THÈSE DE DOCTORAT

de

L'UNIVERSITÉ DE PARIS 6 PIERRE ET MARIE CURIE

Spécialité

SYSTÈMES INFORMATIQUES

présentée par

Matthias GROSSGLAUSER

pour obtenir le grade de DOCTEUR de l'UNIVERSITÉ PARIS 6

Sujet de la thèse:

Contrôle des Ressources de Réseaux sur des Échelles Temporelles Multiples

Soutenue le 21 avril 1998 devant le jury composé de:

MM.	Prof. Jim Kurose	UMass	Rapporteurs
	Prof. Jean-Yves Le Boudec	EPFL	
	Prof. Jean Walrand	UC Berkeley	
MM.	Dr. Jean-Chrysostome Bolot	INRIA	Examinateurs
	Prof. Serge Fdida	U Paris Pierre et Marie Curie	
	Dr. James Roberts	CNET	

Für d'Mère und für e Röbi

Remerciements

Je remercie avant tout **Jean-Chrysostome Bolot** pour avoir encadré ma thèse ainsi que pour son soutien et sa disponibilité. J'ai toujours eu un plaisir énorme à travailler avec lui, tant du point de vue scientifique que humain. Il a trouvé le compromis idéal entre la poursuite de mes propres idées, mes choix de collaborations, et le soutien dans ma quête pour la *victoire de l'esprit sur la matière...* En temps de détresse et frustration, il avait toujours sa *petite prune* pour me détendre et des mots de consolation ("don't worry... [your talk] can only get better!") pour me relancer. Je pars d'ici en espérant qu'il y aura encore beaucoup de "nos papiers" qui verront le jour![1]

Cette thèse a été en grande partie financée par le **Centre National d'Études en Télécommunications (CNET)** de France Telecom. **Jim Roberts**, notre interlocuteur au CNET, était un partenaire idéal. Il nous a laissé poursuivre nos projets de recherche en fonction de leurs intérêts tout en nous offrant un soutien très compétent.

Je suis très reconnaissant envers **Serge Fdida** pour avoir été mon directeur de thèse officiel. J'ai beaucoup apprécié nos interactions scientifiques ainsi que son aide dans les démarches administratives.

Je tiens à remercier **David Tse** pour m'avoir accueilli pour un stage de 4 mois à l'université de Berkeley en automne 1996. Notre collaboration sur le contrôle d'admission basé sur les mesures a été fructueuse et très enrichissante.

Plusieurs séjours dans AT&T Bell Laboratories sous la responsabilité de **S. Keshav** en 1994 et de **K. K. Ramakrishnan** en 1995 m'ont donné le goût de la recherche et ont suscité mon intérêt à faire une thèse. J'espère vivement que nous continuerons à collaborer dans l'avenir.

Je remercie les rapporteurs de thèse, **Jim Kurose**, **Jean-Yves Le Boudec**, et **Jean Walrand** pour leurs efforts. Leurs commentaires ont beaucoup contribué à l'amélioration du présent document, et ont approfondi ma vision du sujet.

Les membres du projet MISTRAL, en particulier **François Baccelli**, **Alain Jean-Marie** et **Philippe Nain**, ont montré une grande disponibilité pour discuter d'aspects théoriques en me faisant profiter de leurs connaissances approfondies dans ce domaine.

Cette thèse ainsi que beaucoup d'autres documents, rapports etc. n'auraient jamais vu le jour sans le support logistique de **Françoise Trucas**, son efficacité et sa persévérance dans ses rappels à respecter des dates limites.

Patrick Cipière a pu résoudre sans faille tous les problèmes techniques rencontrés. Grâce à lui, j'ai pu me munir d'un jargon technique qui me permettait de mieux suivre les discussions entre les nombreux "hackers" de notre groupe.

Les trois années passées parmi les collègues du projet RODEO resteront inoubliables. Il régnait une atmosphère détendue et amicale, à l'exception des nombreuses séances de Quake "deathmatch", Mimaze, F22 Raptor, Screamer... Je remercie particulièrement **Emmanuel Duros** pour avoir été mon "prof de français" pendant la rédaction de cette thèse. S'il reste des erreurs de français (ce qui est fort improbable), prière de prendre directement contact avec lui.

Sophia Antipolis, le 21 Avril 1998.

[1] Avertissement: ce paragraphe ne peut être entièrement apprécié que par J-C. Bolot et l'auteur :-)

Abstract

Real-time multimedia applications require resource allocation because of their quality of service (QoS) requirements. On the other hand, network efficiency depends crucially on the degree of resource sharing inside the network. A key problem in concurrently achieving both goals is caused by the fluctuation over multiple time-scales of the traffic bandwidth emitted by these applications, because it makes it hard to predict resource requirements with sufficient accuracy. This in turn requires a careful design of control mechanisms so that they cover all time-scales.

In this thesis, we examine resource control over three natural time-scales. On the packet time-scale, we evaluate the performance of traffic smoothing as a mechanism to accommodate bandwidth fluctuation. We show that there exists a correlation horizon that separates relevant from irrelevant fluctuation time-scales for the purpose of performance prediction. This illustrates the general principle that the traffic, system, and performance metric time-scales together determine the set of candidate traffic models.

On the burst time-scale, we advocate renegotiation as an efficient mechanism to accommodate fluctuations over time-scales beyond the correlation horizon. A new network service model called RCBR (Renegotiated Constant Bit Rate) combines network simplicity with desirable quality of service guarantees, while achieving much of the potential statistical multiplexing gain of bursty traffic.

On the flow time-scale, we discuss measurement-based admission control as a means of relieving the application of the burden of a-priori traffic specification. The impact of measurement uncertainty and the interplay of burst and flow time-scale dynamics are captured in a single analytical model. This model provides useful guidelines for robust admission control in the presence of unknown traffic statistics. A key result shows that robustness can be achieved by matching the traffic measurement time-scale to the critical time-scale of the system, which depends on the average flow holding time and the system size.

We then address two issues in multicast transmission. First, we extend the RCBR service model to multicast flows to illustrate the tradeoff between network and application complexity. In particular, we compare the complexity of applications in a best effort network and in a network offering controlled sharing through renegotiation. For hierarchically encoded flows, we examine the issue of making the network aware of the dependencies between flows that make up the layers of a single session. Second, we address the problem of feedback implosion in multicast. One approach to feedback implosion avoidance relies on delaying feedback at the receivers. We present a distributed algorithm to compute optimal deterministic timeouts for each receiver in a multicast tree as a function of the tree topology and the sender-to-receiver round-trip delays. This algorithm is fully distributed, because it relies only on local topology information and local communication for the timeout computation. Furthermore, the set of timeouts is optimal with respect to the maximum response time.

Résumé

Les applications multimédia ont des besoins de qualité de service qui nécessitent l'utilisation de réseaux permettant l'allocation de ressources. Cependant, l'efficacité d'un réseau dépend de façon cruciale du degré de partage des ressources. Satisfaire ces deux contraintes est difficile à cause des fluctuations à multiples échelles temporelles du débit du trafic généré par ces applications. En effet, cela complique la prédiction des demandes en ressources avec une précision suffisante. Il faut mettre en œuvre des mécanismes de contrôle de façon à couvrir toutes les échelles temporelles.

Dans cette thèse, nous étudions le contrôle de ressources sur trois échelles temporelles naturelles. A l'échelle du paquet, nous évaluons la performance du lissage du trafic comme mécanisme pour accommoder les fluctuations de débit. Nous montrons l'existence d'un horizon de corrélation; seules les échelles temporelles plus courtes que cet horizon de corrélation sont pertinentes pour la prédiction de performance. Ceci illustre le principe plus général que le trafic, le système, et la métrique de performance doivent être pris en compte pour déterminer l'ensemble de modèles de trafic possibles.

A l'échelle de la rafale, nous proposons la renégociation comme mécanisme pour accommoder les fluctuations sur une échelle temporelle plus longue que l'horizon de corrélation. Un nouveau modèle de service appelé RCBR ("Renegotiated Constant Bit Rate") combine la simplicité du réseau avec un contrôle efficace de la qualité de service, tout en extrayant la quasi totalité du gain de multiplexage statistique du trafic.

A l'échelle du flux (ou de la connexion), nous décrivons le contrôle d'admission basé sur les mesures, qui simplifie le problème de la spécification du trafic pour l'application ou pour l'utilisateur. Nous proposons un modèle qui prend en compte l'impact de l'incertitude dans les mesures et l'interférence de l'échelle temporelle de rafale et de flux. Ce modèle permet d'obtenir des résultats fondamentaux pour la conception de mécanismes de contrôle d'admission basé sur les mesures robuste. Un résultat important issu de cette étude est que cette robustesse peut être atteinte en faisant correspondre l'échelle temporelle de la mesure (la longueur de la fenêtre d'estimation) à l'échelle temporelle critique du système, qui dépend de la durée moyenne des flux ainsi que de la taille du système.

Finalement, nous étudions deux problèmes dans la communication multipoint. Premièrement, nous étendons le modèle de service RCBR au cas des flux multipoint pour illustrer le compromis entre la complexité dans le réseau et la complexité dans les applications. Nous comparons la complexité d'une application qui utilise un réseau "best effort" avec celle d'une application qui utilise un réseau offrant des garanties renégociables. Nous nous intéressons aussi au problème de la dépendance des flux qui résulte de la compression en couches hiérarchiques d'une source d'information.

Deuxièmement, nous abordons le problème de l'implosion de l'information de contre-réaction dans la communication multipoint. Une approche pour éviter ce problème est de retarder l'envoi de l'information aux destinations. Nous présentons un algorithme distribué pour calculer un délai fixe pour chaque destination dans un arbre multipoint, qui est une fonction de la topologie de l'arbre est des délais source-destination. Cet algorithme est entièrement distribué car chaque nœud de l'arbre a seulement besoin d'informations locales et de communication entre nœuds voisins. De plus, l'ensemble des délais calculés par cet algorithme est optimal dans le sens qu'il minimise le temps de réponse maximal.

Contents

1 Résumé de la Thèse **13**

1.1 Chapitre 2: Introduction . . . 13

1.2 Chapitre 3: La Pertinence de la Dépendance à Longue Portée dans le Trafic Réseau . . . 18

1.3 Chapitre 4: RCBR . . . 22

1.4 Chapitre 5: Le Contrôle d'Admission Basé sur les Mesures . . . 23

1.5 Chapitre 6: Sur les Modèles de Service pour la Transmission Multipoint dans un Environnement Hétérogène . . . 27

1.6 Chapitre 7: Contrôle de l'Implosion de Contre-Réaction en Multipoint . . . 29

2 Introduction and Overview **31**

3 Control Time-Scales and Network Traffic Modelling **37**

3.1 Introduction . . . 37

3.2 Model Description . . . 39

3.3 Numerical Results . . . 43

3.4 Discussion . . . 52

3.5 Conclusion . . . 54

4 Renegotiated Constant Bit Rate Service **57**

4.1 Introduction . . . 57

4.2 Performance Problems for Multiple Time-Scale Sources . . . 58

4.3 The RCBR Scheme . . . 60

4.3.1 RCBR Service Description . . . 60

4.3.2 Implementation . . . 61

4.3.3 Scaling . . . 62

4.4 Computation of Renegotiation Schedules . . . 62

4.4.1 Optimal Renegotiation Schedule . . . 62

4.4.2 Causal Renegotiation Schedule . . . 67

4.5 Performance of RCBR . . . 68

4.5.1 Analysis of a Multiple Time-Scale Model . . . 69

4.5.2 Experimental results . . . 72

4.6 Related Work . . . 73

4.7 Discussion and Conclusions . . . 75

5 Measurement-Based Admission Control **79**
5.1 Introduction . . . 79
5.2 Models . . . 82
5.3 Impulsive Load Models . . . 83
5.3.1 Infinite Flow Holding Time . . . 83
5.3.2 Finite Holding Time . . . 89
5.4 The Continuous Load Model . . . 91
5.4.1 Memoryless MBAC . . . 91
5.4.2 Analysis of Overflow Probability . . . 94
5.4.3 MBAC with Memory . . . 96
5.5 Discussions and Simulations . . . 98
5.5.1 Summary of Results . . . 99
5.5.2 Simulation Results . . . 100
5.5.3 Memory Window Size and Robust MBAC . . . 100
5.5.4 Heterogeneous Flows . . . 106
5.6 Related Work . . . 106
5.7 Conclusions . . . 107
5.8 Appendix A: Proof of Proposition 2 . . . 108
5.9 Appendix B: Computation of $\sigma_m^2(t)$ . . . 109

6 Issues in Multicast Transmission in Heterogeneous Environments **111**
6.1 Introduction . . . 111
6.2 Exploring the Design Space . . . 112
6.2.1 Impact of Service Model on Network and Application Complexity . . . 112
6.2.2 The Notion of Resource Waste . . . 114
6.2.3 The Relevance of External Waste . . . 116
6.2.4 Resource Waste and Channel Estimation in the Current Internet . . . 117
6.2.5 Hiding Resource Waste . . . 118
6.2.6 Summary . . . 119
6.3 Application: The Design of the GRCBR Architecture . . . 119
6.3.1 Basic Ideas . . . 119
6.3.2 Signalling Issues in GRCBR . . . 123
6.4 Conclusion . . . 126

7 Multicast Feedback Implosion Avoidance **129**
7.1 Introduction . . . 129
7.2 Related Work . . . 130
7.3 Context . . . 131
7.4 Computing Timeouts . . . 132
7.4.1 How to Choose Consistent Timeouts . . . 132
7.4.2 Optimal Timeouts . . . 135
7.4.3 Computing an Optimal Receiver Allocation . . . 136

7.5 Distributed Optimal Timeout Computation 142
7.6 Discussion 144
7.6.1 Lost NACKs and Delay Variation 144
7.6.2 Timeouts and Response Times 145
7.6.3 Dynamic Membership Changes 148
7.7 Conclusion and Future Work 149

8 Conclusion and Outlook **151**
8.1 Summary of Contributions 151
8.2 Future Directions 152

Chapter 1

Résumé de la Thèse

Ce chapitre contient un résumé en français de cette thèse.

1.1 Chapitre 2: Introduction

Jusqu'à un passé récent, deux infrastructures de communication fondamentalement différentes ont coexisté, à savoir le réseau téléphonique et les réseaux de données comme l'Internet. Pour assurer une qualité perceptuelle satisfaisante pour l'utilisateur, une connexion téléphonique a besoin de garanties de qualité de service telles que des bornes sur le délai, la gigue, et la probabilité de perte d'information. Pour obtenir ces garanties, le réseau téléphonique réserve des ressources pour chaque appel (voir Fig. 2.1). Le trafic émis par un téléphone étant très régulier, puisqu'il s'agit d'un flux à débit constant ("Constant Bit Rate", CBR) de 64kbps, la ressource réservée est de la bande passante, plus précisement une bande passante constante et égale à 64 kb/s pendant toute la durée d'un appel.

Les applications de type données, comme le transfert de fichiers ou l'accès à la Toile, ont des besoins de qualité de service à la fois moins stricts et plus flous. Souvent, il est suffisant d'offrir une bande passante moyenne minimale sur un intervalle de temps suffisant pour assurer une qualité perceptuelle satisfaisante à l'utilisateur. Par conséquent, il n'est pas nécessaire de réserver des ressources de façon statique pour chaque connexion. De plus, le trafic émis par les applications de données a tendance à être très irrégulier, indiquant que les besoins en ressources de ces applications fluctuent dans le temps. Une allocation statique de bande passante (et plus généralement de ressources) dans ce cas n'est donc pas appropriée. Au contraire, les réseaux de données nécessitent des mécanismes pour accomoder ces fluctuations, en particulier des mécanismes de gestion de buffers et des disciplines d'ordonnancement de paquets (voir Fig. 1.1).

La différence fondamentale entre ces deux types de réseau est l'échelle temporelle de la fluctuation de la charge du réseau et des mécanismes de contrôle correspondants (voir Fig. 1.2). La charge du réseau téléphonique fluctue par suite des arrivées de nouvelles connexions et des départs des connexions terminées; le mécanisme de contrôle associé est le *contrôle d'admission*, qui refuse l'établissement d'une nouvelle connexion pendant une période de surcharge, c'est à dire lorsque la bande passante demandée par cette connexion n'est pas disponible dans le réseau. Ainsi, il existe une échelle temporelle naturelle associée à ce mécanisme de contrôle, qui dépend de la fréquence d'arrivée des connexions ainsi que de leur durée de vie. La charge d'un réseau de données fluctue suite aux arrivées et aux départs des paquets, qui sont elles même fonctions des besoins intrinsèques en bande passante des

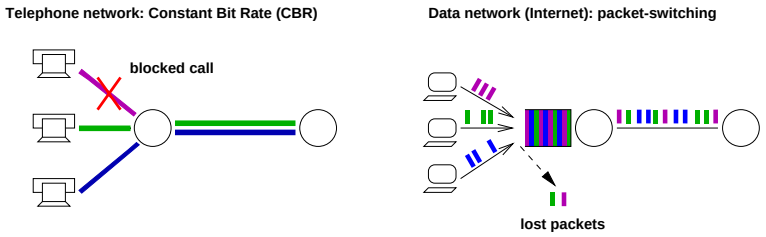

Figure 1.1: Le réseau téléphonique et les réseaux de données.

applications; les mécanismes de contrôle sont la gestion des buffers et l'ordonnancement, qui peuvent rejeter des paquets en période de surcharge. Ainsi, il existe une échelle temporelle naturelle associée à ces mécanismes de contrôle, qui dépend de la fréquence d'arrivée des paquets ainsi que de leur temps de service. Il n'y a pas de concept clair de connexion ou de flux, et donc pas de contrôle d'admission.

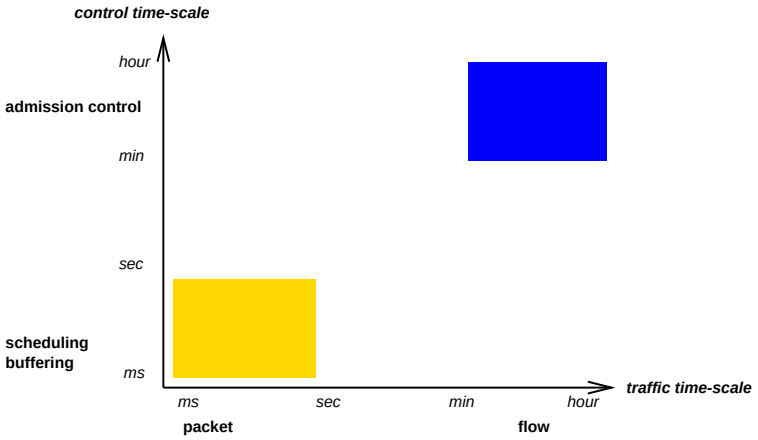

Figure 1.2: Les échelles temporelles du trafic et du contrôle pour le réseau téléphonique et pour les réseaux de données.

La situation a beaucoup évolué ces dernières années. Les nouvelles applications multimédia temps réel ont des besoins de qualité de service similaires à ceux d'une connexion téléphonique. Par exemple, des applications utilisant la voix et la vidéo comprimée comme les outils de vidéoconférence, la télémédecine, et le téléenseignement, nécessitent un faible délai à cause de leur caractère interactif, et une faible probabilité de perte à cause de l'impact perceptuel considérable d'une perte d'information sur la qualité auditive ou visuelle. D'autre part, ces applications génèrent un trafic très irrégulier, similaire à celui généré par les applications de type données. Une solution allouant de façon statique assez de ressources à chaque flux comme dans le réseau téléphonique mènerait à une très faible utilisation de ces ressources. Pour améliorer cette utilisation, les ressources, comme la bande passante ou les buffers, doivent être sur-allouées. L'augmentation en utilisation est appelée *gain de multiplexage statistique*. Mais ceci nécessite l'utilisation judicieuse de mécanismes de contrôle. Ces mécanismes doivent couvrir toutes les échelles temporelles mentionnées plus haut. Dans les chapitres 3 à 5, nous discutons le contrôle de ressources sur trois échelles temporelles naturelles: celle des paquets, celle des rafales, et celle de la

durée de vie des flux.

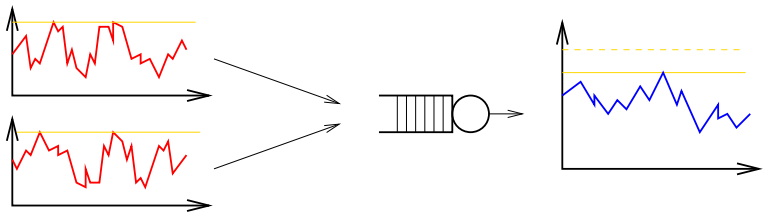

Figure 1.3: Une illustration du gain de multiplexage statistique.

L'échelle temporelle du paquet est la plus petite échelle de temps naturelle dans un réseau. Elle est d'un intérêt particulier depuis qu'un grand nombre d'études récentes ont mis en évidence le fait que le trafic observé dans les réseaux a des propriétés très générales d'auto-similarité et de dépendance à longue portée (LRD). Un processus stochastique possède de la dépendance à longue portée si sa fonction d'autocorrélation n'est pas sommable. Intuitivement, ceci signifie que le processus fluctue sur une large gamme d'échelles temporelles. Cependant, malgré un grand nombre de mesures concluantes, l'observation de la LRD dans de nombreux domaines des sciences naturelles et physiques a souvent été accueillie par la résistance ou tout au moins la surprise. La raison de cette attitude est la difficulté de trouver des explications physiques satisfaisantes pour ce phénomène. En effet, un système physique générant de la LRD transmet de l'information sur l'état du système sur de très longues échelles temporelles. Pour des processus naturels comme le niveau d'un fleuve par exemple, il est difficile de comprendre où réside la "mémoire" du comportement passé du processus.

La situation en télécommunications est comparable à celle existant dans d'autres domaines. La LRD a été observée dans un grand nombre de réseaux et pour un grand nombre de sources de trafic. De plus, des travaux sur les files d'attente avec un trafic LRD en entrée ont montré que la performance asymptotique d'une telle file d'attente est fondamentalement différente d'une file sans LRD. D'autre part, une chaîne de Markov (finie) ne peut pas modéliser un processus LRD, car la fonction d'autocorrélation d'un modèle Markovien est toujours asymptotiquement exponentielle. Mais la quasi-totalité des modèles traditionnels de trafic de télécommunication sont Markovien pour des raisons de facilité mathématique.

Cette différence entre observations et modèles traditionnels pourrait suggérer que les modèles Markovien ne sont pas adéquats pour modéliser le trafic réseau. Cependant, jusqu'à la découverte de LRD dans le trafic, ces modèles semblaient donner des résultats corrects. La situation est donc un peu confuse, et il n'existe pas encore de consensus général sur la meilleure façon de modéliser les propriétés de LRD et leur impact pratique.

Malheureusement, les travaux récents de modélisation n'apportent pas de réponse satisfaisante à ces questions. Nous pensons en particulier qu'ils ne prennent pas en compte deux paramètres importants, qui sont d'une part les statistiques de premier ordre comme les distributions marginales du trafic d'arrivée, et d'autre part le fait que les problèmes d'évaluation de performance ne font intervenir qu'un nombre fini d'échelles de temps (par exemple l'utilisation de files d'attente de taille finie dans les réseaux opérationnels fait intervenir l'échelle de temps naturelle qui est le temps nécessaire à la file pour se vider).

Nous développons et évaluons dans le chapitre 3 un modèle dans lequel ces paramètres

peuvent être facilement contrôlés. Notre modèle est un modèle de type fluide modulé dans lequel la fonction de corrélation du débit du fluide est asymptotiquement autosimilaire (de deuxième ordre) pour un paramètre de Hurst donné, mais tombe brutalement à zéro pour une valeur donnée du décalage, que l'on appelle décalage de coupure. Nous développons une procédure numérique efficace pour évaluer les mesures de performance d'une file d'attente de taille finie alimentée par ce modèle de trafic. Nous utilisons cette procédure pour évaluer une mesure de performance intéressante, à savoir le taux de perte, et pour un grand nombre de distributions marginales, valeurs du paramètre de Hurst, valeurs du décalage de coupure, et tailles maximales de file.

Notre résultat principal est que la mesure de performance dépend non seulement de la structure de corrélation du trafic fluide, mais également de l'échelle temporelle propre au système étudié. Dans le cas d'une file de taille finie, cette échelle temporelle dépend de la taille maximale de la file. En pratique, pour une file de taille maximale finie, nous observons que l'impact de la corrélation sur le taux de perte devient nul au dela d'une échelle temporelle que nous appellons *horizon de corrélation.* Aussi, nous montrons que le taux de perte dépend très fortement de la distribution marginale du processus fluide, et qu'il est difficile de réduire les pertes simplement en augmentant la taille de la file. Nous proposons plutôt d'utiliser des mécanismes de contrôle de trafic et de multiplexage statistique (et donc de partage de buffer entre plusieurs flux LRD).

Puisque les fluctuations du trafic réseau comportent des composantes qui sont sur une échelle temporelle trop longue pour être lissée par un buffer de taille raisonnable, le lissage *par flux* n'est donc pas suffisant pour arriver à une bonne utilisation des ressources. Comme mentionné ci-dessus, nous devons donc partager les ressources entre différents flux. La question principale concerne quelles ressources devraient être partagées. Dans un réseau comme l'Internet actuel, les buffers et la bande passante sont partagés parmi tous les flux de façon non contrôlée (tout au moins de façon explicite). Mais comme nous désirons assurer une qualité de service, nous devons partager ces ressources de manière contrôlée. La plupart des propositions pour des services de transport pour de tels flux, comme le modèle "integrated services" dans la communauté Internet, ou le modèle ATM VBR, font l'hypothèse que le partage des buffers *et* le partage de la bande passante sont nécessaires. Ces services sont typiquement assez complexe et nécessitent des disciplines d'ordonnancement élaborées, des files d'attente par flux, et des tests d'admission non triviaux. L'alternative que nous proposons simplifie considérablement la complexité du réseau en *partageant seulement la bande passante, mais pas les buffers.*

A l'échelle de la rafale, *nous proposons la renégotiation comme mécanisme pour accomoder des fluctuations sur de multiples échelles temporelles au-delà l'horizon de corrélation. Un nouveau modèle de service appelé RCBR (Renegotiated Constant Bit Rate) combine la simplicité du réseau avec des garanties de qualité de service désirables, tout en extrayant la majorité du gain de multiplexage statistique du trafic transporté.*

Les sources utilisant RCBR se voient offertes une abstraction de service, en l'occurence d'un buffer de taille fixe qui est drainé à un débit constant. Elles peuvent dynamiquement renégocier ce débit afin de l'adapter à leurs besoins instantanés. Tout le trafic entrant le réseau est CBR, ce qui signifie qu'un réseau qui implémente RCBR est tout aussi simple qu'un réseau purement CBR au niveau des buffers et de l'ordonnancement. Bien sûr, ce service est parfait pour des applications CBR comme la téléphonie. Mais nous montrons que ce service est également adéquat pour des applications avec trafic très variable comme la vidéo interactive ou la vidéo à la demande.

Une source RCBR doit décider quand elle veut renégocier son débit et quel nouveau débit

demander. Nous présentons un algorithme pour calculer la séquence optimale de renégociations pour une vidéo pré-enregistrée, ainsi que des heuristiques pour le cas interactif. Des résultats de simulation montrent que RCBR est capable d'extraire la quasi-totalité du gain de multiplexage statistique disponible. De plus, des mécanismes de contrôle d'admission simples sont suffisants pour maintenir la probablité de blocage d'une renégotiation endessous d'un seuil désiré. *Les résultats montrent donc que RCBR est un service simple, pratique, et efficace pour transporter du trafic à multiples échelles temporelles.*

A l'échelle du flux, nous étudions le contrôle d'admission basé sur les mesures, afin de simplifier le problème de la spécification du trafic pour l'application ou pour l'utilisateur. Avec le contrôle d'admission traditionnel, les décisions d'admission ou de rejet sont basées sur des descripteurs de trafic *a-priori*, comme la bande passante effective ou les paramètres de "leaky bucket". Le problème de cette approche est qu'il est souvent difficile pour l'utilisateur ou l'application de trouver des descripteurs précis avant l'établissement même de la connexion. Ceci est particulièrement vrai pour les communications interactives. Par exemple, les descripteurs d'une vidéoconférence dépendront des mouvements des participants pendant la conférence, ce qui est difficile à prévoir. Le contrôle d'admission basé sur les mesures (MBAC, "Measurement-Based Admission Control") est un mécanisme efficace pour offrir des garanties de qualité de service sans demander à l'utilisateur la spécification exacte de son trafic. Le MBAC permet ceci *en apprenant les caractéristiques du trafic lui-même.* L'avantage est clair: un descripteur mal choisi est remplacé par une spécification apprise par le MBAC une fois le flux dans le système, ce qui évite que le descripteur initial dégrade la performance du système pendant toute la durée de vie du flux.

Malgré ses avantages clairs, plusieurs aspects d'un tel dispositif doivent être étudiés afin de rendre un MBAC *robuste*, c'est à dire de lui permettre d'offir les mêmes types de garanties qu'un contrôle d'admission traditionnel. Nous proposons un modèle qui prend en compte l'impact de l'incertitude dans les mesures et l'interférence des échelles temporelles de niveau rafale et flux. Ce modèle permet d'obtenir des résultats fondamentaux pour la conception de mécanismes robustes de contrôle d'admission basé sur les mesures. Un résultat fondamental issu de cette étude est que cette robustesse peut être atteinte en faisant correspondre l'échelle temporelle de la mesure (la longueur de la fenêtre d'estimation) à l'échelle temporelle critique du système, qui dépend de la durée moyenne des flux ainsi que de la taille du système.

Jusqu'à présent, nous avons considéré des *aspects temporaux* du problème de contrôle de ressources. Nous discutons dans la suite des *aspects spatiaux* qui se posent dans le contexte de la communication multipoint, qui permet la transmission efficace de données vers plusieurs destinations. Cette efficacité vient du fait que la duplication des données s'effectue à l'intérieur de l'arbre multpoint, pour éviter qu'un paquet ne traverse un lien donné plus d'une fois.

Nous discutons deux problèmes en communication multipoint dans les chapitres 6 et 7. Dans le chapitre 6, *nous étendons le modèle de service RCBR à la communication multipoint pour illustrer le compromis entre la complexité dans le réseau et la complexité dans l'application.* En particulier, nous comparons la complexité des applications dans un réseau "best-effort" avec celle dans un réseau qui offre le partage contrôlé par le biais de la renégotiation. Pour des flux encodés hiérarchiquement, nous examinons s'il est judicieux de mettre de l'état supplémentaire dans le réseau afin de l'informer de la dépendance entre les flux issus d'une même session. Ces applications utilisent souvent un codage hiérachique pour décomposer l'information en sous-couches qui sont transportées par des flux séparés, afin d'accomoder l'hétérogénéité de grands groupes multipoint. Étant donné qu'un service

de transport multipoint doit de toute façon maintenir de l'état dans le réseau, le coût incrémental pour déployer cet état supplémentaire pourrait être acceptable, surtout considérant que les problèmes d'adaptation de bout en bout, comme l'adaptation du "playout point", la "forward error correction" (FEC) et le contrôle de débit, deviennent plus complexe dans une approche multipoint/multicouches.

Dans le chapitre 7, *nous étudions le problème de l'implosion d'information de contre-réaction dans les protocols de contrôle multipoint, comme le contrôle de congestion ou le transport fiable.* Pour éviter ce problème, le nombre de destinations qui renvoyent de l'information de contre-réaction en cas d'un événement comme une perte de paquet doit rester limité. Cependant, les observations aux différentes destinations ont tendance à être fortement corrélées car les destinations partagent des ressources à l'intérieur de l'arbre multipoint. Ainsi, la perte d'un seul paquet près de la racine de l'arbre (et donc près de la source) peut se traduire par un très grand nombre d'observations de pertes au niveau des feuilles de l'arbre, c'est à dire aux destinations, dans le cas d'un grand groupe multipoint.

Une approche pour éviter l'implosion consiste à retarder l'envoi de l'information aux destinations. Nous présentons un algorithme distribué pour calculer un délai fixe pour chaque destination dans un arbre multipoint, qui est une fonction de la topologie de l'arbre et des délais source-destination. Cet algorithme est entièrement distribué car chaque n/oe d de l'arbre a seulement besoin d'informations locales et de communication entre noeuds voisins. De plus, l'ensemble des délais calculés par cet algorithme est optimal dans le sens qu'il minimise le temps de réponse maximal.

1.2 Chapitre 3: La Pertinence de la Dépendance à Longue Portée dans le Trafic Réseau

Une observation d'un système physique est généralement considérée dans un but de modélisation comme la réalisation d'un processus stochastique sous-jacent, qui en pratique est supposé être stationnaire et ergodique. L'analyse de tels processus dans un très grand nombre de systèmes dans des domaines aussi variés que la physique, l'économie, la biochimie, ou l'hydrologie montre que beaucoup de ces systèmes exhibent une propriété de corrélation sur toutes les échelles temporelles. Cette propriété est appellée *dépendance à longue portée* (long range dependence LRD en anglais). Cependant, bien que de nombreuses études confirment son existence, la présence de la LRD en général a généré une controverse. Celle ci est avant tout dûe à l'absence d'une explication physique satisfaisante pour ce phénomène. Deux approches sont alors possibles. Une première approche consiste à mettre en question les hypothèses faites sur le processus sous-jacent, en particulier celle de la stationnarité et de l'ergodicité, et à expliquer la présence de la LRD par exemple par la présence de non-stationnarités. Une autre approche consiste à adopter des modèles de type LRD en dépit de l'absence d'explication physique, en se basant sur le fait que l'utilité d'un modèle vient de sa facilité d'utilisation et de capacité à aider à la prise de décision, et non de l'adéquation parfaite entre le modèle et le phénomène physique modélisé.

Malheureusement, il est impossible en pratique d'être certain si une réalisation est stationnaire ou non, et il est donc difficile d'évaluer la validité de la prmière approche. La controverse autour de ces deux approches persiste donc. Il est clair qu'il est préférable qu'un modèle possède le plus grand nombre des propriétés du processus physique. Mais un modèle est fondamentalement un outil de décision, et sa qualité dépend donc de la qualité des décisions qu'il permet de prendre et non de sa proximité à la réalité physique.

La situation dans le domaine des systèmes de communication, et des réseaux en particulier, est comparable à celle décrite ci-dessus. Un grand nombre d'études récentes, dont la plus connue est l'étude du trafic Ethernet réalisée à Bellcore, ont mis en évidence le fait que le trafic observé dans les réseaux a des propriétés très générales d'auto-similarité et de dépendance à longue portée (ces propriétés sont souvent résumées par la formule un peu abrupte "le trafic est fractal").

Cependant, le débat sur le choix adéquat de modèle pour ce type de trafic réseau persiste, et les deux approches mentionnées plus haut ont été prises. La première approche argumente que la LRD observée pourrait être due à des non-stationnarités dans les données comme la superposition de "level shifts" ou d'impulsions de Dirac avec des processus stochastiques sans LRD. La deuxième approche propose d'utiliser des modèles dont la structure de corrélation contient de la LRD. Toutefois, ces modèles sont difficiles à analyser analytiquement; de plus, ils ne fournissent pas d'indices supplémentaires sur les raisons physiques de l'observation de LRD. Ceci explique en partie le fait qu'une grande partie de l'activité de modélisation de trafic réseau continue à être basée sur des modèles markoviennes. La situation dans le domaine de la modélisation reste donc floue.

D'autre part, il n'existe pas encore de consensus général l'impact pratique des propriétés de LRD. Les travaux récents de modélisation n'apportent pas de réponse satisfaisante à ces questions pour 3 raisons principales. Tout d'abord, ces travaux ne considèrent que des modèles de files d'attente de taille infinie. Ils n'aboutissent donc qu'à des résultats asymptotiques. Ensuite, ils ne prennent pas en compte des paramètres importants comme les distributions marginales du trafic. Enfin, ils ne prennent pas en compte le fait que les problèmes de performance ne font intervenir en pratique qu'un nombre fini d'échelles de temps.

Le but de notre travail consiste donc à évaluer l'impact pratique de la LRD dans une situation pratique, et de comparer cet impact avec celui d'autres paramètres. Deux paramètres s'avèrent particulièrement importants, qui sont i) la distribution marginale du processus d'arrivée, et ii) le spectre fini d'échelles temporelles pertinentes pour le problème d'évaluation de performance considéré.

Nous avons développé un modèle de source qui permet de contrôler ces paramètres. Il s'agit d'un modèle fluide, c'est à dire un processus stochastique $\{X_t\}$ qui décrit le débit de la source (ou taux) au temps t. Nous supposons que X_t peut prendre un ensemble fini de valeurs $\{\lambda_1, \ldots, \lambda_M\}$. De plus, nous supposons que le processus fluide est constant par intervalles. En fait, le taux reste constant sur des intervalles dont la longeur est donnée par les arrivées d'un processus ponctuel stationnaire sous-jacent. Les interarrivées de ce processus, de même que le taux sur les intervalles, sont individuellement et mutuellement indépendents.

Il est possible de montrer que la fonction d'autocorrélation de ce processus est donnée par

$$\phi(t) = p(t)\Pi\Lambda^2(1-\Pi)^T, \tag{1.1}$$

ou $\Pi = (\pi_1, \ldots, \pi_M)$ est le vecteur qui définit la distribution marginale du taux, c'est à dire π_i est la probabilité qu'à un instant arbitraire le taux soit égal à λ_i. La matrice $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_M)$ est la matrice des taux, et $p(t)$ est la probabilité qu'il n'y ait pas d'arrivée dans une intervalle de longeur t dans le processus ponctuel sous-jacent. Il s'avère que $p(t)$, et donc la fonction d'autocorrélation, dépend directement de la distribution d'interarrivée

du processus ponctuel, que nous choisissons comme

$$F_T(t) = \begin{cases} \left(\frac{t+\theta}{\theta}\right)^{-\alpha} & \text{si } t < T_c \\ 0 & \text{sinon} \end{cases} \tag{1.2}$$

Nous appellons cette distribution une distribution *Pareto tronquée*. Pour $T_c = \infty$, la fonction d'autocorrélation est celle d'un processus asymptotiquement auto-similaire de deuxième ordre avec paramètre de Hurst $H = (3-\alpha)/2$. Pour T_c fini, l'autocorrélation disparait au-delà d'une distance de T_c, mais se comporte identiquement pour des distances inférieures à T_c. Nous appellons donc T_c la distance de coupure de corrélation.

Nous voulons ensuite étudier la performance d'une file d'attente de taille finie alimentée par le trafic décrit ci-dessus. La mesure de performance qui nous intéresse est le taux de perte, c'est à dire la fraction de travail (de fluide) en arrivée qui est perdue suite à un débordement de la file. Malheureusement, il s'avère impossible de résoudre le problème analytiquement. Par contre, nous observons qu'il est possible de calculer le taux de perte en connaissant la distribution stationnaire de la file seulement aux instants d'arrivée du processus ponctuel. Cette observation est à la base d'une procédure numérique efficace pour calculer le taux de perte détaillée dans le chapitre 3. sur nos pages Web). Nous avons utilisé cette procédure pour obtenir les résultats présentés dans ce chapitre.

Plus précisément, nous étudions la mesure de performance choisie, à savoir le taux de perte, en fonction de différents paramètres du modèle qui sont la taille de la file, la distance de coupure de la corrélation T_c, le paramètre du Hurst H, et la distribution marginale Π.

Nous identifions les paramètres du modèle à partir de deux traces de trafic réseau qui contiennent de la LRD. La première trace contient une heure de vidéo JPEG (canal MTV), et elle est caractérisée par un paramètre de Hurst $H = 0.83$. La deuxième trace contient les mesures classiques Ethernet de Bellcore, pour lesquelles $H = 0.9$. Nous faisons correspondre la distribution marginale du modèle en calculant un histogramme des débit dans la trace, et en utilisant cet histogramme comme la distribution Π de débit du modèle. Nous calculons ensuite le temps moyen que passe la trace dans un intervalle (*bin*) de l'histogramme avant de changer d'intervalle. Nous faisons correpondre le temps moyen d'interarrivée du modèle, c'est à dire la longueur moyenne des intervalles, à cette valeur, moyennant le paramètre θ de $F_T(t)$. Finalement, le paramètre de Hurst est contenu explicitement dans le modèle.

Une fois les paramètres du modèle identifiés, nous pouvons commencer nos expériences. Nous avons effectué trois types d'expériences.

Le premier type examine l'impact sur le taux de perte de la taille de la file et de la distance de corrélation T_c. Nous avons obtenu deux résultats intéressants. Premièrement, pour chaque taille de file, le taux de perte reste constant quand la distance de corrélation T_c dépasse une certaine valeur. En d'autres termes, au-delà de cette valeur, rajouter de la corrélation n'a plus d'impact sur la performance (et donc le taux de perte reste constant même quand $T_c \to \infty$). Nous appellons cette valeur limite l'horizon de corrélation (correlation horizon CH). Deuxièmement, la décroissance du taux de perte en fonction de la taille de la file dépend de la valeur de la distance de corrélation. Ceci est une manifestation du phénomène "d'inefficacité de buffer" déjà observée par ailleurs. Ce comportement n'est pas surprenant, car un trafic LRD comporte des rafales occasionnelles très longues, qui ne peuvent être absorbés même par des files de taille élevée.

Le deuxième type d'expérience examine le taux de perte en fonction du paramètre de Hurst et de la distribution marginale du débit. Pour étudier l'impact de la distribution marginale, nous transformons cette distribution en effectuant un changement d'échelle autour de sa

moyenne. Ceci peut se faire en gardant le vecteur Π, mais en remplacent λ_i par $\lambda_i' = E[\lambda] + \alpha(\lambda_i - E[\lambda])$, où $E[\lambda]$ est de débit moyen. Varier α revient alors à varier la "largeur" de la distribution. On observe alors que le taux de perte est très sensible à des variations de α, en dépit de la présence de LRD dans le trafic. Et en fait, la variation du paramètre α entre 0.5 et 1.5 a un impact plus grand que n'aurait une variation du paramètre de Hurst entre 0.1 et 1.

Ce résultat est confirmé dans le troisième type d'expérience dans lequel nous étudions l'impact sur le taux de perte de la distribution marginale et de la taille du buffer. Nous observons cette fois que l'impact de la distribution marginale sur le taux de perte domine celui d'un autre paramètre, ici la taille du buffer.

Ces observations ont deux conséquences importantes. Premièrement, la distribution marginale du débit est un paramètre crucial dans la modélisation. Nous ne pouvons pas espérer développer des modèles de trafic qui donnent des résultats de prédiction de performance satisfaisants s'il ne prennent pas en compte cette distribution marginale. Deuxièmement, l'utilisation des mécanismes de contrôle de trafic qui réduisent la largeur de la distribution marginale, comme le multiplexage statistique et le contrôle de débit, sont très efficaces pour réduire le taux de perte de flux LRD; ils sont en tout cas plus efficaces que l'augmentation de la taille du buffer.

Notre observation de l'existence d'un horizon de corrélation aide à résoudre la controverse autour de l'importance de la LRD en modélisation. En effet, il a été montré mathématiquement et par simulation que des systèmes de file d'attente alimentés par un trafic LRD se comportent asysmptotiquement très différemment de systèmes alimentés par un trafic markovien. D'autre part, la littérature sur la modélisation markovienne rapporte une bonne prédiction de performance pour des files d'attente de taille finie même en présence de corrélation à longue portée. Nous avons donc montré qu'il existe un horizon de corrélation qui sépare la corrélation pertinente de la corrélation sans impact sur notre mesure de performance en question. Cet horizon de corrélation dépend 1) de la structure de corrélation du trafic d'entrée, 2) du système (la file d'attente de taille finie), et 3) de la mesure de performance (le taux de perte). La taille finie de la file d'attente en effet limite la mémoire du système, c'est à dire la quantité d'information que le système peut accumuler sur le passé du processus. Nous avons trouvé que l'horizon de corrélation est proportionnel à la taille du buffer. Ceci explique pourquoi la situation des files de taille infinie est fondamentalement différente de celle des files de taille finie. Un tel système a une mémoire infinie, il n'y pas d'horizon de corrélation, et la corrélation sur toutes les échelles temporelles a donc un impact sur la performance.

Nous nous sommes concentrés dans cette étude sur une instance particulière de prédiction de performance, où la mesure de performance est le taux de perte. Nous avons montré que la corrélation pertinente est limitée aux échelles temporelles inférieures à un horizon de corrélation. Ceci pourrait suggérer que seuls les modèles de trafic à mémoire finie ont un sens. Mais cette observation signifie plutôt que pour ce type de problème de performance, un ensemble de modèles s'avère possible, à savoir tous ceux qui modélisent correctement la corrélation jusqu'à l'horizon de corrélation. Parmi cet ensemble de modèles, nous somme libres de choisir celui qui est le plus convenable en termes de tractabilité analytique ou de la facilité d'identification de ses paramètres. Par exemple, les modèles de trafic LRD ou auto-similaires sont tout à fait adaptés à la modélisation de systèmes qui n'ont pas d'échelle temporelle propre clairement identifiable. Ceci est le cas lors de la génération de traces. En effet, on ne connait en général pas à l'avance la longueur nécessaire de la trace, ni l'utilisation précise qui en sera faite.

En conclusion, notre travail illustre le fait que l'observation de la propriété de LRD dans le trafic réseau n'implique pas forcément qu'il faille la modéliser. Mais plutôt que le trafic, le système lui-même, et la mesure de performance choisie, doivent être examinés pour déterminer l'échelle temporelle pertinente dans la modélisation, et donc le type de modèle à utiliser. Il est donc inutile d'essayer d'écarter un type de modèle sans étudier le contexte exact dans lequel le modèle sera utilisé.

1.3 Chapitre 4: RCBR

Nous avons montré dans la section précédente que les fluctuations sur de longues échelles temporelles rendent inefficace un modèle de service qui essaie d'absorber ces fluctuations. Pour la vidéo comprimée par exemple, ces fluctuations sont dues aux changements de scène, et elles peuvent typiquement durer des dizaines de secondes voire des minutes. Dans le chapitre 4, nous argumentons qu'un service *renégocié* est le plus adéquat pour accomoder ces fluctuations tout en offrant une qualité de service donnée à l'application. En particulier, nous proposons un service RCBR (Renegociable Constant Bit Rate), c'est à dire une extension du service CBR (constant bit rate = débit constant) avec renégociation possible de la bande passante.

Une application qui utilise le service RCBR a un débit garanti à tout instant. L'application a le droit de demander au réseau un nouveau débit pour l'adapter à ses besoins instantanés et ainsi "suivre" les fluctuations à long terme par exemple du débit video qu'elle génère. L'application envoit alors un message de signalisation qui suit le chemin de la connexion et qui vérifie que le nouveau débit demandé peut effectivement être alloué. S'il y a assez de bande passante dans le réseau et si donc la requête est satisfaite, l'application a le droit d'utiliser le débit qu'elle a demandé.

Cependant, puisque RCBR est un service qui partage des ressources (la bande passante) entre plusieurs utilisateurs, il est toujours possible qu'une renégociation se bloque. Que se passe-t-il dans ce cas? Premièrement, il est important de remarquer que si une renégociation est bloquée, *le débit avant la renégociation reste garanti, sans délai supplémentaire et sans perte de paquet.* RCBR offre donc un modèle de "partage contrôlé". Deuxièmement, beaucoup de travaux récents sur les applications adaptatives montrent qu'il est possible de concevoir des applications qui peuvent maintenir un niveau de qualité perceptuelle acceptable quand la bande passante disponible est insuffisante. Donc, après un blocage de renégociation, la source peut s'adapter au débit réduit jusqu'à la prochaine renégociation. Troisièmement, le contrôle d'admission (discuté au prochain chapitre) permet de maintenir la probabilité de blocage suffisamment basse.

Nous étudions ensuite comment une application qui utilise le service RCBR peut décider quand renégocier et quel nouveau débit demander. Nous distinguons deux cas. Dans le cas "off-line", où l'évolution du débit en fonction du temps est connue à l'avance (par exemple pour une application de type vidéo à la demande où la video est stockée et peut être analysée *a priori*), il est possible de calculer la séquence des renégociation de façon optimale. Pour ceci, il faut définir une fonction de coût. Nous présentons un algorithme qui calcule cette séquence pour la fonction de coût suivante:

$$\phi \cdot \sum_{i=1}^{N-1} (1 - I_{\{s_{i-1}, s_i\}}) + \gamma \cdot \sum_{i=0}^{N-1} s_i, \tag{1.3}$$

où s_i est la quantité de donées générées dans l'intervalle i, et où ϕ est un coût constant *par renégociation* et γ un coût par unité de bande passante et unité de temps.

Dans le cas "on-line", où l'évolution du débit n'est pas connue à l'avance (par exemple pour une vidéoconférence), il faut utiliser une prédiction de débit nécessaire pour générer la séquence de renégociation. Nous avons étudié un algorithme qui se base sur un modèle AR(1) très simple pour montrer qu'une performance raisonnable peut être atteinte pour ce cas (voir Figure 4.3).

Finalement, nous avons fait l'expérience suivante afin de vérifier que le gain de multiplexage statistique de RCBR est proche de celui d'un scénario de référence idéalisé (voir Figure 4.4). On considère n flux qui sont de versions arbitrairement décalées d'une même trace de vidéo et qui sont multiplexés sur un lien de capacité c. Dans le scénario RCBR, un buffer de taille B est utilisé afin d'absorber les fluctuations sur des échelles temporelles courtes. Le débit après ce buffer est un débit constant par intervalle, qui pourrait donc être transporté par RCBR. Dans le scénario de référence, tous les flux partagent un buffer de taille nB. La taille de ce buffer est donc la même que la somme des tailles des buffers individuels dans le cas RCBR.

La Figure 4.7 montre l'utilisation du lien en fonction du nombre de flux n pour une probabilité de dépassement du lien fixée à 10^{-5}. On note que l'utilisation, et donc le gain de multiplexage statistique, dans le cas RCBR est inférieur à, mais remarquablement proche de l'utilisation du scénario référence. Intuitivement, ceci est du au fait que la majorité du gain de multiplexage statistique vient du partage de la bande passante du lien et non du partage du buffer, car les fluctuations à longues échelles temporelles "ne voient pas" le buffer.

Nous pensons donc que RCBR est un modèle de service *simple*, parce qu'il nécessite un réseau CBR sans buffer ni disciplines d'ordonnacement compliquées, mais seulement avec un mécanisme de renégociation simple et *efficace*, car il permet d'extraire presque autant de gain de multiplexage statistique qu'un modèle de service plus complexe qui partage les buffers entre flux.

1.4 Chapitre 5: Le Contrôle d'Admission Basé sur les Mesures

Le but du contrôle d'admission est de limiter le nombre de flux admis dans un réseau afin de satisfaire les demandes de qualité de service (QoS) des utilisateurs ou applications, tout en partageant les ressources du réseau (bande passante, buffer) de façon efficace.

On peut distinguer deux classes générales de garanties de qualité de service. Les garanties de type déterministe se basent sur une description du "pire" comportement possible du trafic; les descripteurs de trafic associés sont simples et la conformité d'un trafic observé avec sa description est facile à vérifier; par contre, il n'est pas possible d'obtenir de gain de multiplexage statistique. Le descripteur de trafic le plus couramment utilisé est le "leaky bucket".

Les garanties de type statistique se basent sur une description stochastique du trafic, donc une description du comportement "moyen" du trafic. Il est plus difficile dans ce cas que dans le cas déterministe de vérifier et de faire respecter la conformité du trafic observé avec sa description. Mais il est possible d'obtenir un gain de multiplexage statistique si l'utilisateur se contente d'une garantie statistique. Tous les modèles de trafic se basant sur une bande passante effective sont de ce type.

Le contrôle d'admission basé sur les mesures (MBAC = Measurement Based Admission Control) est intrinsèquement une procédure statistique, et ne peut donc s'appliquer uniquement dans le cadre de guaranties statistiques et de modèles de trafic stochastiques. Dans l'approche traditionnelle du contrôle d'admission, l'utilisateur fournit le descripteur de son trafic, et c'est donc lui qui est responsable de la caractérisation de son trafic. Ceci est difficile en général, surtout pour le trafic issu d'applications interactives, car ces caractéristiques peuvent être impossible à prédire. Par exemple, dans une vidéoconférence, le comportement du trafic dépend des mouvements des interlocuteurs, ce qui rend un tel comportement difficile à spécifier à priori.

Le MBAC, contrairement à l'approche traditionnelle, estime lui-même les paramètres du modèle de trafic. Ceci a l'avantage de simplifier le descripteur fourni par l'utilisateur considérablement, et par exemple de se limiter au seul débit crête. Une fois qu'un flux est admis, le réseau estime lui-même les caractéristiques de ce flux. Il n'est donc plus nécessaire de vérifier la conformité d'une spécification statistique du trafic; notons également que des spécifications trop conservatrices ne réduisent plus l'utilisation du système, car le réseau met à jour (grâce aux mesures sur le trafic observé) la spécification une fois qu'un flux est admis.

Le but de ce travail est de de rendre un MBAC robuste, c'est à dire capable de donner les mêmes garanties de QoS qu'un contrôle qui se base sur la connaissance exacte des paramètres. Ceci revient à comprendre l'impact de l'incertitude implicite dans chaque procédure d'estimation sur la performance d'un MBAC. Nous montrons en particulier qu'un contrôle d'admission qui fait une hypothèse de certitude équivalente, et donc suppose que les paramètres mesurés sont les vrais paramètres, mène à une dégradation de la qualité de service. Nous cherchons donc à compenser cette dégradation.

Notre but est de trouver des directives pour le développement de MBAC robustes, et non de développer un algorithme MBAC complet. En particulier, nous voulons comprendre l'impact sur la qualité de service i) de l'incertitude due à l'estimation des paramètres, ii) des caractéristiques du trafic (en particulier des corrélations temporelles), et iii) de la dynamique du système causée par les arrivées et les départs de flux. Le modèle que nous proposons se veut donc aussi simple que possible pour étudier les facteurs mentionnés ci-dessus.

Voici brièvement les hypothèses que nous faisons dans ce modèle:

1. La bande passante utilisée par chaque flux est modélisée par un processus fluide; les bandes passantes des flux sont i.i.d.

2. Les durées des flux, une fois admises dans le réseau, sont i.i.d., suivant une distribution exponentielle avec moyenne T_h.

3. La fonction d'autocorrélation de la bande passante de chaque flux est exponentielle décroissante avec constante de temps T_c, c'est à dire de la forme $\exp(-t/T_c)$.

4. Pour éviter de faire une hypothèse quelquonque sur le processus d'arrivée de flux, et donc de lier la robustesse à un processus d'arrivée donné, nous faisons l'hypothèse du taux d'arrivée infini de flux; en d'autres termes, il existe à tout moment des flux désirant être admis dans le système.

5. Le serveur a une capacité $c = n\mu$, où μ est le débit moyen d'un flux, et n est la capacité normalisée; le serveur n'a pas de file d'attente.

6. Estimation: Le système connaît le débit instantané de chaque flux, et construit les estimateurs habituels de moyenne et de variance.

7. Qualité de service: la qualité de service est spécifiée en terme de la probabilité souhaitée de dépassement de la capacité p_q.

Nous étudions en fait une séquence de modèles de complexité croissante. Nous nous limitons ici à donner un bref aperçu de la théorie que nous avons développé. L'idée de base est de linéariser la relation entre les paramètres de trafic estimés (moyenne et variance) et l'estimation du nombre de flux admissibles. Ceci nous permet d'obtenir le processus stochastique M_t qui donne l'estimateur du nombre de flux admissibles au temps t (cf. Fig. 5.3). Ensuite, grâce à l'hypothèse du taux d'arrivée infini, nous avons $N_t \geq M_t$, où N_t est le nombre de flux dans le système au temps t. Finalement, nous pouvons obtenir la vraie probabilité de dépassement p_f à partir du processus N_t. Le but du contrôle d'admission robuste est évidemment d'obtenir $p_f = p_q$.

On remarque que lorsque $N_t > M_t$, c'est à dire lorsque le nombre de flux est supérieur au nombre estimé de flux admissibles, le système rejette les nouveaux flux arrivant. A partir de ce moment, c'est à dire pour $t \geq s^*$, N_t diminue car les flux déjà acceptés se terminent les uns après les autres (nous supposons en effet une durée de vie finie des flux).

Appelons m^* le nombre idéal de flux, c'est à dire le nombre que l'on admettrait pour faire en sorte que la qualité de service obtenue soit égale à celle voulue (c'est à dire que $p_f = p_q$). Si on suppose donc que le système a admis trop de flux jusqu'à l'instant $t = s^*$, mais n'admet plus de flux à partir de s^*, les départs de flux après s^* permettent de "réparer" cette erreur et de ramener le nombre de flux à m^*. Ce phénomène de "réparation" définit le temps moyen de relaxation du système. Nous appelons ce temps *l'échelle temporelle critique*; il définit le temps moyen dont le système a besoin pour "oublier" des erreurs d'admission. Cette échelle temporelle critique s'avère être l'échelle naturelle pour étudier la dynamique de ce système.

Considérons d'abord le cas de l'estimation sans mémoire, c'est à dire quand le système n'a pas à disposition de l'information sur le comportement passé des débits des flux. Nous montrons que la probablilité de dépassement p_f est alors

$$p_f = \frac{\widetilde{T_h}}{2T_c} \cdot \frac{\sigma}{\mu} \alpha_q Q\left(\frac{\alpha_q}{\sqrt{2}}\right) \tag{1.4}$$

où

- T_h: durée moyenne d'un flux.
- $\widetilde{T_h} = T_h/\sqrt{n}$: échelle temporelle critique.
- T_c: paramètre de la fonction de corrélation.
- μ, σ^2: moyenne et variance du débit d'un flux.
- $\alpha_q = Q^{-1}(p_q)$, où p_q est la probabilité de dépassement souhaitée.

Il est utile de noter qu'avec une connaissance exacte des paramètres μ est σ^2, la probabilité de dépassement p_f serait

$$p_f = p_q = Q(\alpha_q). \tag{1.5}$$

On constate donc que p_f est supérieure à p_q, et ce indépendamment de la taille du système (n). Cette différence est dûe à l'erreur d'estimation. On observe qu'elle se manifeste de deux façons. Premièrement, l'argument de la fonction $Q(.)$ est réduit d'un facteur de $1/\sqrt{2}$. Cette réduction est due à la non-linéarité de la probabilité de dépassement en fonction de l'estimateur de la moyenne. Son impact peut être considérable: si par exemple $p_q = 10^{-5}$, alors $Q(\alpha_q/\sqrt{2}) \approx 10^{-3}$, donc une détérioration d'environ deux ordres de grandeur dans la probabilité de dépassement.

Deuxièmement, il y a un préfacteur supplémentaire $\frac{\widetilde{T_h}}{2T_c} \cdot \frac{\sigma}{\mu}\alpha_q$ qui est dû à la dynamique du système, et plus particulièrement le rapport entre la fluctuation de l'estimateur donné par T_c et l'échelle temporelle critique $\widetilde{T_h}$. Ce préfacteur peut être expliqué comme suit: quand $T_c \approx \widetilde{T_h}$, alors en moyenne le système fluctue entre périodes de surestimation et sousestimation du nombre de flux admissibles environ une fois par échelle temporelle critique, que l'on peut comprendre comme le temps de relaxation du système. Le système arrive donc la plupart du temps à suivre cette fluctuation. Quand $T_c << \widetilde{T_h}$, alors les changements entre périodes de surestimation et de sousestimation sont beaucoup plus fréquents, et le système n'arrive plus à suivre l'évolution de l'estimateur du nombre de flux admissibles. Le danger de dépassement est donc plus grand.

Regardons maintenant le cas avec mémoire d'estimation, c'est à dire, l'évolution de la bande passante des flux dans une fenêtre de longueur T_m est prise en compte pour l'estimation (cf. (1.6)). Nous observons qu'une longueur de fenêtre d'estimation T_m se manifeste de deux façons. Premièrement, l'exposant est réduit, ce qui correspond à l'amélioration des estimateurs, c'est à dire la réduction de leur variance. Deuxièmement, le préfacteur est réduit, ce qui correspond au "ralentissement" des fluctuations de l'estimateur du nombre de flux admissibles grâce à la mémoire.

$$p_f(p_q, T_h, T_c, T_m) = \frac{\widetilde{T_h}}{\sqrt{(T_c + T_m)(2T_c + T_m)}} \cdot \frac{\sigma}{\sqrt{2\pi}\mu} \left(\sqrt{2\pi}\alpha_q p_q\right)^{\frac{T_c+T_m}{2T_c+T_m}} \tag{1.6}$$

Nous avons vérifié ces formules par des simulations. Nous trouvons que

- La probabilité de dépassement p_f donnée par la théorie correspond relativement bien à la probabilité expérimentale; en général, la théorie surestime légèrement la valeur expérimentale.
- Un contrôle robuste, c'est à dire qui respecte la qualité de service souhaitée, peut-être réalisé en calculant un paramètre de qualité de service p_{ce} plus conservateur que p_q, en inversant (1.6) avec $p_f(p_{ce}, T_h, T_c, T_m) := p_q$.

Jusqu'ici, nous avons supposé que le paramètre de la fonction de corrélation T_c ainsi que la durée moyenne d'un flux T_h étaient connus. Or, en pratique, alors qu'il est relativement facile d'avoir une bonne estimation de T_h, il n'en est pas de même pour T_c. De plus, p_f est plus sensible à T_c qu'à T_h. Nous discutons donc comment obtenir un contrôle d'admission qui est robuste *aussi* envers le paramètre T_c. Le paramètre qui nous permettra d'obtenir cette robustesse est la longueur de la fenêtre d'estimation T_m.

Plus précisement, nous suggérons que si la taille de mémoire T_m est choisie du même ordre de grandeur que l'échelle temporelle critique $\widetilde{T_h}$, alors nous obtenons un contrôle qui est robuste par rapport à la valeur de T_c. Nous devons distinguer deux cas: si $T_c < T_m$, alors le comportement dynamique de T_m est dominé par T_m, et la valeur exacte de T_c n'a pas d'impact sur p_f. Par conséquant, nous appelons ce régime le "régime de masquage".

si $T_c > \underset{\sim}{T_m}$, alors la fluctuation de la superposition de flux est plus "lente" que l'échelle critique $\widetilde{T_h}$. L'effet de réparation (les départs de flux) évite donc les dépassements et rend p_f très petit.

Cette étude nous permet d'apprécier l'impact de l'incertitude due à l'estimation de paramètres et de la dynamique du système. Cette dernière est régie par la corrélation dans le trafic, la durée des flux, et la longueur de la fenêtre d'estimation. Nos formules explicites permettent d'obtenir un contrôle d'admission robuste. Nous avons reconnu l'importance fondamentale de l'échelle temporelle critique, ce qui motive le choix de la longueur de la fenêtre de mémoire de l'ordre de $\widetilde{T_h}$. Un point remarquable est le fait que cette longueur dépend donc de la capacité n du système; en d'autres termes, on utilisera une fenêtre plus courte dans un réseau d'interconnexion à haute capacité que dans un réseau local.

Ce travail ouvre plusieurs perspectives de recherche, par exemple:

- Mesures de la superposition: il est souhaitable en pratique de ne pas devoir estimer de paramètres *par flux*. Nous voulons donc étudier la performance du système, moyennant une approche similaire, quand seul le débit de la superposition des flux est disponible.

- Non-stationnarités: nous avons, jusqu'à présent, étudié le système dans le régime stationnaire. Il est évident que dans ce régime, agrandir la longueur de la fenêtre d'estimation est toujours avantageux. Pour montrer que la réactivité du système souffre quand cette fenêtre devient trop longue, il nous faut introduire des non-stationnarités dans le comportement des flux et étudier la performance du MBAC dans ce cas là.

- Dépendance à longue portée: il est intéressant et important d'étudier la performance du système quand la fonction d'autocorrélation n'est pas exponentielle comme supposée ci-dessus; en particulier, dans le cas d'une autocorrélation non-sommable, ce qui correspond à une dépendance à longue portée (LRD).

1.5 Chapitre 6: Sur les Modèles de Service pour la Transmission Multipoint dans un Environnement Hétérogène

Le modèle de service de "best-effort" de l'Internet actuel facilite énormément l'interconnexion de réseaux de différents types, et permet une administration décentralisée. Cette flexibilité vient au prix d'un modèle de service qui n'est pas le plus adéquat pour toutes les applications. Il y a deux remèdes possibles pour ceci. Soit les applications essaient de s'adapter au mieux au canal offert par le réseau, par le biais de mécanismes comme l'adaptation du "playout point", le contrôle de débit, ou le contrôle d'erreur, soit le réseau est modifié de façon à offrir d'autres classes de service avec des caractéristiques plus prévisibles que "best effort". Il est clair que cette complexité supplémentaire du réseau entraîne une réduction de complexité du coté application. Par exemple, les commutateurs du réseau téléphonique sont des systèmes extrêmement complexes, alors qu'il n'y a pas besoin de contrôle d'erreur, de délai ou de débit du coté application.

Nous suggérons dans le chapitre 6 que ce compromis entre complexité dans le réseau et dans l'application est modifié dans le cas de la communication multipoint et multicouches. Ceci est du au fait que même dans l'architecture actuel de l'Internet, les protocoles de routage multipoint ont besoin d'état dans les routeurs. L'Internet a aussi besoin de protocoles de

signalisation comme DVMRP pour manipuler cet état. Vu que ces protocoles sont donc déjà en place, le coût incrémental pour déployer un modèle de service plus sophistiqué est sensiblement plus petit que dans le cas unicast.

De l'autre coté, plusieurs mécanismes d'adaptation dans les applications deviennent beaucoup plus complexes s'il doivent fonctionner dans en environnement multipoint et multicouches. Nous discutons ces complications pour plusieurs mécanismes.

L'estimation de canal est nécessaire quand une application n'a pas de connaissance explicite (après une réservation préliminaire ou par une notification par le réseau) des propriétés du canal. L'estimation de canal est plus difficile pour le cas multipoint, car le canal est maintenant un arbre partagé parmi plusieurs destinations. Aussi, l'utilisation de multiples couches signifie que l'interdépendance entre ces couches doit être estimée.

Le contrôle d'erreur désigne plusieurs types de mécanismes pour limiter l'impact d'erreurs de transmission comme par exemple la perte de paquets. Les mécanismes de reprise sur erreur essaient de réparer des lacunes par retransmission ou par un codage aproprié. Les deux approches sont fortement compliquées dans le cas multipoint/multicouches. La retransmission dans un arbre multipoint peut entraîner un grand coût si l'arbre est grand. La "forward error correction" (FEC) pose le problème de trouver la bonne quantité de redondance pour plusieurs couches à la fois, ce qui revient à résoudre un problème d'optimisation multidimensionnel. La résistance aux erreurs est une propriété du système qui tente de minimiser l'impacte des erreurs, par exemple par un mapping judicieux des informations de l'application sur des paquets. Ce problème est plus difficile dans le cas multicouche car l'interdépendance implicite des couches a tendance à rendre le système plus sensible aux erreurs.

La spécification de trafic nécessite que l'application puisse estimer son trafic à l'avance. Ce problème est plus difficile dans le cas multicouche car il y a plus de paramètres à estimer.

En résumé, les tâches de l'application pour fonctionner avec un modèle de "best effort" sont fortement compliquées dans un environnement multipoint/multicouche. Un autre aspect préoccupant de l'approche multicouche est le phénomène de *perte de ressources externe*. De façon générale, nous appelons *perte de ressources* l'allocation d'une ressource à une tâche inutile. Un exemple est une perte d'un paquet qui a parcouru une partie du chemin de la source à la destination. La perte de ressources externe est une perte de ressource qui ne peut pas être identifiée en tant que telle par le réseau, parce qu'il manque l'information nécessaire. Un exemple est la transmission d'un paquet sur une couche d'un niveau supérieur quand le paquet correspondant du niveau de base est perdu. Dans le chapitre 6, nous discutons un modèle très simple qui illustre que même pour des probabilités de perte de paquet assez faibles, la perte de ressources externe peut devenir considérable dans la communication multicouche si le réseau n'a pas d'information sur l'interdépendance des flux.

L'Internet actuel ne fournit aucune indication explicite de congestion aux applications. Les applications doivent donc elles-mêmes détecter la congestion, ce qui revient au problème d'estimation de canal mentionné plus haut. Cette détection implicite de congestion limite les régimes possibles dans lesquels le réseau peut fonctionner. En effet, un régime où la probabilité de dépassement désirée est très basse n'est pas réalisable, car il est trop difficile d'estimer si la probabilité de dépassement instantanée est au-dessus ou au-dessous du seuil désiré, et donc de concevoir un contrôle stable autour de ce point d'opération. Une autre approche consiste à essayer de cacher ce régime de l'application moyennant des techniques de "forward error correction" (FEC), de façon à ce que la probabilité de perte vue

par l'application soit beaucoup plus basse que celle du canal. Cependant, nous montrons dans l'appendix du chapitre 6 que cette approche est problématique si son utilisation est généralisée à tout le réseau.

En d'autres mots, pour limiter la perte de ressources externe, il faudrait opérer dans un régime qui n'est pas réalisable avec le contrôle de congestion actuel de l'Internet de type TCP. Basé sur cette observation, nous proposons un modèle de service appelé GRCBR (Generalized RCBR) qui est une extension de RCBR au cas multipoint/multicouche. Le but de ce modèle est d'illustrer que la perte externe dans un cadre multipoint/multicouche peut être évitée moyennant un mécanisme très simple à mettre en œuvre et une quantité d'état supplémentaire très limitée. GRCBR utilise deux types de renégociation. Le premier type ("closed renegotiation") ressemble à celui de la version unicast de RCBR. Il est accepté si tous les liens dans l'arbre peuvent accepter le nouveau débit. Le deuxième type ("open renegotiation") sert à permettre à des couches de basse priorité de fonctionner dans un régime de haute probabilité de blocage. Il est toujours accepté; les liens qui ne peuvent pas accommoder le nouveau débit suppriment temporairement le flux en question jusqu'à la prochaine requête de renégociation. La combinaison de ces deux types de renégociation permet d'utiliser un régime à basse probabilité de blocage pour une couche de base (haute priorité) et de traiter les couches hautes (faible priorité) plus comme un "best effort renégocié". Nous introduisons aussi la renégociation atomique de plusieurs flux, ce qui permet de respecter l'interdépendance entre les flux et donc d'éviter la perte de ressources externe.

1.6 Chapitre 7: Contrôle de l'Implosion de Contre-Réaction en Multipoint

Les protocoles de contrôle de bout-en-bout pour les communications multipoint, comme par exemple le contrôle de congestion et le transport fiable, reposent sur des information de contre-réaction envoyées par les destinations vers la source pour signaler leurs observations, comme la congestion ou la perte d'un paquet. Cependant, les observations aux différentes destinations ont tendance à être fortement corrélées car les destinations partagent des ressources à l'intérieur de l'arbre multipoint. Ceci peut entraîner, pour une seule perte de paquet sur un lien dans l'arbre, une implosion d'information de contre-réaction émise par les destinations. Pour éviter ce problème, le nombre de destinations qui renvoyent de l'information de contre-réaction en cas d'un événement comme une perte de paquet doit rester limité.

Il existe deux approches pour résoudre ce problème: l'approche *structurelle* et l'approche *temporelle*. Dans l'approche structurelle, les noeuds intérieurs du réseau traîtent l'information de contre-réaction. Par exemple, les nœuds intermédiaires peuvent comprimer plusieurs messages en provenance des destinations dans un seul message envoyé vers la source. Dans l'approche temporelle, l'implosion est évitée en retardant l'envoi de l'information aux destinations. L'avantage de cette deuxième approche est qu'elle ne nécessite pas de support réseau spécifique pour éviter l'implosion, mais le désavantage potentiel d'un temps de résponse élevé.

Nous présentons un algorithme distribué pour calculer un délai fixe pour chaque destination dans un arbre multipoint, qui est une fonction de la topologie de l'arbre est des délais source-destination. Cet algorithme est entièrement distribué car chaque nœud de l'arbre a seulement besoin d'informations locales et de communication entre nœuds voisins. De plus, l'ensemble des délais calculés par cet algorithme est optimal dans le sens qu'il minimise le

temps de réponse maximal.

Chapter 2

Introduction and Overview

Traditionally, two classes of fundamentally different networking infrastructures have coexisted, namely telephone networks and data networks, each one built for a specific type of applications. A telephone voice call requires stringent quality of service (QoS) guarantees such as minimal delay, delay jitter, and loss, in order to ensure a satisfactory perceptual voice quality for the user. In order to make such guarantees, the telephone network reserves resources (bandwidth) inside the network on behalf of a phone call during the entire duration of the call (cf. Fig. 2.1). The traffic emitted by a telephone is very regular: it is a constant bit rate (CBR) stream with a bandwidth of 64kbps.

Data applications such as file transfers or web accesses, on the other hand, have considerably less stringent quality of service requirements. Some minimal long-term bandwidth is often sufficient for a satisfactory perceptual quality for the user. Therefore, it is not necessary to reserve resources inside the network for each call. Furthermore, given the diverse set of data networking applications, the concept of a call or *flow* is often not cleary identifiable. The traffic emitted by data applications tends to be very bursty in nature, i.e., the bandwidth fluctuates over time. As a consequence, the resource requirements of the flow are not constant during its lifetime. Data networks therefore need mechanisms to cope with these bandwidth fluctuations, such as buffer management and link scheduling disciplines (cf. Fig. 2.1).

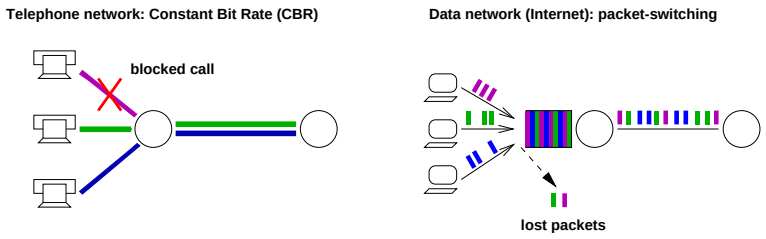

Figure 2.1: Telephone and data networks.

The fundamental differences between the two types of networks are the time-scales of the load flucation and the associated control mechanism to accommodate them (cf. Fig. 2.2). The load of the telephone network fluctuates due to call arrivals (new call setup) and call departures (call termination); the associated control mechanism is *call admission*

control, which takes control actions on a call by call basis. It can turn some calls away during overload. The traffic generated by active flows is very regular, and consequently, only very simple mechanisms (Sonet/SDH) are necessary to carry the traffic. The load of data networks fluctuates due to the bursty traffic emitted by applications; the associated control mechanisms are *buffer management* and *scheduling*, which takes control actions on a packet-by-packet basis. It can drop data packets during overload. There is no clear concept of a call or a flow, and no admission control mechanism. In summary, both types of networks match the time-scales of control to the time-scales of load fluctuation: call by call admission decisions for the telephone network, per packet buffering and scheduling for data networks. This is illustrated in Figure 2.2.

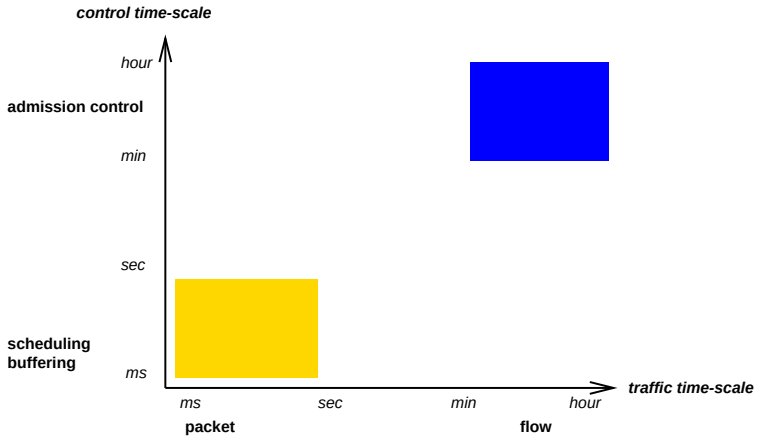

Figure 2.2: The traffic and control time-scales of telephone and data networks.

This situation has changed because for an important class of emerging applications, there is a mismatch between the time-scale of load fluctuation and the time-scale over which a quality of service has to be guaranteed. New applications such as real-time multimedia applications have quality of service requirements similar to telephone voice communication. For example, compressed audio and video applications such as videoconferencing, telemedicine and teleteaching require low delay because of their interactive nature, and a small loss probability because the perceptual impact of loss tends to be considerable. However, these applications emit bursty traffic, similar to data applications. This precludes simply allocating the necessary resources to each flow, as resource utilization would be prohibitively low. For network efficiency, resources such as buffers and bandwidth have to be overbooked; the resulting increase in resource utilization is referred to as *statistical multiplexing gain (SMG)*. This in turn requires a careful design of control mechanisms so that they cover all time-scales. In Chapters 3 to 5, we discuss resource control over three natural time-scales: the packet, the burst, and the flow holding time-scales, respectively.

The **packet time-scale** is the shortest time-scale we study. Our interest stems from the mounting experimental evidence that packet arrival processes exhibit ubiquitous properties of self-similarity and long range dependence (LRD). A random process exhibits long-range dependence if it has a non-summable autocorrelation function. Intuitively, this means that the process exhibits fluctuations over a wide range of time-scales.

In spite of much statistical evidence, the existence of LRD has often been met with resistance of at least puzzlement. This was caused in large part by the absence of physical

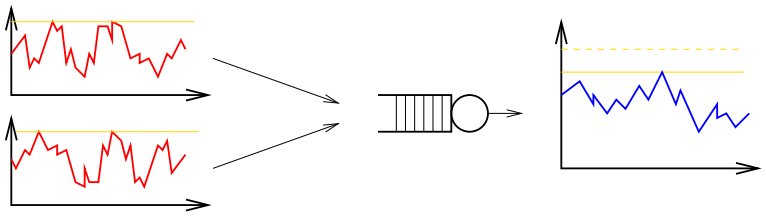

Figure 2.3: An illustration of statistical multiplexing gain.

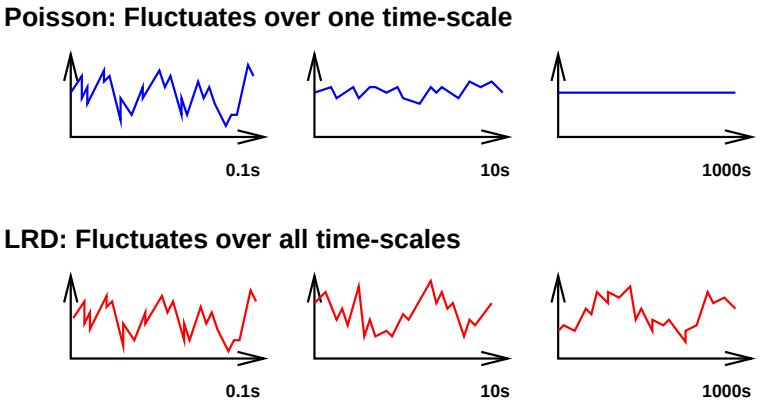

Figure 2.4: Long-range dependent processes exhibit fluctuations over a wide-range of time-scales, as opposed to short-range dependent processes such as the Poisson process.

explanations for the phenomenon. Intuitively, the surprising aspect of a physical system generating observations with long-range dependence is the transmission of state information about the process over very long time-scales. Especially in natural processes (e.g., the levels of the Nile river), it is usually hard to imagine where the "memory" maintaining this information about the past behavior of the process is located.

The situation in the networking community parallels the situation in other fields, such as hydrology and econometry. The excitement about the finding of LRD in many empirical studies of network traffic has been further stimulated by various queueing-theoretic studies of the overflow probability of infinite-buffer queues fed by a long-range dependent process. It was found that the asymptotic performance of such queueing systems is fundamentally different from those without long-range dependence. Furthermore, no (finite-state) Markovian traffic model can give rise to a long-range dependent process, as the autocorrelation function of Markovian models is always asymptotically exponential. In the teletraffic community, markovian models were used almost exclusively because of their mathematical tractability.

This situation has led many researchers to suggest that Markovian models are not adequate for teletraffic modeling. On the other hand, until the discovery of LRD, such models seemed to be accurate, and no systematic discrepancy between measured and predicted performance could be observed. Therefore, there is still considerable debate about how to model such processes and about their impact on network and application performance. *In Chapter 3, we argue that much recent modeling work has failed to consider the impact of*

an important parameter, namely the finite range of time scales of interest in performance evaluation and prediction problems.

In our contribution, we introduce and evaluate a model in which these parameters can be easily controlled. Specifically, our model is a modulated fluid traffic model in which the correlation function of the fluid rate is asymptotically second-order self-similar with given Hurst parameter, then drops to zero at a cutoff time lag. Intuitively, the cutoff lag corresponds to the longest time-scale of fluctuation in the traffic process. We develop a very efficient numerical procedure to evaluate the performance of the single server queue fed with the above fluid input process. We use this procedure to examine the fluid loss rate for a wide range of marginal distributions, Hurst parameters, cutoff lags, and buffer sizes.

Our main result is that the amount of correlation that needs to be taken into account for performance evaluation depends not only on the correlation structure of the source traffic, but also on time scales specific to the system under study. For example, the time scale associated with a queueing system is a function of the maximum buffer size. Thus for finite buffer queues, we find that the impact on loss of the correlation in the arrival process becomes nil beyond a time scale we refer to as the correlation horizon. also, we find that loss depends in a crucial way on the marginal distribution of the fluid rate process. Furthermore, our results suggest that reducing loss by buffering is hard.

Given that network traffic fluctuations have components that are too long-term to be smoothed out by buffering, per-flow buffering is not sufficient to reduce fluctuation and to achieve good resource utilization. As mentioned above, *resource overbooking* (sharing) is necessary to increase utilization. We then face the question of what resources should be shared. In best-effort network such as the Internet, both buffers and link bandwidth are shared among flows. As we have to respect quality of service constraints of flows, a controlled way of sharing is needed. Most proposals for transport services for such flows, such as the Internet integrated-service model or the ATM VBR class, make the assumption that sharing both buffers and bandwidth is necessary. These services are typically quite complex and need elaborate scheduling disciplines, per-flow queueing, and non-trivial admissability checks. The alternative we propose considerably simplifies network complexity by *only sharing bandwidth*, but not buffers.

On the **burst time-scale**, *we advocate renegotiation as an efficient mechanism to accommodate fluctuations over time-scales beyond the correlation horizon. A new network service model called RCBR (Renegotiated Constant Bit Rate) combines network simplicity with desirable quality of service guarantees, while achieving much of the potential statistical multiplexing gain of bursty traffic.*

Sources using RCBR service are presented with an abstraction of a fixed-size buffer which is drained at a constant bit rate. They may dynamically renegotiate the drain rate to match their workload. Because all traffic entering the network is CBR, RCBR requires minimal buffering and scheduling support in switches. As an example, we show that the service is suitable for both stored and online video sources.

An RCBR source must decide when to renegotiate its service rate, and what the new service rate should be. We present an algorithm to compute the optimal renegotiation schedule for stored (off-line) traffic, and a heuristic to approximate the optimal schedule for online traffic.

Simulation experiments show that RCBR is able to extract almost all of the statistical multiplexing gain available by exploiting slow time-scale variations in traffic. Moreover, simple admission control schemes are sufficient to keep the renegotiation failure probability

below a small threshold, while still offering high link utilization. *Thus, we believe that RCBR is a simple, practical, and effective service for carrying multiple time-scale traffic.*

On the **flow time-scale**, we discuss measurement-based admission control as a means of relieving the application of traffic specification. With "traditional" admission control, admission decisions are based on a-priori traffic descriptors such as effective bandwidths or leaky-bucket parameters. The problem is that it is often hard for the user or for the application to come up with good descriptors; this is especially true for on-line communication. For example, the traffic descriptor of a videoconference might depend on the movements of participants during the conference, which is hard to predict in advance. This can lead to poor resource utilization or poor quality, depending on if the specification was overly conservative or overly optimistic. Measurement-based Admission Control (MBAC) is an attractive mechanism to concurrently offer Quality of Service (QoS) to users, without requiring a-priori statistical traffic specification and on-line policing of this specification. MBAC achieves this by *learning* the traffic characteristics of existing flows in the system. Admission decisions are then based on a-priori traffic descriptors only for the new flow. The advantage is obvious: a poorly chosen a-priori descriptor is replaced by a better measurement-based traffic characterization once the flow is in the system.

Despite its obvious advantages, several aspects of such a system need to be clearly understood in order to devise robust MBAC schemes that are able to make quality of service guarantees. The impact of measurement uncertainty and the interplay of burst and flow time-scale dynamics are captured in a single analytical model. *We use this model to provide useful guidelines for robust admission control in the presence of unknown traffic statistics.*

We show that a *certainty equivalence* assumption, i.e., assuming that the measured parameters are the real ones, can grossly compromise the target performance of the system. We quantify the improvement in performance as a function of the memory size of the estimator and a more conservative choice of the certainty-equivalent parameters. Our results yield new insights into the performance of MBAC schemes, and represent quantitative guidelines for the design of robust schemes. A key result shows that robustness can be achieved by matching the traffic measurement time-scale to the critical time-scale of the system, which depends on the average flow holding time and the system size. We validate our results through simulations.

So far, we have focused on efficient network control for fluctuations on three different time-scales. We have focused on the temporal aspects of the resource control problem. We next turn to *spatial* aspects of the resource control problem, which arise in the case of multicast delivery. **Multicast transmission** allows the efficient delivery of data to several receivers. It is efficient because contrary to unicast, data is duplicated inside the delivery tree instead of at the source, thereby avoiding that the same packet traverses some links more than once. Multicast is already widely used in the Internet MBone for the broadcasting of technical conferences. Multicast applications such as audio, video and text distribution have become popular, with much further increase expected in the future, for example with distributed gaming and web pushing.

We address two issues in multicast transmission. *In Chapter 6, we extend the RCBR service model to multicast flows to illustrate the tradeoff between network and application complexity.* We first examine service models and their impact on application design for large-scale multicast applications in heterogeneous environments. Using simple models, we illustrate the tradeoff between network and application complexity. In particular, we compare the complexity of applications in a best effort network and in a network offering controlled sharing through renegotiation. We then examine the issue of making the network

aware of the dependencies between flows that make up the layers of a single session.

These applications typically rely on layered transmission of prioritized information in order to accommodate the heterogeneity of large multicast groups. As multicast distribution requires some per-flow state anyway, we argue that the tradeoff between service model and application complexity should be reconsidered. All the more so because the complexity of end to end adaptation mechanisms such as adaptive playout adjustment, forward error correction, and rate control, is exacerbated when using a multilayer/multicast approach. The insights we obtain motivate the design of a renegotiated constant-rate service. This generalized version of RCBR shares with RCBR the simplicity of the data path. We show that minor additions to the signalling system are sufficient to take flow dependence into account.

In Chapter 7, we address the problem of feedback implosion in multicast. Multicast control protocols for reliability or congestion control suffer from the problem of feedback implosion. To avoid this problem, the number of receivers sending feedback must be small. However, observations of different receivers are strongly correlated, since receivers share common resources in the multicast tree. For example, Figure 2.5 illustrates how multiple receivers in a tree can observe a single packet loss somewhere in the delivery tree.

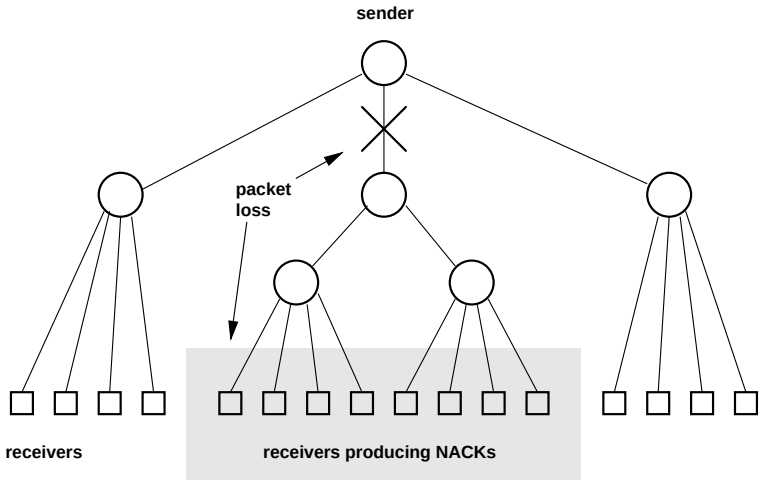

Figure 2.5: An example of a feedback implosion due to a single packet loss.

One approach to feedback implosion avoidance relies on delaying feedback at the receivers. We present DTRM (Deterministic Timeouts for Reliable Multicast), a distributed algorithm to compute optimal deterministic timeouts for each receiver in a multicast tree as a function of the tree topology and the sender-to-receiver round-trip delays. *DTRM has several desirable properties. First, feedback implosion is provably avoided for a single loss anywhere in the tree, provided delay jitter is bounded. Second, the computation of the timeouts can be entirely distributed; receivers and intermediate nodes only rely on local topology information and only need to communicate with neighboring nodes. Third, the timeouts computed by DTRM are optimal with respect to the maximum response time.*

Chapter 3

Control Time-Scales and Network Traffic Modelling

Notation

X_t	fluid rate at time t
$\{\lambda_1, \ldots, \lambda_M\}$	set of possible fluid rates
τ_n	modulating point process
T_n	length of interval n: $T_n = \tau_{n+1} - \tau_n$
$F_T(t)$	ccdf of the interarrival time
$\{\pi_i\}$	marginal fluid rate distribution
$\phi(t)$	autocorrelation function of process X_t
T_c	cutoff lag
θ	shape parameter of Pareto distribution
H	Hurst parameter
Q_n	queue occupancy at $t = \tau_n$
W_n	work arriving within $[\tau_n, \tau_{n+1})$
Q_n^L, Q_n^H	lower and upper bounds of queue occupancy at $t = \tau_n$
W_l	work lost between two arrivals
T_{CH}	correlatin horizon

The work in this chapter has appeared in [44].

3.1 Introduction

Experimental data obtained from the observation of systems is typically considered for modeling purposes as a realization, or sample path, of an underlying stochastic process. In practice, statistical analysis of the data proceeds with the additional hypotheses that the process is stationary and ergodic. Such analysis has shown that many systems of interest in the physical world exhibit a property of correlation over many different time scales, often referred to as long range dependence (LRD) or long memory. Some of the better known examples of such systems are found in hydrology [59]. However, the phenomenon of LRD occurs in many other systems including chemical, astronomical, and biological systems (see [6] for references).

In spite of much statistical evidence, the existence of LRD has often been met with resistance or at least puzzlement. This was caused in large part by the absence of physical

explanations for the observed phenomenon. Hydrologists for example wondered "By what sort of physical mechanism can the influence of, say, the mean temperature of this year at a particular geographic location be transmitted over decades and centuries?" [69]. Two approaches then are possible. One approach is to argue that the LRD observed in the measurement data is a consequence of inadequate hypotheses, in particular the stationarity hypothesis, made about the underlying process that (it is assumed) did generate the data. For example, the superposition of a process with short range dependence (SRD) and an appropriately chosen on/off trend [69] or a hyperbolically decreasing trend [8] is difficult to distinguish from a stationary process with LRD. Another approach is to not worry about a physical explanation and to develop and use models that do exhibit LRD, on the grounds that a model is good not because it explains a phenomenon correctly, but rather because it provides good prediction ability and it is numerically and/or analytically tractable.

Unfortunately, it is not possible to tell with certainty whether or not a realization is stationary. Therefore, the jury is still out on which of the above two approaches is "the right one". Clearly, it is better for a model to match more properties of the data. However, a model is a tool for decision making. Thus, its quality depends on the quality of the decisions it leads to rather than on its closeness to physical reality.

The situation in the area of communication systems in general, and computer networks in particular, is no exception to that described above. Careful statistical analysis of data collected over a wide variety of networks has provided ample evidence that network traffic processes exhibit properties of self-similarity and LRD [80, 73, 29, 70, 89, 7]. However, there is still considerable debate about how to model such processes. Different approaches have been taken that parallel those taken in other areas and described earlier. One approach has been to argue that the observed LRD may be due to non-stationarity in the data caused by the superposition of level shifts [28] or Dirac pulses [41] with short range dependent (SRD) stationary processes. Another approach has been to use stochastic models (such as fractional Brownian motion [82], zero-rate renewal processes [111] and various other point processes [96]) or deterministic models (such as chaotic maps [34]) that exhibit the LRD observed in the experimental data. However, these models are analytically difficult to handle. Furthermore, they do not provide much insight into why they are meaningful on physical grounds. This explains in part that much modeling work still relies on more traditional multi-state Markovian models (e.g., [74, 4]).

However, recent work has shown that the superposition of many on/off sources with heavy-tailed on- and off-periods results in aggregate traffic with LRD [117, 13]. Furthermore, there is widespread evidence that human as well as computer sources of traffic do tend to behave as heavy tailed on/off sources [20, 70, 117]. Thus, LRD in network traffic can be explained simply in terms of the nature of the traffic generated by individual sources.

This intuitively appealing explanation suggests that LRD will remain a salient feature of network traffic even as network characteristics such as bandwidth and topology evolve over time. Thus, the fundamental question for both current and future networks is that of the practical impact of LRD on network and application performance. Not surprisingly, much effort has focused on trying to answer this question. The main result is that the performance of queueing systems with LRD in the input or service processes can be radically different from the performance of usual Markovian systems [82, 33, 87].

However, consider for example the asymptotic behavior of an infinite queue fed with three different arrival processes that all exhibit the LRD property. First, if the arrival process is a fractional Brownian motion, then the queue length distribution is Weibullian. Second, if the arrival process is a single on/off source with heavy-tailed on and off periods, then the

queue length distribution is hyperbolic. Third, if the arrival process is a single on/off source in which the off periods only are heavy-tailed, then the queue length distribution decays exponentially [13, 87]. Thus, processes with the same correlation structure can generate vastly different queueing behavior. Therefore, it is important to consider parameters other than the correlation of the input process for accurate performance prediction.

Two such parameters stand out, namely the marginal distribution of the arrival process, and the finite range of time scales of interest in performance evaluation and prediction problems. The main goal of this chapter is to evaluate the impact of these parameters, as well as the correlation structure of traffic sources, on network and application performance.

To achieve this goal, we develop a model in which all three parameters can be easily controlled. Specifically, our model is a modulated fluid traffic model in which the correlation function of the fluid rate is asymptotically second-order self-similar with given Hurst parameter, then drops to zero at a cutoff time lag. We then consider the behavior of a finite-buffer queue fed with the above fluid input process. We cannot describe this behavior with closed-form analytic expressions. However, we develop a very efficient numerical procedure to evaluate various performance measures. In this chapter, the measure of interest is the fluid loss rate, i.e., the amount of work lost because of buffer overflow to the amount of work arriving at the queue.

Our main results are as follows. First, we find that the amount of correlation that needs to be taken into account for performance evaluation depends not only on the correlation structure of the source traffic, but also on time scales specific to the system under study. For example, the time scale associated to a queueing system is a function of the maximum buffer size. Thus for finite buffer queues, we find that the impact on performance of the correlation in the arrival process becomes nil beyond a time scale we refer to as the correlation horizon.

Second, we find that the loss rate depends in a crucial way on the buffer size, but more importantly on the marginal distribution of the fluid arrival process. An obvious but important consequence is that the marginal distribution must be taken into account for accurate loss prediction. Another consequence is that controlling the loss rate by increasing the buffer size is much less efficient than controlling the loss rate by adjusting the marginal distribution. Statistical multiplexing and source traffic control are two efficient ways to do this, and to achieve high utilization while keeping loss low.

The rest of the chapter is organized as follows. In Section 3.2, we describe the model and the numerical solution procedure. In Section 3.3, we describe the behavior of the loss rate as as function of system parameters. In Section 3.4, we discuss the implications of our results. Section 3.5 concludes the chapter.

3.2 Model Description

In this section, we describe our modulated fluid traffic model and the numerical procedure we developed to evaluate the behavior of a finite buffer queue fed with this input traffic.

Recall that the goal of the model is to examine the impact on the performance measure of interest of parameters such as time scales and the marginal distribution of the traffic process. Thus, we need a traffic model in which these parameters can be controlled easily.

Specifically, the source traffic model is described by a random process $\{X_t\}$ which represents the fluid rate at time t. We assume that X_t takes on a finite set of possible rates

$\{\lambda_1, \ldots, \lambda_M\}$. Furthermore, we assume that the fluid rate process is piecewise constant. Thus, the rate remains constant over intervals the lengths of which are determined by arrivals of a stationary point process $\{\tau_n\}$. We denote $X_t = \lambda(n)$ for $\tau_n \leq t < \tau_{n+1}$. The interarrival times $T_n = \tau_{n+1} - \tau_n$ are i.i.d. with survival function $\Pr\{T_n > t\} = F_T(t)$. Furthermore, the constant fluid rate $\lambda(n)$ is i.i.d. with distribution $\Pr\{\lambda(n) = \lambda_i\} = \pi_i$. For i.i.d. random variables, we drop the subscript if this does not lead to confusion. Note that this model can be specialized into the familiar on/off source model with identically distributed on and off periods.

The rest of this section proceeds in three steps. In the first step, we derive the covariance function of the fluid process $\{X_t\}$ in terms of the interarrival time distribution $F_T(t)$, the rate matrix $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_M)$, and the marginal distribution of the fluid rate $\Pi = (\pi_1, \ldots, \pi_M)$. In the second step, we derive the occupancy distribution at time τ_n of a queue fed with $\{X_t\}$. In the third step, we use this distribution to derive the performance measure of interest here, namely the stationary fluid loss rate.

The autocovariance function of $X(t)$ is defined by

$$\phi(t) = \mathrm{E}\left[(X_0 - \mu)(X_t - \mu)\right] \tag{3.1}$$

where

$$\mu = \Pi\Lambda\mathbf{1}^T \tag{3.2}$$

is the mean rate.

Let A_t denote the event of on arrival occurring within $[0, t]$. Then, noting that conditional on A_t, X_0 and X_t are independent, and conditional on A_t^c, $X_0 = X_t$, it follows

$$\phi(t) = \mathrm{E}\left[(X_0 - \mu)(X_t - \mu)|A_t\right] + \mathrm{E}\left[(X_0 - \mu)(X_t - \mu)|A_t^c\right] = \sigma^2 p(t) \tag{3.3}$$

where

$$\sigma^2 = \mathrm{E}\left[(X_0 - \mu)^2\right] = \Pi\Lambda^2\mathbf{1}^T - (\Pi\Lambda\mathbf{1}^T)^2 \tag{3.4}$$

is the variance of X_t.

Furthermore, it follows from renewal theory that the probability $p(t)$ is equal to the probability that the residual life τ_{res} of the interarrival time T_n exceeds t, which is given by [68, p. 172]

$$\Pr\{\tau_{res} \geq t\} = \int_t^\infty \frac{F_T(x)}{E[T_n]} dx. \tag{3.5}$$

We now consider the special case when the interarrival time distribution $F_T(t)$ is a truncated Pareto distribution defined by

$$F_T(t) = \begin{cases} \left(\frac{t+\theta}{\theta}\right)^{-\alpha} & \text{if } t < T_c \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

where $1 < \alpha < 2$. We refer to this distribution as truncated Pareto because $F_T(t)$ is a Pareto distribution with parameters θ and α for $0 < t < T_c$. We refer to the parameter T_c as the *cutoff lag*.

Since the length T_n of an interval cannot exceed T_c, and since the rates in consecutive intervals are independent, it follows that there is no correlation in the fluid rate process beyond lag T_c.

We now compute the covariance function $\phi(t)$ for the above distribution. We have

$$\Pr\{\tau_{res} \geq t\} = \begin{cases} \frac{(t+\theta)^{-\alpha+1}-(T_c+\theta)^{-\alpha+1}}{\theta^{-\alpha+1}-(T_c+\theta)^{-\alpha+1}} & \text{if } t < T_c \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

Therefore

$$\phi(t) = \Pi\Lambda^2(\mathbf{1} - \Pi)^T \times \begin{cases} \frac{(t+\theta)^{-\alpha+1}-(T_c+\theta)^{-\alpha+1}}{\theta^{-\alpha+1}-(T_c+\theta)^{-\alpha+1}} & \text{if } t < T_c \\ 0 & \text{otherwise} \end{cases}. \tag{3.8}$$

We observe that $\phi(t)$ behaves asymptotically as $t^{-\alpha+1}$ when $T_c = \infty$. This implies that $\{X_t\}$ is asymptotically second-order self-similar with Hurst parameter H such that $-\alpha+1 = -(2-2H)$, i.e., $H = \frac{3-\alpha}{2}$ [73]. When T_c is finite, the correlation drops to zero at lag T_c.

In summary, our source model allows us to specify the marginal fluid distribution Π, the time-scale T_c over which the correlation structure matches that of an asymptotic second-order self-similar model, and the Hurst parameter H.

We next examine the performance of a queue with constant service rate c and a finite buffer B fed with the above source model. This queueing model does not appear to be analytically tractable. However, we have developed a very efficient numerical procedure to determine the queue occupancy at the arrival instants τ_n. Although this queue occupancy is not equal to the queue occupancy at a random point in time, we can derive our chosen performance metric from it, namely the long-term loss rate.

Let Q_n be the continuous random variable describing the queue occupancy at arrival instant τ_n, and let $W_n = T_n \times (\lambda(n) - c)$. The continuous random variable W_n is the difference between arriving and departing work in interarrival interval n (including arriving work that is lost due to buffer overflow). Note that the $\{W_n\}$ are i.i.d. because T_n and $\lambda(n)$ are i.i.d. and jointly independent. The queue occupancy is recursively given by

$$Q_{n+1} = \max(0, \min(B, Q_n + W_n)). \tag{3.9}$$

(W_n, Q_n) is a two-dimensional Markov process because W_n and W_{n-1} are independent. Assuming ergodicity, (W_n, Q_n) converges to (W_∞, Q_∞) as $n \to \infty$. Our goal is to derive upper and lower bounds of the stationary density $f_Q(x)$ of the queue occupancy. For convenience of notation, we assume that $\lambda_1 < \lambda_2 < \ldots < \lambda_L < c < \lambda_{L+1} < \lambda_{L+2} < \ldots < \lambda_M$. We exclude the trivial case when one of the fluid rates is equal to c, in which case the queue occupancy does not change.

The probability density of W_n is given by

$$f_W(w) = \sum_{i=1}^{M} \pi_i dF_T\left(\frac{w}{\lambda_i - c}\right) \tag{3.10}$$

From (3.9), we can compute the density of Q_{n+1} given the densities of Q_n and W_n. Let $U_n = Q_n + W_n$. Then

$$f_{U_n}(x) = f_{Q_n}(x) * f_W(x) \tag{3.11}$$

where $*$ denotes convolution. It follows that

$$f_{Q_{n+1}}(x) = \begin{cases} f_{U_n}(x) & +[\int_{-\infty}^{0} f_{U_n}(y)dy]\delta(x)+ \\ & +[\int_{B}^{\infty} f_{U_n}(y)dy]\delta(x-B) \quad \text{if } x \in [0, B] \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

where $\delta(.)$ is a Dirac impulse.

We know that $f_{Q_{n+1}}(x) = f_{Q_n}(x)$ in steady state. Furthermore, Q_n converges to this distribution as $n \to \infty$. Our numerical solution relies on computing an approximation of the stationary queue occupancy distribution by discretizing (3.12).

We define two discrete random variables Q_n^L and Q_n^H in the following way. Let $d = B/N$ be the quantization interval size, where B is the buffer size and N a positive integer. Let

$$\begin{aligned} Q_n^L &= d\lfloor Q_n/d \rfloor \\ Q_n^H &= d\lceil Q_n/d \rceil \end{aligned} \tag{3.13}$$

so that $Q_n^L \leq Q_n \leq Q_n^H$. Let

$$\begin{aligned} q_n^L(i) &= \Pr\left\{Q_n^L = id\right\} = \Pr\left\{Q_n \in [id,(i+1)d)\right\}, i = 0,\ldots,N \\ q_n^H(i) &= \Pr\left\{Q_n^H = id\right\} = \Pr\left\{Q_n \in ((i-1)d,id]\right\}, i = 0,\ldots,N \end{aligned} \tag{3.14}$$

denote the corresponding densities. We define discrete upper and lower bounds for W_n in the same way. Then we can use equation (3.11) above to find upper and lower bounds for U_n as follows

$$\begin{aligned} u_n^L(i) &= \sum_{j=0}^{N} q_n^L(j) w^L(i-j) \\ u_n^H(i) &= \sum_{j=0}^{N} q_n^H(j) w^H(i-j). \end{aligned} \tag{3.15}$$

Finally, as the probability mass of U_n in $(-\infty, 0]$ and $[B, \infty)$ is concentrated in 0 and B of Q_{n+1}, we can actually concentrate the probability mass of W outside of $[-B, B]$, and instead define

$$w^L(i) = \begin{cases} \Pr\{W \in (-\infty,(i+1)d)\} & i = -N \\ \Pr\{W \in [id,(i+1)d)\} & -N < i < N \\ \Pr\{W \in [id,\infty)\} & i = N \end{cases} \tag{3.16}$$

$$w^H(i) = \begin{cases} \Pr\{W \in (-\infty,id]\} & i = -N \\ \Pr\{W \in ((i-1)d,id]\} & -N < i < N \\ \Pr\{W \in ((i-1)d,\infty)\} & i = N \end{cases} \tag{3.17}$$

$$\tag{3.18}$$

A simple sample path argument shows that the queue occupancy at τ_n for an initially full queue is greater than or equal to the queue occupancy at τ_n for an initially empty queue. Thus, we obtain a lower and an upper bound on the limiting queue occupancy by initializing Q_0^L and Q_0^H to an empty and full queue, respectively, and by iterating (3.15). We have found that in this approach, the lower and upper bounds converge rapidly to the limiting occupancy (a graphical illustration of this is shown below in Figure 3.1). Nevertheless, it is possible to improve the efficiency of the discrete convolution (3.15) by using a fast Fourier transform (FFT) with appropriate zero-padding [83]. This reduces the complexity of the convolution from $O(n^2)$ to $O(n \log n)$.

We derive a numerical procedure to compute the long-term stationary loss rate, defined as the ratio of work lost to work arriving, from the stationary queue occupancy at arrival instants. This is done as follows. The random variable $W_l = (W - (B - Q))^+$ represents

the amount of work lost in an interarrival interval. The loss rate l is then

$$l = \frac{\mathrm{E}[W_l]}{\overline{\lambda}\mathrm{E}[T_n]} \tag{3.19}$$

with

$$\mathrm{E}[W_l] = \int_0^B f_Q(x)\mathrm{E}[W_l|Q=x]\,dx \tag{3.20}$$

and

$$\begin{aligned}
\mathrm{E}[W_l|Q=x] &= \int_0^\infty F_{W_l|Q=x}(y)dy \\
&= \int_0^\infty \sum_{\{i:T_c(\lambda_i-c)-B+x>0\}} \pi_i F_T\left(\frac{y+B-x}{\lambda_i-c}\right)dy \\
&= \frac{\theta}{\alpha-1}\sum_{\{i:T_c(\lambda_i-c)-B+x>0\}} \pi_i(\lambda_i-c)\times\left[\left(\frac{B-x}{\theta(\lambda_i-c)}+1\right)^{1-\alpha}-\left(\frac{T_c}{\theta}+1\right)^{1-\alpha}\right]
\end{aligned}$$

We obtain an upper and lower bound on l using the upper and lower bounds on the queue occupancy derived above. We obtain

$$\frac{\sum_{i=0}^N q^L(i)\mathrm{E}[W_l|Q=id]}{\overline{\lambda}\mathrm{E}[T_n]} \le l \le \frac{\sum_{i=0}^N q^H(i)\mathrm{E}[W_l|Q=id]}{\overline{\lambda}\mathrm{E}[T_n]} \tag{3.21}$$

We have found that the above iterative procedure is quite efficient in terms of computational complexity and that it yields tight lower and upper bounds on the loss rate with a reasonable number of iterations. This is illustrated in Figure 3.1, which shows an example of the upper and lower bounds of the queue occupancy distribution after 5, 10, and 30 iterations when the queue has been discretized using $N = 100$ bins (i.e., $d = B/100$).

3.3 Numerical Results

In this section, we present results of numerical experiments we have conducted using the model and the analysis technique described above. The goal of these experiments is to evaluate the impact on performance of various parameters of the model.

The performance metric we consider is the loss rate, i.e., the ratio of work lost due to buffer overflow to total work arriving at the server. Other performance metrics would be of interest as well. However, the loss rate is a very natural metric for finite buffer queues. Furthermore, it relates to the tail distribution of the queue occupancy, which is the metric considered in most of the analyses pertaining to infinite buffer queues[1]. We also note that the loss rate has tremendous impact on both application performance (e.g., on the quality of the audio or video delivered from a source to a destination) and network performance (e.g., on the number of retransmissions required to achieve reliable communication).

The parameters of interest here are the buffer size B (or rather the normalized buffer size, which is equal to the actual buffer size divided by the service rate c), the cutoff lag T_c,

[1]The overflow probability, i.e., the probability that the queue occupancy exceeds some amount, in an infinite buffer queue is an upper bound to the loss rate in the corresponding finite buffer queue.

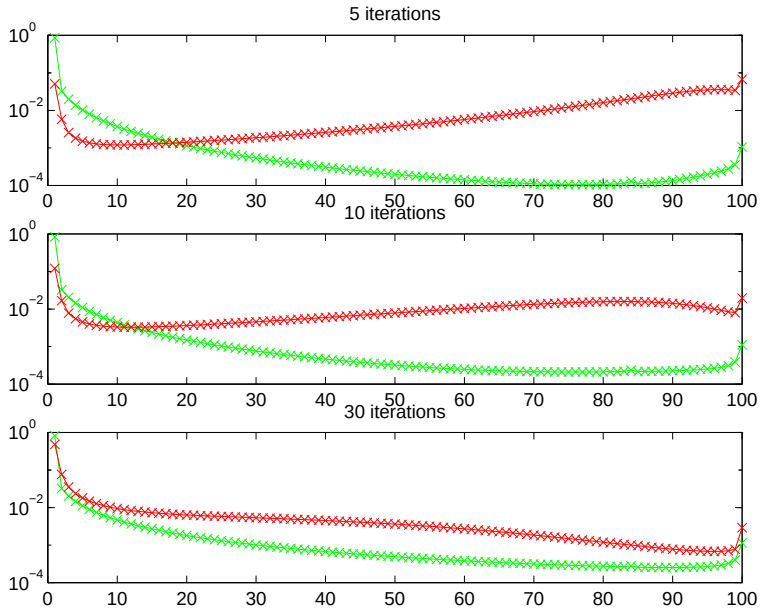

Figure 3.1: The upper and lower bounds on the arrival instant queue occupancy after 5, 10, and 30 iterations.

and the marginal fluid rate distribution Π. We let these parameters vary within ranges consistent with practical networking situations. We use normalized buffer sizes of up to a few seconds. These values are typical of currently available switches. For example, the Fore ATM 200BX/1000 switch has a per-port buffer of 13312 cells. Since the slowest available link on this switch is a 1.5 Mb/s T1 link, the maximum delay in the buffer is equal to 3.3 s. Higher link speeds would yield correspondingly smaller delays.

We use marginal distributions of fluid rates obtained from traces of various traffic sources. In this chapter, we consider two traces. The first trace has been generated by JPEG-encoding an NTSC TV channel (MTV) for one hour. We have recorded this trace on June 11, 1995, 14:59 EST. It includes 107892 frames, with a mean rate of 9.5222 Mb/s. The second trace is based on the August 1989 "purple-cable" Ethernet trace collected at Bellcore [73]. Each trace element is a rate averaged over a 10 ms interval.

Both traces have been found to exhibit long-range dependence (refer to [73] for a detailed analysis of the Bellcore trace). This is visible in Figure 3.2, which shows the variance-time plot of the MTV trace. Estimations of the Hurst parameter from the variance-time plot yield $H_{MTV} \approx 0.83$ and $H_{BC} = 0.9$, this latter value being compatible with the findings in [73]. The variance-time plot estimator is a simple but relatively poor estimator for H. We have used it only to make sure that the value of H in the traces is greater than 0.5. Accurate estimators [2] would confirm this result.

There still remains to match the marginal rate distribution of the traces to the fluid rate vector Π. Figure 3.3 shows the marginal fluid rate distributions for both traces. Recall that the traces represent the amount of work arriving within constant-length time intervals (33ms for the MTV trace, 10ms for the Bellcore trace). Thus, the marginal distribution vectors Π and the rate matrices Λ are simply obtained from a constant bin-size histogram of the traces. We set the number of bins to 50 in all experiments. We determine θ in (3.6) as follows. We first compute the average number of consecutive samples in the trace that fall within the same histogram bin. We then set θ such that the mean interval duration,

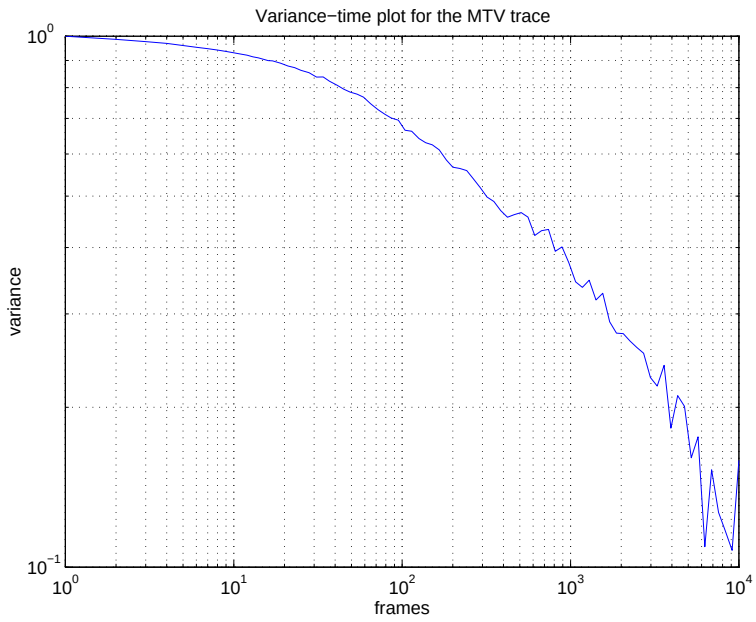

Figure 3.2: The variance-time plot of the MTV trace.

which is given by

$$\mathrm{E}\left[T_n\right] = \frac{\theta}{\alpha - 1}\left[1 - \left(\frac{T_c}{\theta} + 1\right)^{1-\alpha}\right] \tag{3.22}$$

matches this empirical mean for $T_c = \infty$. We find from the trace data that the mean epoch durations are quite short, specifically about 80ms for the MTV trace and 15ms for the Bellcore trace.

All the numerical results reported in this section have been obtained as follows. Recall that the numerical procedure described in Section 3.2 provides an upper and a lower bound on the loss rate. The numbers reported below are the average of the upper and lower bounds. We require that the length of the interval between the upper and the lower bound be less than 20% of the average value. If the upper bound of the loss rate falls below 10^{-10}, we consider that the loss rate is simply below a value that is of practical importance, and report zero loss. We perform iterations until either the above confidence criterion is fulfilled, in which case we stop, or until the two bounds do not move any more, which means that the discretization step is too large to achieve sufficient accuracy. In this case, we double the number of "bins" to discretize the queue, and resume iterating. Note that when doing this, we can use the previous estimations Q_n^L and Q_n^H of the stationary queue distribution for initialization; we do not have to restart with an empty and full queue for the lower and upper bound, respectively. This considerably increases the efficiency of the procedure for the cases where the discretization step has to be quite small.

We have now completed the description of the setup for our numerical experiments. We next describe the experiments themselves.

We have carried out three sets of experiments. In each set, we examine our chosen performance metric (namely the loss rate) as a function of two of the four parameters B, Π, H, and T_c. In the first set, we consider the impact of the buffer size and the cutoff lag

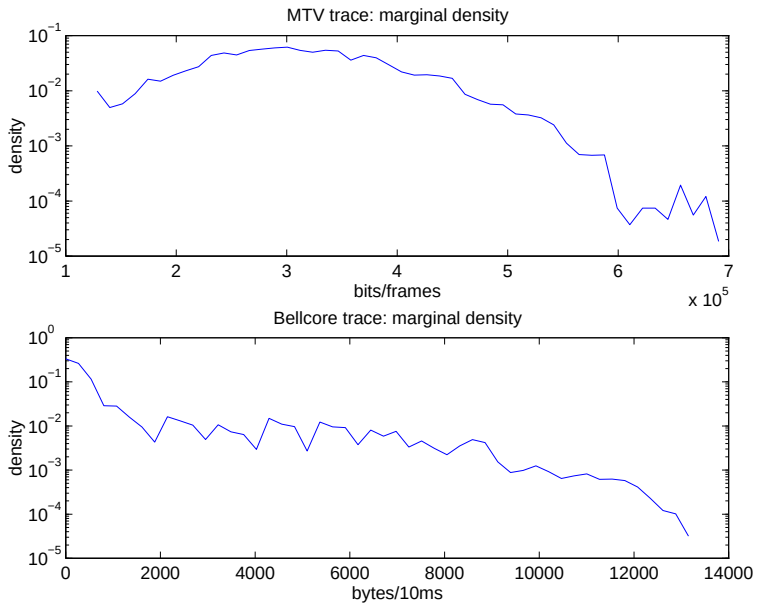

Figure 3.3: The marginal distributions of the MTV and the Bellcore traces.

on the loss rate. In the second set, we consider the impact of the Hurst parameter and the marginal distribution on the loss rate. In the third set, we consider the impact of the buffer size and the marginal distribution on the loss rate.

In the *first set of experiments*, we examine the impact of buffer size (or rather normalized buffer size) and cutoff lag on the loss rate. Figure 3.4 shows the loss rate for the MTV trace with a utilization equal to 0.8. Figure 3.5 shows the loss rate for the Bellcore trace with a utilization equal to 0.4[2].

The figures bring out two important results. First, we observe that for each buffer size, the loss rate is not significantly affected if the cutoff lag increases beyond some value. We refer to this value as the *correlation horizon (CH)*. An important consequence of this is that it is sufficient for a model to take into account correlation up to this correlation horizon to accurately predict loss.

Second, we observe that the rate at which the loss rate decreases as the buffer size increases depends on the value of the cutoff lag. For small cutoff lags, the decrease is approximately exponential. However, as the cutoff lag increases, the rate of decrease actually decreases. This is an illustration of the "buffer ineffectiveness" phenomenon also reported elsewhere (e.g., [60]), whereby increasing buffer sizes beyond a certain value only slightly decreases loss rates. Note that this phenomenon is not unexpected, since an input process with non negligible correlation over long lags generates occasional bursts of traffic that cannot be absorbed even by very large buffers.

Recall that the cutoff lag T_c eliminates correlation in the input fluid process beyond a lag equal to T_c. Thus, its impact is similar to that of the "external shuffling" procedure described in [33]. In this procedure, a time series representing a realization of a process is

[2]These utilizations were chosen to obtain loss rates in a relevant range between approx. 10^{-10} (loss probability of an optical fiber) up to approx. 10^{-1} (loss probability during Internet congestion).

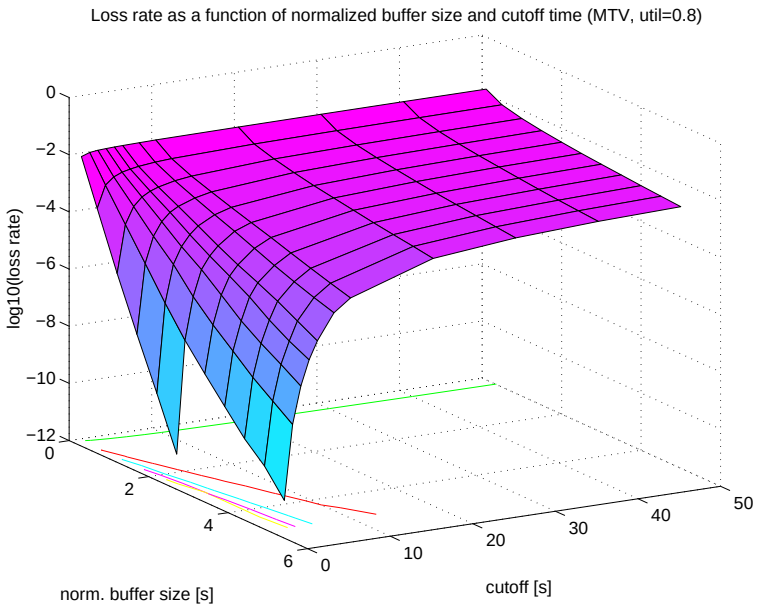

Figure 3.4: The loss rate predicted by the model for the MTV trace as a function of normalized buffer size and cutoff lag, at utilization 0.8.

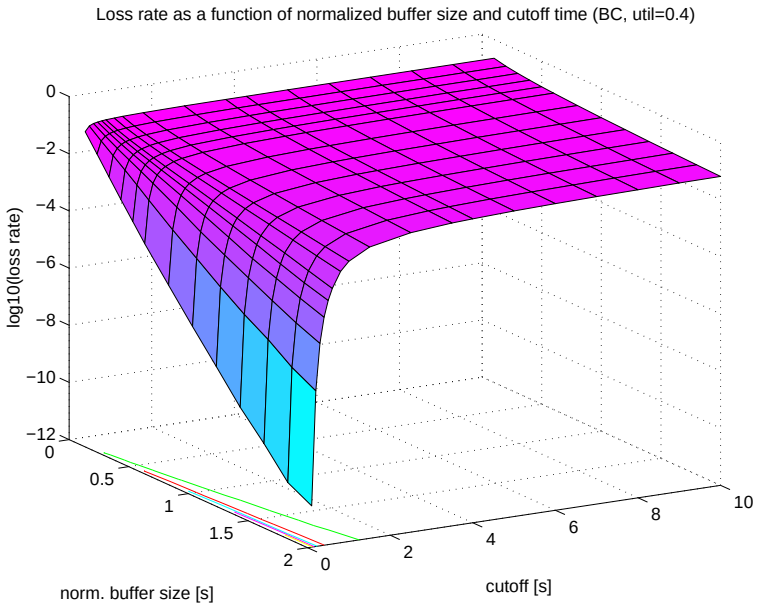

Figure 3.5: The loss rate predicted by the model for the Bellcore trace as a function of normalized buffer size and cutoff lag, at utilization 0.4.

divided up into blocks and the blocks are shuffled (cf. Fig. 3.6). However, the structure of the time series inside a block remains unchanged. Thus, external shuffling removes correlation from the series beyond a lag equal to the length of a block.

Therefore, it is natural to compare the numerical results above to those obtained by external shuffling of the input traces. We have carried out this comparison for both the MTV and the Bellcore traces. Figures 3.7 and 3.8 show the loss rate as a function of buffer size for

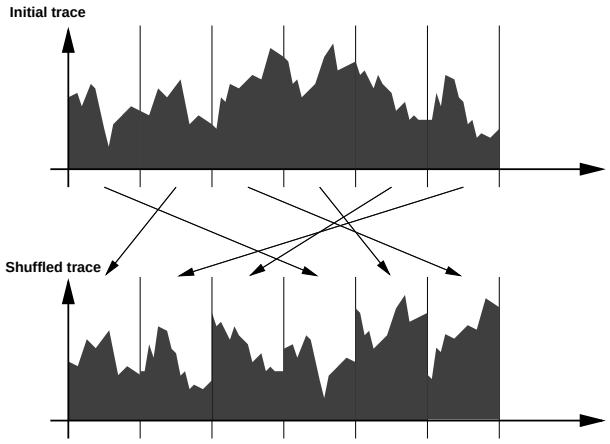

Figure 3.6: Shuffling can be used to destroy correlation beyond a given cutoff lag in a trace.

different values of the shuffle block size (which is referred to as "cutoff" in the figures). We observe that the loss predicted by the model is very close to that obtained with shuffling and simulation for the MTV trace. The agreement is not so good with the Bellcore trace. The discrepancy is probably due to the fact that the residence time distribution for the fluid rates is not matched well enough in our model for this trace. The main results, however, namely the existence of a correlation horizon and the buffer ineffectiveness phenomenon for large values of the cutoff lag, still hold in both figures.

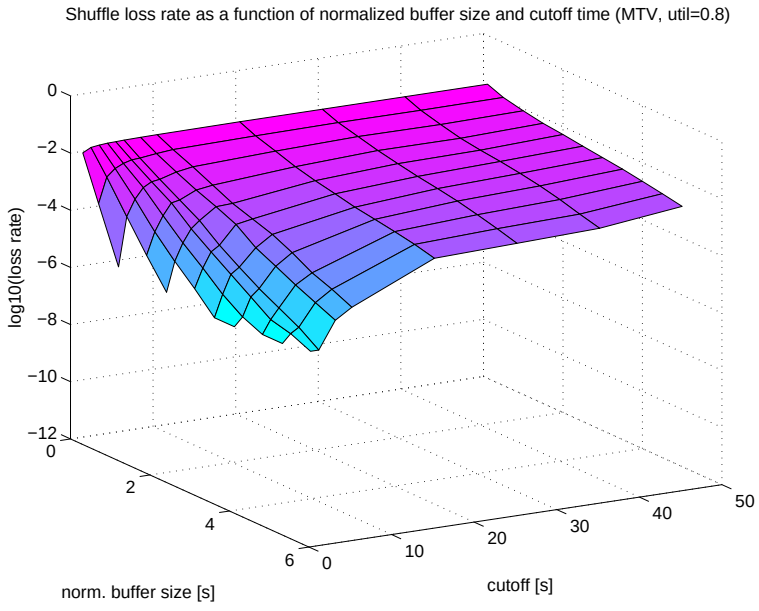

Figure 3.7: The loss rate obtained with shuffling for the MTV trace as a function of normalized buffer size and cutoff lag, at utilization 0.8.

In the *second set of experiments*, we examine the impact of the marginal distribution and the Hurst parameter on the loss rate. To do this, we examine two transformations of

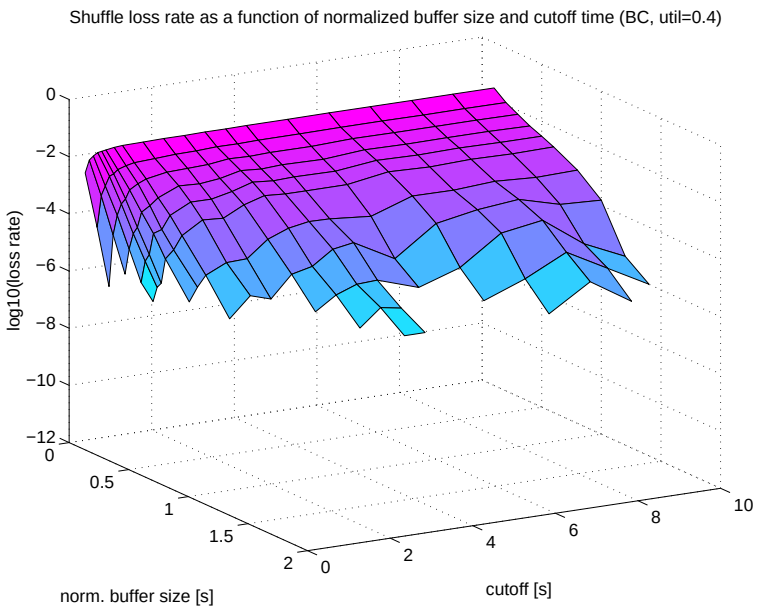

Figure 3.8: The loss rate obtained with shuffling for the Bellcore trace as a function of normalized buffer size and cutoff lag, at utilization 0.4.

the marginal distribution, namely a scaling and a convolution transformation. The first transformation scales the density of the marginal distribution by a constant factor α, while keeping the mean $\overline{\lambda} = \Pi\Lambda\mathbf{1}^T$ constant. Thus, we simply replace λ_i with $\lambda_i' = \overline{\lambda} + \alpha(\lambda_i - \overline{\lambda})$. Figure 3.9 shows the loss rate for H in the range $(0.55, 0.95)$ and α in the range $(0.5, 1.5)$. The normalized buffer size is set to 1 s. The cutoff lag is set to infinity, i.e., there is no cutoff. Note that we use the same θ in the entire experiment, by matching the average interval length for the *nominal* Hurst parameter. We do so in order to avoid that a larger Hurst parameter decreases short-range dependence, which would perturb the assessment of the impact of the Hurst parameter. Referring to Figure 3.4, we note that this cutoff lag is considerably larger than the CH.

The second transformation convolves the original distribution n times and renormalizes it to the original mean. Note that this transformation amounts to considering the superposition of n of the original streams, where the buffer size and the service rate per stream are kept constant. Figure 3.10 shows the loss rate for H in the range $(0.55, 0.95)$ and n in the range $1, \ldots, 10$.

The figures bring out a very interesting result, namely that the impact of the transformations on the loss rate is much greater than that of the Hurst parameter. For example, we observe in Figures 3.9 and 3.10 that changing α from 1.0 to 0.5 or superposing 5 streams decreases the loss rate by more than an order of magnitude. In contrast, changing the value of H has much less of an impact on the loss rate.

This result is confirmed in our *third set of experiments*, in which we examine the impact of the marginal distribution and the normalized buffer size on the loss rate. Figures 3.11 and 3.12 show that small changes in the marginal scaling factor again yield dramatic changes in the loss rate, more so than changes in the buffer size would do. For example, reducing the width of the marginal distribution by a factor of two (from $\alpha = 1$ to $\alpha = 0.5$) decreases the loss rate more than increasing the buffer size even up to 5 s (which is an extremely large value in practice).

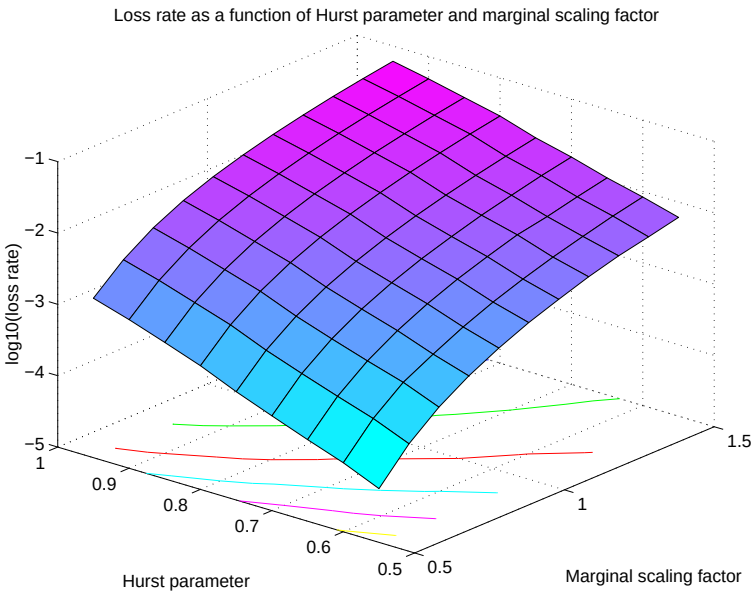

Figure 3.9: The loss rate predicted by the model for the MTV trace as a function of the Hurst parameter and the marginal scaling factor, at utilization 0.8.

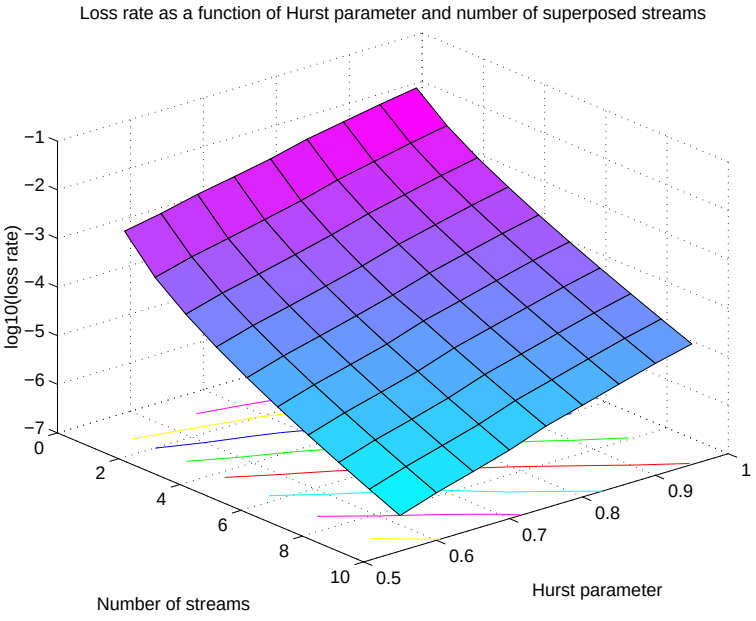

Figure 3.10: The loss rate predicted by the model for the MTV trace as a function of the Hurst parameter and the marginal distribution obtained by n convolutions of the original marginal distribution, at utilization 0.8.

The consequences of this are threefold.

The first consequence relates to modeling. Clearly, the marginal distribution is a crucial parameter and it must be taken into account for accurate loss prediction. This is in agreement with results obtained by others using analytic approaches regarding the impact of the marginal distribution on the tail of the queue occupancy in infinite buffers (e.g.,

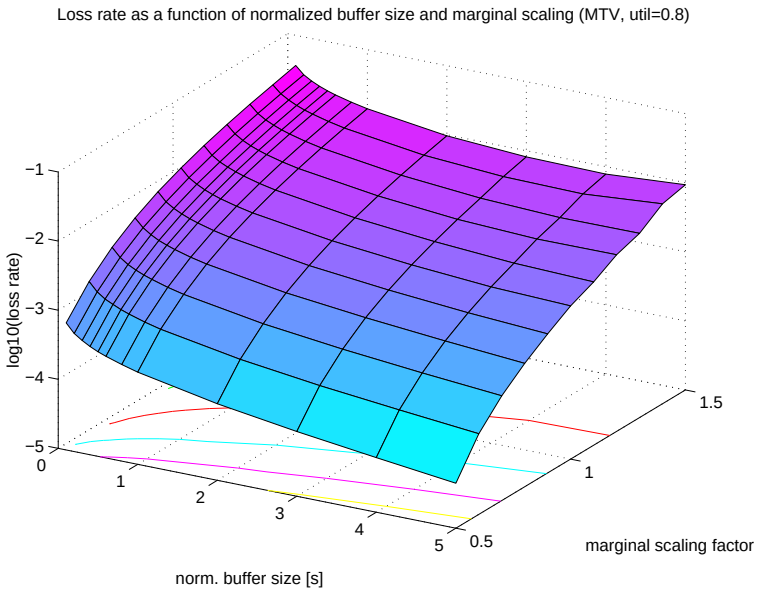

Figure 3.11: The loss rate for the MTV trace as a function of normalized buffer size and marginal scaling factor for a utilization of 0.8.

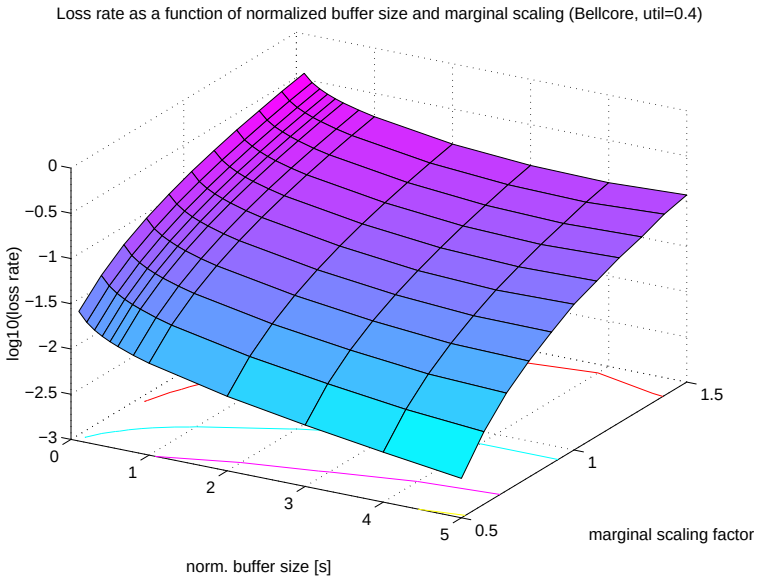

Figure 3.12: The loss rate for the Bellcore trace as a function of normalized buffer size and marginal scaling factor for a utilization of 0.4.

[87]).

The second consequence relates to multiplexing. Our result above, namely that superposing even a moderate number of streams sharply decreases the loss rate, indicates that statistical multiplexing is an efficient mechanism (more so than buffering) to achieve high utilization while keeping loss low [46].

The third consequence relates to traffic control. Indeed, source traffic control mechanisms can be thought of as modifying the scaling of the marginal distribution. Our result then

indicates that such control mechanisms (which might typically be feedback-based) are effective for loss control with LRD sources.

3.4 Discussion

We start this section by summarizing the key results from Section 3.3. These are

- There exists a correlation horizon (CH) such that the loss rate is not affected if the cutoff lag increases beyond CH. Thus, CH separates relevant and irrelevant correlation with respect to the loss rate.
- Large buffers are helpful to significantly reduce the loss rate only for short-range dependent traffic; for long-range dependent traffic, increasing the buffer size has little impact.
- The marginal scaling factor has considerable impact on the loss rate, even more so than the Hurst parameter or the buffer size.
- Adjusting the marginal scaling factor by statistical multiplexing several streams or by using source traffic control mechanisms is a very efficient way of reducing loss while keeping utilization high.

Our observation of the existence of the correlation horizon CH helps resolve the seemingly contradictory conclusions that have been drawn in the literature from experiments with long-range dependent traffic. Indeed, mathematical analysis and simulation of queueing systems with LRD input shows that the queue occupancy exhibits an asymptotic behavior very much different from that observed with Markov sources [82, 28, 33]. However, the literature on Markov modeling reports good performance prediction for finite buffer systems even when input traffic streams are correlated over many time-scales [31, 53, 104]. We have shown that there exists a correlation horizon which separates relevant and irrelevant correlation with respect to the performance measure of interest (namely the loss rate). Intuitively, the CH depends on the correlation structure of the input traffic, and on the system under study, namely the finite buffer queue. Indeed, the finite buffer queue sets a limit on the memory of the system since the the buffer "forgets" about the past as soon as it is either empty or full (this is referred to as the resetting effect in [53]). Therefore, we expect the CH to depend on the maximum queue size (and we provide evidence for this below). Furthermore, while correlation on all time-scales has an impact on performance for the infinite queue, only the correlation up to the CH has an effect in the finite buffer queue. Ryu and Elwalid, in [97], independently came to similar conclusions. They use large deviations theory to derive an expression for the correlation horizon (called *Critical Time Scale*). Their traffic process is a superposition of a LRD and a SRD process. While their approach is computationally less demanding than ours, they do not study the impact on the correlation horizon of the marginal distribution and of different Hurst parameters. Also, their performance metric is the buffer overflow probability for an infinite queue, not the loss rate.

We next describe a simple way to estimate CH in terms of the various system parameters. The estimation procedure is based on the resetting argument mentioned earlier, i.e., we assume that when the buffer becomes empty or full, information about the past is lost. Then, we take the correlation horizon estimate T_{CH} to be the time interval for which the probability that the buffer empties or overflows at least once is close to one.

We make the assumption that the correlation horizon is much longer than the average interarrival time μ. If the converse were true, then the buffer would either empty or overflow with very high probability within a single interarrival interval. This would mean that the utilization is close to zero, or that the loss rate is extremely high.

Let $\Gamma_n = \sum_{i=1}^{n} W_n$ denote the sum of the excess work in n consecutive intervals, and $\Theta = \sum_{i=1}^{n} T_n$ denote the sum of the lengths of the n intervals. Now assume that we are looking at the queue at an arrival instant τ_m and that the queue occupancy at that instant is equal to Q. Note that the probability of either emptying or overflowing the buffer at some point during the next n intervals is bounded below by $\Pr\{Q + \Gamma_n \leq 0\} + \Pr\{Q + \Gamma \geq B\}$. We wish to find n such that the probability $p = 1 - \Pr\{Q + \Gamma_n \leq 0\} - \Pr\{Q + \Gamma \geq B\}$ of no reset occurring during n intervals is very small.

Given our assumption that $T_{CH} \gg \mu$, we look for n large. Then, the central limit theorem says that for large values of n, $Q + \Gamma_n$ is approximately normally distributed with mean $Q + n\mu(\overline{\lambda} - c)$ and variance $n\sigma_T^2\sigma_\lambda^2$, where μ is the average interval length, $\overline{\lambda}$ is the average rate, σ_T^2 is the variance of the interarrival time, and σ_λ^2 is the variance of the marginal distribution. Furthermore, Θ is approximately normally distributed as well, with mean $n\mu$ and variance $n\sigma_T^2$.

As we do not explicitly know the distribution of Q, we find an upper bound on p by setting the mean of the normal distribution to $B/2$, which corresponds to selecting the value of Q that is least likely to result in a reset during the next n intervals. With this assumption, the upper bound on p becomes $p \leq \mathrm{erf}(\frac{B}{2\sqrt{2}n\sigma_T\sigma_\lambda})$, where erf(.) denotes the error function. Let us now choose T_{CH} such that the probability that n intervals fit into T_{CH} is high, i.e., $T_{CH} = n(\mu + \beta\sigma_T)$, where β is a small constant (e.g., $\beta = 2$ results in this probability being ≈ 0.9.) Then we may write T_{CH} as

$$T_{CH} = \frac{B}{2\sqrt{2}\sigma_T\sigma_\lambda \mathrm{erf}^{-1}(p)}(\mu + \beta\sigma_T) \tag{3.23}$$

The above estimate provides valuable insight into the impact on CH of the various parameters of the model. In particular, as suspected earlier, we find that T_{CH} scales linearly with the buffer size. We also note that T_{CH} is inversely proportional to the standard deviation of the marginal distribution. This is intuitively not surprising. On the other hand, scaling the marginal distribution by a factor α has exactly the opposite impact (refer to Figure 3.9 and the discussion there). Finally, we observe that, as σ_T increases, T_{CH} converges to a limiting value that depends only on the buffer size and on the width of the marginal distribution. This is in agreement with the "flattening out" of the loss rate with increasing cutoff lag observed in Figures 4 to 7.

We have focussed in this chapter on a particular instance of performance prediction relying on traffic modeling, where the performance metric has been defined as the long-term loss rate. We have then shown that the relevant correlation is limited to a time-scale smaller than the correlation horizon. This might seem to support the claim that only models with limited memory (up to CH) make sense, and hence that self-similar models have no place in cases like these. Instead, this means that for this type of performance problem, we may choose any model among all the available models as long as the chosen model captures the correlation structure up to CH. The choice can be based on analytic tractability, on ease of parameter identification from traces, etc. In this chapter, we chose the truncated Pareto model because it is a parsimonious model which provides a simple way to control its correlation structure (via T_c), and because it is self-similar when T_c is set to infinity. However,

Markov models would be another possible choice since they can capture correlations up to a given value CH (and since the numerical procedure developed in Section 3.2 can be used independent of the particular model).

And indeed, several studies have used Markov models to approximate traffic sources with long range dependence. This approach can be used to obtain accurate performance predictions since a power law decay can be approximated arbitrarily closely by enough exponential decay functions [74]. However, the resulting Markov models are complex multi-state models that do not follow the principle of parsimonious modeling because every state added to such a model also adds several free parameters. This presents two problems, namely that of identifying the parameters (states and state transition rates) from experimental data, and that of obtaining closed-form analytic expressions for performance measures. The first problem is the more important one because it is often difficult and time consuming in practice to collect the data required for parameter estimation. This problem is generally used to promote instead parsimonious long-range dependent models.

However, we should note that parsimonious modeling and LRD are really orthogonal issues. Indeed, it is possible to reduce the impact of the parameter-explosion problem with multi-state Markov models, where each state models a different time scale. We describe the details of one such model in the Appendix.

3.5 Conclusion

We have focussed in this chapter on a particular instance of performance prediction relying on traffic modeling, where the performance metric has been defined as the long-term loss rate. We have then shown that the relevant correlation is limited to a time-scale smaller than the correlation horizon CH, meaning that for this type of performance problem, we may choose any model among the panoply of available models (including Markovian and self-similar models) as long as the chosen model captures the correlation structure up to CH.

However, we would like to conclude by stressing that the amount of correlation that must be taken into account in a model depends in general on the performance metric of interest. We illustrate this with a simple example. Consider the problem of comparing the performance of closed-loop (ARQ) and open-loop (FEC) error control schemes for reliable point-to-point communications. Let us try to guess the relevant time-scales of this problem. ARQ schemes perform well when losses are bursty because they can accumulate information about a loss burst and request retransmission of all packets lost in the burst in one go. FEC schemes perform well when losses are spread out over time because they can correct errors of the type "among n packets, $k \leq k_{\max} < n$ have been lost". The probability that k exceeds $k_{\max}$ is smaller for independent losses than for correlated losses. This suggests that extending the time-scale of the correlation structure of the packet arrival process in a model of error control schemes amounts to increasing the advantage of ARQ over FEC. Therefore, it seems necessary in this problem to accurately model the arrival and loss processes over a wide range of time-scales. Thus, a self-similar model would be appropriate here.

In general, models of self-similar processes capture the behavior of these processes over *all* time-scales using a small number of parameters (i.e., they are parsimonious models). They are particularly well suited in a number of different cases. For example, in cases such as that described above for the ARQ/FEC evaluation when the system under study does not have a clearly bounded time-scale. Also, in cases when we wish to generate traces that

match LRD behavior observed in actual networks. Since we do not want to restrict the use of these traces to a specific modeling task, we cannot make any assumptions about the relevance of the model parameters. Thus, traces should be derived from a self-similar traffic model. Of course, our results in Section 3.3 show that trace generation based on self-similar models (or any other model for that matter) must also take into account statistics such as the marginal distribution in order to achieve accurate performance prediction.

Appendix: A Parsimonious Markov Model

We would like to model transitions occurring over K different time-scales using a K state model with appropriate transition rates. Desirable properties for the transition rates are i) they should be relatively easy to determine from experimental data and ii) they should cover as wide a range as possible. In our model, we choose these rates so that they form a geometric progression.

Specifically, we obtain our Markov model by replacing the truncated Pareto interarrival process with a Markov chain as follows. The chain is initially in a state S. There are K intermediate states, labeled $Z_0, Z_1, \ldots, Z_{K-1}$. The only possible transitions from the intermediate states Z_i are into a state E (cf. Fig. 3.13). The transition rates from S to Z_i are set to $\rho^{-j(3-2H)}$, where $\rho > 1$ is a constant. The transition rates from Z_i to E are set to $\beta\rho^{-j}$. We compute the distribution of the time between the transition out of state S and the transition into state E. This distribution has the following survival function.

$$F_T(t) = \frac{1-\rho^{-(3-2H)}}{1-\rho^{-K(3-2H)}} \sum_{j=0}^{K-1} \rho^{-j(3-2H)} \exp(-\beta\rho^{-j}t). \tag{3.24}$$

We have shown previously (cf. (3.3)) that the model's covariance function is proportional to $p(t)$, the survival function of residual life. For our Markov model, we find

$$p(t) = A \sum_{j=0}^{K-1} \rho^{-j(2-2H)} \exp(-\beta\rho^{-j}t), \tag{3.25}$$

where $A = (1-\rho^{-(2-2H)})/(1-\rho^{-K(2-2H)})$ is a normalization constant.

We show that the covariance function of this Markov model indeed exhibits the desired scaling behavior over a finite, but arbitrary set of time-scales. For this, note that

$$\begin{aligned} p(\rho t) &= A \sum_{j=0}^{K-1} \rho^{-j(2-2H)} \exp(-\beta\rho^{-j+1}t) \\ &= A\left[\rho^{-(2-2H)}p(t) + \rho^{(2-2H)}\exp(-\beta\rho t) - \rho^{-(K-1)(2-2H)}\exp(-\beta\rho^{-(K-1)}t)\right] \end{aligned} \tag{3.26}$$

The first term corresponds to the scaling behavior of the covariance function of a second-order self-similar process with Hurst parameter H. As we approximate an ideal, hyperbolically decreasing covariance function with a sum of exponentials, this only holds, of course, for scalings that are multiples of ρ. A smaller ρ yields a better approximation, at the expense of more states in the model. Note that the two other terms in (3.26) vanish as $K \to \infty$ and $t \to \infty$. In other words, as the number of states grows, the model becomes asymptotically second-order self-similar. The parameter β can be used to match,

for example, the average residence time $\mathrm{E}[T_n]$ per bandwidth level, given by

$$\mathrm{E}[T_n] = (A\beta)^{-1} \frac{1-\rho^{-(3-2H)}}{1-\rho^{-K(3-2H)}}. \tag{3.27}$$

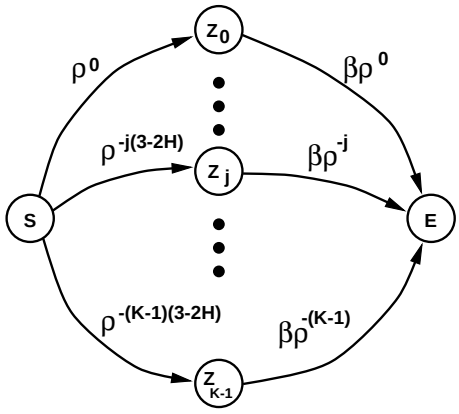

Figure 3.13: The building block for the parsimonious Markov model that exhibits LRD behavior over a limited range of time-scales.

The fact that fluid rates in different intervals are iid allow us to associate with each possible fluid rate a "copy" of this Markov chain. We use superscripts to identify what fluid rate a state of the above chain is associated with. We now remove the start and end states of each level and set the transition rate from state Z_i^m to state Z_j^n such that the total fluid rate out of state Z_i^m is still $\beta\rho^{-i}$, and such that the probability that the resulting total chain is at fluid level λ_n is π_n. This can be achieved by setting this transition rate to

$$A\beta\pi_n\rho^{-[i+j(3-2H)]}. \tag{3.28}$$

A similar model has been proposed in [95]. Our model differs in a significant way. While our model explicitely contains the Hurst parameter H, a numerical fitting algorithm must be used for the model in [95] to match a given Hurst parameter.

Chapter 4

Renegotiated Constant Bit Rate Service

Notation

r_i	amount of data entering queue during time slot i
$s_i \in \{c_0, c_1, \ldots, c_{K-1}\}$	service rate during time slot i
N	session duration in time slots
ϕ	renegotiation cost
γ	bandwidth cost
b_i	buffer occupancy at the end of time slot i
$\widehat{r}_i$	predicted rate at time slot i
Δ	bandwidth allocation granularity
s_{new}	new bandwidth request
B_l, B_h	lower and upper buffer threshold

The work in this chapter has appeared in [47, 46, 48].

4.1 Introduction

Video traffic is expected to be a significant component of the traffic mix in integrated services networks. Video is invariably compressed with either constant quality (and variable bit rate) or constant bit rate (and variable quality). Constant bit rate compressed video, which is predominant in current networks, may exhibit visual glitches in information-rich scenes. To minimize these glitches, the coding rate has to be large enough to encode all but a few of the scenes in the video stream, leading to a reduction in the available statistical multiplexing gain [112]. This has led to great interest in variable bit rate video compression, and techniques for carrying such traffic in computer networks [22, 23, 85, 86, 71, 1]. [1]

A key characteristic of a compressed video source is its *burstiness*. A bursty source occasionally transmits at a peak rate significantly larger than its long term average rate. Recent research has determined another key characteristic: the presence of traffic variations over multiple time-scales [91, 92, 37, 38]. Intuitively, there is a variation in source rate not only over a period of milliseconds to seconds, corresponding to variations within a scene, but

[1] For notational convenience, we will refer to a variable bit rate compressed video source simply as a 'compressed video source'.

also over a period of tens of seconds to minutes, corresponding to scenes with differing information content. Taken together, these facts imply that a compressed video source can transmit at its peak rate over multiple time scales.

We discuss in Section 4.2 how burstiness at multiple time scales, in conjunction with a traditional one-shot (static) traffic descriptor (such as a leaky bucket), leads to performance problems. Instead, we argue that a *renegotiated service* best addresses the presence of burstiness over multiple time-scales. This motivates the design of Renegotiated Constant Bit Rate (RCBR) service for carrying compressed video traffic. Sources using RCBR service are presented with an abstraction of a fixed-size buffer drained at a constant bit rate called the *drain rate*. Sources typically choose the drain rate to match their current short-term average rate and renegotiate this rate in response to changes in their workload. Because all traffic entering the network is CBR, RCBR requires minimal buffering and scheduling support in switches. We show in Section 4.3 that the service is suitable for both stored and online video sources and that its signaling overhead is likely to be manageable with current technology. Note that RCBR is the simplest possible renegotiated service.

An RCBR source must determine *when* to renegotiate and *what* rate to choose during a renegotiation. These decisions constitute a per-source *renegotiation schedule*. We present algorithms for choosing renegotiation schedules in Section 4.4.

We evaluate RCBR both analytically and through simulation. Our results in Section 4.5 indicate that RCBR allows a network operator to extract almost all of the statistical multiplexing gain achievable by multiplexing large numbers of compressed video sources. For example, if an MPEG-1 compressed version of the "Star Wars" movie is transferred through RCBR service, and if the average service rate over the lifetime of the connection is 5% above the average source rate of 374kbps, then 300kbit worth of buffering at the end-system and an average renegotiation interval of about 12s are sufficient for RCBR. In contrast, a non-renegotiated service with the same service rate would require about 100Mbit of buffering at the end-system (cf. Fig. 4.6).

A natural question is how to admit RCBR sources into the network while still allowing network operators to stochastically bound performance metrics such as the renegotiation failure probability and link utilization. We discuss simple measurement-based admission control schemes suitable for RCBR sources in Chapter 5. We show that a memoryless scheme is not robust. We advocate the use of memory, i.e., history about the past bandwidth of calls, to achieve satisfactory robustness.

While our focus is on compressed video traffic, our results are applicable to multiple time-scale traffic in general. Sections 4.6 and 4.7 present related work and place our work in context of other services for carrying variable bit rate traffic.

4.2 Performance Problems for Multiple Time-Scale Sources

It has been observed by several researchers [91, 92, 37, 38] that compressed video traffic typically exhibits burstiness over multiple time-scales. While the short term burstiness of MPEG sources due to the I, B, and P frame structure is well known, they have found fairly long duration, as long as 30 seconds, when the data rate of the video source is continuously near its peak rate. This is due to scenes with considerable motion or rapid chrominance and luminance changes such as those caused by flashing lights, where, independent of the coding algorithm, the coder generates traffic near its peak rate. Unfortunately, these peak rates are much higher than the long term average rate. For example, we find that for an

MPEG-1 compressed version of the "Star Wars" movie, there are episodes where a sustained peak of five times the long term average rate lasts over 10 seconds.

Compressed video traffic is expected to be carried in ATM networks using either CBR or VBR service, and in the Integrated Services Internet using Guaranteed Service [16] (The Integrated Services Internet also allows for a VBR-like Controlled Load service, but this service is too imprecisely defined at the time of this writing to map to any of the ATM classes. Therefore, our remarks apply only to the Internet's Guaranteed Service class.) With CBR service, a source is restricted to a bit rate that it chooses at the time of connection setup. With VBR or Guaranteed Service, the source chooses both a *token bucket size* and a *token rate*. These correspond roughly to the largest size burst allowed from the source into the network and its long-term average rate.

We model all three services as follows. Traffic from a source is queued at a buffer at the end-system, and the network drains the buffer at a given *drain rate*. The drain rate for a CBR source is the connection rate, and for a VBR or Guaranteed Service source is the token rate. With VBR service, data may leave the buffer at a rate greater than the drain rate if the token bucket is non-empty. The key fact is that with a non-renegotiated service, as is the case in both ATM and Integrated Services Internet proposals, a source chooses the drain rate exactly once, at the time of connection establishment. (The Integrated Services Internet proposal– specifically the RSVP resource reservation protocol– does require a source to periodically refresh its reserved rate, and renegotiation could be piggybacked with a refresh. However, refreshes are currently viewed primarily as a mechanism for state management, rather than for rate adaptation. Sources are therefore expected to choose a token rate once and to merely repeat this request when refreshing their reservation [121]).

If sources exhibiting bursts at multiple time scales are allowed only a single drain rate to describe their behavior, they are faced with a set of poor choices. Assume, for the moment, that the drain rate is chosen close to the long term average rate in order to maximize the statistical multiplexing gain in the network. Then, during sustained peaks, the source buffer fills up at the peak rate and is drained at the drain rate. If the peak rate is much higher than the average rate, either the data buffer has to be very large, or the loss rate will be unacceptably high. If the loss rate is made small by provisioning large data buffers, this leads to expensive buffers at end-systems and long delays for the sources. Even if the data buffering costs are not excessive, the ensuing delays may not be tolerable for interactive applications.

With VBR or Guaranteed service, we can deal with sustained bursts by choosing a large token bucket, thus admitting part or all of the burst into the network. We call this the *unrestricted sharing* approach to dealing with bursts. The problem with this approach is that unless intermediate switches and the receiver have large data buffers (which, in some cases, may need to be on the order of tens of megabytes), sources have no assurance that their data will not be lost if bursts coincide. We call this loss of *protection*. Providing protection with unrestricted sharing is expensive and can potentially lead to excessive queueing delays. Note that there is a tradeoff between the drain rate and the largest-size burst that may enter the network. A source can minimize delay and reduce the probability of cell loss, but only at the expense of a reduced statistical multiplexing gain.

Thus, burstiness at slow time-scales with a non-renegotiated drain rate leads either to a) loss of statistical multiplexing gain, b) large data loss rate, c) large buffers in end systems or switches, leading to delays and expensive line cards or d) loss of protection. Current (non-renegotiated) services cannot simultaneously avoid all four problems because sustained peaks in workload are not adequately captured by a static descriptor, such as a

leaky bucket. We argue that these peaks are better captured by renegotiation of the drain rate at a slower time-scale. A more detailed discussion of the effectiveness of renegotiation in solving these problems can be found in Section 4.7.

4.3 The RCBR Scheme

4.3.1 RCBR Service Description

With static CBR service, during call setup, a source requests a constant bandwidth from the network [61, 62]. Because a source is described by a single number, the admission control test is trivial. Moreover, because traffic entering the network is smooth, internal buffers can be small and packet scheduling need only be FIFO [45]. With RCBR, a source can renegotiate its service rate. Renegotiation consists of sending a signaling message requesting an increase or decrease in the current service rate. If the request is feasible, the network allows the renegotiation. Upon successful completion of the request, the source is free to send data at the new CBR rate. RCBR therefore retains the simplicity and small buffer requirements of CBR service at the expense of the signaling overhead needed to implement renegotiation.

Renegotiation failure What happens if a renegotiation fails? A trivial solution is to try again. Of course, data will build up in the source's data buffer while the second request proceeds and there is the possibility of excessive delay, and even data loss. This may not be acceptable for some sources. Such sources might reserve resources at or close to the peak rate, so that the frequency of renegotiation is highly reduced and so is the possibility of renegotiation failure. There is a three-way tradeoff between buffer size (and delay), requested rate and the frequency of renegotiation. In any case, note that even if the renegotiation fails, *the source can keep whatever bandwidth it already has.*

Second, during admission control, a switch controller might reject an incoming call even if there is available capacity, if the resources used by the new call will make future renegotiations more likely to fail. This allows the network operator to trade off call blocking probability and renegotiation failure probability. We consider admission control in more detail in Chapter 5.

Finally, the signaling system could ask the user or application (perhaps out of band) to reduce its data rate. Since the network interface (i.e. the session layer) is expected to be no more than a few milliseconds away from the end point, the control loop between the network interface and the user will be tight, so that responding to such signals should be easy, particularly for adaptive codecs [65]. Recent work suggests that even stored video can be dynamically requantized in order to respond to these signals [98, 30].

Thus, there are several viable alternatives for dealing with renegotiation failures. With an appropriate combination, some users can choose to see few or no renegotiation failures, while others might trade off a non-zero renegotiation failure rate for a lower cost of service.

Stored and Interactive Sources Stored video (off-line) and interactive (on-line) applications use RCBR services differently. Off-line sources can compute the *renegotiation schedule* in advance and can initiate renegotiations in anticipation of changes in the source rate. Moreover, if all systems in the network share a common time base, advance reservations could be done for some or all of the data stream [118]. Interactive applications must compute the renegotiation schedule on-the-fly. For such applications, an active component could monitor the buffer between the application and the network and initiate renegotia-

tions based on the buffer occupancy. This monitor could be part of the session layer in an ISO protocol stack, or reside in the Network Interface Unit (NIU) for "dumb" endpoints. It would need to be activated only when data is written to or drained from the buffer. Note that in both the on-line and off-line cases, renegotiation signaling and data transfer occur in parallel. Algorithms for computing the renegotiation schedule for off-line and on-line applications are presented in Section 4.

4.3.2 Implementation

During renegotiation, a switch controller need *not* recompute routing, allocate a connection identifier or acquire housekeeping records. Thus, signaling for renegotiation is much less expensive than signaling for call setup and need not use the same protocol. This allows us to exploit lightweight signaling mechanisms for renegotiation. A hardware implementation of signaling for renegotiation is described in [12].

In an ATM network, sources can reuse the Resource Management (RM) cell mechanism, originally proposed for ABR service, for lightweight signaling. An RCBR source sets the Explicit Rate (ER) field in the RM cell to the difference between its old and new rates [2]. On receiving an RM cell, a switch-controller (or a dedicated hardware module, as in ABR) determines the output port of the VCI in one lookup, and the utilization and capacity of the output port in a second lookup. With this information, it checks if the current port utilization plus the rate difference is less than the port capacity. If this is true, then the renegotiation request succeeds at this switch, and the VCI and port statistics are updated. Otherwise, the controller modifies the ER field to deny the request, and the renegotiation is canceled on the reverse path. Note that the logic to modify the ER field with RCBR is simpler than that required for fair-share computation in ABR. Thus, the deployment of ATM switches with ABR support is an existence proof that RCBR support in ATM switches is feasible.

In the Integrated Services Internet, sources and receivers periodically refresh their network reservation state using the RSVP signaling protocol [121]. A source periodically emits a PATH message describing its characteristics, and each receiver periodically emits a RESV message requesting a reservation. To renegotiate its service rate, a source should change its traffic description (flowspec) in the PATH message, and the receivers should correspondingly change their reservation in the RESV message. We do not have enough experience with RSVP to determine whether this mechanism is sufficiently lightweight for renegotiation. If this is not the case, we may need to augment RSVP with a lightweight renegotiation protocol. In any case, we anticipate that renegotiations will happen only around every ten seconds or so (see Section 4.5.2), so the overhead for RCBR at each source is inherently small.

Note that in order to limit the renegotiation rate, it is likely that a user will be charged for each renegotiation, just as users are now charged per call setup. This affects the choice of renegotiation schedule, as discussed in Section 4.

[2] We use a difference because this simplifies the computation at the switch controller, which need not keep track of the source's rate. This has the problem of parameter drift in case of RM cell loss. To overcome this, we can resynchronize rates by periodically sending an RM cell with the true explicit rate, instead of a difference.

4.3.3 Scaling

We now consider how well RCBR scales with latency in the path, number of sources, and number of hops. Scaling with path latency is different for on-line and off-line applications. Off-line applications are insensitive to path latency because they can compensate for an increased latency by initiating renegotiation earlier. However, the performance of applications with on-line RCBR decreases with an increase in latency, because these applications must predict their future data rate, and prediction accuracy decreases with increased latency. This can be compensated for by increasing the end-system buffer or by asking for more bandwidth than needed, thus reducing the statistical multiplexing gain. We do not yet have analytical expressions or simulation results studying the effect of renegotiation delay on RCBR performance.

Signaling load increases linearly with the number of RCBR sources in the network. With hardware support, we believe that an ATM switch can support several tens of thousands of simultaneous RCBR sources [46]. The bottleneck in RM cell processing is the time taken to lookup per-VCI state. Since RCBR support does not require per-VCI state, we do not anticipate difficulties in scaling ATM switches to handle more renegotiating sources. Scaling of Integrated Service routers is still a matter of speculation.

As the mean number of hops in the network increases, the probability of renegotiation failure is likely to increase, since each hop is a possible point of failure. Moreover, the net renegotiation signaling load on the network also increases. However, if there is a simultaneous increase in the number of alternate routes in the network, then load balancing at the call level might reduce the load at each hop, thus compensating for this increase. This is still an open area for research.

4.4 Computation of Renegotiation Schedules

In this section, we address the problem of deciding when to request a bandwidth renegotiation from the network, and how much bandwidth to ask for. We present two algorithms that transform a given data rate function into a stepwise CBR data rate function. The first algorithm determines an optimal schedule for a playback application based on total knowledge of the user's data rate function and a pricing model discussed below. The second algorithm is a causal heuristic that could be used for interactive users, where the rate function is not known in advance.

4.4.1 Optimal Renegotiation Schedule

We model the problem with a slotted time queue. For video, a time slot would typically be the duration of a frame. Renegotiations occur on the boundary between slots. Let $r_i, i = 0, 1, \ldots, N-1$ denote the amount of data entering the queue during time slot i, and let s_i denote the service rate during time slot i. The session duration is N time slots. We assume the service rate during any time slot is in a given set $\mathcal{C} = \{c_0, c_1, \ldots, c_{K-1}\}$.

We have assumed a constant cost per renegotiation ϕ and a cost γ per *allocated* bandwidth and time unit. Therefore, the total cost is given by

$$\phi \cdot \sum_{i=1}^{N-1} (1 - \delta(s_{i-1}, s_i)) + \gamma \cdot \sum_{i=0}^{N-1} s_i \tag{4.1}$$

with

$$\delta(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

For a given r_i, the optimal allocation minimizing the total cost, has to be found, subject to the buffer constraint

$$0 \leq b_i \leq B \text{ for } i = 0, 1, \ldots, N-1 \tag{4.2}$$

where b_i is the queue size at the end of time slot i, with

$$b_i = \begin{cases} 0 & \text{if } i < 0 \\ \max\{b_{i-1} + r_i - s_i, 0\} & i = 0, 1, \ldots, N-1 \end{cases} \tag{4.3}$$

We solve this optimization problem with a Viterbi-like algorithm [115]. Let us first introduce some notation (cf. Fig. 4.1). A *node* is a 4-tuple (i, k, b, w), where i denotes (discrete) time, $k \in \{0, \ldots, K-1\}$ denotes a bandwidth allocation $c_k \in \mathcal{C}$, $b \in \{0, \ldots, B\}$ denotes a buffer occupancy, and w denotes the weight, which equals the partial cost of the best path to this node. A *branch* connects a node (i, k, b, w) to another node $(i+1, k', b', w')$ if $b' = \max\{b_i + r_{i+1} - c_{k'}, 0\}$. It has an associated weight of $\gamma \cdot s_{i+1} + \phi \cdot (1 - \delta(s_i, s_{i+1}))$. A branch represents one step in the evolution of the system state, given a choice of the new rate allocation $c_{k'}$. A *path* is a sequence of branches. The cost of a path is the sum of the cost of its branches. All possible paths form the *trellis*. A *full path* is a path connecting a node with $i = 0$ with a node with $i = N-1$, and corresponds to a feasible renegotiation schedule.

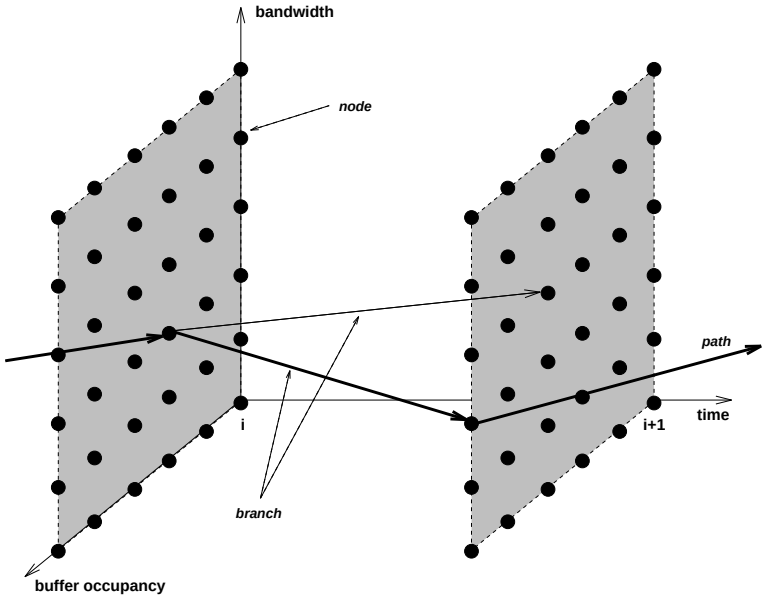

Figure 4.1: An illustration of the trellis to be used for the Viterbi-like algorithm.

Now we can formulate the optimization problem as follows: find the shortest path from some node at time zero to some node at time $N-1$. The algorithm to do this is presented below.

1. Set $i = 0$. Create the initial set of nodes $(0, k, 0, 0)$ for $k \in \{0, \ldots, K-1\}$.

2. Create all the branches between nodes of slot i and nodes of slot $i+1$. Set the weight according to (4.1) for the nodes of slot $i+1$.

3. Prune paths according to Lemma 1 given below.

4. Increment i and repeat steps 2 and 3 as long as $i < N$.

5. Choose one of the paths with the minimum weight as the solution.

We now present a lemma that governs the pruning of paths.

Lemma 1 *A path X going through a node $x = (i, k_x, b_x, w_x)$ is not optimal if there exists a path Y through a node $y = (i, k_y, b_y, w_y)$ such that* [3]

$$b_y \leq b_x \text{ and } w_y \leq w_x - \begin{cases} \phi & \text{if } k_y \neq k_x \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

Proof: Assume the condition is true. First, if $k_x = k_y$, then path Y has smaller or equal buffer occupancy and smaller or equal weight than path X. Due to the buffer constraint, for all future time slots, the best full path containing X must have a bandwidth allocation that is at least the bandwidth allocation of the best full path containing path Y. Therefore, it cannot have a lower weight than the best full path containing Y. Second, if $k_x \neq k_y$ then for any $k \in \{0, \ldots, K-1\}$ such that a branch from x to a node $x' = (i+1, k, b_{x'}, w_{x'})$ exists, there exists a branch from y to a node $y' = (i+1, k, b_{y'}, w_{y'})$ such that $b_{y'} \leq b_{x'}$, as the service rate in interval $i+1$ is the same and by assumption, $b_y \leq b_x$. As the difference in cost of the branch connecting y to y' and the branch connecting x to x' cannot be larger than ϕ, the first part of the proof applies to x' and y'. ■

In the sequel, we will say that node y *prunes* node x if (4.4) is true. Assuming that path pruning (i.e. deleting non-optimal paths) according to Lemma 1 is done at each time step (including time step i), call *plausible nodes* those nodes that lie on a path. The remainder of this section discusses how to make the optimization algorithm more efficient, on one hand by reducing the number of comparisons necessary to do pruning (cf. Lemma 2 and 3 below), on the other hand by establishing a criterion that allows us to reduce the number of branches that have to be considered from certain nodes (cf. Theorem 1).

Lemma 2 *For any non-plausible node $x = (i, k_x, b_x, w_x)$, there exists at least one plausible node $y = (i, k_y, b_y, w_y)$ that prunes x.*

Proof: First, observe the transitivity of the *prunes*-relation: if a prunes b and b prunes c, then a prunes c. This follows readily from lemma 1 by considering the two cases $k_a = k_c$ and $k_a \neq k_c$.

Now, because x is non-plausible, there must be at least one node y that prunes x. If y is plausible, then the assertion is true. If it is not plausible, then there must exist another node z that prunes y, and therefore also prunes x. We can repeat the previous argument for z. As the set of nodes is finite, we must necessarily find a plausible node pruning x. This completes the proof. ■

[3] *Note that this allows us to do more than the "standard Viterbi" pruning, i.e. among paths terminating in a common node, keep only the one with the lowest weight. We can also prune across nodes.*

In other words, once we know a node is not plausible, we do not need to attempt to prune other nodes with it. In particular, this means that for a given level k, we can immediately discard nodes that are pruned by some other node on the same level. If we only look at nodes at a given level (without considering pruning between nodes of different levels), we only keep nodes $x = (i, k, b_x, w_x)$ such that for any other node $y = (i, k, b_y, w_y)$, we have $b_x < b_y$ and $w_x > w_y$, or vice versa. Other nodes are necessarily non-plausible. Note that $x < y \stackrel{\text{def}}{=} w_x < w_y$ is a total ordering on the nodes not pruned by a node of the same level.

Next, we want to show that if we organize the nodes at time step i and a level k in a chain (ordered set) based on the previous observation, then we can directly derive a new chain for time step $i+1$ and some level k' and be sure that all the nodes in this new chain are "locally" plausible, i.e. no node in this chain can prune another node in the chain.

Definition 1 *Let $f(x,k)$ for a plausible node $x = (i, k_x, b_x, w_x)$ and a rate level $k \in \{0, \ldots, K-1\}$ be the node $(i+1, k, b, w)$ connected to node x through a branch, if it exists. Otherwise, the function is undefined.*

Note that the function is undefined if serving at rate c_k during interval $i+1$ leads to a buffer overflow.

Lemma 3 *Let $x = (i, k_0, b_x, w_x)$ and $y = (i, k_0, b_y, w_y)$ two plausible nodes at a same level k_0, such that $x < y$. Assuming that $u = f(x,k)$ and $v = f(y,k)$ both exist for some $k \in \{0, \ldots, K-1\}$, then $x < y$ implies $u < v$, provided $b_u > 0$.*

Proof: From (4.1) we have $w_u = w_x + \gamma \cdot s_k$ and $w_v = w_y + \gamma \cdot s_k$. Therefore, $w_u < w_v$. From (4.3) we have $b_u = \max(b_x + \alpha, 0)$ and $b_v = \max(b_y + \alpha, 0)$, with $\alpha = r_{i+1} - c_k$. As $b_x > b_y$, we have either $b_u > b_v \geq 0$, and therefore $u < v$, or $b_u = b_v = 0$, in which case u prunes v. ■

The following theorem will allow us to determine for certain nodes that the only plausible nodes they can be connected to at the next time step are on the same level. In other words, the only path we have to consider through such a node continues on the same level. For this purpose, let us first define the *set of breakpoints* of a node.

Definition 2 *For a plausible node x, the* set of breakpoints *$B(x)$ contains all the nodes (i, k, b, w) on the (unique) path leading to x such that the node $(i-1, k', w', b')$ on the same path has $k' \neq k$.*

Colloquially, the set of breakpoints of a node x is the set of nodes on the path leading to x at which a new service rate comes into effect.

Theorem 1 *Assume a plausible node $x = (i, k_x, w_x, b_x)$, and let $x_B = (i_B, k_x, w_B, b_B)$ be the element of $B(x)$ with largest time index i. Call y the node that has $B(y) = B(x) \setminus x_B$. In other words, $y = (i, k_y = k_B, w_y, b_y)$ has the same breakpoints as x except the last one. If y exists, then there is no plausible node connected to x at time step $i+1$ except at the level k_x if either $c_{k_x} < c_{k_y}$ and $\Delta < \Phi$ or $c_{k_x} > c_{k_y}$ and $b_x = b_y = 0$, with*

$$\Delta = \gamma(i - i_B + 1)(c_{k_x} - c_{k_y}) \tag{4.5}$$

Proof: We prove this theorem by comparing the partial weight of a path through x with the partial weight of a path through y to a same node $a = (i+1, k_a, w_a, b_a)$.

In the case where $c_{k_x} < c_{k_y}$, note that the buffer occupancy b_a is smaller for the path going through y than for the one through x. As for the weights, the path through x has one more renegotiation of cost Φ than y, but a bandwidth cost that is smaller by Δ. Therefore, as long as $\Delta < \Phi$, the path through x cannot have lower weight than the path through y, except if $k_a = k_x$. By virtue of lemma 1, the path through x cannot be optimal if $k_a \neq k_x$.

In the case where $c_{k_x} > c_{k_y}$, note that the weight of path x to reach a is always higher than for path y if $k_a \neq k_x$. On the other hand, the buffer occupancy b_a through y is either larger than the one through x (they had the same buffer occupancy at the end of time step $i_B - 1$; from then on, the service rate for path x was always higher), or both are zero. If both are zero, then lemma 1 again states that path x cannot be optimal. ■

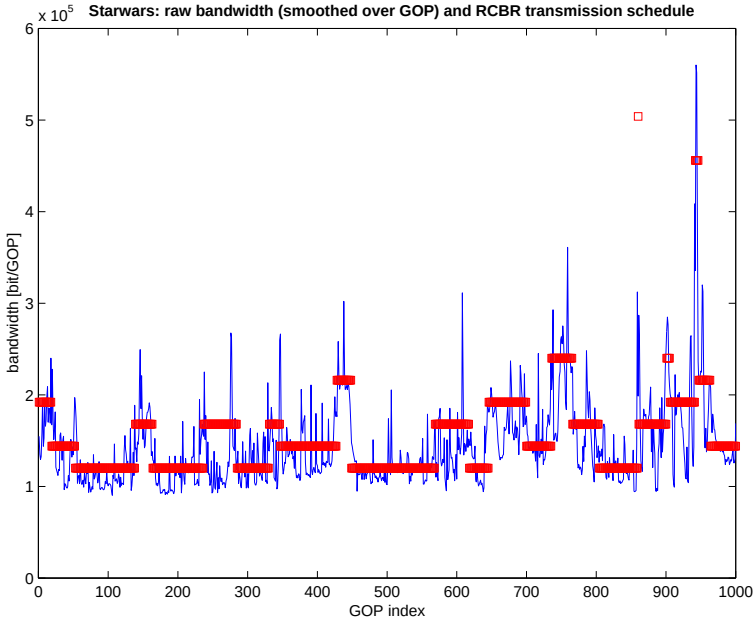

Figure 4.2: An example of the optimal segmentation algorithm.

Instead of the buffer bound (4.2), it is also possible to enforce a delay bound. This might be desirable in real-time applications, if sufficient buffer space is available, but the Quality of Service still requires delays to be kept low. The condition for all data entering during time slot $i - D$ to have left at the end of time slot i is

$$b_{i-D} \leq \sum_{j=-D+1}^{0} s_{i+j} \quad i = D, \ldots, N-1 \tag{4.6}$$

The runtime complexity of the optimization algorithm very much depends on the cost ratio ϕ/γ, the buffer size B and above all the number of bandwidth levels K. Also, the user rate function $\{r_i\}$ has has an impact on how many candidate paths remain valid at each time slot. We have found that if we restrict K to about 20, optimizations can be done in reasonable time, even for long traces like the Star Wars movie (approx. 174000

samples) [37]. For larger K, e.g. 100, it quickly becomes impracticable, because of an explosion in the number of paths that have to be considered. For example, with $K = 20$ (with the bandwidth levels chosen uniformly within $c_0 = 48$kbps and $c_{K-1} = 2.4$Mbps), the computation took about 20 minutes on an Sun UltraSparc 1, while with $K = 100$, the computation took more than a day.

We call *bandwidth efficiency* the ratio of the original stream's average rate to the average of the piecewise constant service rate, i.e.

$$\frac{\sum_{i=0}^{N-1} r_i}{\sum_{i=0}^{N-1} s_i}.$$

The graph "OPT" in Figure 4.3 shows the mean renegotiation interval and the bandwidth efficiency for various choices of the cost ratio ϕ/γ, for a buffer size $B = 300$kbit, which represents a buffering delay of slightly less than 1 second (recall that the average rate of the trace is 374 kbps). It is clear from Fig. 4.3 that there exists a tradeoff between bandwidth efficiency and renegotiation frequency. This tradeoff depends on the cost ratio ϕ/γ: raising the price for renegotiation results in a lower renegotiation frequency, but also in a lower bandwidth efficiency, and vice versa. The network operator can announce these prices to the user, and the user optimizes his network usage accordingly. Note how close the bandwidth efficiency gets to one with very reasonable renegotiation frequencies; for example, with one renegotiation every 7 seconds, we achieve over 99% of bandwidth efficiency. This is a clear manifestation of the slow time-scale behavior of compressed video streams.

4.4.2 Causal Renegotiation Schedule

For interactive (online) sources, the optimization algorithm described above cannot be used to determine optimal renegotiation points. For such sources, causal heuristics have to be used to make decisions about requesting new rates. Such heuristics predict the future bandwidth requirement based on some statistics collected in the past. The goal of this section is to show that heuristics resulting in satisfactory performance do indeed exist, although their derivation is somewhat *ad hoc*.

The heuristic we present is based on a AR(1) bandwidth estimator and on buffer thresholds. Three parameters have to be tuned: a high and a low buffer threshold B_h and B_l, respectively, and a time constant T, which should reflect the long-term rate of change of the rate function. The rate predictor we have used is

$$\widehat{r}_{i+1} = (1 - T^{-1})\widehat{r}_i + T^{-1}(r_i + \max\{b_i - B_h, 0\}) \tag{4.7}$$

where r_i is the actual incoming rate during slot i, and b_i is the buffer size at the end of slot i. The additional term $T^{-1} \cdot \max\{b_i - B_h, 0\}$ in the estimator adds the bandwidth necessary to flush the current buffer content within T. This is necessary to have a sufficiently fast reaction to sudden large buffer buildups.

The algorithm is very simple. Let

$$s_{new} = \left\lceil \frac{\widehat{r}_{i+1}}{\Delta} \right\rceil \Delta \tag{4.8}$$

with Δ the bandwidth allocation granularity. A new bandwidth s_{new} is then requested if

$$(b_i > B_h \text{ and } s_{new} > s) \text{ or } (b_i < B_l \text{ and } s_{new} < s) \tag{4.9}$$

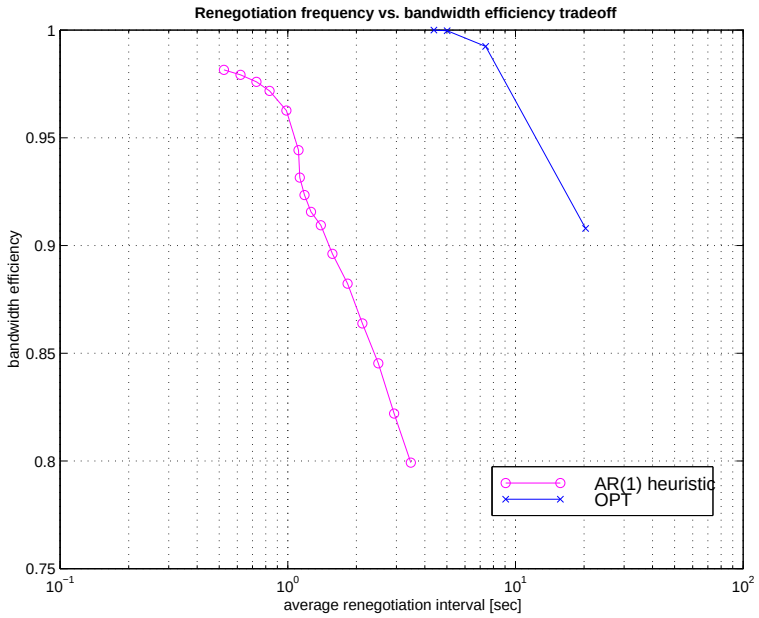

Figure 4.3: The tradeoff between bandwidth efficiency and renegotiation frequency for the AR(1)-based heuristic, compared to the optimum, for the "Star Wars" trace. The parameters for the AR(1) heuristic are: $B_l = 10$kbit, $B_h = 150$kbit, $T = 5$ frames, and Δ is varied from 25kbps (left) to 400kbps (right). In this example, the buffer occupancy never exceeds $B = 300$kbit.

It can be seen in Fig. 4.3 that using the heuristic, we need about one renegotiation a second to achieve 95% of bandwidth efficiency (with $B_l = 10$kbit, $B_h = 150$kbit, $T = 5$ frames, and $\Delta = 150$kbps). Although this is considerably less than what can be achieved with the optimal allocation, it still represents a relatively small load on the signaling system. However, this gap suggests a potential for better heuristics, and we hope to address this problem in future research. For example, the prediction quality could be improved by taking into account the inherent frame structure of MPEG encoded video.

4.5 Performance of RCBR

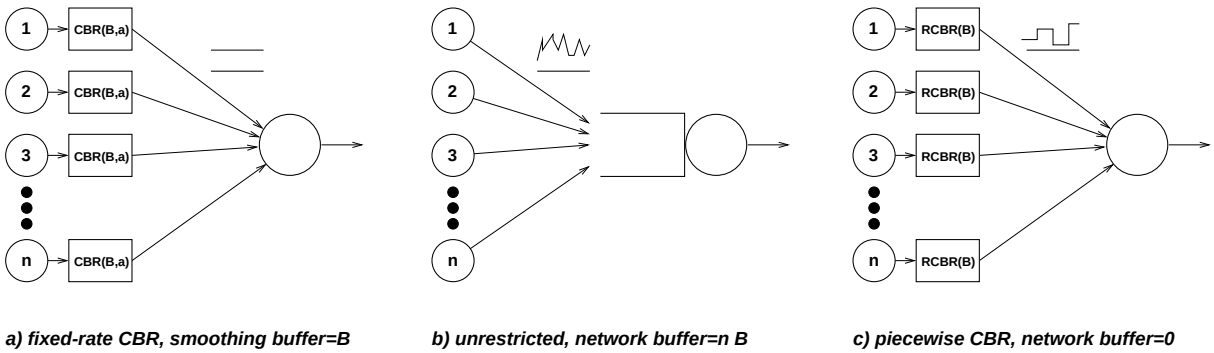

Figure 4.4: The three scenarios to assess statistical multiplexing gain (SMG) of our proposed service.

In this section, we examine the statistical multiplexing gain (SMG) achievable using the RCBR scheme, by means of both a theoretical analysis of a multiple time-scale source model as well as simulation experiments on real traffic. More specifically, we compare the SMG of RCBR with that of two other scenarios (cf. Fig. 4.4). The first scenario (a) represents traditional CBR service, with a smoothing buffer of size B at the network entry and a fixed CBR rate $\bar{c}$ for each source. Here, there is no multiplexing between traffic of different sources. The second scenario (b) multiplexes n streams without any restriction on a server with rate c and buffer size nB. This gives the maximum achievable SMG for the given sources. The third scenario (c) represents the RCBR approach. Each source is smoothed by a dedicated buffer of size B and transformed into a stepwise CBR stream, which is then transported without further buffering in the network (except some cell level buffering). The total service rate is c and the total amount of buffering is fixed at nB in all three scenarios. While the theoretical analysis gives insights as to the nature of the SMG captured by the RCBR, the experimental results quantify the amount of gain for video traffic.

4.5.1 Analysis of a Multiple Time-Scale Model

We consider the following discrete-time traffic model for an individual video source. Let X_t be the amount of data (measured in bits, bytes, cells etc.) generated per time-slot (duration of a frame,etc.). The process $\{X_t\}$ is *modulated* by an irreducible finite state Markov chain such that the value of X_t is a function of the current state. The Markov structure models the correlation in the data generation rate over time. The state space $\mathcal{S}$ is decomposed into a union of disjoint subsets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_K$; each $\mathcal{S}_k$ can be interpreted as the state space of a *fast time-scale sub-chain*. The dynamics within each sub-chain model fast time-scale behavior (such as correlations between adjacent frames). Transitions between various sub-chains, on the other hand, happen very rarely compared with the transitions inside each sub-chain; these transitions model the slow time-scale dynamics of the traffic stream (such as scene change). Let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be the probabilities of these rare transitions; these are very small parameters. Thus, the source would typically spend a long time in a sub-chain, and then occasionally jump to a different sub-chain. In the analysis below, we are interested in the regime when the buffer time-scale is large enough to smooth out the fast time-scale fluctuations of the traffic, but is small compared to the slow transition time-scale.

This multiple time-scale Markov-modulated model has been used in several video traffic studies [101, 81]. The sustained peak observed by several researchers corresponds to remaining in a high-rate sub-chain for a long time in this multiple time-scale model. See Figure 4.5 for an example of a source with three sub-chains.

We shall now characterize the resource requirements for multiple time-scale sources under the three scenarios in Fig. 4.4, for given loss probability requirements. Consider first scenario (a), when each individual stream is smoothed by a buffer B and allocated a fixed CBR rate of $\bar{c}$. The minimum drain rate $\bar{c}$ required to achieve a target QoS buffer overflow probability p_{qos} is known as the *equivalent bandwidth* $e(p_{qos})$ of the source, and for *single time-scale* Markov sources, this has been explicitly computed in terms of the statistics of the source [39, 32, 67]. This equivalent bandwidth is based on a large deviations estimate of the buffer overflow probability, in the regime of large buffer size B. It can be shown that the equivalent bandwidth is between the mean and peak rates of the stream, and it measures the amount of smoothing of the stream by buffering. A large buffer B in this context means that the buffer is sufficiently large to smooth out the fluctuation of the

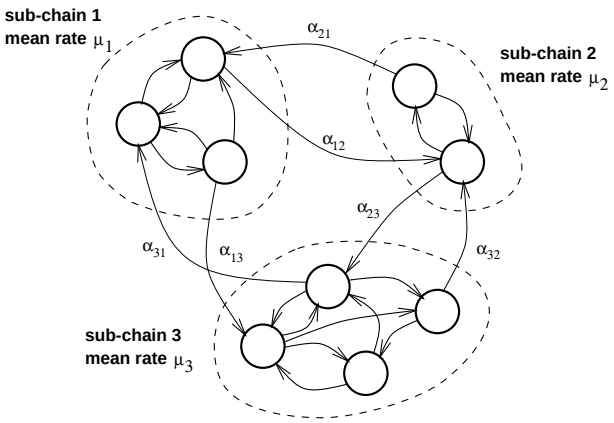

Figure 4.5: A multiple time-scale source with 3 sub-chains.

traffic stream.

Analogous results have been obtained for multiple time-scale Markov traffic [108]. For multiple time-scale sources, one now has to look at the *joint* asymptotic regime when simultaneously the rare transition probabilities α_i's are close to zero and the buffer size B is large enough to absorb the fast time-scale fluctuations of the stream. It is shown in [108] that the equivalent bandwidth $e(p_{qos})$ of the multiple time-scale stream is given by

$$e(p_{qos}) = \max_{1 \leq k \leq K} e_k(p_{qos}), \tag{4.10}$$

where $e_k(p_{qos})$ is the equivalent bandwidth of the kth fast time-scale sub-chain when considered in isolation. The intuition is that buffer overflows are due mainly to the effects of the most bursty sub-chain, and thus the drain rate needed for the entire stream is the drain rate of that particular sub-chain. In particular, the drain rate needed is greater than $\hat{\mu}$, the maximum of the average rates of the sub-chains. This implies that the gain due to buffering alone is rather limited for multiple time-scale traffic, as the CBR rate needed for the stream is determined by the worst-case sub-chain. The theoretical result also makes precise the intuition we presented in Section 4.2, that static traffic descriptors (in this case, the CBR rate) leads to a wasteful allocation of resources for multiple time-scale traffic.

To get significant multiplexing gain beyond that obtained by buffer smoothing, the limitation imposed by the slow time-scale dynamics can be overcome by multiplexing many independent streams. By a law of large number effect, the probability that many streams are simultaneously in a bursty sub-chain is small, so that a small loss probability can be guaranteed even if the capacity allocated per stream is less than $\hat{\mu}$. This is shown in scenario (b) in Fig. 4.4, where n independent and statistically identical streams are multiplexed. If we scale the total link rate $c := n\bar{c}$ and the total buffer as nB (i.e. the link capacity and buffer space *per stream* is fixed in this scaling), an estimate of the buffer overflow probability, in the regime of large n, can be obtained in terms of *only* slow time-scale statistics of the individual stream (with the fast time-scale dynamics averaged out.) [108]. Specifically, consider a random variable which takes on the value μ_k with probability π_k, where π_k is the steady-state probability that the stream is in sub-chain k and μ_k is the mean rate of

sub-chain k. Let L be the log moment generating function of this random variable:

$$L(r) \equiv \log \sum_{k=1}^{K} \pi_k \exp(\mu_k r).$$

and define L^* by:

$$L^*(\mu) = \max_{r>0}[\mu r - L(r)],$$

the *Legendre* transform of L. Then the asymptotic estimate of the loss probability when there are many sources and the buffering B per source large is given by:

$$p \approx \exp(-L^*(\bar{c}) \cdot n) \tag{4.11}$$

Note that (4.11) is simply the Chernoff's estimate of the probability that the streams are in a combination of sub-chains whose total mean rate exceeds the channel capacity [116, 58]. Note that this estimate does not depend on the fast time-scale statistics of the streams nor on the specific value of buffer size B, provided that it is large enough to absorb the fast time-scale variations of the streams. This result can be interpreted as a decomposition of the gain from multiplexing large number of multiple time-scale streams in a *buffered* node into two components. The first component is the gain obtained from *buffering*: its effect is essentially to remove the time-scale fluctuations of the sources. The second component is the gain from *averaging* between sources: it only depends on the slow time-scale statistics, and is the same as that obtained in a *bufferless* system with the fast time-scale fluctuations removed from the traffic. At the slow time-scale, the buffer is too small to have any significant effect. Note also that for a target overflow probability p_{qos}, the total link rate needed can be computed from eqn. (4.11).

Finally, we consider the RCBR scenario, (c) in Fig. 4.4, where the multiplexing node is bufferless and users have dedicated buffer. We characterize how much of the multiplexing gain in the shared buffer case (scenario (b)) our proposed scheme can capture. Assume that the scheme does an ideal job in separating the slow and fast time-scales, such that it renegotiates a new CBR rate whenever the source jumps from a fast time-scale sub-chain to another. For a buffer overflow probability requirement p_{qos}, the new CBR rate it should renegotiate for is the equivalent bandwidth $e_k(p_{qos})$ of the sub-chain k the source enters. Since π_k is the steady-state probability that the stream is in sub-chain k, the stream will demand a CBR rate of $e_k(p_{qos})$ for π_k long-term fraction of time. The probability of renegotiation failure is roughly the probability that the total CBR bandwidth demand exceeds the available capacity; for large n, we can use Chernoff's estimate to approximate this as

$$\exp(-L_e^*(\bar{c}) \cdot n), \tag{4.12}$$

where

$$L_e(r) = \log \sum_{k=1}^{K} \pi_k \exp(e_k(p_{qos}, B) \cdot r) \qquad L_e^*(\mu) = \max_{r>0}[\mu r - L_e(r)]$$

and $\bar{c}$ is the link capacity per source. Comparing this to the loss probability (4.11) when there is a shared buffer of size nB, we see that this renegotiation failure probability is larger since the equivalent bandwidth $e_k(p_{qos})$ of every sub-chain is greater than its mean rate μ_k. Viewed in another way, the capacity per stream needed for the same level of performance is greater in the RCBR scheme. This discrepancy in bandwidth requirement is due to the fact that the RCBR scheme does not take advantage of a large shared buffer to effectively absorb all fast time-scale variations through statistical multiplexing. Thus, out

of the two components of the SMG in the shared buffer case, RCBR extracts the component obtained from averaging between sources. Our scheme essentially focuses on the gain in the averaging of the slow time-scale dynamics rather than the smoothing of the fast time-scale dynamics. However, for sources with small fast time-scale fluctuations superimposed on larger slow time-scale variations, the equivalent bandwidths of the sub-chains will be close to the mean-rates for reasonably sized buffers, and the discrepancy will be small. This is further substantiated by the experimental results presented next.

4.5.2 Experimental results

We shall now present simulations results comparing the performance in the three scenarios in Fig. 4.4. The stream we have used is the MPEG-1 encoded trace of the Star Wars movie [37]. The n sources are randomly shifted versions of this trace. The buffer size B was chosen as 300kbit, slightly more than the maximum size of three consecutive frames in the trace. This approximately corresponds to the buffering of current video codecs. The renegotiation schedule used in the experiments is computed using the off-line optimization algorithm described in Section 4.4.1, with a bandwidth granularity of 1 kbps, which results in one renegotiation every 12 seconds on average.

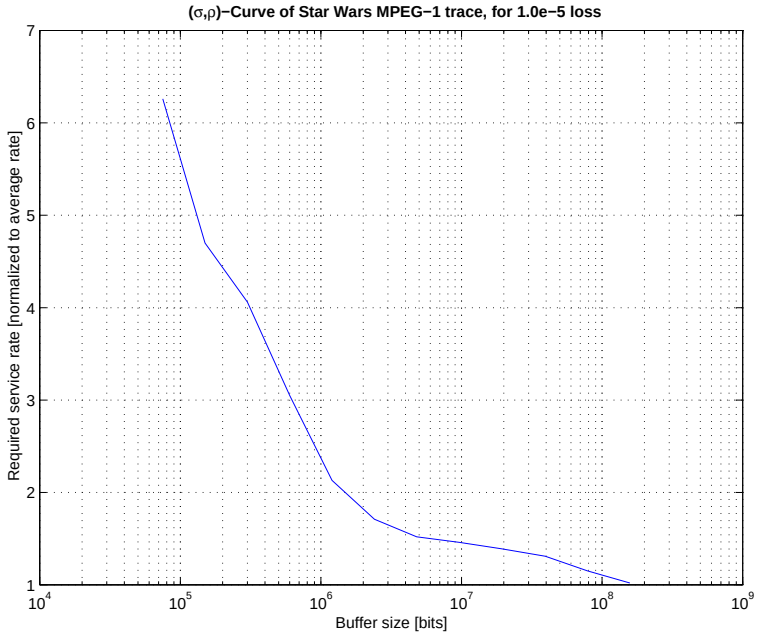

Figure 4.6: The (σ, ρ)-curve of the video trace for 10^{-5} loss.

To assess the SMG for all three scenarios, we have determined the channel service rate per stream c/n, as a function of n, needed to guarantee a desired bit loss probability. In scenario (a) and (b), bits are lost due to buffer overflow. In scenario (c), bits are lost due to failure in renegotiating for a higher CBR rate (in which case we assume the source has to temporarily settle for whatever bandwidth remaining in the link until more bandwidth becomes available). Determining c is straightforward for scenario (b). For scenarios (a) and (c), we find for each n the *minimum* c that guarantees the desired loss probability: for

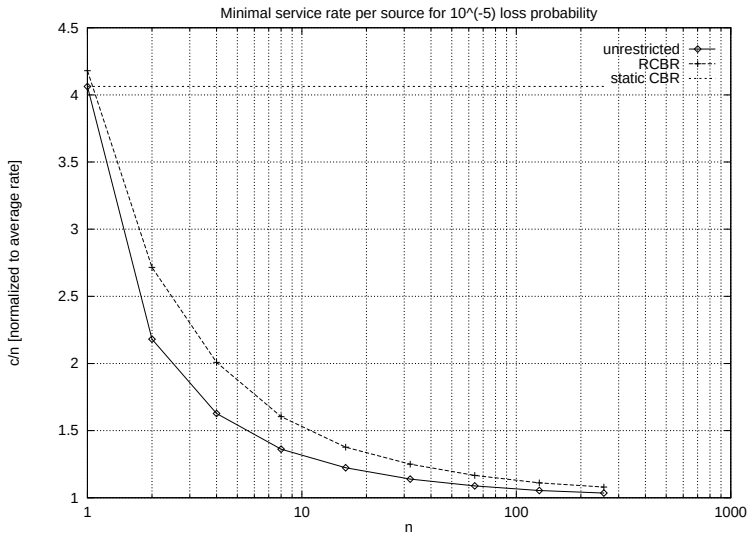

Figure 4.7: Statistical multiplexing gain (SMG) achievable for 10^{-5} loss probability.

each n, we do a binary search on c; for each step in the search, we do many simulations, where each simulation has a randomized phasing of the sources, and compute the average fraction of bits lost as an estimate of the loss probability. At each step, we repeat the simulations until the sample standard deviation of the estimate is less than 20% of the estimate. Results for 10^{-5} loss probability requirement are depicted in Fig. 4.7.

In the CBR case (a), the bandwidth per stream is $\bar{c}$, of course, regardless of the number of streams n. Note that $\bar{c}$ can be determined from the corresponding (σ, ρ) curve of this trace in Fig. 4.6 (For a given buffer size σ, this curve gives the minimum service rate ρ such that the fraction of bits lost is less than 10^{-5}.) As has been previously observed in the literature, this is close to the peak rate [92, 38]. For the given buffer size and loss ratio, $\bar{c}$ is 4.06 times the trace's average rate of 374kbps.

Our scheme achieves slightly less SMG than the unrestricted case because buffers are not shared and the fast time-scale multiplexing gain is not exploited, as explained in the theoretical analysis. Nevertheless, we are able to extract most of the SMG, especially for a large number of multiplexed streams. For example, for $n = 100$ streams, we require less than a third of the bandwidth of the static CBR approach, and less than 10% more than in the unrestricted case. Recall that the gain that offsets the slightly higher bandwidth requirement of RCBR is the simplicity of mechanism in RCBR. Asymptotically, the value for c/n for the stepwise CBR function approaches the inverse of the bandwidth efficiency obtained in the optimization algorithm.

4.6 Related Work

The key contributions in this chapter are: (a) to note that compressed video traffic has significant burstiness in the slow time-scale, (b) showing that renegotiation allows us to extract almost all the SMG available from exploiting this variation, and (c) admission control for loosely-constrained traffic sources. Recently Chong et al [15] and Zhang and

Knightly [120] have independently published work that comes to the same conclusions. Zhang and Knightly present a renegotiated VBR service. Chong et al have concentrated on the online prediction problem using artificial neural networks. Our work differs from theirs in some important aspects. First, our work is based on the theoretical foundation of large deviation analysis of multiple time-scale sources, which gives us deeper insight into the nature of the multiplexing gain and allows us to formally study the renegotiation failure probability for ensembles of renegotiating sources, which is asymptotically correct in the regime stated in the theorems. In contrast, Chong et al [15] based their analysis solely on the power spectral density of the traffic and moreover does not consider the statistical multiplexing issues. As pointed out by Hajek and He [52], second-order statistics alone do not uniquely specify loss probabilities and thus it is important to understand the regime in which these approximations are valid. Second, we have obtained an optimal off-line renegotiation algorithm. Third, we have considered admission control for renegotiating sources. Finally, we have considered the system aspects of the problem in more detail. Nevertheless, we feel that their work complements ours, in that it reinforces the importance of renegotiation for multiple time-scale sources.

The two core mechanisms for RCBR are renegotiation and rate prediction. In-call renegotiation has been proposed for bursty data traffic by Hui [58], Turner [110], Doshi and Dravida [27], and Boyer and Tranchier [12]. In their work, a traffic source sets up a burst level reservation before sending, or in some cases during a burst. However, since data traffic bursts can occur every tens of milliseconds the reservation process has to be fast. This speed is not essential for RCBR, where renegotiations happen once every tens of seconds. In addition, we believe that renegotiation is effective mainly as a mechanism to extract SMG from slow time-scale variations in source traffic. Data traffic exhibits burstiness in the fast time-scale, and thus renegotiation for data traffic is not likely to be economical in practice. Nevertheless, the mechanisms for renegotiation proposed in the literature can be used for RCBR with minor changes.

De Veciana and Walrand have proposed a *periodic averaging of rate* scheme to smooth traffic at the network edge [25]. Like RCBR, the output of their traffic shaper is also a piecewise CBR stream. The basic difference, however, is that they do not model the multiple time-scale nature of the traffic stream, and their scheme is not designed to capture the SMG from multiplexing many sources with slow time-scale dynamics.

The off-line schedule computation problem has also been addressed in Salehi et al's recent work [100]. They propose to use a client buffer and *work-ahead* smoothing, i.e., sending data ahead of schedule, in order to achieve an additional reduction in the flow's bandwidth fluctuation. They present an *optimal smoothing* algorithm that transforms an arbitrary data stream into a piecewise-CBR stream that minimizes both the peak rate and the rate variance, and show that this approach allows to considerably reduce the renegotiation frequency under RCBR service. Their work provides an interesting alternative for computing an optimal renegotiation schedule. Rexford et al [94] study the smoothing problem under the assumption that only limited knowledge about the future frame sizes is available. McManus and Ross discuss heuristics for the same problem setting [79]. They show the condition on the bandwidth b, the number of prefetched frames d, and the client-side memory size B, under which a sequence of VBR video can be transmitted at a constant rate without overflowing or underflowing the client buffer. Based on these conditions, heuristics are developed that yield a piecewise constant-rate transmission schedule. The client-side memory size and the number of prefetched frames are shown to decrease with the number of intervals, i.e., the number of renegotiations.

The on-line rate prediction problem has been extensively studied from several different perspectives in the past. Adas has addressed it using adaptive linear filtering [3], and he reports good prediction performance over a range of compressed video sequences. Other promising methods are described in [56]. Chong et al [15] have proposed an Artificial Neural Network based approach for prediction, and have shown that it compares well with more traditional alternatives. Reininger et al [93] have investigated methods to renegotiate Variable Bit Rate (VBR) parameters, including the peak rate, the sustained rate, and the burst size. The focus of their work is on the on-line prediction problem. A drawback of their scheme is the large number of parameters to be tuned (sliding window size $\mathrm{UPC}_{\mathrm{window}}$, aggressiveness factor k, four buffer thresholds D_1, D_2, I_1 and I_2, target quantization Q_{target}, and renegotiation delay parameters S and L).

Current proposals in the ATM Forum for dealing with ABR traffic are similar in spirit to RCBR in that a source obtains a stepwise-CBR rate allocation from the network. However, in the ABR framework, there is an assumption that the source has an intrinsically infinite data rate that is modulated by the fair share of the available network capacity. Thus, the data rate from a source is dynamically adapted to the available capacity in the network. This is the opposite of our situation, where the source has an intrinsic data rate that the network tries to accommodate. In other words, in the ABR case the rate information flows from the network to user; in the RCBR case, the information flows from the user to the network.

4.7 Discussion and Conclusions

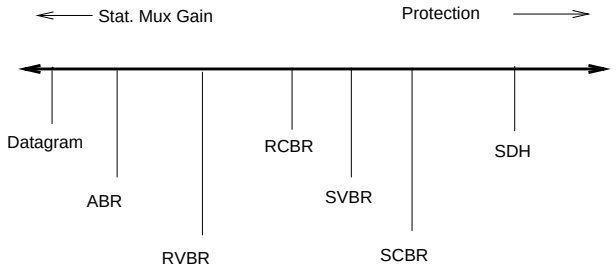

Figure 4.8: Design space for traffic management policies.

We believe that the performance tradeoff space for traffic management policies looks something like Figure 4.8. Starting from the right and moving to the left, we have the Synchronous Digital Hierarchy (SDH) for telephony, Static CBR, Static VBR, Renegotiated CBR, Renegotiated VBR (RVBR), ABR, Controlled Load, and finally, unrestricted datagram service. In SDH, each call is associated with a time slot, and thus a corresponding bandwidth, that cannot be shared with any other call. Static CBR and static VBR are described in Section 4.2 and have one-shot traffic descriptors. RCBR and its corresponding service, RVBR, add renegotiation to CBR and VBR respectively. ABR service in ATM networks guarantees a connection zero loss, but its service rate changes as a function of other traffic in the network. With this service, the network agrees to perform admission control such that a source's performance does not substantially degrade, but the degree of degradation is not quantified. Finally, "datagram" refers to unrestricted sharing of all

network resources.

As we move from right to left, the statistical multiplexing gain (SMG) achievable increases, but if the network resources allocated to a stream are kept the same, the protection between streams decreases. That is, one stream can more adversely affect another's performance in terms of its service rate and loss rate. For example, as one moves from Static CBR to Static VBR, more SMG is possible, but there is a greater loss of protection. This is because with a fixed amount of buffering a VBR source could experience packet loss due to a coincident burst from another source. Note that in moving from Static CBR to Static VBR similar protection can be obtained, but only at the cost of increased buffering, or by describing source traffic with more parameters [71]. Similarly, as we move from Static VBR to RCBR, we incur renegotiation overheads, but can potentially exploit slow time-scale variations in the source rate to get increased SMG. RVBR allows more SMG, since both slow and fast time-scale variations are exploited. However, there is more overhead for renegotiation, per-stream regulation, and larger buffers at each switch. The next step along the spectrum is to ABR, where there is much less protection between streams, because each user's bandwidth depends on the demand of the others. However, even more SMG is possible, since SMG is extracted at the burst level. Controlled Load service offers potentially even more SMG than ABR service, but at the expense of a non-zero loss rate. Finally, with datagram service, the most SMG is available, since call level, burst level, and cell level statistical multiplexing is possible. Unfortunately, datagram service also has the least protection - a single burst from a malicious or ill-behaved source can affect all the others.

The point is that RCBR is not a panacea. It is one choice in a spectrum of choices for carrying compressed video. We feel that RCBR is best suited to traffic whose variation is not confined to the fast time-scale. This seems to match at least the subset of the compressed video traffic workload that has been measured in the literature. Other services could also be used to carry compressed video traffic: ABR, Static VBR, RVBR, and Static CBR have all been proposed in the literature. Ultimately, a network provider and user must choose a service based on their relative costs, efficiencies and afforded qualities of service.

Nevertheless, we feel that RCBR has some clear benefits. First, it is relatively easy to implement, since we are adding a renegotiation component to the well-understood Static CBR service. While RCBR admission control is potentially complex, this is more than balanced by the fact that *neither complex scheduling disciplines nor large buffers are required in the network switches* [45]. RCBR allows us to keep the network core fast, cheap and dumb (at least in the data path), and put intelligence in the edges to extract the SMG from slow time-scale variations.

Second, an RCBR network is always stable, in the sense that the sum of arrival rates to a multiplexing point is always smaller than the corresponding service rate. Each admitted call or burst moves the system from a stable configuration to another stable configuration. Thus, the network operator can easily guarantee zero loss and small queueing delays within the network.

We have already shown that RCBR gets more SMG than a static service. There is another significant advantage. Users of a static service get only one chance to provide the network with a traffic descriptor. If they guess wrong, they either get poor SMG, or suffer from large delays, which might be unacceptable. With RCBR, a source has the option to modify its traffic descriptor over time. The danger is that the network might admit too many ill-described users, so that at some future time, the renegotiation failure rate may be too high. This is because there really is no free lunch. If a user is admitted into a network

before its traffic is characterized, then there is always the possibility that mistakes will be made by admitting too many users. However, Section 5 indicates that we might be able exploit the law of large numbers to make this risk acceptably small.

It is instructive to compare RCBR with unrestricted sharing. With unrestricted sharing ("datagram service") we achieve the maximum SMG, but the least protection. In practical terms, with unrestricted sharing, a source must always be prepared to deal with data loss (for example, by using forward error correction or retransmitting data). Moreover, data loss is unpredictable. The analogue to loss in RCBR is renegotiation failure. With RCBR, however, a source retains its existing bandwidth even if renegotiation fails. Besides, a source is explicitly informed about renegotiation failure so that it can take corrective measures. This makes it easier to integrate RCBR with techniques such as dynamic requantization of stored video, adaptive coding and multilevel scalable coding.

To conclude, we have shown that a source with slow time-scale variations would suffer performance problems when carried over a static service. Large deviation analysis provides theoretical insight into this problem and motivates the design of the RCBR service. We have considered the system aspects of implementing RCBR and have carried out several experiments to measure its performance. The results in Section 4.5.2 show that RCBR obtains most of the slow time-scale SMG with a fairly small load on the signaling system. Further, it is possible to compute the optimal renegotiation schedule for a real traffic source in a reasonable amount of time. Finally, we have studied the call admission problem and come up with admission control tests based on a large deviation analysis. Thus, our analysis and experiments show that RCBR service is efficient and well suited for multiple time-scale traffic.

Chapter 5

Measurement-Based Admission Control

Notation

$\phi(x)$	probability density of a standard Gaussian RV
$Q(x)$	complementary cumulative distribution function of a standard Gaussian RV
$X_i(t)$	bandwidth of flow i at time t
μ, σ^2	flow bandwidth mean and variance
$\widehat{\mu}$, $\widehat{\sigma^2}$	estimators of flow bandwidth mean and variance
c	link capacity
n	normalized link capacity ($n = c/\mu$)
m^*	admissible number of flows under perfect knowledge
S_t	aggregate bandwidth
p_q	target overflow probability
α_q	$\alpha_q = Q^{-1}(p_q)$
p_f	overflow probability
M_t	estimated admissible number of flows at time t
Y_t	normalized estimated admissible number of flows at time t
N_t	actual number of flows in the system at time t
$\rho(t)$	correlation function of the flow bandwidth process
T_h	mean flow holding time
T_c	burst time-scale
T_m	estimation window time-scale
$\widetilde{T_h}$	critical time-scale ($\widetilde{T_h} = T_h/\sqrt{n}$)
β	$\beta = \frac{\mu}{\sigma\widetilde{T_h}}$
γ	burst/critical time-scale separation parameter ($\gamma = \frac{\widetilde{T_h}}{T_c}\frac{\sigma}{\mu}$)
$h(t)$	estimation window

The work in this chapter has appeared in [109, 49].

5.1 Introduction

Integrated-services networks are expected to carry a class of traffic that requires Quality of Service (QoS) guarantees. One of the main challenges consists in providing QoS to users

while efficiently sharing network resources through statistical multiplexing. The role of Admission Control (AC) is to limit the number of flows admitted into the network such that each individual flow obtains the desired QoS.

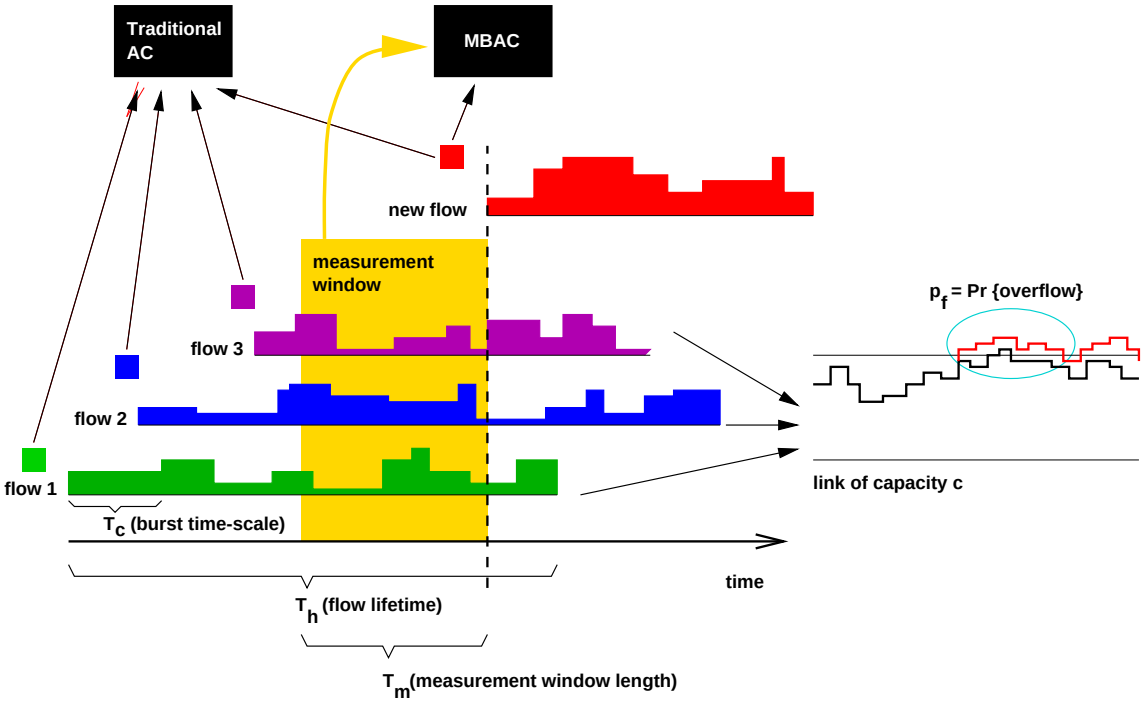

Figure 5.1: Traditional admission control and measurement-based admission control.

Traditional approaches to admission control require an *a priori* traffic specification in terms of the parameters of a deterministic or stochastic model. The admission decision is then based on the specifications of the existing and the new flow. However, it is usually difficult for the user to tightly characterize his traffic in advance [92]. For many types of traffic, such as variable-bit-rate (VBR) compressed video, it is very difficult to define adequate traffic descriptors that take into account the widely varying source characteristics occurring at a slow time-scale (such as due to scene changes). This is true even for stored media such as video-on-demand, as the user is expected to be able to exercise interactive control (such as pause, fast-forward etc.) Traffic descriptors such as the leaky bucket that has been proposed in standard bodies like the ATM Forum is only adequate for describing the short time-scale burstiness of the variable-rate traffic. As a result, traffic specifications can be expected to be quite loose, resulting in conservative use of resources.

Stochastic models such as those based on effective bandwidth [58] are better suited to achieve good statistical multiplexing gain. However, they suffer from two problems. First, it is difficult for the user to come up with the model parameters *a priori*. If he overestimates his requirements, then resources will be wasted in the network. This reduces the network utilization. If he underestimates his requirements, then insufficient resources will be allocated to his flow. The user has to abort the flow or try to adapt to this situation, for example by increasing the degree of compression of a video flow, thereby lowering its perceived quality. Second, it is hard to police traffic according to a statistical model [58]. It is not clear how to ensure that a traffic flow correspond to the specified parameters,

without which admission control can easily be "fooled".

Measurement-based Admission Control (MBAC) avoids these problems by shifting the task of traffic specification from the user to the network. Instead of the user explicitly specifying his traffic, the network attempts to "learn" the statistics of existing flows by making on-line measurements. This approach has several important advantages. First, the user-specified traffic descriptor can be trivially simple (e.g. peak rate). Second, an overly conservative specification does not result in an overallocation of resources for the entire duration of the session. Third, when traffic from different flows are multiplexed, the QoS experienced depends often on their *aggregate* behavior, the statistics of which are easier to estimate than those of an individual flow. This is a consequence of the law of the large numbers. It is thus easier to predict aggregate behavior rather than the behavior of an individual flow.

Relying on measured quantities for admission control raises a number of issues that have to be understood in order to develop robust schemes.

- **Estimation error**. There is the possibility of making errors associated with any estimation procedure. In the context of MBAC, the estimation errors can translate into erroneous flow admission decisions. The effect of these decision errors has to be carefully studied, because they add another level of uncertainty to the system, the first level being the stochastic nature of the traffic itself. Assuming *certainty equivalence* up-front, i.e. assuming that the estimated parameters are the real parameters, is dangerous, as we simply ignore its impact on the quality of service.

- **Dynamics and separation of time-scale**. A MBAC is a dynamical system, with flow arrivals and departures, and parameter estimates that vary with time. Since the estimation process measures the in-flow burst statistics, while the admission decisions are made for each arriving flow, MBAC inherently links the flow and burst time-scale dynamics. Thus, the question of impact of flow arrivals and departures on QoS arises. Intuitively, each flow arrival carries the potential of making a wrong decision. We therefore expect a high flow arrival rate to have a negative effect on performance. On the other hand, the impact of a wrong flow admission decision on performance also depends on how long it takes until this error can be corrected - that is, on flow departure dynamics.

- **Memory**. The quality of the estimators can be improved by using more past information about the flows present in the system. However, memory in the estimation process adds another component to the dynamics of a MBAC. For example, it introduces more correlation between successive flow admission decisions. Moreover, using too large a memory window will reduce the adaptability of MBAC to non-stationarities in the statistics. A key issue is therefore to determine an appropriate memory window size to use. For this, a clear understanding of the impact of memory on both estimation errors and flow dynamics is necessary.

The goal of this work is to study the above issues - the impact of estimation error, of flow arrival and departure dynamics, and of measurement memory - in a unified framework. We wish to gain an understanding about how these aspects of a MBAC system interact. To do so, we consider a sequence of increasingly sophisticated models, adding one of the above issues at a time. This sequence culminates in the *continuous load model*, which allows us to derive analytical approximations, as well as an intuitive understanding, about how the above issues fit together. The ultimate goal is to shed insights on the design of robust

MBAC schemes which can provide the appropriate QoS to the user even in the presence of the additional uncertainty due to measurements.

The rest of the chapter is organized as follows. In Section 5.2, we describe the models that will be studied. The analysis of these models is explained in Section 5.3 and 5.4. In Section 5.5, we summarize the insights obtained and use them to study the problem of choosing the appropriate memory window size. We also report some simulation results on real and synthetic traffic. In 5.6, we discuss how our results relate to previous work in measurement-based admission control. We conclude the chapter in Section 5.7.

5.2 Models

We begin by briefly describing the basic model. The network resource considered is a bufferless single link with capacity c. Flows arrive over time and, if admitted, stay for a random time. The bandwidth requirements of a flow fluctuate over time while in the system. We assume that the statistics of the bandwidth fluctuations of each flow are identical, stationary and independent of each other, with a mean bandwidth requirement of μ and variance σ^2. An important system parameter is the normalized capacity $n := \frac{c}{\mu}$, which measures the system size in terms of the mean bandwidth of the flows. Resource overload occurs when the instantaneous aggregate bandwidth demand exceeds the link capacity, and the quality of service is measured by the steady-state overflow probability p_f.

To study the various issues outlined in the introduction, we will analyze three variations of this basic model of increasing complexity. In the first variation, an infinite burst of flows arrives at time 0 and admission control decisions are made then, based on the initial bandwidths of the flows. After time 0, no more flows will be accepted and moreover the flows already admitted will stay in the system forever. We call this the *impulsive load* model with *infinite flow holding time*. This model permits us to study the impact of the measurement errors on the number of admitted flows and on the overflow probability, without the need to worry about flow dynamics.

In the second variation, we consider a similar model with flows admitted only at time 0, but now the admitted flows have holding times exponentially distributed with mean T_h. Thus, they will gradually depart from the system. We call this the impulsive load model with *finite* flow holding time. This model allows us to study the impact of flow departures on the overflow probability.

The last variation is the *continuous load* model, where the full flow arrival and departure dynamics are considered. In this model, flows arrive continuously over time with effectively *infinite* arrival rate, i.e. there are always flows waiting to be admitted into the network. Once they are admitted, they stay for an exponentially distributed holding time with mean T_h. The motivation for this model is that a well-designed robust MBAC should work well even for very high flow arrival rates, to cater for times when there is a surge in user demand of the service. Thus, the continuous-load model provides the most stringent test for MBACs.

Several comments about the model are in order. First, we observe that the traffic model is a stationary one. In practice, one of the main reasons for using a measurement-based scheme is to adapt to non-stationarities in the statistics of the traffic, either due to the change in the nature of the flows or change in the statistics within a flow itself. The approach taken in this chapter is to use a stationary model to evaluate the performance of schemes with *limited memory*. Thus, the results are valid if the traffic statistics are stationary within the

memory time-scale. We view this as a first step towards a full understanding of adaptivity issues.

Second, we consider a resource model without buffers. There are several motivations for this. First, the dynamics leading to the overflow event in a bufferless system is much simpler than that of overflowing in a buffered system, as the event occurs whenever the instantaneous aggregate traffic load exceeds the link capacity. This simplification allows us to focus on the measurement problem that is of central interest in this chapter. Second, our work in Chapter 4 has indicated that a significant bulk of the statistical multiplexing gain can be obtained by a Renegotiated Constant Bit Rate (RCBR) service. In this service model, buffering only occurs at the network edge, while sources renegotiate CBR rates from the network over the duration of a flow. Thus, the rates of the users fluctuate over time. Bandwidth renegotiations fail when the current aggregate bandwidth demand exceeds the link capacity, and the renegotiation failure probability is the QoS measure of this service. Thus, our bufferless model is directly applicable to this problem. In any case, the performance of schemes for the bufferless model is a conservative upper bound to the case when there are buffers.

Third, the flows are assumed to have homogeneous statistics. How this assumption can be relaxed will be discussed in Section 5.5.4.

Before we begin the analysis of these models, a few words about the notations in this chapter. We use capital letters to denote random variables. The Gaussian distribution will play a central role in our analysis; the probability density function of a zero mean, unit variance Gaussian random variable ($N(0,1)$) is denoted by

$$\phi(x) := \frac{1}{\sqrt{2\pi}} \exp(\frac{-x^2}{2}) \tag{5.1}$$

and the complementary cumulative distribution function denoted by

$$Q(x) := \int_x^\infty \phi(u)du. \tag{5.2}$$

5.3 Impulsive Load Models

5.3.1 Infinite Flow Holding Time

In this subsection, we study the impulsive load model with infinite flow holding time, when flows are admitted at time 0 and stay in the system forever. The goal here is twofold. First, we wish to illustrate the importance of the additional uncertainty due to *measurement* or *estimation error*, by comparing schemes with perfect knowledge and measurement-based schemes. Second, we wish to lay the groundwork for the more sophisticated models discussed in subsequent sections.

Suppose the stationary bandwidth distribution of each flow i has mean μ and variance σ^2. The number of admissible flows m^* is the largest integer m such that

$$\Pr\left\{\sum_{i=1}^{m} X_i(t) > c\right\} \leq p_q. \tag{5.3}$$

where $X_i(t)$ is the bandwidth of the ith flow at time t. (Recall that $c := n\mu$ is the total

capacity of the link.) For large system size n, the number of admissible calls will be large, and by the Central Limit Theorem,

$$\frac{1}{\sigma\sqrt{m}}\left[\sum_{i=1}^{m} X_i(t) - m\mu\right] \sim N(0,1)$$

Thus, if the parameters μ and σ^2 are known *a priori*, then the number of flows m^* to accept should satisfy:

$$Q\left[\frac{n\mu - m^*\mu}{\sigma\sqrt{m^*}}\right] = p_q. \tag{5.4}$$

where $Q(\cdot)$ is the ccdf of a $N(0,1)$ Gaussian random variable as defined in eqn. (5.2). [1] Because the AC has perfect knowledge of the statistics, the actual steady state overflow probability

$$p_f := \Pr\left\{\sum_{i=1}^{m^*} X_i(t) > c\right\}$$

satisfies the QoS requirement. For reasonably large capacities, it follows from solving (5.4) that m^* is well approximated by:

$$m^* = n - \frac{\sigma\alpha_q}{\mu}\sqrt{n} + o(\sqrt{n}) \tag{5.5}$$

where $\alpha_q := Q^{-1}(p_q)$ and $o(\sqrt{n})$ denotes a term which grows slower than $\sqrt{n}$. Note that n is the number of flows that can be carried on the link if each has constant bandwidth μ. Thus, the term $\frac{\sigma\alpha_q}{\mu}\sqrt{n}$ in the above expression can be interpreted as the safety margin left to cater for the (known) burstiness of the traffic.

Now, consider the situation when a MBAC does not know μ and σ *a priori*, but relies on an estimation of these parameters from the initial bandwidth of the flows and use the estimates in a *certainty equivalent* fashion. More specifically, we assume there are an infinite number of flows waiting for admission at time 0 due to a burst of arrivals. Invoking again the central limit approximation for large systems, the number of flows M_0 the MBAC admits should satisfy:

$$Q\left[\frac{n\mu - M_0\hat{\mu}}{\hat{\sigma}\sqrt{M_0}}\right] = p_q, \tag{5.6}$$

where

$$\widehat{\mu} = \frac{1}{n}\sum_{i=1}^{n} X_i(0) \qquad \text{and} \qquad \widehat{\sigma} = \left[\frac{1}{n-1}\sum_{i=1}^{n}(X_i(0) - \widehat{\mu})^2\right]^{\frac{1}{2}} \tag{5.7}$$

The criterion (5.6) is the same as (5.4), but with the true mean μ and standard deviation σ replaced by the *estimated* mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$ respectively.[2] Note that the number of flows M_0 admitted under the MBAC is now random, depending on the random bandwidths of the flows at time 0. This is a consequence of the fact that the admission control decisions are made based on measurements rather than known parameters. Also, the scheme considered here is an example of a *memoryless* MBAC, since the admission

[1] Note that here, as in the sequel, we are ignoring the fact that m^* is an integer and therefore eqn. (5.4) cannot be satisfied exactly in general. In the regime of large capacities, however, the approximation is good and the discrepancy can be ignored.

[2] Observe here that the estimation is based on n flows. In a more precise model, the estimation should be based on M_0 flows, the number to be admitted. However, in a large system, M_0 will be close to n and the discrepancy in replacing M_0 by n in the estimators are small.

control decisions are made based on the current bandwidths only.

We now want to approximate the average overflow probability

$$p_f := \Pr\left\{\sum_{i=1}^{M_0} X_i(t) > c\right\}$$

in steady state (i.e. for t large) and compare it to the target p_q. To do this, we first find an approximation for the distribution of M_0, the number of flows admitted.

For large capacities, by the law of large numbers, the estimated mean $\hat{\mu}$ will be close to the true mean μ, and the estimated variance $\hat{\sigma}^2$ will be close to the true variance σ^2. A more precise approximation of the deviation of these estimated quantities from the true values is given by the Central Limit Theorem:

$$\begin{aligned}\hat{\mu} &= \frac{1}{n}\sum_{i=1}^{n} X_i(0) = \mu + \frac{1}{\sqrt{n}}\left\{\frac{1}{\sqrt{n}}\left[\sum_{i=1}^{n} X_i(0) - n\mu\right]\right\} \\ &= \mu + \frac{\sigma Y_0}{\sqrt{n}} + o(\frac{1}{\sqrt{n}}) \qquad (5.8)\end{aligned}$$

for large n. Here, $Y_0 \sim N(0,1)$, and can be interpreted as the scaled aggregate bandwidth fluctuation at time 0 around the mean. Similarly, the estimated standard deviation can be written as:

$$\hat{\sigma} = \sigma + \frac{Z_0}{\sqrt{n}} + o(\frac{1}{\sqrt{n}}) \qquad (5.9)$$

where Z_0 is Gaussian. These two approximations imply that the deviation of the estimates from the respective true quantities is of order $\frac{1}{\sqrt{n}}$. Now, as mentioned earlier, if the estimates were *exactly* equal to their true values, then the number of flows admitted M_0 will be precisely m^*. This suggests that we can approximate the distribution of M_0 by a *linearization* of the relationship (5.6) around a nominal operating point (m^*, μ, σ) (i.e. the operating point under perfect knowledge):

$$\frac{n\mu - (m^* + \Delta_M)(\mu + \frac{\sigma Y_0}{\sqrt{n}})}{(\sigma + \frac{Z_0}{\sqrt{n}})\sqrt{m^* + \Delta_M}} = \alpha_q$$

Expanding the left hand side, using (5.4) and neglecting terms $o(1)$[3], we get[4]

$$\frac{\Delta_M}{\sqrt{n}} + \frac{\sigma}{\mu} Y_0 = 0$$

and hence

$$M_0 \approx m^* - \frac{\sigma}{\mu} Y_0 \sqrt{n}. \qquad (5.10)$$

Thus, we see that the effect of estimation error is an order $\sqrt{n}$ Gaussian fluctuation around m^*, the number of sources admitted under perfect knowledge (cf. top part of Fig 5.2). Note also that the randomness in the number of flows admitted is due mainly to the error in estimating the mean (Y_0) rather than the error in estimating the standard deviation (Z_0).

[3]The notation $o(1)$ means terms going to zero as n goes to infinity

[4]Here and in the sequel, the notation "$\approx$" refers to an approximation with an error $o(\sqrt{n})$.

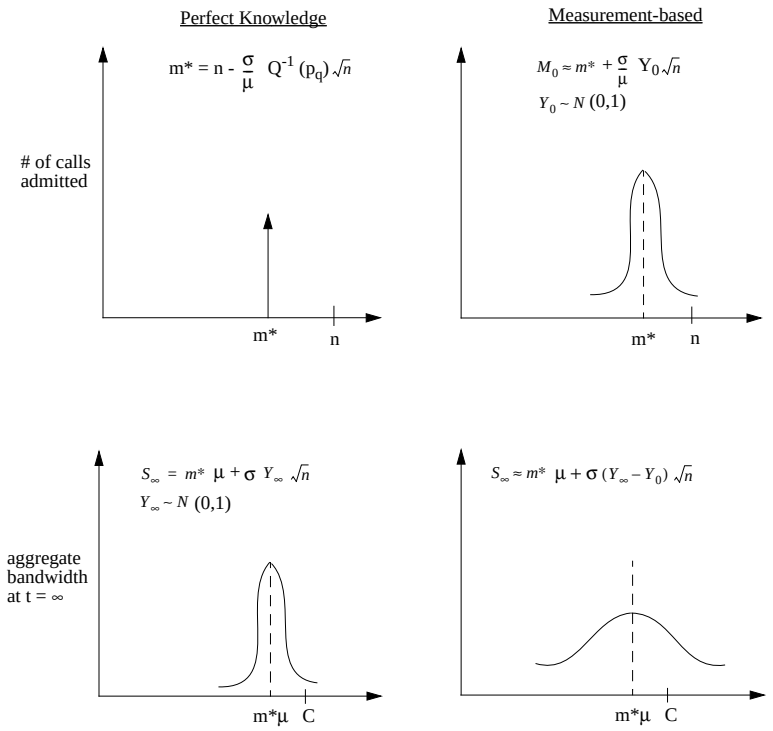

Figure 5.2: Uncertainty due to fluctuation in the number of flows (top) and in the aggregate bandwidth (bottom), for an admission controller with perfect knowledge (left) and for an MBAC (right).

Substituting eqn. (5.5) into (5.10), we get M_0 in terms of the system size n:

$$M_0 \approx n - \frac{\sigma}{\mu}(Y_0 + \alpha_q)\sqrt{n} \tag{5.11}$$

Although we have derived the result somewhat heuristically, it can be made precise by the following result, which is proved in Appendix A.

Proposition 2 *For each system size n, let $M_0^{(n)}$ be the random number of flows admitted under the MBAC when the capacity is $n\mu$. Then the sequence of random variables $\{\frac{M_0^{(n)}-n}{\sqrt{n}}\}$ converges* in distribution *to the random variable $-\frac{\sigma}{\mu}(Y_0 + \alpha_q)$.*

We now proceed with an explicit approximation of the overflow probability. The randomness in the aggregate traffic load at some future time is due both to the randomness in the number of flows admitted as well as the randomness in the bandwidth demands of those flows. This can be approximated with the help of the following lemma, which is an extension of the Central Limit Theorem for a sum of a random number of random variables:

Lemma 4 *[10, p. 369, problem 27.14] Let $X_1, X_2, \ldots$ be independent, identically distributed random variables with mean μ and variance σ^2, and for each positive n, let V_n be a random variable assuming positive integers as values; it need not be independent of the X_m's. Let $W_n = \sum_{i=1}^{V_n} X_i$. Suppose as $n \to \infty$, $\frac{V_n}{n}$ converges to 1 almost surely. Then as $n \to \infty$,*

$$\frac{W_n - V_n\mu}{\sigma\sqrt{n}}$$

converges in distribution to a $N(0,1)$ random variable.

Applying this lemma, the aggregate load at time t can approximated by:

$$S_t := \sum_{i=1}^{M_0} X_i(t) \approx M_0\mu + \sigma Y_t\sqrt{n} \tag{5.12}$$

Here $Y_t \sim N(0,1)$ and can be interpreted as an approximation for the scaled aggregate bandwidth fluctuation at time t:

$$\frac{1}{\sigma\sqrt{n}}\left[\sum_{i=1}^{n} X_i(t) - n\mu\right] \tag{5.13}$$

Intuitively, eqn. (5.12) means that the fluctuation of the aggregate load is approximately the linear superposition of two effects: the random number of flows together with the random bandwidth fluctuation around the mean. Substituting eqn. (5.10), we get

$$S_t \approx n\mu + \sigma(Y_t - Y_0 - \alpha_q)\sqrt{n}$$

Thus, for large n, the overflow probability at time t is:

$$\Pr\{S_t > n\mu\} \approx \Pr\{Y_t - Y_0 > \alpha_q\}$$

This expression gives us an interpretation of how overflow occurs in a MBAC system: it is a combination of an aggregate bandwidth estimation error at admission time (Y_0) and a fluctuation of the aggregate bandwidth (Y_t) at time t after the flows have been accepted. Contrast this with the case with perfect knowledge, where the overflow probability at time t is simply $\Pr\{Y_t > \alpha_q\}$, due to bandwidth fluctuation at time t.

To get the overflow probability in steady state, we set $t = \infty$, in which case Y_∞ is independent of Y_0. Therefore, the difference $Y_\infty - Y_0$ is a Gaussian random variable with mean 0 and variance $2\sigma^2$. The overflow probability is therefore

$$p_f \approx Q\left(\frac{\alpha_q}{\sqrt{2}}\right). \tag{5.14}$$

We summarize this result more formally in the following proposition:

Proposition 3 *Suppose the target overflow probability QoS is p_q. Let $p_f^{(n)}$ be the actual average steady state overflow probability using the certainty equivalent MBAC for capacity $n\mu$. Then as the system size grows:*

$$\lim_{n\to\infty} p_f^{(n)} = Q\left(\frac{Q^{-1}(p_q)}{\sqrt{2}}\right)$$

Note that for the AC with perfect knowledge, the overflow probability is exactly p_q. This is because the aggregate bandwidth fluctuation stems only from the fluctuation of the individual flows' bandwidths (cf. lower left part of Fig. 5.2). On the other hand, in the measurement-based case, the variance of the aggregate bandwidth is doubled because the number of flows also fluctuates due to measurement error (cf. lower right part of Fig. 5.2). The $\sqrt{2}$ factor is therefore the effect of measurement error, and has quite a tremendous impact on the overflow probability p_f. For example, if $p_q = 1.0e-5$, then

the actual performance in the MBAC system would be $p_f \approx 1.3e-3$, a difference of two orders of magnitude. In other words, if we want to achieve $p_f = p_q$ using a MBAC in this impulsive load model, then we have to adjust the target overflow probability under certainty equivalence.

$$p_{ce} = Q\left(\sqrt{2}\alpha_q\right) \qquad \text{or} \qquad \alpha_{ce} := Q^{-1}(p_{ce}) = \sqrt{2}\alpha_q. \tag{5.15}$$

Using the approximation $Q(x) \approx \frac{\phi(x)}{x}$ for small $Q(x)$, we see that

$$p_{ce} \approx \frac{\alpha_q}{2\sqrt{\pi}} p_q^2$$

Thus, we see that to achieve a target p_q in this setting, we need to set p_{ce} roughly to be the square of the target probability. This conservatism leads to a loss in system *utilization* compared to the scheme with perfect knowledge of the statistics. The average utilization (in terms of bandwidth) for the certainty equivalent scheme using parameter p_{ce} instead of p_q is given by $E(M_0)\mu$, or $c - \sigma\alpha_{ce}\sqrt{n}$, as implied by eqn. (5.10). The average utilization for the perfect knowledge scheme, on the other hand, is given by $m^*\mu$, or $c - \sigma\alpha_q\sqrt{n}$, as inferred from (5.5). Thus, if we pick α_{ce} to be $\sqrt{2}\alpha_q$, this translates to a loss of utilization of $(\sqrt{2}-1)\sigma\alpha_q\sqrt{n}$.

Proposition 3 has several surprising aspects. First, it is a *universal* result in the sense that the performance of the certainty equivalent scheme does not depend on the stationary distribution of the flow nor its mean and variance. Second, although the estimators are unbiased, the net impact on the performance of the system is negative. Thus there is an inherent asymmetry between the effects of over-estimation and under-estimation. Third, the impact of the estimation error does not vanish as the system size becomes large, even though the estimates become more and more accurate. Fourth, for a large system, the degradation in performance of the certainty equivalent scheme is due mainly to the estimation error in the *mean* μ of the bandwidth distribution and not to that in the *standard deviation* σ.

To get more insights into the last two phenomena, let us perform the following deterministic sensitivity analysis. Define the following function:

$$p_f(\mu, \sigma, m) := Q\left[\frac{c - m\mu}{\sigma\sqrt{m}}\right]$$

which is the overflow probability when there are m flows in the system each with mean rate μ and variance σ^2. Suppose first that we measure only μ, but that σ is known exactly. The number of flows admitted $m(\widehat{\mu})$ depends on the measured value $\widehat{\mu}$ and is given by the certainty-equivalent admission criterion (compare with (5.6)):

$$p_f(\widehat{\mu}, \sigma, m(\widehat{\mu})) = p_q. \tag{5.16}$$

Note that the *actual* overflow probability p_f for a given $m(\widehat{\mu})$ is $p_f(\mu, \sigma, m(\widehat{\mu}))$. The *sensitivity* of the overflow probability with respect to the measured $\widehat{\mu}$ is the deviation of p_f from its target value p_q if $\widehat{\mu}$ deviates slightly from its target value μ. For small deviations, we can simply use the derivative of p_f with respect to $\widehat{\mu}$.

$$s_\mu := \frac{\partial}{\partial\widehat{\mu}} p_f(\mu, \sigma, m(\widehat{\mu}))\bigg|_{\widehat{\mu}=\mu}.$$

Using (5.16), this derivative can be computed as:

$$s_\mu = -\frac{\phi(\alpha_q)\mu}{\sigma}\sqrt{m^*}.$$

Similarly, the sensitivity with respect to measured $\widehat{\sigma}$, assuming μ known, is given by:

$$s_\sigma = -\frac{\alpha_q \phi(\alpha_q)}{\sigma}$$

Now observe that the sensitivity of the system performance on the knowledge of the standard deviation, s_σ, does not depend on the system size. Therefore, increasing the system size, and therefore improving the quality of the estimator $\widehat{\sigma}$, results in a *diminishing* net impact on the overflow probability. On the other hand, the sensitivity s_μ *increases* with the system size, approximately as $\sqrt{n}$, while the variance of the estimator $\widehat{\mu}$ decreases approximately as $1/\sqrt{n}$. This suggests that the net impact of the uncertainty in the mean bandwidth estimate does not diminish as the system size grows, and also explains why the deviation from p_f from the target overflow probability p_q is asymptotically independent of n: both effects, less estimation error but increased sensitivity to estimation error, cancel out. The increased sensitivity to the mean estimate arises because when there are more flows in the system, and therefore more statistical regularity in the aggregate bandwidth, the system is driven closer to full utilization, which makes it more susceptible to admission mistakes.

The approximations used here are based in the *heavy traffic regime*, where the system size is large and when scaling up the size of the system, we exploit the additional statistical regularity by increasing the system utilization, while keeping the QoS constant. This is in contrast to the *large deviations regime*, where the system utilization is asymptotically constant, but where the QoS-requirement is scaled with the system size. The heavy traffic approximations allow us to linearize the dynamics of the system and to use Gaussian statistics. This will prove even more valuable as we analyze more complex models in the next sections. A large deviations analysis of a related measurement-based admission control problem can be found in [109].

5.3.2 Finite Holding Time

Now that we have convinced ourselves that estimation error can have an impact that should not be neglected, we want to refine the previous model. More specifically, we now assume that the time-scale separation is finite. There still is a burst of flows arriving at time 0 and demanding admission into the system. However, these flows are now assumed to have *finite duration*. In fact, we assume that the holding time of a flow (i.e., the time between the flow's admission and the time when it departs from the system) is an exponential random variable with mean T_h, and the holding times of different flows are assumed independent. We let p_t denote the probability that a flow has not departed from the system at time t. It is given by

$$p_t = \exp\left(-\frac{t}{T_h}\right). \tag{5.17}$$

Furthermore, we let $\rho(t)$ denote a flow's auto-correlation function, where

$$\rho(t) := \frac{E[(X_i(0)-\mu)(X_i(t)-\mu)]}{\sigma^2}.$$

If N_t is the number of flows left in the system at time t, and M_0 is the initial number of flows admitted into the system, then expected number of flows $E[N_t]$ at time t is $p_t E[M_0]$. Using eqn. (5.10), this implies that

$$E[N_t] \approx p_t n - \frac{p_t \sigma \alpha_q}{\mu} \sqrt{n}$$

We observe that the system size is n, and so approximately a fraction p_t of the total capacity is used at time t. The law of large number suggests that as n becomes large and everything else fixed, the overflow probability at time t actually goes to zero!

Intuitively, this can be explained as follows. When performing certainty-equivalent admission control, we set aside some bandwidth in order to accommodate fluctuations of the aggregate bandwidth. This spare bandwidth is on the order of $\sqrt{n}$ (cf. (5.5)). On the other hand, the flow departure rate is *proportional* to the number of flows in the system, approximately proportional to n/T_h. Now suppose that at some time instant, the system is close to overloading. How much time is necessary to restore the "safety margin" of $\sqrt{n}$ by letting flows depart? This restore time is on the order of $\sqrt{n}/(n/T_h) = T_h/\sqrt{n}$. Thus, the larger the system, the faster can the safety margin be restored. This means that to cause an overload, the aggregate bandwidth must fluctuate fast enough so that this fluctuation cannot be compensated for by just letting flows depart. However, as the time-scale gets shorter, the aggregate bandwidth tends to be more correlated, thus making such a quick change more and more unlikely.

While the above suggests that for large enough n, the overflow probability gets close to zero, it is clear that the longer the duration T_h of the flows, the larger the system size has to be for this effect to kick in. The above asymptotic analysis is crude in the sense that the flow duration, which may be quite long, does not enter the picture, since all other parameters are kept fixed while n grows large. On the other hand, it can be seen from the above discussion that the restore time $T_h/\sqrt{n}$ is the natural time-scale to analyze the dynamics due to flow departure. To make such analysis more convenient, let us rescale the flow holding time:

$$T_h = \widetilde{T_h}\sqrt{n}$$

where we view $\widetilde{T_h}$ fixed as n grows large. The advantage of this scaling is that it allows us to make approximations for large n but at the same time taking into consideration the actual duration of the flows. More specifically, it can be shown, under this scaling, the flow departure rate can be thought of as constant equal to $\sqrt{n}/\widetilde{T_h}$. Letting $D[0,t]$ be the number of flows departing in $[0,t]$, we have the approximation:

$$D[0,t] \approx \frac{t}{\widetilde{T_h}} \sqrt{n} \tag{5.18}$$

Using eqn. (5.10), the number of flows left in the system at time t can therefore be given by:

$$N_t = M_0 - D[0,t] \approx n - \left[\frac{\sigma}{\mu}(Y_0 + \alpha_q) + \frac{t}{\widetilde{T_h}} \right] \sqrt{n} \tag{5.19}$$

Using Lemma 4, the aggregate load at time t is:

$$S_t \;=\; \sum_{i=1}^{N_t} X_i(t) \approx N_t \mu + \sigma Y_t \sqrt{n}$$

$$\approx \quad n\mu + \sigma\left(Y_t - Y_0 - \frac{\mu t}{\sigma\widetilde{T_h}} - \alpha_q\right)\sqrt{n} \tag{5.20}$$

where Y_t is an approximation of the scaled fluctuation of the aggregate bandwidth

$$\frac{1}{\sigma\sqrt{n}}\left[\sum_{i=1}^{n} X_i(t) - n\mu\right].$$

By the Central Limit Theorem applied to pairs of random variables [10], Y_0 and Y_t are jointly Gaussian random variables with zero means, unit variances and covariance $\rho(t)$ (i.e. same as an individual flow). Thus, $Y_t - Y_0 \sim N[0, 2(1-\rho(t))]$.

The overflow probability $p_f(t)$ at time t is given by

$$\begin{aligned} p_f(t) &\approx \Pr\left\{Y_t - Y_0 > \frac{\mu t}{\sigma\widetilde{T_h}} + \alpha_q\right\} \\ &= Q\left(\frac{1}{\sqrt{2(1-\rho(t))}}\left[\frac{\mu}{\sigma}\frac{t}{\widetilde{T_h}} + \alpha_q\right]\right) \end{aligned} \tag{5.21}$$

From (5.21), we can see clearly the two effects affecting the overflow probability. For small t, the denominator $\sqrt{2(1-\rho(t))}$ is close to zero, making the overflow probability very small. This is because shortly after the admission decision, due to correlation in the bandwidth of the flows, the aggregate bandwidth does not change much. For large t, $t/\widetilde{T_h}$ makes the argument of the Q-function large as well, i.e. the overflow probability small. This is because enough flows have departed to make overflow unlikely. Intuitively, $\widetilde{T_h}$ defines the *critical time-scale* for this system: it is unlikely that an overflow event occurs at times significantly after $\widetilde{T_h}$. Thus, in the study of this system, we can concentrate on what happens between times of the order of $\widetilde{T_h}$. It is interesting that since $\widetilde{T_h} = T_h/\sqrt{n}$, this critical time-scale depends not only on the average holding time but also the size of the system.

5.4 The Continuous Load Model

We shall now consider a full-blown dynamical model, where flows arrive *continuously* over time. We assume a worst-case scenario, where the effective arrival rate is infinite, i.e. there are always flows waiting to be admitted into the network. Thus, admission control decisions are made continuously at all times. Clearly, the performance of any admission control algorithm under finite arrival rate will be no worse than its performance in this model. Another advantage of this model is that we need not worry about the specific flow arrival process which may be hard to model in practice. As before, when flows are admitted, they stay for a duration exponentially distributed with mean T_h. In this section, we will look at both memoryless MBAC schemes and schemes with memory and compare their performance.

5.4.1 Memoryless MBAC

We first look at the scheme that was considered in the impulsive load model, where admission control decisions are made based on estimates of the mean and variance using the current bandwidths of the flows. Assume that the system starts at time 0. Our goal is to find the overflow probability at an arbitrary time t, particularly at $t = \infty$ which yields

the steady-state overflow probability. We do this by first analyzing the dynamics of the number of flows in the system.

Let M_t be the number of flows that the MBAC determines should be in the network at time t; as in (5.22), M_t is given by:

$$Q\left[\frac{n\mu - M_t\hat{\mu}(t)}{\hat{\sigma}(t)\sqrt{M_t}}\right] = p_q, \tag{5.22}$$

where

$$\widehat{\mu}(t) = \frac{1}{n}\sum_{i=1}^{n} X_i(t) \qquad \text{and} \qquad \widehat{\sigma}(t) = \left[\frac{1}{n-1}\sum_{i=1}^{n}(X_i(t) - \widehat{\mu}(t))^2\right]^{\frac{1}{2}} \tag{5.23}$$

Observe that M_t is random and depends only on the current bandwidths $X_i(t)$'s of the flows. Call M_t the *estimated admissible* number of flows at time t. The *actual* number of flows N_t in the system at time t is no less than M_t since there are always flows waiting to be admitted and thus the system is always filled to the limit as currently determined by the MBAC. On the other hand, N_t can be strictly greater than M_t as flows that were admitted earlier stay for a certain duration and thus N_t cannot perfectly track the fluctuations of M_t (see Fig. (5.3)). To compute N_t, first observe that if s^* is the last time at or before time t that flows were admitted, then the number of flows in the system at time s^* is precisely the same as number of flows admissible at time s^*, i.e. $N_{s^*} = M_{s^*}$. In between time s^* and time t, no new flows were admitted. Hence, if we let $D[s,t]$ be the number of flows departed in time interval $[s,t]$, then

$$N_t = N_{s^*} - D[s^*,t] = M_{s^*} - D[s^*,t] \tag{5.24}$$

On the other hand, for *any* $s \le t$,

$$N_t = N_s + A[s,t] - D[s,t] \ge N_s - D[s,t] \ge M_s - D[s,t] \tag{5.25}$$

where $A[s,t]$ is the number of flows *admitted* during $[s,t]$. Thus we conclude from (5.24) and (5.25) that

$$N_t = \sup_{0\le s\le t}\{M_s - D[s,t]\} \tag{5.26}$$

Eqn. (5.19) in the previous section tells us that $M_s - D[s,t]$ is the number of flows in the system at time t if there were only a single impulse of flow arrivals at time s. Thus, the effect under a continuous load can be thought of as the superposition of the effects from all impulsive arrival times, starting at time 0.

Using formula (5.26), we can approximate N_t using our approximations for M_s and $D[s,t]$ as discussed in the previous section. Eqn. (5.10) gives an approximation for M_s:

$$M_s \approx n - \frac{\sigma}{\mu}(Y_s + \alpha_q)\sqrt{n} \tag{5.27}$$

where $\{Y_t\}$ is a stationary zero-mean Gaussian process with unit variance and auto-correlation function $\rho(t)$ (that of an individual flow), and can be interpreted as the scaled aggregate bandwidth fluctuation of the flows around the mean. Eqn. (5.18) suggests an approximation for $D[s,t]$:

$$D[s,t] \approx \frac{(t-s)}{\widetilde{T_h}}\sqrt{n}$$

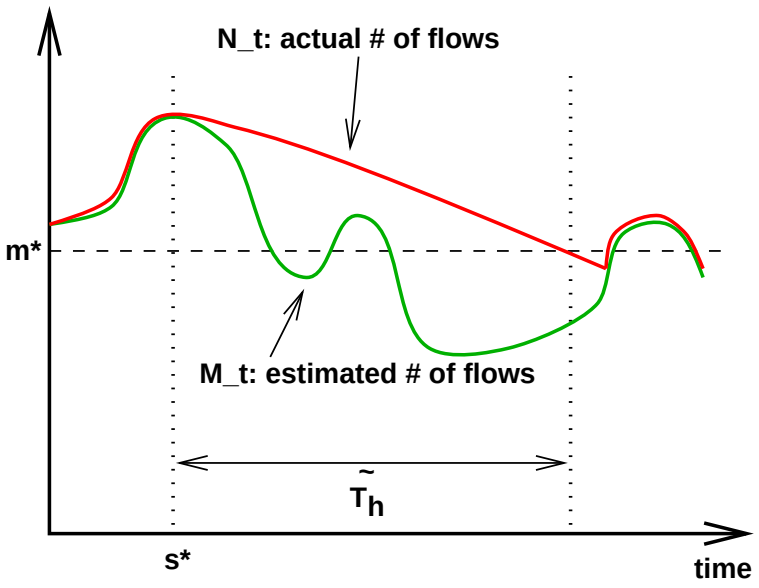

Figure 5.3: The relationship between the current estimate of admissible number of flows M_t and the actual number of flows N_t. The time-scale $\widetilde{T_h}$ is the typical time for the system to recover from admission errors.

An approximation for the number of flows in the system is then given by:

$$\begin{aligned} N_t &\approx \max_{s \le t} \left\{ n - \left[\frac{\sigma}{\mu}(Y_s + \alpha_q) + \frac{(t-s)}{\widetilde{T_h}} \right] \sqrt{n} \right\} \\ &= n + \frac{\sigma\sqrt{n}}{\mu} \max_{s \le t} \left\{ -Y_s - \frac{\mu(t-s)}{\sigma\widetilde{T_h}} - \alpha_q \right\} \end{aligned} \tag{5.28}$$

Once we obtained an approximation for N_t, we can immediately deduce an approximation for the aggregate load S_t at time t and hence the steady-state overflow probability p_f, using the same argument as for the impulsive load model.

$$\begin{aligned} S_t &\approx N_t\mu + \sigma Y_t\sqrt{n} \\ &\approx n\mu + \sigma \max_{s \le t} \{Y_t - Y_s - \beta(t-s) - \alpha_q\} \sqrt{n} \end{aligned}$$

where we define for brevity

$$\beta := \frac{\mu}{\sigma\widetilde{T_h}}. \tag{5.29}$$

The steady-state overflow probability is therefore:

$$p_f = \Pr\{S_t > n\mu\} \approx \Pr\left\{ \max_{s \le t} \{Y_t - Y_s - \beta(t-s)\} > \alpha_q \right\} \tag{5.30}$$

Interestingly, one can interpret the limiting overflow probability at time t as that of the length of a certain *queue* at time t exceeding α_q. The queue is one which has a constant service rate of β, with the amount of work arriving in time interval $[s,t]$ given by $Y_t - Y_s$, with the notable difference that $Y_t - Y_s$ *cannot* be negative in the case of a queue.

5.4.2 Analysis of Overflow Probability

Our next step is to analyze the approximation to the overflow probability given by eqn. (5.30). Since the process $\{Y_t\}$ is stationary and symmetrically distributed around 0, we can rewrite that as

$$p_f \approx \Pr\left\{\max_{t\geq 0}\{Y_{-t} - Y_0 - \beta t\} > \alpha_q\right\}.$$

This can be interpreted as the *hitting* probability of a Gaussian process $\{Y_{-\tau} - Y_0\}$ on a moving boundary $y = \beta\tau + \alpha_q$. While there is no known closed-form solution to this problem, an approximation can be obtained by applying results due to Braker [50, 51] on hitting probabilities of locally stationary Gaussian processes, extending the results by [24] for stationary processes. Define

$$\sigma^2(t) := E[(Y_{-t} - Y_0)^2] = 2[1 - \rho(t)]$$

to be the variance of $Y_{-t} - Y_0$. (Recall that Y_t has zero mean and unit variance.) Assume the single-sided derivatives of $\rho(t)$ at $t = 0$ exist and are finite, let $v^+(0)$ be the right derivative of the function $\sigma^2(t)$ at $t = 0$. [5] Then an approximation to the hitting probability is given by:

$$\Pr\left\{\sup_{t\geq 0}\{Y_{-t} - Y_0 - \beta t\} > \alpha_q\right\} \approx$$
$$\approx \frac{1}{2}\int_0^\infty v^+(0)\frac{\alpha_q + \beta t}{\sigma^3(t)}\phi\left(\frac{\alpha_q + \beta t}{\sigma(t)}\right)dt \tag{5.31}$$

where $\phi(x)$ is the $N(0,1)$ probability density function. The integrand above can be viewed as an approximation to the first hitting time density at time t; integrating over all t yields the probability that hitting occurs at all. This is an approximation in the sense that as $\alpha_q \to \infty$, the ratio of the left-hand and the right-hand sides approaches 1. Hence this approximation is good when p_q is small.

While this yields an approximation that can be computed numerically for general auto-correlation functions, we would like to get more analytical insights. To that end, consider the specific auto-correlation function:

$$\rho(t) = \exp(-\frac{|t|}{T_c}). \tag{5.32}$$

With this choice of the auto-correlation function, $\{Y_t\}$ is the well-known Ornstein-Uhlenbeck process. The parameter T_c governs the exponential drop-off rate of the correlation function, and is a natural *correlation time-scale* for the burst dynamics of the traffic. Substituting this into the approximation (5.31) and rescaling the time variable, we get:

$$p_f \approx \gamma\int_0^\infty \frac{(\alpha_q + t)}{[2(1 - \exp(-\gamma t))]^{\frac{3}{2}}}\phi\left(\frac{\alpha_q + t}{\sqrt{2(1 - \exp(-\gamma t))}}\right)dt \tag{5.33}$$

[5] i.e. $v^+(0) := \lim_{t\to 0^+}\frac{\sigma^2(t) - \sigma^2(0)}{t}$.

where

$$\gamma := \frac{1}{\beta T_c} = \frac{\widetilde{T_h}}{T_c} \cdot \frac{\sigma}{\mu}.$$

One can think of γ as the separation between the flow and burst scales, although note that $\widetilde{T_h}$ is the scaled holding time. If we make a time-scale separation assumption, i.e. $\gamma \gg 1$, then

$$p_f \approx \gamma \int_0^\infty \frac{(\alpha_q + t)}{2^{\frac{3}{2}}} \phi\left(\frac{\alpha_q + t}{\sqrt{2}}\right) dt = \frac{\gamma}{2\sqrt{\pi}} \exp\left(-\frac{1}{4}\alpha_q^2\right) \tag{5.34}$$

Note that the first approximation is via $\exp(-\gamma t) \approx 0$ for $\gamma \gg 1$.

It is interesting to compare this overflow probability for the continuous-load model with the corresponding result for the impulsive load model under long flow durations, given in Proposition (3). To do this, we first use the approximation $\frac{\phi(x)}{x} \approx Q(x)$ and rewrite (5.34) in terms of the flow parameters as

$$p_f \approx \frac{\widetilde{T_h}}{2T_c} \cdot \frac{\sigma \alpha_q}{\mu} Q\left(\frac{\alpha_q}{\sqrt{2}}\right) \tag{5.35}$$

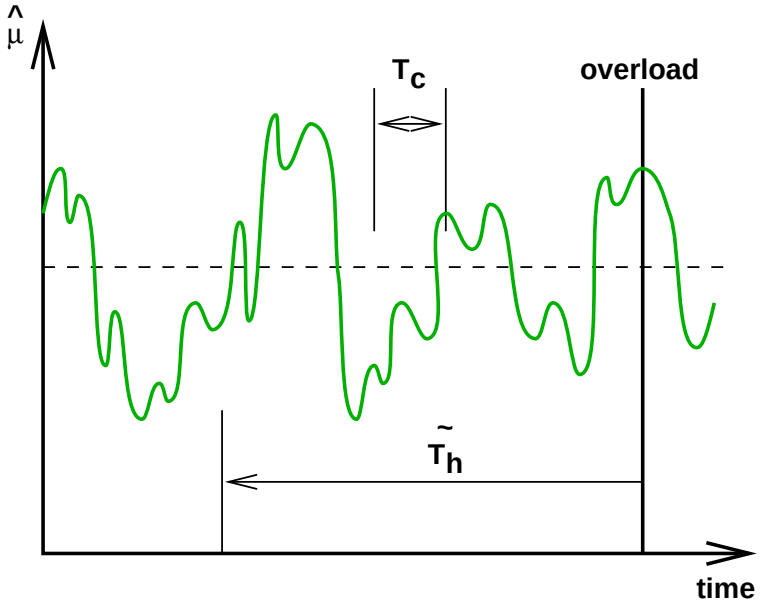

Figure 5.4: The ratio of correlation time-scale T_c and of the critical time-scale $\widetilde{T_h}$ determines the overflow probability.

For the impulsive load model, the overflow probability is approximately $Q(\frac{\alpha_q}{\sqrt{2}})$. Eqn. (5.35) tells us that in the regime of separation of time-scales, the corresponding overflow probability can be much worse in the continuous-load model. This is because while estimation errors can occur only at a *single* point of time in the impulsive load model (time 0), in the continuous-load model estimation errors occurring at *any* time in an interval of size roughly $\widetilde{T_h}$ before time t will have a significant impact on the number of flows at time t. The shorter the traffic correlation time-scale T_c, the faster the memoryless mean bandwidth estimates fluctuates, and the larger the probability of having an under-estimation at some time in the interval. Hence, the overflow probability in the continuous-load model increases with the separation of time-scale $\frac{\widetilde{T_h}}{T_c}$. For example, note the multiple peaks (underestimations of

μ) within the interval of length $\widetilde{T_h}$ in Fig. 5.4: each of these peaks could potentially cause overload within the critical time-scale $\widetilde{T_h}$. The lesson is that it's not only important to consider the estimation error at a single time-instant, but also the chance of making error any time in the interval defined by the effective flow holding time-scale $\widetilde{T_h}$. Note also since $\widetilde{T_h}$ decreases as $\frac{T_h}{\sqrt{n}}$, where T_h is the actual mean holding time, the overflow probability decreases roughly as $\frac{1}{\sqrt{n}}$.

We can also write the above approximation as (using again $\frac{\phi(x)}{x} \approx Q(x)$),

$$p_f \approx \frac{\widetilde{T_h}}{\sqrt{2}T_c} \frac{\sigma}{\sqrt{2\pi}\mu} \left(\sqrt{2\pi}\alpha_q p_q\right)^{\frac{1}{2}} \tag{5.36}$$

5.4.3 MBAC with Memory

We see that the memoryless scheme suffers from two problems . First, the estimation error at a specific admission time instant is large, and in fact has impact which is of the same order of magnitude as that due to the statistical fluctuations of the bandwidths when the correct number of flows are admitted. Second, the correlation time-scale of the estimation errors is the same as that of the traffic itself; thus, in the regime when the flow holding time is much larger than the traffic correlation time-scale ($\widetilde{T_h} \gg T_c$), the probability of having a large under-estimation of mean bandwidth at *some time* during the time-scale $\widetilde{T_h}$ is high. A strategy which, as we will see, counters both these difficulties is to use more memory in the mean and variance estimators.

To be more concrete, let us consider using the first-order auto-regressive filter with impulse response:

$$h(t) := \frac{1}{T_m} \exp\left(-\frac{t}{T_m}\right) u(t)$$

to estimate both the mean and the variances. (Here, $u(t)$ is the unit step function.) Thus, in place of the memoryless estimators in eqn. (5.23), the MBAC would use:

$$\begin{aligned} \widehat{\mu}_m(t) &= \int_0^\infty \left[\frac{1}{n}\sum_{i=1}^n X_i(t-\tau)\right] h(\tau)d\tau \\ \widehat{\sigma}_m^2(t) &= \int_0^\infty \left[\frac{1}{n-1}\sum_{i=1}^n \left(X_i(t-\tau) - \widehat{\mu}_m(t)\right)^2\right] h(\tau)d\tau \end{aligned}$$

Note that the estimates are obtained by an exponential weighting of the past bandwidths of the flows. The parameter T_m governs how the past bandwidths are weighted; it can thought of as a measure of the *memory size* of the estimators. The relationship between $\widehat{\mu}_m(t)$ and the memoryless estimator $\widehat{\mu}(t)$ is simply $\widehat{\mu}_m = \widehat{\mu} * h$, where $*$ is the convolution operation.

Corresponding to (5.28) in the memoryless case, we can show that the number of flows N_t in the system at time t under the MBAC with memory is approximately

$$N_t \approx n + \frac{\sigma\sqrt{n}}{\mu} \max_{s \le t} \left\{-Z_s - \frac{\mu(t-s)}{\sigma\widetilde{T_h}} - \alpha_q\right\} \tag{5.37}$$

One can interpret Z_t as the error in the *filtered* estimate of the mean bandwidth of a flow at

time t. The steady-state overflow probability under the MBAC with memory can therefore be approximated by:

$$p_f \approx \Pr\left\{\sup_{t\geq 0}(Z_{-t} - Y_0 - \beta t) > \alpha_q\right\}$$

This is again a hitting probability of a Gaussian process ($\{Z_{-t} - Y_0\}$) on a moving boundary, and an approximation of such a probability is given by [50, 51]:

$$p_f \approx \frac{\gamma T_c}{T_c + T_m}\int_0^\infty \frac{(\alpha_q + t)}{\left[\sigma_m(\frac{t}{\beta})\right]^3}\phi\left(\frac{\alpha_q + t}{\sigma_m(\frac{t}{\beta})}\right)dt + Q\left(\alpha_q\sqrt{1 + \frac{T_c}{T_m}}\right) \tag{5.38}$$

where

$$\sigma_m^2(\frac{t}{\beta}) := E[(Z_{-\frac{t}{\beta}} - Y_0)^2] = \frac{2T_c + T_m}{T_c + T_m} - \frac{2T_c}{T_c + T_m}\exp(-\gamma t).$$

The computation of $\sigma_m^2(t)$ is shown in Appendix B.

Now, under separation of time-scales, $\gamma \gg 1$, we have the approximation that

$$\sigma_m^2(\frac{t}{\beta}) \approx \frac{2T_c + T_m}{T_c + T_m}$$

in which case the above integral can be explicitly computed as:

$$\begin{aligned} p_f \;\approx\; & \frac{\gamma T_c}{\sqrt{(T_c + T_m)(2T_c + T_m)}} \cdot \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{T_c + T_m}{2(2T_c + T_m)}\alpha_q^2\right) \\ & + Q\left(\alpha_q\sqrt{1 + \frac{T_c}{T_m}}\right) \end{aligned} \tag{5.39}$$

To compare this result to the memoryless case, let us first use the approximation $Q(x) \approx \frac{\phi(x)}{x}$ to rewrite (5.39) in terms of p_q and also the flow parameters:

$$\begin{aligned} p_f \;\approx\; & \frac{\widetilde{T_h}}{\sqrt{(T_c + T_m)(2T_c + T_m)}} \cdot \frac{\sigma}{\sqrt{2\pi}\mu}\left(\sqrt{2\pi}\alpha_q p_q\right)^{\frac{T_c + T_m}{2T_c + T_m}} \\ & + Q\left(\alpha_q\sqrt{1 + \frac{T_c}{T_m}}\right) \end{aligned} \tag{5.40}$$

Comparing eqn. (5.39) to eqn. (5.34), we can see explicitly the effect of memory. Let us look at the first term in (5.39), which corresponds to (5.36). The exponent is $\frac{T_c + T_m}{(2T_c + T_m)}$ which is $\frac{1}{2}$ when there is no memory (as we had in the memoryless scheme), monotonically increases with T_m, and reaching a value of 1 for infinite memory. This effect can be explained by the fact that the variance of the mean bandwidth estimate, $E[Z_t^2]$, is $\frac{T_c}{T_c + T_m}$ and decreases monotonically to zero with more memory. Thus the inaccuracy in the estimates and hence the inaccuracy in the number of flows accepted decreases (cf. Fig. 5.5). Furthermore, increasing the amount of memory has an additional effect of *smoothing* the mean bandwidth estimates; thus, not only are the *individual* bandwidth estimates more accurate, they also fluctuate less so that the probability of having an under-estimation at *some time* over an interval of length $\widetilde{T_h}$ is reduced. This is reflected in the smaller pre-factor $\frac{\widetilde{T_h}}{\sqrt{(T_c + T_m)(2T_c + T_m)}}$ in the first term of (5.40) replacing the factor $\frac{\widetilde{T_h}}{\sqrt{2T_c}}$ in the memoryless case. This can be

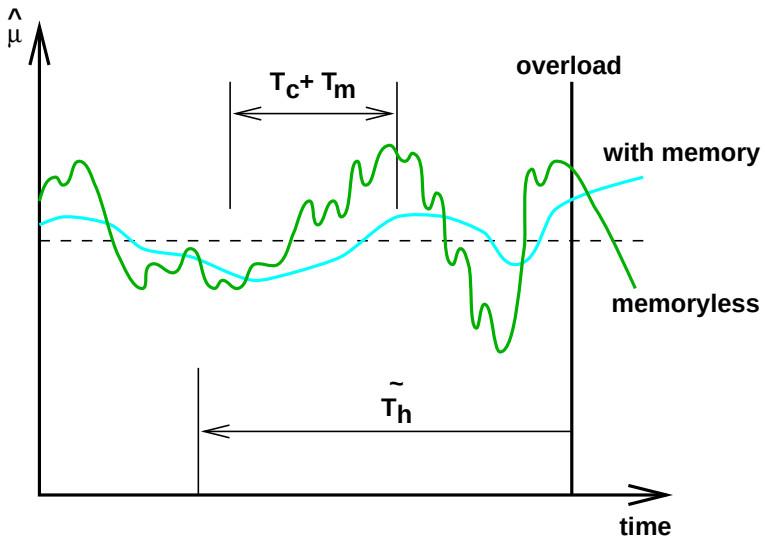

Figure 5.5: Estimation memory reduces the variance of the bandwidth estimator, and also smoothes its fluctuation.

interpreted as increasing the correlation time-scale by T_m, the estimator memory size.

In the limit for large T_m, we always have exactly the right number of flows in the system and the overflow occurs due only to the fluctuation of bandwidth requirements of flows in the system, and not to the fluctuation of the number of flows in the system. This is now given by the second term in (5.40).

The shorter T_m, the more conservative the choice of p_{ce} has to be, resulting in a loss of utilization. This loss of utilization can be quantified. The average utilization (in terms of bandwidth) of the system is given by $\mu E[N_t]$, where N_t is the (stationary) number of flows in the system at time t. Eqn. (5.37) allows us to approximate this when p_{ce} is used as the certainty-equivalent parameter:

$$\mu E[N_t] \approx n\mu + \sigma\sqrt{n}E\left[\sup_{s\leq t}\left\{-Z_s - \frac{\mu(t-s)}{\sigma\widetilde{T_h}}\right\}\right] - \sigma Q^{-1}(p_{ce})\sqrt{n}$$

Since the other terms do not depend on p_{ce}, we see that the difference in utilization in using p_{ce} and p'_{ce} is simply

$$\sigma\sqrt{n}\left[Q^{-1}(p_{ce}) - Q^{-1}(p'_{ce})\right] \tag{5.41}$$

This allows us to quantify the impact on the utilization on using a more conservative certainty-equivalent parameter.

5.5 Discussions and Simulations

We begin this section by summarizing the results obtained from our model. We then report some simulation results. We then illustrate how robust measurement-based admission control can be performed using our results. Finally, we investigate the sensitivity of our approach to several of the assumptions in our model.

5.5.1 Summary of Results

Our framework yields several interesting qualitative insights about the measurement-based admission control issues we discussed in the introduction:

- Memoryless certainty-equivalent admission control can have very poor performance due to estimation error. The target QoS overflow probability can be missed by several orders of magnitude. The impact of the estimation errors does not diminish as the system gets larger. This is in spite of the fact that the estimation errors are unbiased. There exists a fundamental asymmetry associated with the uncertainty of the flow parameters: the negative effect on QoS of an underestimation of flow parameters - and therefore of an overestimation of the number of permissible flows - far exceeds the positive effect on QoS of an overestimation of flow parameters.

- Estimation errors of different statistical parameters can have very different impact on the performance of an MBAC scheme. In the heavy traffic regime, the effect of error in estimating the mean is much more significant than the error in estimating the standard deviation.

- Flow departure dynamics have a significant impact on the performance of an MBAC scheme. The parameter $\widetilde{T_h} = T_h/\sqrt{n}$, where T_h is the average flow holding time and n the system size, defines a *critical time-scale* for which the effect of an admission error persists. This critical time-scale decreases with a shorter holding time or a bigger system because flows can leave the system more rapidly to repair a wrong decision.

- A high flow arrival rate has a detrimental effect on the performance of an MBAC scheme. A robust MBAC not only has to make sure that the estimation error for *each* decision is small, but also that the *worst* estimation error over the critical time-scale is small. Thus, a memoryless scheme which makes decisions based only on estimating *current* bandwidths is not robust; if the traffic correlation time-scale is short compared to the critical time scale $\widetilde{T_h}$, then the bandwidth estimates fluctuate too wildly.

- Increasing the amount of memory in the estimator reduces the overflow probability in two ways. First, the individual bandwidth estimates are more accurate because of averaging over a larger number of samples. Second, it smoothes the bandwidth estimates so that they fluctuate less over time. This provides more control to the worst estimation error over the critical time-scale.

These insights are obtained from our analysis, which culminated in *explicit formulas* for evaluating the performance of MBAC schemes in terms of key parameters such as estimator memory size, traffic correlation time scale and average flow duration. Specifically, the main results are the general formula (5.38) for the overflow probability, and the formula (5.40) specialized to the regime of separation of flow and burst time-scales Moreover, formula (5.41) yields the impact of a more conservative MBAC scheme on the utilization of the system, and, together with the previous formulas on overflow probability, quantifies the tradeoff between estimator memory size and the conservativeness of the MBAC for a given target QoS.

5.5.2 Simulation Results

We use RCBR (Renegotiated Constant Bit Rate [48]) traffic sources, i.e., the traffic rate produced by a source is constant over time intervals. Rate changes (renegotiations) are source-initiated and occur only on interval boundaries. We use independent homogeneous sources where the marginal rate distribution is Gaussian with $\sigma/\mu = 0.3$. The interval lengths are i.i.d. following an exponential law with mean T_c, which implies that the auto-correlation function of the traffic rate process is precisely as in (5.32).

We simulate the admission controller under infinite load and we measure the resulting overflow probability p_f. We terminate simulations when (a) the 95% confidence interval is less than +/- 20% of the estimated mean, or (b) the estimated mean plus the confidence interval is at least two orders of magnitude below the target overflow probability. The latter criterion is to terminate simulations within a reasonable time for very small p_f. In that case, we report an estimated p_f obtained by keeping track of the empirical mean $\widehat{\mu}$ and variance $\widehat{\sigma^2}$ of the aggregate bandwidth at the sample points and computing p_f as $Q(\frac{c-\widehat{\mu}}{\widehat{\sigma}})$.

We sample p_f at regular intervals of length $2\max(\widetilde{T_h}, T_m, T_c)$. This sample period is long enough to give approximately independent samples of the system, as the "memory" due to flow dynamics, estimation memory, and traffic correlation is taken into account. We also let the system initially warm up to steady state without collecting samples.

We now describe some simulations we have performed to validate the above insights. In particular, we wish to verify that our formulas can be used to perform robust measurement-based admission control. We proceed in two steps. First, we compare the overflow probability p_f obtained through simulation to the value predicted by theory. Second, we invert (5.39) to obtain an adjusted target overflow probability p_{ce} such that $p_f(T_m', T_c', p_{ce}) = p_q$. We then simulate the system with this adjusted target overflow probability in order to check if the overflow probability p_f is indeed close to the target overflow probability p_q regardless of the other parameters.

Figure 5.6 shows the overflow probability p_f as a function of the memory window size T_m. We observe that the overflow probability predicted by theory is conservative with respect to the simulated value. We attribute this offset to the assumptions in our model, such as ignoring the discreteness of the number of flows. However, the shape of the graphs correspond very well; in particular, the knee, corresponding to the value of T_m beyond which using a longer memory window size has little additional benefit, is well matched. Figure 5.7 and 5.8 demonstrate that our formulas can be used to perform robust measurement-based admission control. We see that with an adjusted overflow probability target, the actual overflow probability is slightly smaller than p_q over the whole range of parameters (cf. Fig. 5.8). Note that for small T_m, the adjusted target overflow probability p_{ce} can be very small ($< 1e-10$) with respect to the target overflow probability p_q of $1e-3$.

5.5.3 Memory Window Size and Robust MBAC

So far, we have assumed that the correlation time-scale parameter T_c and the flow holding time T_h are known. In practice, it is usually not very difficult to obtain a good estimate of the average holding time T_h of flows. On the other hand, the correlation time-scale T_c and more generally the correlation structure of the traffic is hard to estimate in practice, as realistic auto-correlation functions are more complex than a pure exponential. Therefore,

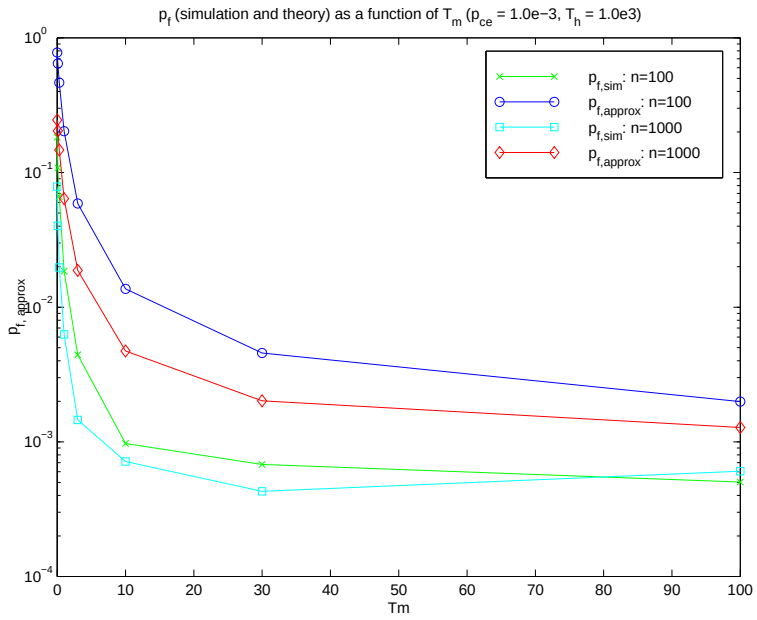

Figure 5.6: The overflow probability p_f as predicated by theory (eqn. (5.39)) and obtained by simulation ($T_h = 1000$, $T_c = 1.0$, $p_{ce} = 1.0e-3$).

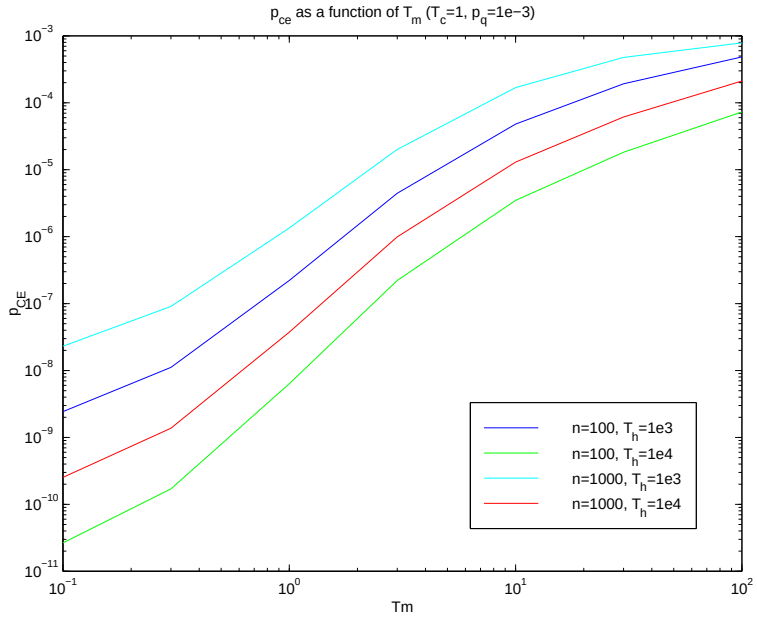

Figure 5.7: The adjusted target overflow probability by inversion of (5.39), for $n = \{100, 1000\}$, $T_h = \{1e3, 1e4\}$, and $p_q = 1.0e-3$.

we would like to design the MBAC such that its performance is good over a wide range of values for T_c. We claim that this can be accomplished by choosing the memory window length T_m on the order of the critical time-scale $\widetilde{T_h}$. For concreteness, let us pick the window size T_m to be $\widetilde{T_h}$ and examine the performance of the system for a range of T_c.

First, assume T_c is small with respect to $\widetilde{T_h}$. This is the separation of time-scale regime

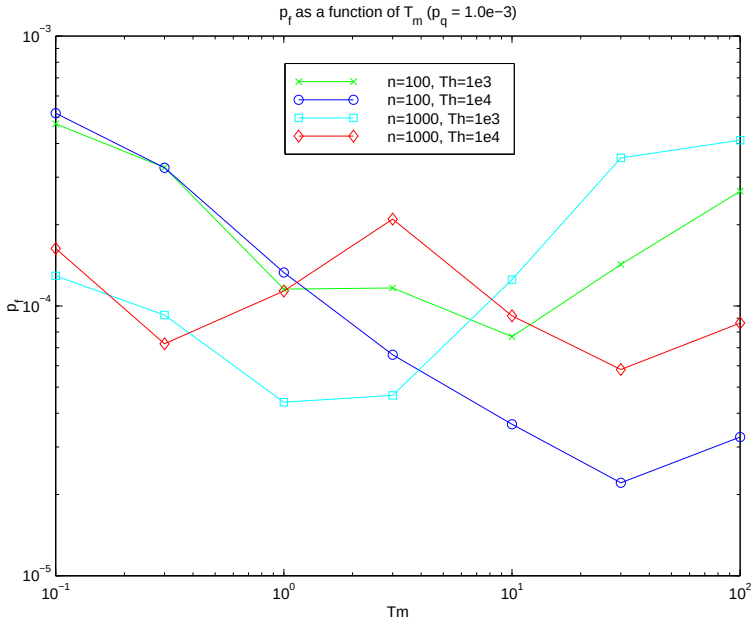

Figure 5.8: The simulated overflow probability p_f using the adjusted target overflow probability shown in the previous graph.

and formula (5.40) applies and holds for all T_m. Using the fact that $T_m = \widetilde{T_h} \gg T_c$, we get the further approximation:

$$p_f \approx \left(\frac{\sigma\alpha_q}{\mu} + 1\right) p_q \tag{5.42}$$

which is of the order of p_q. In this regime, the effect of the estimator memory effectively smoothes the fluctuations of the traffic and obtain a reliable estimate of the mean traffic rate. Although this result is derived using the simple exponential auto-correlation function (5.32), it can be easily shown that in this regime, the detailed correlation structure is not relevant and a similar approximation holds for other auto-correlation functions. We call this the *masking regime* (cf. Figure 5.9) because the memory window size masks the impact of the parameter T_c on the overflow probability p_f; the fluctuation time-scale of the mean estimator is determined by T_m alone.

Next, let us consider the other extreme, when T_c is much longer than $\widetilde{T_h}$. In this case, $\gamma = \frac{\widetilde{T_h}}{T_c}\frac{\sigma}{\mu} \ll 1$, and we have the approximation:

$$\sigma_m^2(t) \approx \frac{T_m}{T_c + T_m}.$$

Substituting this into the general formula (5.38) and evaluating the integral, we get:

$$p_f \approx \frac{1}{\sqrt{2\pi}} \frac{T_c}{\widetilde{T_h}} \frac{\sigma}{\mu} \exp\left[-\left(\frac{T_c}{\widetilde{T_h}}\right)^2 \alpha_q^2\right]$$

which definitely meets the target QoS since $T_c \gg \widetilde{T_h}$ in this regime. In contrast to the masking regime, the time-scale of the estimator fluctuation is dominated by T_c. The memory window is effectively useless in this regime, as it does not reduce estimation errors. However, the fluctuation of the estimators around their mean is at a time-scale longer than

the critical time-scale. This is precisely the regime where the repair effect makes overflow unlikely. Therefore, we call this the *repair regime* (cf. Fig. 5.9).

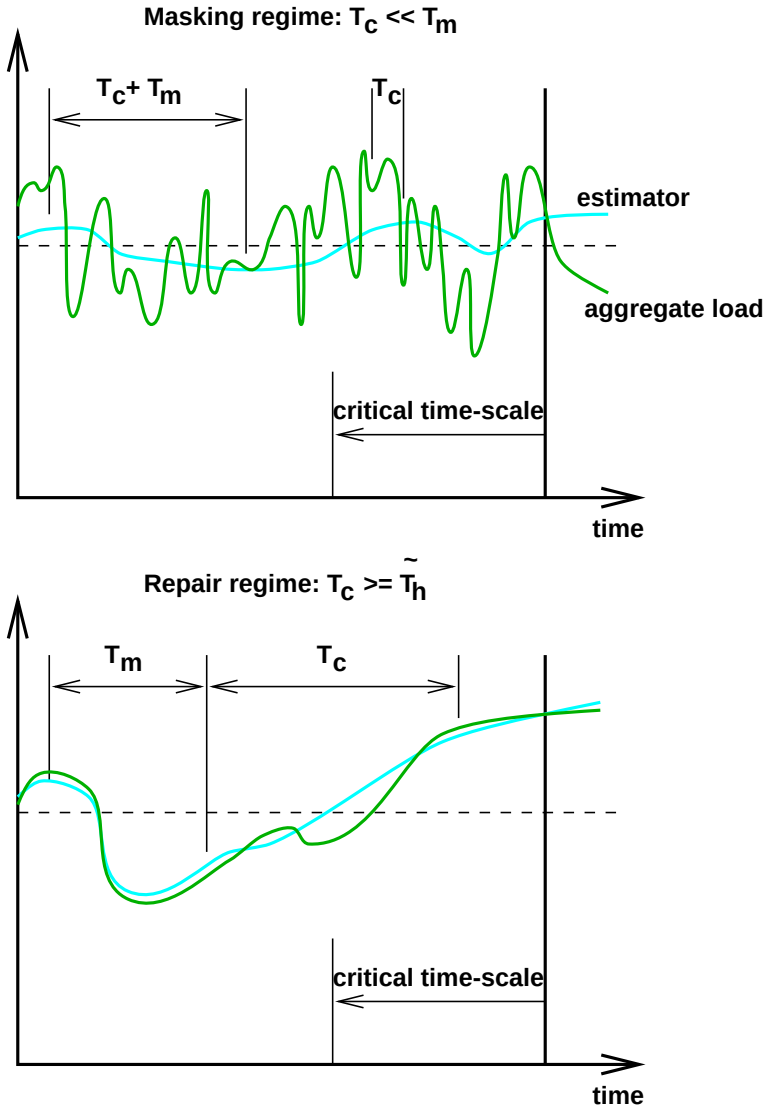

Figure 5.9: An illustration of the masking regime and of the repair regime.

For T_c in between the two extremes, there is no closed-form expression for the overflow probability, and we resort to a numerical integration of the formula (5.38) to study the performance of the MBAC. This is shown in Fig. 5.10, where we plot the overflow probability as a function of $T_m/\widetilde{T_h}$ and T_c. We see that while for small $T_m/\widetilde{T_h}$ the performance is not robust, the QoS is satisfied over a wide range of T_c once the memory window size is chosen to be a significant fraction of $\widetilde{T_h}$. This is further corroborated by simulation results using RCBR shown in Fig. 5.11.

Another way of understanding these two regimes is through a slightly different viewpoint of the MBAC. So far, we have assumed that the goal of the measurement process is to estimate stationary traffic parameters (μ and σ^2) in order to keep the number of admitted

flows as close as possible to the number of flows m^* we would admit if μ and σ^2 were known exactly. The goal of this process is to maintain enough spare bandwidth such that the target overflow probability is not exceeded. However, we have seen that flow departures reduce the time-scale over which admission errors persist to the critical time-scale $\widetilde{T_h}$. The system has a relaxation period over which it "forgets" past admission mistakes. What we effectively do by setting the memory window size to $\widetilde{T_h}$ is to let the estimators *track* the traffic fluctuation over this relaxation time-scale. This is appropriate as we only need to accurately *predict* the parameters over a time-scale of $\widetilde{T_h}$. Longer-term fluctuations are implicitly absorbed by the system.

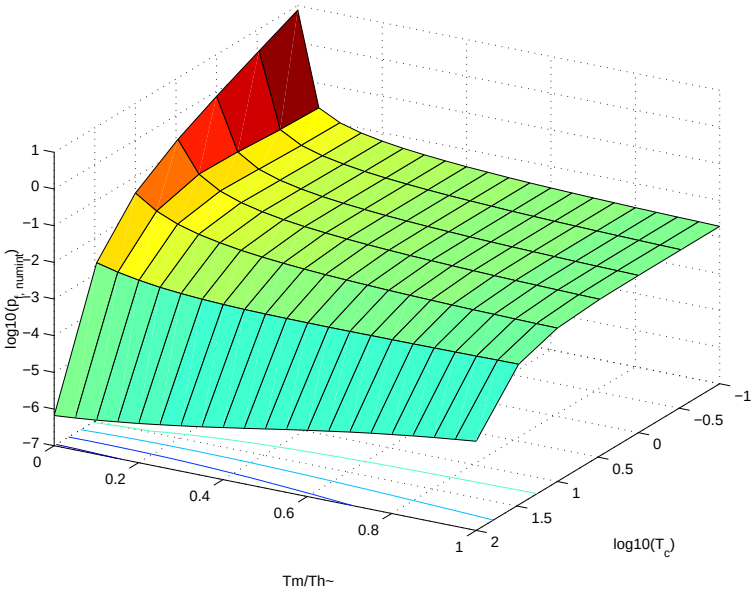

Figure 5.10: The overflow probability p_f obtained by numerical integration of (5.38), as a function of the normalized memory window size $T_m/\widetilde{T_h}$ and of the correlation time-scale T_c.

The above analysis and simulations are based on a traffic model with correlation at a single time-scale. In practice, traffic fluctuations may occur at multiple time-scales. In particular, several studies of various types of network traffic have found phenomenon of long-range dependence (LRD) [73, 38, 7, 20]. However, based on the intuition gained from the single time-scale model , we expect that a memory window size on the order of $\widetilde{T_h}$ is again appropriate here. As before, flow departures dictate a critical time-scale $\widetilde{T_h}$ over which the statistics of the future behavior of the traffic has to be predicted. A memory window of $\widetilde{T_h}$ allows the simultaneous *smoothing* of the fluctuations faster than $\widetilde{T_h}$ for reliable estimation and the *tracking* of fluctuations at a time-scale larger than $\widetilde{T_h}$. The statistics of the long-term fluctuations of long-range dependence is therefore irrelevant.

To provide some support for this hypothesis, we present simulation results on an actual traffic trace. Figure 5.12 and 5.13 show the overflow probability when the flow is a piecewise CBR version of the MPEG-1 encoded Starwars movie [48]. This particular trace has been shown to exhibit long-range dependence [38]. We vary the average flow holding time and plot the overflow probability as a function of $1/\widetilde{T_h}$. As with the synthetic traffic above, we see that the performance is not robust under memoryless estimation. When $\widetilde{T_h}$ is large (corresponding to small T_c in Fig. 5.10), the performance misses the target by 1 or 2 orders of magnitude. On the other hand, we note that with the choice of memory window

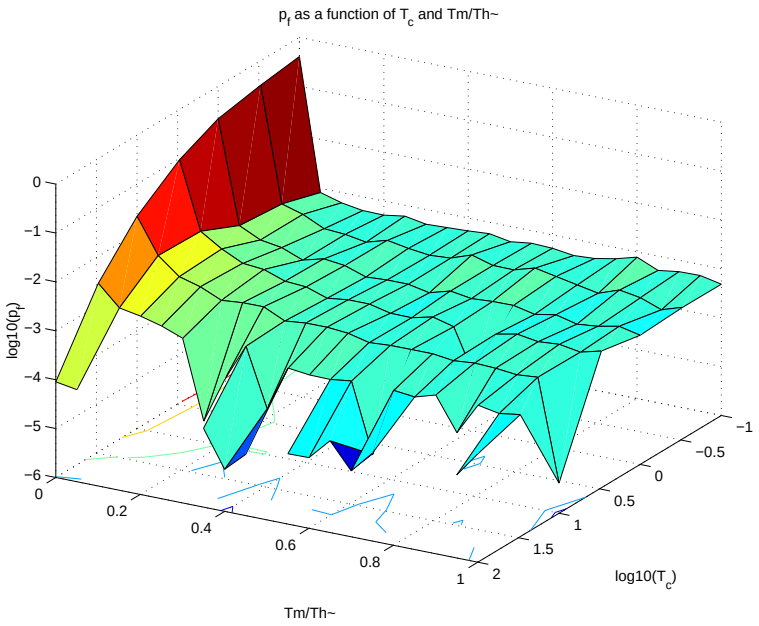

Figure 5.11: The simulated p_f over the same parameter range as in Fig. 5.10.

size $T_m = \widetilde{T_h}$, the MBAC is robust (cf. Fig. 5.13). Apparently, the strong long-term fluctuations of this traffic do not degrade the performance of the MBAC.

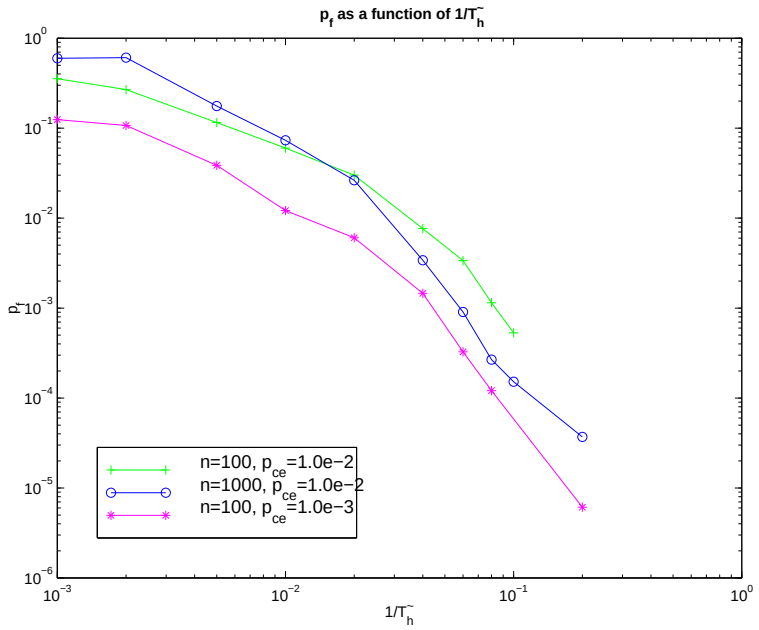

Figure 5.12: The overflow probability for Starwars sources with memoryless estimation ($T_m = 0$).

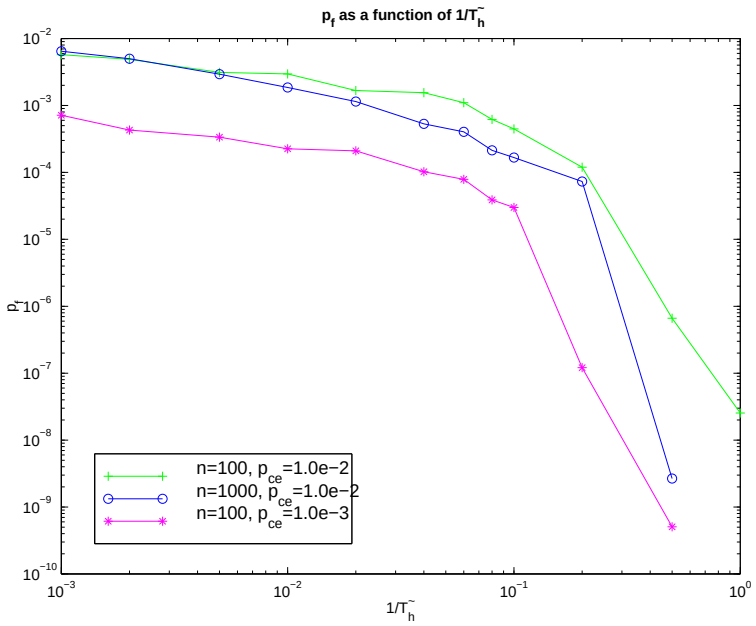

Figure 5.13: The overflow probability for Starwars sources with $T_m = \widetilde{T_h}$.

5.5.4 Heterogeneous Flows

The approach taken in this chapter is to use a Gaussian approximation (justified by a heavy-traffic limit) for the *aggregate* flow and focus on estimating the mean and variance of the aggregate fluctuation. We did however make the assumption of *homogeneous* individual flows, each with same mean μ, variance σ^2 and mean holding time T_h. A straightforward extension of our analysis shows that the results carry through for the situation when the average holding times of the flows are different, with the critical time-scale $\widetilde{T_h}$ given by the reciprocal of the departure rate averaged over all flows in the system.

Heterogeneous flows with different mean bandwidth requirements has also no effect on the mean estimator used in this chapter, provided that a Gaussian approximation of the aggregate is still valid. This is because the mean estimator only makes use of the aggregate and not the individual rates anyway. However, this is not true for the *variance* estimator, as it makes use of the individual rates in the estimate. In fact, it is not difficult to show that the variance estimator (5.7), which treats each flow as though they have the same mean, is always biased when the flows have different mean and over-estimates the variance. Thus, in a heterogeneous environment, the scheme proposed here will be conservative, leading to some loss in utilization. However, it is robust with respect to the homogeneity assumption. Of course, if classification of the flows is available to the MBAC, one can modify the variance estimator, using a different mean estimate for each class.

5.6 Related Work

Past work on measurement-based admission control [19], [99], [57] have either ignored measurement errors or assumed a static situation where calls do not arrive or depart the

system and there is arbitrarily long time to make accurate measurements. Here we discuss two more recent papers which are closer in spirit to our work.

Jamin *et al.*, in [64], presented a specific algorithm for measurement-based admission control of predictive traffic, and evaluated its performance through simulation. The algorithm relies on measurements of the maximum delay and maximum bandwidth over a measurement interval. There are several parameters in the algorithm (sampling window size S, measurement window size T, utilization target, back-off factor λ) that are found to have a significant impact on performance. However, clear guidelines on how to set these parameters are lacking. We believe that our work offers some insight into the impact of these system parameters. In particular, the measurement window size T is very similar to our measurement time-scale T_m. Also, λ is a parameter that controls an *overestimation* of the actual measured delay - in other words, it controls conservativeness, which in our work is represented through the parameter p_{ce}. Therefore, while the details of the models and metrics are not exactly identical, we think that our work helps understand the issues that govern the tuning of the above parameters. Our work has the further advantage that we use a much simpler service model so that we can focus on the issues associated with the measurement process.

Gibbens *et al.* [40] studied *memoryless* measurement-based admission control in a decision-theoretic framework. Their work takes into account the impact of measurement errors on performance and also considers the call dynamics. However, there are some significant differences between theirs and our work. First, a perfect time-scale separation is explicitly built into their model by assuming that the network states seen by successive call arrivals are independent. This makes it difficult to evaluate the performance of MBAC schemes with memory and also the effect of traffic correlation on a system with very high call arrival rates. Indeed they only focused on *memoryless* schemes. Moreover, our results show that the condition for time-scale separation is rather subtle, as it depends, among others parameters, on the system size. Second, while they also observed that a memoryless certainty equivalent scheme can perform poorly, their remedy is quite different. They relied on essentially two mechanisms: the use of a Bayesian prior on the call statistics and network state-independent call rejection. The first mechanism serves to smooth out the fluctuation in successive memoryless estimates, as the observations are weighted by a fixed prior. The second mechanism counters very high call arrival rates, by not accepting calls until one has left the system. In contrast, we propose the use of an appropriate amount of memory in the estimator , which as we have seen deals with both these problems. Our framework, without *a priori* assuming time-scale separation, allows us to evaluate the performance as a function of the amount of memory used. We believe the appropriate use of memory is a natural and effective strategy, particularly when no reliable prior exists.

5.7 Conclusions

Measurement-based Admission Control simplifies the contract between the user and the network, at the expense of having to deal with additional uncertainty in the system. The benefit of relieving the user of the burden of a-priori traffic specification, and of relieving the network of the burden of policing, far outweighs the costs of this uncertainty, if it can be prevented from compromising the quality of service experienced by the user. This problem has motivated the present work.

In this chapter, we have presented a framework for studying the performance of admission control schemes under measurement uncertainty and flow dynamics. Using heavy-traffic

approximations, the analysis of the resulting dynamical systems is simplified via linearization around a nominal operating point and by Gaussian approximations of the statistics via central limit theorems. We believe that the insight derived from our models, and the engineering guidelines on the choice of memory and certainty-equivalent target overflow probability, should be directly applicable in the design of robust MBAC schemes. However, there are also some additional issues that merit attention.

First, there is increasing interest in *adaptive* applications, i.e., applications that are capable of functioning properly even if the QoS falls below the desired level [16]. This interest stems from the inability of the current Internet to guarantee any level of QoS. The QoS metric used here, i.e., the probability that a flow cannot get at least its target bandwidth at time t, is extreme in the sense that it does not account for the fact that getting part of that target bandwidth is still useful to an adaptive application. We are therefore working on a generalization of the QoS metric based on utility functions, inspired by Shenker's work [102]. The goal is to assess the impact of application adaptivity on the admission problem.

Second, we have assumed that *individual flows* are available for measurement. This might actually not be desirable or feasible in practice. Aggregate measurements can be expected to be easier to implement, because no per-flow information has to be maintained. While using only aggregate measurement does not affect the mean estimator, the accuracy of the variance estimator is hampered without per-flow information. We plan to study the effect on QoS of having only aggregate estimates available.

5.8 Appendix A: Proof of Proposition 2

We use the notations $\stackrel{\mathcal{D}}{\rightarrow}$ and $\stackrel{a.s.}{\rightarrow}$ to denote convergence in distribution and almost sure convergence respectively. The following theorems are standard results in the theory of convergence in distribution.

Theorem 4 *(Continuous-Mapping Theorem) Let $\{\vec{Y}^{(n)}\}$ be a sequence of random vectors on $\Re^k$. If $h : \Re^k \rightarrow \Re$ is continuous and $\vec{Y}^{(n)} \stackrel{\mathcal{D}}{\rightarrow} \vec{Y}$, then $h(\vec{Y}^{(n)}) \stackrel{\mathcal{D}}{\rightarrow} h(\vec{Y})$.*

Theorem 5 *Let $\vec{Y}^{(n)}$'s and $\vec{Z}^{(n)}$'s be random vectors defined on the same probability space. If $\vec{Y}^{(n)} \stackrel{\mathcal{D}}{\rightarrow} \vec{Y}$ and $\vec{Z}^{(n)} \stackrel{a.s.}{\rightarrow} \vec{a}$ where $\vec{a}$ is a constant vector, then $(\vec{Y}^{(n)}, \vec{Z}^{(n)}) \stackrel{\mathcal{D}}{\rightarrow} (\vec{Y}, \vec{a})$.*

Proof of Proposition 2:

For each system size n, let $\hat{\mu}_n$ and $\hat{\sigma}_n$ be the estimates of the mean and standard deviation of the bandwidth distribution of the flow, respectively. By definition of the MBAC,

$$M_0^{(n)} = \frac{1}{4\hat{\mu}_n^2}\left(\sqrt{\hat{\sigma}_n^2\alpha_q^2 + 4n\mu\hat{\mu}_n} - \hat{\sigma}_n\alpha_q\right)^2$$

which is obtained by solving eqn. (5.6) for each n. Thus,

$$\frac{M_0^{(n)} - n}{\sqrt{n}} = \frac{\sqrt{n}(\mu - \hat{\mu}_n)}{\hat{\mu}_n} + \frac{\hat{\sigma}_n^2\alpha_q^2}{2\hat{\mu}_n^2\sqrt{n}} - \frac{\hat{\sigma}_n\alpha_q}{2\hat{\mu}_n^2}\sqrt{\frac{\hat{\sigma}_n^2\alpha_q^2}{n} + 4\mu\hat{\mu}_n}$$

By the strong law of large numbers $\hat{\mu}_n \stackrel{a.s.}{\rightarrow} \mu$ and $\hat{\sigma}_n \stackrel{a.s.}{\rightarrow} \sigma$. For the first term above, $\sqrt{n}(\mu - \hat{\mu}_n) \stackrel{\mathcal{D}}{\rightarrow} -\sigma Y_0$ by the Central Limit Theorem, where $Y_0 \sim N(0,1)$. Also, $\hat{\mu}_n \stackrel{a.s.}{\rightarrow} \mu$

and hence by theorems (5) and (4) above, the first term converges to $-\frac{\sigma}{\mu}Y_0$ in distribution. The second term converges almost surely to 0, while the third term converges almost surely to $-\frac{\sigma\alpha_q}{\mu}$. Applying the above theorems we now get the desired result:

$$\frac{M_0^{(n)} - n}{\sqrt{n}} \xrightarrow{\mathcal{D}} -\frac{\sigma}{\mu}(Y_0 + \alpha_q)$$

■

5.9 Appendix B: Computation of $\sigma_m^2(t)$

We have

$$\sigma_m^2(t) = \mathrm{E}\left[(Z_{-t} - Y_0)^2\right], \tag{5.43}$$

where $Z_t = Y_t * h_t$, and $h(t) = 1/T_m \exp(-t/T_m)\mathrm{I}_{t\geq 0}$.

Define $V_s = Z_{s-t} - Y_s$. Let $g(s)$ be the impulse response of the filter with input Y and output V.

$$g(\tau) = -\delta(\tau) + \frac{1}{T_m}\exp\left[-\frac{\tau - t}{T_m}\right]\mathrm{I}_{\tau\geq t}. \tag{5.44}$$

Then $\sigma_m^2(t) = \sigma_V^2 = \rho_V(0)$, where $\rho_V(s) = \rho_Y(s) * r_g(s)$ and $r_g(s) = g(s) * g(-s)$. Let us compute $r_g(.)$.

$$\begin{aligned}
r_g(s) &= \int_{-\infty}^{\infty} g(\tau)g(\tau - s)d\tau && (5.45)\\
&= \delta(s) - \frac{1}{T_m}\left\{\exp\left[-\frac{-s-t}{T_m}\right]\mathrm{I}_{-s\geq t} + \exp\left[-\frac{s-t}{T_m}\right]\mathrm{I}_{s\geq t}\right\} && (5.46)\\
&\quad + \frac{1}{T_m^2}\exp\left[-\frac{-s-2t}{T_m}\right]\int_{t+\max(0,s)}^{\infty}\exp\left[-\frac{2\tau}{T_m}\right]d\tau && (5.47)\\
&= \delta(s) + \frac{1}{T_m}\left\{\frac{1}{2}\exp\left[-\frac{|s|}{T_m}\right] - \mathrm{I}_{|s|\geq t}\exp\left[-\frac{|s|-t}{T_m}\right]\right\} && (5.48)
\end{aligned}$$

Let us compute $\sigma_m^2(t)$ when the autocorrelation function of the process Y_t is exponential with time-constant T_c, i.e.,

$$\rho_Y(s) = \exp\left[-\frac{|s|}{T_c}\right]. \tag{5.49}$$

$$\begin{aligned}
\sigma_m^2(t) &= \rho_V(0) = \int_{-\infty}^{\infty}\rho_Y(s)r_g(s)ds && (5.50)\\
&= 1 + \frac{1}{T_m}\int_0^{\infty}\exp\left[-\frac{s}{T_c} - \frac{s}{T_m}\right]ds - \frac{2}{T_m}\int_t^{\infty}\exp\left[-\frac{s}{T_c} - \frac{s}{T_m} + \frac{t}{T_m}\right]ds && (5.51)\\
&= \frac{2T_c + T_m}{T_c + T_m} - \frac{2T_c}{T_c + T_m}\exp\left[-\frac{t}{T_c}\right] && (5.52)
\end{aligned}$$

Chapter 6

Issues in Multicast Transmission in Heterogeneous Environments

Notation

p_{loss}	link loss probability
$u(.)$	fraction of useless traffic
L	number of hops
n	number of layers
m	number of data packets
k	number of redundancy packets
B	buffer size

6.1 Introduction

The current Internet provides a simple service, namely a single class best effort service, which relies on simple mechanisms in the network, namely datagram switching and stateless FIFO queueing. The model is attractive because it facilitates the interconnection of networks with different architectures, and it provides flexible resource allocation and good reliability against node and link failure [17]. The underlying network mechanisms are attractive because they are extremely simple and they do not rely on complex control mechanisms or signaling protocols.

However, from a connection's point of view, best effort service amounts in practice to offering a channel with time-varying and hard-to-estimate characteristics such as delay and loss distributions. This makes it hard to predict or guarantee performance such as maximum delay or maximum loss rate to applications. Two approaches to this problem are possible. For clarity of presentation, let us first consider the case of unicast connections.

One approach is to adapt application behavior to the service provided by the network, i.e. to the time-varying characteristics of the channel over which the application data packets are sent. This amounts in practice to incorporating in the application, control mechanisms that attempt to minimize the negative impact of channel characteristics on the quality of the data delivered at the destination. Examples of such mechanisms include playout adjustment (to control delay jitter), rate control (to control the amount of packet loss), error recovery (to control the impact of packet loss on quality), etc. Clearly, the complexity of applications increases as applications incorporate more such mechanisms.

The other approach is to augment best effort service with other services providing various degrees of performance guarantees, i.e., providing channels with more predictable (or even fixed) characteristics. It is clear that the complexity of the control mechanisms will decrease as the predictability of the channel characteristics increases. For example, in a network providing constant bit rate (CBR) channels, applications need not include any kind of playout adjustment scheme (since delay jitter is zero) or error recovery scheme (since packets are not lost due to buffer overflow once admitted).

Thus, we observe a clear tradeoff between the complexity of the applications (or rather the complexity of the control mechanisms included in the applications) on one hand, and the complexity of the mechanisms in the network that provide a service with stricter or looser guarantees, i.e. with more or less predictable characteristics.

The balance of the tradeoff discussed above turns out to be somewhat different in the case of multicast connections. This is because even in the current Internet architecture, the multicast routing protocols do require that some per-connection state be kept in routers. Furthermore, the state is updated using protocols that essentially act as signaling protocols. The question then is whether this state information and the underlying signaling capabilities, which *do* exist even in the stateless Internet and are *required* by the multicast routing algorithms, could not be taken advantage of and used to improve the performance of multicast delivery of layered data. *More generally, the question is whether a "small" amount of state information in the network can improve overall performance, and if so how small is small enough, and how much of a performance improvement can be expected.*

We discuss this question in this chapter. We consider two issues in particular, and we illustrate our points using two kinds of service models and network architectures. The *service models* we examine are 1) the best effort model in which no state is kept in intermediate routers, and 2) a controlled sharing model with renegotiation similar to the RCBR model in Chapter 4. We pick best effort and CBR-like services because they are in a way the two extreme points in the space of service models.

The *specific issues* we examine are 1) the tradeoff between complexity at the application level (i.e. at the network edge) versus complexity inside the network, and 2) the impact on performance and architecture complexity of information related to the different flows sent over multiple multicast groups, specifically information about dependencies between flows.

6.2 Exploring the Design Space

6.2.1 Impact of Service Model on Network and Application Complexity

In this section, we study the tradeoff between application and network complexity, by comparing the functionality required in applications and in network nodes for several classes of service models[1]. We identify several core mechanisms both in the application and inside the network (cf. Table 6.2.1). Table 6.2.1 gives an overview of the mechanisms required to use and implement each class of service model. The order of the service model in the table is roughly given by the degree of "predictability" of the service to the application. To the left, best-effort service is completely unpredictable, making no implicit or explicit guarantees at all. To the right, guaranteed rate/constant bit rate (CBR) service guarantees

[1] At the time of this writing, there is no complete consensus on the future DiffServ service model yet; for the sake of this discussion, we assume that DiffServ is a service that offers a set of network-wide QoS classes, without allocating resources to individual flows.

Service model		BE	DS	VBR	RCBR	CBR
Application mechanisms	Channel estimation	●	●			
	Error control	●	●	◒		
	Traffic specification		◒	●		●
Network mechanisms and properties	Buffer management		◒	●		
	Scheduling		◒	●		
	Admission control			●	●	●
	Signalling			●	●	●
	Per-flow state			●	●	●
	Peak reservation					●

Table 6.1: Five classes of service models (BE=Best Effort; DS=Differentiated Service; VBR=Variable Bit Rate; RCBR=Renegotiated Constant Bit Rate; CBR=Constant Bit Rate) and their associated application and network functionality.

a fixed bandwidth allocated to the application, without any bandwidth sharing through statistical multiplexing.

Our main focus here is not a direct comparison of these service models; rather, we are interested in the shift in relative costs when we assume that the service is used as a multicast/multilayer service. We argue that the *incremental cost* of adding functionality to the network is smaller than in the unicast case, whereas the cost of application functionality is increased.

Let us look at application functionality first. **Channel estimation** is necessary whenever an application has no explicit knowledge, through an explicit prior reservation or through an explicit notification from the network, of the end-to-end channel characteristics such as available bandwidth or delay jitter. The channel estimation problem is harder in multicast, as the channel is now a multicast tree shared with other receivers. Furthermore, the use of multiple layers also makes it necessary to estimate the interdependence of these layers. For example, in multicast congestion control schemes that rely on receiver-initiated joins and leaves to/from layers [14, 77, 75, 114], it is not enough to estimate each per-layer channel, but the relative impact of each layer on other layers as well; this is a very hard estimation problem, and there is so far scant evidence that such approaches are feasible in large groups.

Error control encompasses several types of mechanisms to limit the impact of transmission errors such as packet loss. *Error recovery* attempts to "fill in the gap" either by retransmissions (ARQ) or through forward-error correction. While ARQ is often not feasible in delay-constrained real-time communication, the problem of determining the optimal amount of FEC information is already hard in a single-flow case [54]. While the joint optimization of FEC on multiple layers is essentially an open problem, it can be expected that its complexity is much higher, as the dimension of the parameter space increases with the number of layers. *Error resilience* is a system property that minimizes the impact of transmission errors, for example by appropriately mapping the source information onto packets to limit error propagation. This precludes the use of certain coding mechanisms such as interframe coding, as these mechanisms are too sensitive to errors; thus, error resilience is bought at the price of coding efficiency [78]. Error resilience is harder to achieve with multilayer transmission, as errors in the base layer can affect other layers as well. For example, if a frame (or a block) of a video transmission is lost in the base layer, then the corresponding information in the enhancement layer becomes useless, thereby making it

harder to limit the propagation of the error.

Traffic specification requires that the application be able to specify the expected behavior of the traffic it generates. Traffic specification of multiple layers is more difficult and expensive than specification of a single layer, because of the larger number of parameters to be determined.

Let us now look at network functionality. **Buffer management and scheduling** is not fundamentally harder in a multicast context, as they are essentially local mechanisms managing local resources. The same holds for **admission control**, which computes local schedulability of new flows demanding admission[2]. **Signalling** and **per-flow state**, on the other hand, require network-wide conventions for protocols, formats, and policy. For a signalling protocol to be useful, all the nodes in the network have to implement it. Even in the absence of an explicit signalling protocol such as RSVP [121], a node needs to be aware of packet formats and semantics in order to be able to implicitly set up meaningful per-flow state; for example, in order to perform Fair Queueing, nodes must know how they are supposed to classify packets (e.g., source-destination, destination only, type-of-service (TOS) field, etc.) The deployment of signalling and per-flow state is therefore considerably more difficult than the deployment of local mechanisms mentioned above such as scheduling and buffer management. Thus, the major hurdle when deploying a more elaborate network service really is the deployment of signalling and of per-flow state. Their absence represents the major cost[3] advantage of the best-effort service model of the current Internet.

The simplicity of the service model of the Internet, which essentially limited the exchange of control information between nodes to routing updates, was a major factor in its rapid spread. However, multicast delivery requires a signalling protocol such as DVMRP [26] anyway in order to establish, maintain, and tear down multicast trees[4]. In order to correctly forward multicast packets, nodes must know what tree a packet belongs to. For this, per-flow state is necessary in nodes. *Therefore, as signalling and per-flow state are in place anyway in a multicast capable network, the incremental cost of using this network functionality to implement a more elaborate service model may be acceptable.* This is especially true since we have observed that the use of multicasting and hierarchical coding of multiple layers considerably increases the complexity of application adaptation mechanisms such as error control, channel estimation, and traffic specification.

In summary, in a multicast/multilayer environment, the balance in the application-network complexity tradeoff might well be shifted in favor of a more elaborate service model and more complexity in the network. Therefore, we argue that the blind extension of the best-effort service model of the current Internet to a multicast/multilayer environment should be reconsidered.

6.2.2 The Notion of Resource Waste

In this section, we formalize in a simple model the main complication arising from multilayer communication. This complication is the *flow dependence* that arises between flows that represent the same information at various qualities. For example, a raw video stream can be compressed into a sequence of information flows [66]. The quality of the decoded signal

[2] Assuming that the mapping from end-to-end to local QoS is straightforward [35]

[3] By "cost", we mean not only the cost of development and of equipment, but also the "political" cost of having to establish standards.

[4] We assume *sparse mode multicast* here, i.e., the number of receivers of a typical multicast group is small w.r.t. the overall network size.

increases with the number of contiguous flows available to the decoder. However, a flow cannot be used if it is not part of a contiguous subset of flows, and is therefore effectively wasted. We focus in this section on the *resource waste* that arises if the network is not aware of these flow dependencies.

We define resource waste as a resource that is allocated to an entity (packet, flow, task etc.) that does not contribute to the "utility" of the application [102]. The reason for resource waste can be either (i) the utility for the application depends on the availability of multiple resources, all of which must be available to be useful, or (ii) the allocation of a redundant resource (e.g., when transmitting the same information in different representations multiple times, such as with FEC or simulcast of multimedia streams encoded with different quality levels).

Let us focus on (i). We can distinguish two subtypes of resource waste due to dependence.

- **Internal waste:** Internal waste arises when the network does not complete a task, but nevertheless allocates resources to it. These resources are then wasted.

  Conceptually, internal waste is due to the distributed nature of resource allocation. It could be completely avoided if resource allocation were done in a centralized manner. As an example, consider a packet that is dropped somewhere in the network; the buffer space and bandwidth consumed up to the drop location is wasted. Internal waste could in principle be avoided by the network: *the necessary information is available* (e.g., credit-based flow control, open-loop deterministic service).

- **External waste:** External waste arises when the network completes a task that is useless to the application due to external conditions. This happens when the task of which the network is aware (e.g. transmitting a packet from application to application) is completed successfully, but with no utility to the application due to these external conditions, such as the dependence between flows. External waste cannot be avoided by the network, as by definition the network has no knowledge about the external conditions.

  The following examples illustrate the concept of external waste: (i) cell vs. packet loss problem in ATM: if the transmission unit of which the network is aware is the cell, then all cells belonging to the same packet as a dropped cell become useless to the application; (ii) layered coding: the resulting flow dependence can result in flows being transported, without being useful to the application; (iii) FEC: transmitting redundant FEC packets that end up not being used, because no original packets were lost.

External waste can be transformed into internal waste by giving the network additional information about the dependence between resource requests, i.e., by giving the network partial or complete knowledge of the application utility as a function of the resources allocated to an application. For example, by making a cell-switched network aware of packet boundaries, we can avoid the transport of useless cells following the lost one.

Clearly, we would like to minimize the amount of external waste. To do this, we need to examine (i) whether external resource waste occurs easily, (ii) assess its cost when it does occur, and (iii) derive ways to prevent it.

6.2.3 The Relevance of External Waste

Below is a simple thought experiment to illustrate some of the issues arising from flow dependence. Assume that we transmit a (unicast) session through a network. The session consists of n flows $\{F_1, \ldots, F_n\}$ that depend on each other due to hierarchical coding of the source information in the sense that a packet belonging to flow i is only useful to the application if the corresponding packets on flows $1, \ldots, i-1$ have been received as well.

Suppose that the network does not know about flow dependence and that packets are dropped at random with probability p_{loss}, independent of each other, and independent of what flow the packet belongs to. Then the probability that a packet gets through L links successfully is $p = (1 - p_{loss})^L$.

We study the following situation: one packet per layer is sent. A packet arriving at the destination is useful to the destination only if all the packets on layers with smaller index have arrived as well. These n packets might represent a hierarchically encoded video frame, for example.

We are interested in the *fraction of useful traffic* arriving at the destination as a function of the per-link probability p_{loss} and the number of L of links traversed. Let $p_L = (1 - p_{loss})^L$ be the probability that a packet gets through L links. Then the expected number of the n packets making it to the destination is np_L. The number of *useful* packets making it to the destination is

$$\sum_{i=1}^{n} p_L^i = p_L \frac{1 - p_L^n}{1 - p_L} \tag{6.1}$$

as the probability that packet i gets through *and* is useful is p_L^i.

Thus, the fraction of useless or wasted traffic at the destination is[5]

$$u(L, n, p_{loss}) = 1 - \frac{1 - (1 - p_{loss})^{nL}}{n[1 - (1 - p_{loss})^L]} \tag{6.2}$$

Fig. 6.1 shows this fraction after $L = 20$ hops, as a function of the per-link loss prob. p_{loss} and n, the number of dependent layers. We can see that the fraction of useless traffic u is quite sensitive to both p_{loss} and L: for example, for $n = 10$ layers and $p_{loss} = 1.0e-2$, about 50% of the traffic arriving at the destination is useless. So although about $p_L \approx 82\%$ of the packets reach the destination, only about 40% of the original packets can be used by the application.

The underlying problem is that the network assumes a utility function, which it is (hopefully) designed to maximize, considerably different from the utility function of the application. Without knowledge of flow dependence, the network tries to optimize the wrong thing, and performs very suboptimally from the application's point of view in terms of utility w.r.t. to the resources allocated [102].

This difference in the utility functions implies that we really only have two choices: either we operate in a regime where the network and the application utility functions are close, or change the network utility function.

[5]Note that part of this useless traffic is internal waste, and part is external waste: if the network knew about flow dependence, it could drop packets belonging to layers with higher index than the packet lost, thereby avoiding the higher-index packets travel all the way to the destination; however, we would still have internal waste, due to the packets transmission up to the point where they get dropped. This is independent of flow dependence.

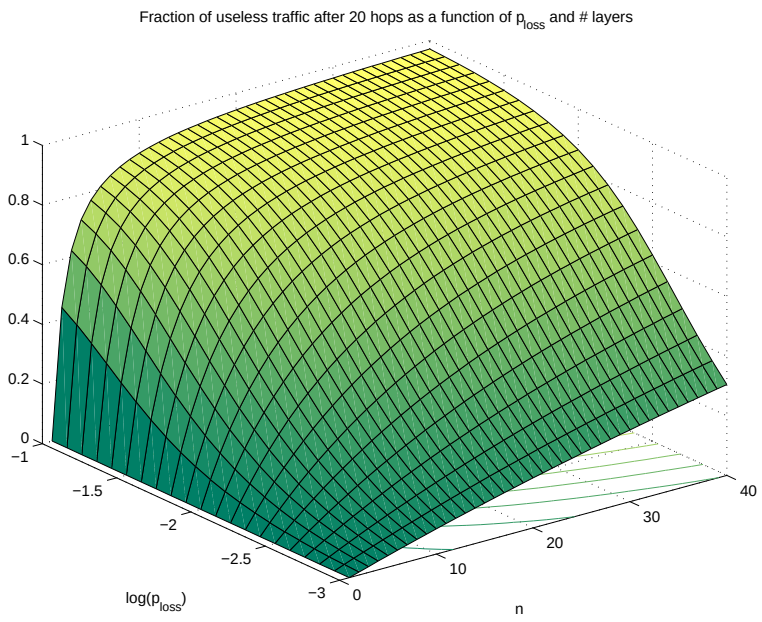

Figure 6.1: The fraction of useless traffic after $L = 20$ hops as a function of the per-link loss probability p_{loss} and the number of layers n.

In the present example, the former implies operating in the regime of very small per-link loss probabilities, thereby potentially reducing the achievable statistical multiplexing gain. The latter implies making the network aware of the utility function, and in particular, of the non-additivity of the per-flow utilities. This implies that there is no way around somehow making the network aware of flow dependence.

6.2.4 Resource Waste and Channel Estimation in the Current Internet

As noted in the introduction to this chapter, best-effort networks can *collapse* in the absence of appropriate end-to-end congestion control. Such a collapse manifests itself through high internal waste; most packets do not make it to the destination, thereby using up resources without ultimately being delivered to the application. The current Internet with its best-effort service model has no network-internal congestion control mechanism, such as hop-by-hop flow control, and no mechanism to convey *explicit congestion notification* to applications. As a result, end-to-end congestion control mechanisms have to rely solely on the observation of the end-to-end packet transmission process to know about the congestion in the network. In other words, the congestion control mechanism has to *estimate* the characteristics of the end-to-end channel.

We argue that relying on channel estimation constrains the possible operating points at which the network can function. The most prominent channel characteristic that congestion control algorithms are based upon is *packet loss*. For example, TCP views a packet loss as an indication that the network is overloaded and reacts by reducing the size of its congestion window [63]. Although TCP is not based on an a-priori specified target operating point from which a control algorithm is derived, which makes it difficult to characterize that operating point, simulation studies [103] and analytical models [72] as well as evidence from Internet measurements [88] clearly show that TCP's operating point is at rather high packet loss rates ($> 1\%$). We claim that this is not an artifact of TCP, but instead a direct

consequence of congestion control based on loss estimation.

Suppose we want to devise an end-to-end congestion control algorithm similar to that of TCP which keeps the network at a much lower packet loss rate (say, 10^{-5}). How would such an algorithm be designed? Assume that at any point in time, packets are metered out into the network at a certain rate. The congestion control algorithm then has to decide if this rate is too high or too low in order to continually adjust the rate. However, in order to make this decision with a certain confidence, the controller must be able to *observe* enough samples of the relevant channel property (here, the bottleneck bandwidth) in order to avoid making wrong decisions too often. But the number of samples required for this depends in turn on the operating point, in the sense that many fewer samples are required to reliably estimate a high loss rate than a low loss rate. For example, a TCP-like algorithm would have to observe on the order of 10^5 packets before being able to reliably decide that the current loss rate is probably lower than the desired loss rate of 10^{-5}, and that it is therefore safe to increase the packet rate. It is clear that this is not feasible in practice, as such a controller would be prohibitively slow to react to changes in network conditions.

The reason why an operating point where the packet loss rate is quite high is not problematic in TCP is that TCP only incurs internal resource waste. When a packet is missing from a TCP session, it is simply retransmitted, and the gap is eventually filled. Thus, the cost of a lost packet is roughly equal to the cost of transmitting an additional packet. However, external resource waste is much more sensitive to the loss rate in the presence of flow dependencies and when retransmission is not an option, such as with real-time video and audio communication. The example in the previous section shows that the implicit operating point of TCP would incur unacceptably high external resource waste. Other control mechanisms competing with TCP for network resources then have no choice but to operate at the same operating point; if they attempted to operate at a lower loss rate, they would be starved by TCP[6]. Thus, competition with TCP effectively precludes applications from operating at a loss rate where external resource waste is low.

6.2.5 Hiding Resource Waste

One way of getting around the above problem of excessive external resource waste at TCP's operating point is to use forward error correction (FEC) techniques to hide packet loss from the application. FEC achieves this by adding redundancy information to each block of source information. The original message can then often be reconstructed even in the presence of loss, as the redundant information can replace the lost packet.

The motivation for an application to use FEC is clear if one assumes that the application attempts to use a given fixed channel in an optimal way. However, suppose all applications sharing the network use FEC; this increases the total load of the network, and may actually worsen the channel characteristics as seen by each application. Therefore, it is not evident if the generalized use of FEC is beneficial or detrimental.

In the appendix, we discuss this problem using a simple model. In this simplified setting, we can make two interesting observations. First, if an application views the channel as *fixed*, i.e., independent of its own packet rate, then increasing the amount of FEC information appears beneficial to that application. However, if all the applications behave in the same way, then there exists an optimal amount of FEC beyond which all applications are worse off. Second, there exist regimes where this optimal amount of FEC information is actually

[6]This is one of the reasons behind the recent interest in "TCP-friendly" applications [113].

zero; in other words, it is best not to use any FEC at all. Therefore, the overall benefit of the generalized use of FEC in a network needs careful examination; its potential may have been somewhat overestimated recently [90].

6.2.6 Summary

In summary, the best-effort service model of the current Internet, combined with end-to-end congestion control based on channel estimation, appears ill-suited to efficiently support multicast/multilayer applications. Indeed, we saw that several problems arise in such an environment. From the application standpoint, the channel estimation procedure scales in complexity with the size of the tree. Furthermore, the channel is shared with other receivers, which makes it necessary to estimate the relative impact of each layer on other layers. This amounts to carrying out a very hard joint estimation procedure.

Error control procedures are also complex. The optimal use of FEC for error recovery amounts to solving a constrained optimization problem, which scales with the number of layers. Furthermore, achieving some sort of error resilience requires that carefully designed coding schemes be used that do not suffer from error proliferation.

In the next section, we illustrate that a departure from the best-effort service model can solve these problems. We present a service model and an overview of its implementation that considerably reduces application complexity.

6.3 Application: The Design of the GRCBR Architecture

6.3.1 Basic Ideas

The Renegotiated Constant Bit Rate (RCBR) service discussed in Chapter 4 simply enhances a pure guaranteed rate/constant bit rate (CBR) flow with a renegotiation mechanism that allows the source to modify the guaranteed rate. An application relying on the RCBR service retains many of the advantages it would get with a CBR service, namely a very low loss probability and no queueing delay. Inside the network, RCBR requires only small buffers and no complex scheduling disciplines. Furthermore, it has been shown that RCBR can extract most of the statistical multiplexing gain of traffic with bandwidth fluctuations on multiple time-scales (cf. Fig. 4.7).

The goal of this section is to illustrate our insight about the tradeoff between network and application complexity by generalizing the RCBR service model. We demonstrate that the Generalized RCBR (GRCBR) protocol, which requires little more network complexity in terms of state and processing than a simple multicast routing protocol, considerably simplifies the design of multicast/multilayer applications and avoids external resource waste by taking flow dependence into account.

Renegotiations are of course at the core of RCBR. Let us then first examine the semantics of renegotiation in GRCBR, keeping in mind that we require the unicast RCBR model to be a subset of its generalized version. Recall that in RCBR renegotiation succeeds only if all the links on the path as well as the receiver itself can accommodate the new requested bandwidth. We therefore extend this model to a multicast tree simply by requiring that for a renegotiation to succeed in GRCBR, all links and receivers making up the multicast tree can accommodate the new bandwidth. We call this type of renegotiation a *closed renegotiation* (**CRNG**).

We now discuss the necessity for a second type of renegotiation called *open renegotiation* (**ORNG**). In a large multicast tree, the overall renegotiation failure probability can be significantly higher than the failure probability on an individual link. This could be compensated for by requiring very low per-link renegotiation failure probabilities. However, it would limit the amount of statistical multiplexing gain, as enough spare bandwidth would have to be reserved to limit this overflow probability. Actually, one of the goals of using multiple hierarchical layers is precisely to use this spare bandwidth as efficiently as possible. One way to achieve this is to limit the "fate sharing" in the multicast tree. An open renegotiation achieves this as follows: When an open renegotiation request arrives on a link, the link attempts to reserve the requested bandwidth; if it cannot, then the link and the whole subtree fed by this link is renegotiated to zero bandwidth and effectively disconnected. The bandwidth can be reinstated and the subtree connected back with the next renegotiation for this flow.

The role of admission control is to ensure that a flow is only admitted on a link if the renegotiation failure probability after admission of the new flow remains small for the new flow and for all previously admitted and still active flows. However, we saw in the Introduction that flows carrying the enhancement layers of a hierarchically encoded video source are useful, i.e. they increase the video quality perceived by the destination, even if their failure probability is much higher than what would be acceptable for the base layer. This means that such flows could grab unused resources when available, and thus maximize network utilization and receiver utility. Therefore, we propose the following rule: If a flow has been admitted on a link, then both open and closed renegotiation can be performed on this link; insufficient unused bandwidth on this link can therefore cause a failure of a **CRNG** for this flow; If a flow has *not* been admitted on a link, then only open renegotiations (**ORNG**) can be performed on this link. Insufficient unused bandwidth can therefore affect only receivers *downstream* from this link.

This rule partitions the multicast tree into a guaranteed (G) tree rooted at the sender[7], and a set of non-guaranteed (NG) subtrees. hanging off the G-tree. A **CRNG** issued by the sender is changed into an **ORNG** when the request traverses from the G-tree into one of the NG-subtrees.

In Section 6.2.2, we discussed the advantage of making the network aware of flow dependencies to avoid external resource waste. Let us now examine how to exploit a knowledge of flow dependence in the context of the GRCBR protocol. Consider the case of an application sending hierarchically encoded data over n flows. We would like a renegotiation request for layer i to be satisfied only if those for layers $1, \ldots, i-1$ are satisfied as well. More generally, if the available bandwidth on this link is insufficient to accommodate all of the n requests, they should be satisfied in the order desired by the application. We seen then that the only functionality we have to add to the renegotiation protocol is the ability to atomically process a sequence of renegotiation requests. The semantics of such an atomic renegotiation are to first free the bandwidth held by the respective flows, and then request the bandwidth in sequential order.

The network limits the number of guaranteed flows on a link to keep the renegotiation failure probability at some small value. On the other hand, the non-guaranteed flows are in a sense "renegotiated best-effort" flows; the renegotiation failure probability of such flows may be high. Therefore, this service model requires preemptability of non-guaranteed flows

[7]We also require that if a flow has not been admitted on a link, then it will not be admitted on any link downstream from it.

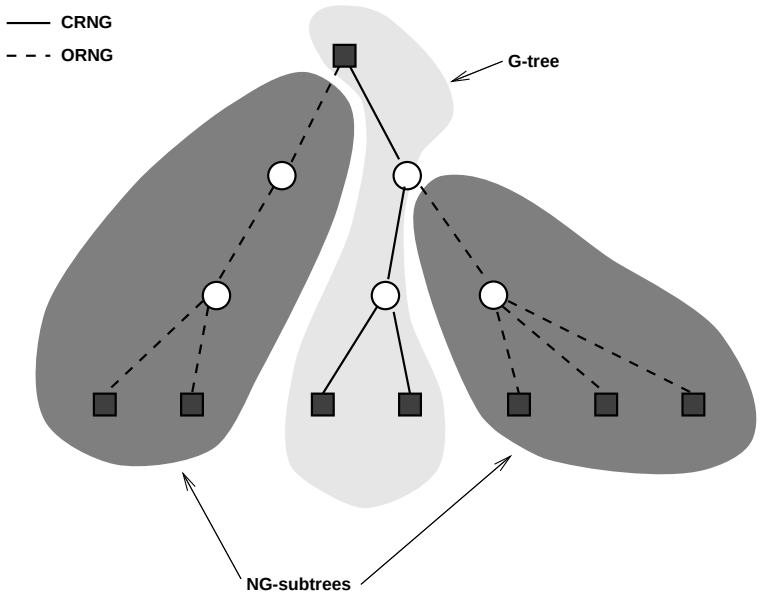

Figure 6.2: The multicast tree consists of an admitted or guaranteed part (G-tree) and a "renegotiated best-effort" part (set of NG-subtrees).

in order to respect the priority of guaranteed flows. Admission control of the guaranteed flows assumes that they can potentially use up all of the available bandwidth[8]. Therefore, when insufficient unused bandwidth is available for an (open or closed) renegotiation request for a guaranteed flow, then this bandwidth should be obtained by preempting non-guaranteed flows, if any. While the selection criterion of which NG-flow(s) to preempt depends on local policy (such as fixed priority or smart market bidding [106]), it is clear that preempting NG-flows without taking flow dependencies into account may again lead to external resource waste. Since the preemption of NG-flows happens asynchronously, it is necessary to introduce a *flow dependence state* for NG-flows to avoid external resource waste due to preemption. This state can take the form of a pointer for an NG-flow that points to a lower-priority flow belonging to the same multilayer session.

Fortunately, setting up this state does not require an additional session control signalling message. It is sufficient to rely on the bundled renegotiation requests for a set of flows to set these pointers. As the order of the flows in the bundled renegotiation request implicitly contains the flow dependence information, the pointers can be set every time a bundled renegotiation request is processed by a node. The NG-flow's dependence information is therefore maintained as *soft state* in the sense there is no need for an additional *session management protocol* to convey flow dependence information from applications to the network. When one or several NG-flows are preempted on a link, then the bandwidth they hold on the subtree fed by this link should be released.

We do not include a mechanism for reinstatement of preempted NG-flows. The reason is that in a cyclic network such as a general mesh network, there can be cycles in the resource dependency graph. Consider for example Figure 6.3. Suppose that initially, flow $A \rightarrow C$ has bandwidth 2, flow $B \rightarrow A$ has bandwidth 3, but it is initially blocked

[8] Although it is conceivable that a service provider would set the available bandwidth for guaranteed flows to less than the link capacity, in order to set aside some minimal bandwidth for non-guaranteed (renegotiated best-effort) flows.

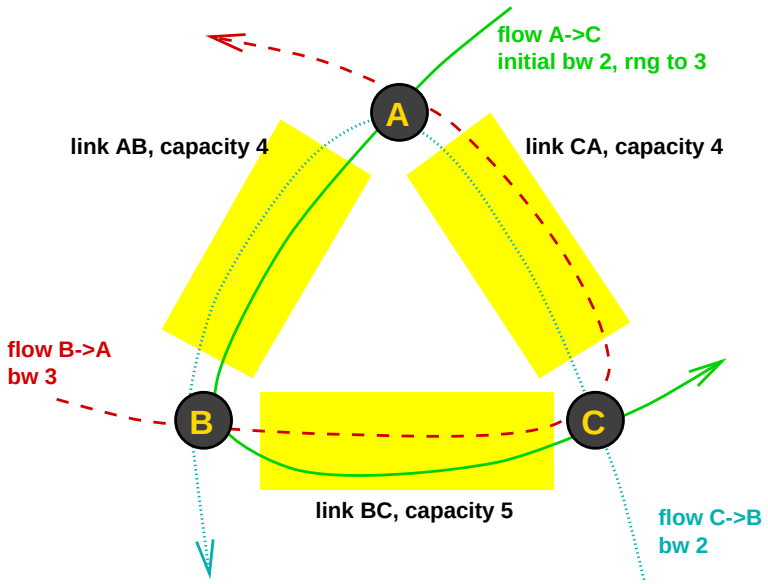

Figure 6.3: An example of a preemption/reinstatement thrashing cycle.

because of insufficient bandwidth on link CA, and flow $C \rightarrow B$ has bandwidth 2. Suppose further that a renegotiation request for flow $A \rightarrow C$ arrives at A. Now the following preemption/reinstatement cycle might result without any further renegotiation request.

1. Flow $A \rightarrow C$ preempts flow $C \rightarrow B$ on link AB.
2. Flow $B \rightarrow A$ is reinstated on link CA.
3. Flow $A \rightarrow C$ is preempted on link BC.
4. Flow $C \rightarrow B$ is reinstated on link AB.
5. Flow $B \rightarrow A$ is preempted on link CA.
6. Flow $A \rightarrow C$ is reinstated on link BC.
7. And so forth...

This example shows that without additional mechanisms such as imposing a total order on flows or centralized control of preemption/reinstatement, the cyclic resource dependency that may arise among flows sharing multiple resources in the network, *thrashing* may occur. Thrashing can dramatically increase the load of the signalling system, and decrease network throughput, as the flows involved in the thrashing are not in a coherent state most of the time[9]. Note that multicast may exacerbate the likelihood of such cycles occurring, as a preemption or reinstatement of a flow can be "seen" by a potentially large subtree. We break these resource dependence cycles by only allowing the network to preempt flows, but not to reinstate them. Reinstatement of a flow can only occur upon its next renegotiation.

[9]A flow is in a coherent state if no internal waste arises from information flowing over links without reaching receivers downstream from this link.

As renegotiations are issued by the source, and are therefore independent of preemption inside the network, resource dependency cycles cannot arise.

Let us summarize the GRCBR multicast service model. A source can send a number of flows to a set of receivers. By grouping renegotiations for these flows in the order of their relative priorities, the GRCBR service guarantees to transport a contiguous set of flows to the receivers, such that no external waste arises. For each flow, a receiver may be connected directly to the G-tree, or it may be connected to an NG-subtree. If the receiver is connected to the G-tree of a flow, then this flow is never disrupted, provided the source only issues **CRNG**s for this flow; if the receiver is connected to the NG-subtree, then it may experience disruption of the flow, depending on network congestion.

We do not discuss flow setup and teardown in detail here, as it is essentially an orthogonal issue. Let us just remark that flow setup and teardown (e.g., receiver-initiated join and leave requests) should be bundled in the same way as the corresponding renegotiations, in order to ensure that (i) the flows belonging to a given session follow the same path in the network, and (ii) flow dependence is respected in admission control (layer i is guaranteed only if layers $1, \ldots, i-1$ are guaranteed as well). Furthermore, let us remark that flow join/leave will typically occur much less frequently than in schemes that rely on receiver-initiated join/leave as a congestion control mechanism, such as those proposed in [14, 77, 75], as flow preemption and reinstatement happen implicitly in GRCBR.

6.3.2 Signalling Issues in GRCBR

The signalling protocol for RCBR is a simple two-phase reservation protocol. The first phase makes a temporary reservation on all the links it traverses in the forward direction (from the source towards the receivers). The second phase in the reverse direction (towards the source) either commits or cancels this temporary reservation. We call the signalling message in the first phase *renegotiation request* (**RNG-request**), and in the second phase *renegotiation response* (**RNG-response**).

GRCBR requires a two-phase reservation protocol similar to its unicast counterpart. However, because of the different semantics of open and closed renegotiation, the paths that signalling messages would take in the case of a renegotiation failure for both types of messages may be different. With closed renegotiation (**CRNG**), all the links in the G-tree must be able to accommodate the new bandwidth for the renegotiation to succeed and a renegotiation request terminates when it encounters a link that cannot accommodate the requested bandwidth. A negative renegotiation response originating at that link then travels back up to cancel the tentative reservation. In case of a renegotiation failure, it is also necessary to undo tentative reservations in the subtrees hanging off the path leading from the source to the failed link. For this, an additional third signalling message of type **CRNG-undo** is used.

With open renegotiation (**ORNG**), when a link has insufficient bandwidth to accept a renegotiation request, the subtree fed by this link is temporarily cut off. Of course, it may later be reinstated by the next renegotiation for this flow. The renegotiation request upon encountering such a link has to travel further down the tree, in order to release the bandwidth reservation held by this flow in the blocked subtree. For example, in Figure 6.4, if an **ORNG** for a flow on link DE fails, then forwarding for that flow on link DE is suspended. However, the bandwidth reservation on ER_5 and ER_6 must be canceled. In this case, a negative **RNG-response** should be sent upstream to release reservations that have become useless as a result (in the previous example, the link SD).

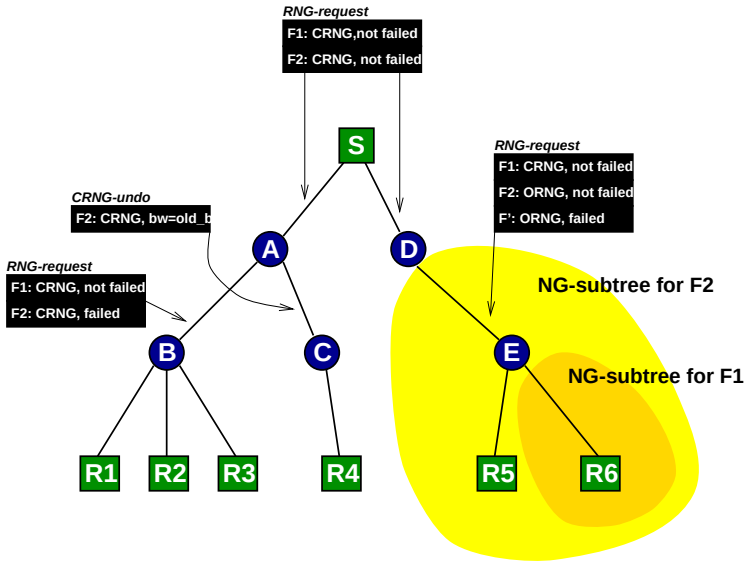

Figure 6.4: An example tree with two flows F_1 and F_2 emitted by source S, where F_1 has higher priority than F_2. A failed **CRNG** on link AB with corresponding **CRNG-undo** on link AC is shown, as well as the transformation of a **CRNG** into a **ORNG** at the boundary between the G-tree and the NG-subtree for F_2, and finally the preemption of flow F' by the **RNG-request** for (F_1, F_2) on link DE.

Unfortunately, **CRNG** and **ORNG** requests cannot follow different paths if we are to bundle renegotiations for multiple flows and process them atomically on each link in order to respect their dependencies. If a source initiates a bundled renegotiation request, possibly combining **ORNG** and **CRNG** (some of which may also be changed into **ORNG** if the corresponding flow has NG-subtrees), then the processing of this request must remain atomic throughout the tree in order to respect the priorities of the flows. Therefore, the bundled renegotiation must follow a common path that permits the correct processing of both **CRNG** and **ORNG**.

This can be achieved as follows. The renegotiation request spans the entire tree from the source to all the receivers. A renegotiation response is generated only when the request reaches a receiver. The responses are merged such that there is only a single (bundled) response on each link.

We now define the GRCBR signalling protocol in more detail. A node maintains the following per-flow data structures. For the upstream link:

flow= (**flow-id**, **guaranteed**, **failed**, **high-prior**, **low-prior**).

For the downstream links:

flow= (**flow-id**, **failed**, **awaiting-response**).

There are three types of messages: the renegotiation request message **RNG-request**, the renegotiation response message **RNG-response**, and the closed renegotiation cancelling message **CRNG-undo**. Each of the three types of messages (**RNG-request**, **RNG-response**, and **CRNG-undo**) contain one or several components that apply to one flow each. Such a component is defined as:

component= (**flow-id**, {**CRNG**, **ORNG**}, **bw**, **failed**).

The order of the components in the bundled renegotiation request defines the dependence

among the renegotiated flows. The first flow in the message has highest priority.

For an **ORNG** component request, the **failed** flag indicates that this is a bandwidth release message because of renegotiation blocking further upstream. A link receiving an (**ORNG**, **failed**) request therefore releases the bandwidth reservation for the corresponding flow. In the upstream direction, the response is initialized to **ok**. When the response gets to the link that caused the **ORNG**-failure, the flag in the response is set to **failed**, indicating that further bandwidth release might be possible upstream[10].

For a **CRNG** component request, the **failed** flag indicates that the existing bandwidth should not be changed, as the request has already failed upstream. The request therefore travels downstream without doing anything. As for **ORNG** the response is initialized to **ok**Ẇhen the response reaches the blocking link, the **failed** flag for the response is set. Further upstream, this flag indicates that **CRNG-undo**-messages should be generated[11].

The purpose of the **high-prior** and **low-prior** pointers is for a link to remember the dependence among flows, as defined by the last bundled **RNG-request** involving these flows, by forming a doubly linked list. This allows the correct merging of upstream bundled **RNG-response**, by ensuring that an upstream **RNG-response** is sent off only when the response for all the flows is the doubly linked list have received all their responses. The **awaiting-response**-flag is used simply to remember on which downstream links the **RNG-response** for the corresponding flow has already arrived.

When an NG-flow is preempted as a result of a **CRNG**-request by another flow for which sufficient bandwidth is not available, then the situation is as if an **ORNG**-request for this NG-flow had failed on this link. Therefore, a **ORNG**-request with a set **failed**-flag can simply be appended to the bundled renegotiation request that caused the preemption. Note that the **high-prior** and **low-prior** pointers contain the flow dependence information and allow to avoid external resource waste upon preemption of a flow. In fact, suppose a flow A is a candidate for preemption. If flow A's **low-prior** is not NIL, then the preemption of A would make A.**low-prior** useless. Therefore, we should also preempt A.**low-prior**.

Let us summarize the GRCBR scheme. The service model offers a renegotiable CBR flow from a source to a set of receivers. Some receivers can be connected to a GRCBR multicast tree only through one or several "renegotiable best-effort" (NG) links. These receivers can experience interruption in the reception of such a flow. If a receiver is connected directly to the G-tree, then it will not see interruption of the flow if the source only issues **CRNG**s.

The data path of the network is as simple as in RCBR. Network buffering is minimal, and no sophisticated scheduling algorithms are necessary. The signalling path essentially receives bundled renegotiation messages from an upstream link, forwards them to the appropriate downstream links, and awaits the responses. It merges the responses and passes them upstream. The **CRNG-undo** message is only necessary when **CRNG** fails, which is a rare event.

[10] Further release is possible on a link if all its adjacent downstream NG-links have released the reservation for this flow. This is true if all the **RNG-response**have the **failed** flag set.

[11] At each node, an **CRNG-undo**-message should be generated for each downstream link that (i) carries this flow as a G-flow, and that (ii) receives a successful response (**failed** not set).

6.4 Conclusion

We have identified several problems applications face when the best-effort service model is applied to a multicast/multilayer environment. We have shown that *external resource waste* can arise if the network is not aware of flow dependencies that arise from hierarchical encoding of multiple flows. External resource waste decreases the overall goodput of the network, and overall application utility. We have seen that external waste tends to be very sensitive to the packet loss probability. Therefore, a dependence-unaware network must stay at an operating point where the loss probability is very low in order to keep external resource waste at an acceptable level.

On the other hand, a pure best-effort service model forces applications to estimate the properties of the end-to-end channel through observations of packet losses in the packet arrival process. This requires an operating point where the loss probability is sufficiently high. This would therefore require an operating point where external resource waste is high.

Finally, we have illustrated that forward error correction (FEC) is problematic if its use is widespread; in a simple model, we have shown that there are actually regimes where using no FEC at all is best from a global viewpoint.

On the network side, we have identified signalling and per-flow state in network nodes as the major obstacles hampering the deployment of more elaborate network service models. However, with multicast delivery, both a signalling protocol and per-flow state are required anyway. Therefore, the incremental cost for deploying a more elaborate service model is considerably reduced.

These insights have motivated a service model and the associated network protocols, which minimizes end-to-end loss and delay variation, and which is able to respect flow dependencies. Thus, external resource waste is avoided. The protocol complexity is only marginally higher than that of a simple multicast routing protocol with receiver-initiated join/leave capabilities. The flow dependence state is established and maintained as soft state; therefore, no explicit session signalling to establish and tear down flow dependence state is necessary. As even under high load, flow dependencies are respected, an operating point where open renegotiation failure and NG-flow preemption is frequent is acceptable; therefore, channel estimation is not wasteful.

By deploying multicast capability in the Internet, the groundwork has already been laid for a shift in the balance between network and application complexity. It appears that this shift has gone largely unnoticed because the signalling and per-flow state is hidden in the multicast routing protocols. We hope that our proposed service model will trigger a more fearless reevaluation of the tradeoffs we have identified.

Appendix: Does FEC Make Sense to Hide Resource Waste?

We consider the following simple model. We wish to transmit a message of size n packets. This message is mapped onto $n+k$ packets, where $k \geq 0$ is the number of redundancy packets per message. We assume that an erasure code is used, so that a loss of up to k packets can be tolerated, i.e., the message can be reconstructed perfectly. If more than k packets are lost, then the message can be reconstructed only partially.

We can express this partial reconstruction as a *distortion*. We assume that if $i > k$ packets are lost, then the distortion is the same as if $n+k-i$ packets of the original message were available for reconstruction. This corresponds to assuming that the code mapping the n message packets onto the $n+k$ network packets is perfect in the sense that for any loss pattern, the remaining packets contain as much information as the best loss pattern[12].

Let p denote the fraction of packets lost. Then $(1-p)(n+k)$ packets are available for reconstruction of the message. Rate distortion theory asserts the existence of a non-increasing (and convex) rate distortion function $D(.)$ such that the distortion of the reconstructed message is given by $D((1-p)(n+k))$. Minimizing the distortion is thus equivalent to maximizing $(1-p)(n+k)$.

Let us consider the distortion as a function of k, the number of redundancy packets. First, let us consider the ideal choice of k if p is fixed, which can be viewed as finding the ideal amount of FEC for a single flow in a system that is shared among a large number of flows, such that the loss probability is not affected by this individual flow's bandwidth. As $(1-p)(n+k)$ increases monotonically with k, we see the obvious incentive of a flow to use as much FEC information as possible.

Second, let us consider the ideal choice of k if p depends on k, which can be viewed as finding the ideal amount of FEC if all flows sharing the system use the same amount of redundant information. We assume that $p(k)$ is non-decreasing, i.e., the fraction of packets lost cannot decrease if more redundant information is transmitted. Then, if $p(k) \to 1$ quickly enough if $k \to \infty$, there exists a k^* such that distortion is minimized. This k^* satisfies the condition

$$1 - p(k^*) - (n+k)p'(k^*) = 0. \tag{6.3}$$

An upper bound for k^* is given by

$$p'(k) > \frac{1}{n+k}. \tag{6.4}$$

As an illustration, we can approximate the fraction of lost packets p with the buffer overflow probability of an $M/M/1$-queue with buffer size B and "base utilization" ρ_0 (i.e. the utilization due to only non-redundant information). The overflow probability of this queue is given by

$$p = \rho^{B+1}. \tag{6.5}$$

We find that condition (6.4) for $k = 0$ becomes

$$1 - p_0 - n(B+1)\rho_0^{B+1}\frac{1}{n} < 0 \tag{6.6}$$

$$p_0 > \frac{1}{B+2}, \tag{6.7}$$

[12] Shannon's channel coding theorem asserts the existence of such codes for large n; the capacity of the memoryless erasure channel is $1-p$, where p is the loss probability.

where p_0 is the overflow probability for $k = 0$. Thus, if the overflow probability p without FEC for this queue exceeds $1/(B + 2)$, *minimum overall distortion is achieved using no FEC at all.* Note that $1/(B+2)$ is a packet loss probability frequently exceeded in Internet routers.

Chapter 7

Multicast Feedback Implosion Avoidance

Notation

d_α	round trip delay (RTD) of receiver α
t_α	timeout value of receiver α
A, B	nodes of the multicast tree
τ	set of nodes of a tree
ρ	set of leaves of a tree
ϕ, ψ	receiver allocations
ϵ	delay jitter offset
$s_A[\phi]$	offset for subtree A under receiver allocation ϕ
$T_A[\phi]$	maximum response time for subtree A under receiver allocation ϕ
$(s_A[\phi], T_A[\phi], \phi)$	element of subtree A's characteristic set

The work in this chapter has appeared in [42, 43].

7.1 Introduction

Some applications require a *reliable multicast service*, i.e., the error-free transport of information to a group of recipients. Such applications include distribution of "non fault-tolerant" information (such as software, news, or web-pages), distributed computing, such as distributed interactive simulation (DIS), or network management. We believe that the importance of this type of service will increase in the future, when distributed computing becomes commonplace.

It has been observed [107] that a receiver-based multicast protocol achieves better scalability than a sender-based one. But even a receiver-based scheme suffers from the NACK[1]-implosion problem if the losses at different receivers are correlated. This is very likely due to the resource-sharing in a multicast tree. When a packet is lost on some link, then all

[1] Negative Acknowledgment

receivers on the subtree fed by this link experience this loss.

There are two approaches to solve this problem: *structure-based* and *timer-based*. In structure-based approaches [76, 55], intermediate nodes in the tree process and combine feedback information. Timer-based approaches, such as the one discussed in this chapter, do not rely on processing by network nodes. Rather, they rely on *delayed feedback* to avoid an implosion [21, 36, 84]. They have the advantage of not requiring network support for implosion avoidance, but the potential disadvantage of higher application-to-application delays.

Another dimension in the solution space is receiver collaboration: if receivers are to collaborate, then they have to buffer correctly received data in order to be prepared to respond to other receivers' NACKs [36]. This also implies that feedback has to be multicast to all receivers, i.e. that each receiver is also a multicast source. In this work, we assume that receivers are greedy in the sense that they immediately consume data they have already received. Only the sender buffers data for possible repairs. The receivers *unicast* feedback to the source. Receiver collaboration can be undesirable because (1) the cost of maintaining copies of data at all the receivers instead of only at the source is too high, or because (2) receivers do not trust each other or want to withhold their identity from each other, or because (3) the underlying network does not offer (scalable) multipoint-to-multipoint communication.

Proposals have been made to use random NACK delay timers, which allows to reduce the expected number of NACKs [21, 105, 36]. One reason why only these heuristic approaches have been proposed is that the current Internet does not provide for any Quality of Service guarantees. Next generation networks are expected to offer a more elaborate service model with delay and delay jitter guarantees. Currently, various bodies are working on standardizing such services, e.g. the ATM Forum or the Internet Engineering Task Force (IETF).

This chapter presents a rigorous approach to the NACK-implosion problem, based on the assumption that end-to-end delay variation is bounded and known to the end-points. We compute *deterministic* timeouts as a function of the multicast tree topology and end-to-end delays. We establish an optimality criterion and present an algorithm, called DTRM (for Deterministic Timeouts for Reliable Multicast), that computes a timeout value for every receiver. In a networking environment with delay guarantees, NACK-implosion is provably avoided. The timeout computation is efficient and can be distributed, in which case neither sender nor receivers need to have knowledge of the group membership or of the complete tree topology. Each receiver only needs to know its round-trip delay to the sender.

The remainder of this chapter is structured as follows. Section 7.2 discusses related work. Section 7.3 presents the context of the problem in more detail. Section 7.4 explains how timeouts have to be set to fulfill the *single-NACK condition*, and establishes an optimality criterion. Section 7.5 presents a distributed algorithm to compute an optimal timeout allocation. Section 7.6 discusses the robustness and scalability of the algorithm, and Section 7.7 concludes the chapter.

7.2 Related Work

In [107], Towsley *et al.* present an analytical comparison of three generic reliable multicast protocols. They conclude that a receiver-based approach, where the receivers carry the burden of detecting losses and requesting retransmissions, is preferable to a sender-based

approach. They also compare two receiver-based schemes, called N1 and N2. In N1, a receiver unicasts a NACK to the sender immediately after it experiences a loss; the sender then retransmits the packet. In N2, the receiver, upon seeing a loss, waits for a random time and then broadcasts the NACK to the entire multicast group. The authors conclude that N2 performs better than N1, but unfortunately, there is no indication as to whether this advantage stems from the fact that N1 immediately responds to a loss, whereas N2 waits for possible NACKs from other receivers, or from the fact that N2 broadcasts the NACK, whereas N1 unicasts it to the sender. Furthermore, it is argued that only processing overhead, but not bandwidth overhead, is costly, which seems to apply more to LANs than to WANs.

Crowcroft and Paliwoda discuss the implosion problem in the context of group communication, where multiple servers reply to a request from a client [21]. They analyze the performance of randomly delaying responses if all members share the same transmission medium. The idea of randomly delaying feedback by receivers (slotting) and having receivers suppress redundant feedback (damping) has been introduced in the XTP protocol [105]. The same ideas are used in the SRM protocol [36]. The authors of SRM present an adaptive algorithm that adjusts the intervals from which the random timers are chosen to the network condition. This heuristic scheme seems to work well in a networking environment without guarantees. However, scalability is unfortunately not fully achieved, as each multicast group member has to maintain a delay estimate between itself and each other group member.

Papadopoulos and Parulkar [84] define a reliable multicast taxonomy around an hypothetical optimal algorithm. In their taxonomy, our approach is called *sender-controlled implosion control*. They also outline an algorithm that organizes receivers into *buckets*, based on their round trip delay, but not on topology. Before requesting retransmission upon loss, each receiver has to make sure that all the receivers in buckets with smaller round trip delays have not already requested this retransmission. The maximum number of requests can therefore be limited by limiting the maximum number of receivers in each bucket.

Two approaches for structure-based implosion avoidance are discussed in [76, 55]. The fundamental difference between the two approaches is that in [76], a group member (called *Designated Receiver*) is responsible for processing a region's feedback, while in [55], dedicated servers (called *Logging Servers*) perform this function.

The feedback implosion problem has also been addressed in a probabilistic context, such as in congestion control for video distribution [11] and under a relaxed form of multicast reliability called *multicast-to-some* [5].

7.3 Context

The advantage of timer-based approaches to the feedback implosion problem is that no network processing of feedback is necessary. The solution is entirely *end-to-end*. The network's only role is in forwarding the feedback from the receivers back to the sender.

We assume that the end-to-end delay for a packet sent from the sender to a receiver α is constant and equal to d_α^P. Also, we assume that the receiver-to-sender delay for NACKs is constant and equal to d_α^N. We call $d_\alpha = d_\alpha^P + d_\alpha^N$ the *round trip delay* (RTD) of receiver α. Furthermore, we assume that processing at the sender and at the receiver is immediate (retransmission due to a NACK packet at the source, detection of loss at a receiver upon

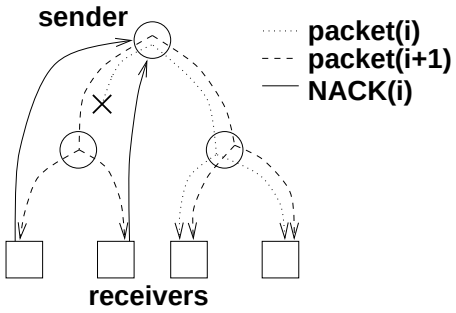

Figure 7.1: A packet loss experienced by multiple receivers results in a NACK being sent back to the sender by each of these receivers.

reception of an out-of-sequence packet).

Figure 7.1 shows how the NACK-retransmission scheme works: if packet i is lost on some link, then each receiver in the subtree attached to that link observes this loss upon reception of packet $i+1$, and starts a timer. When a receiver's timer expires, it sends a NACK(i) packet back to the sender, requesting retransmission of packet i. When a receiver that has started a retransmission timer for packet i receives the retransmitted packet i, it stops the timer without further action.

A NACK-implosion can be avoided if we can make sure that for any loss, one NACK arrives early enough such that the retransmission of the lost packet caused by this NACK prevents further NACKs from other receivers. For this, the retransmitted packet has to arrive at these other receivers before their timers expire. We refer to this requirement as the *single-NACK condition*. The implicit assumption here is that it is reasonable to prevent NACK-implosion at the cost of increased delays in the case of loss.

7.4 Computing Timeouts

In this section, we first discuss how to determine *some* set of timeouts such that the single-NACK condition is met in the entire multicast tree. Then we establish an optimality criterion. Finally, we describe a distributed algorithm to determine an optimal set of timeouts.

Let us introduce some notation. We assume the existence of a multicast tree. The root of this tree is the *sender*. Internal nodes are called *switches*, and leaf nodes *receivers*. Finally, by *downstream* we mean "towards the receivers", and by *upstream* we mean "towards the sender".

7.4.1 How to Choose Consistent Timeouts

Assume a packet is lost on a link, and call the subtree fed by this link the *loss subtree*. All receivers in the loss subtree, and only they, experience this loss. Consider two receivers α and β in the loss subtree with round trip delays d_α and d_β and timeouts t_α and t_β, respectively.

Assume that an out-of-sequence packet, i.e. the packet following the lost packet, is sent at time 0. The out-of-sequence packet reaches receiver α at time d_α^P and receiver β at time

d^P_β. The timer of receiver α therefore expires at time $t_\alpha + d^P_\alpha$, the one of receiver β at $t_\beta + d^P_\beta$. The resulting NACK from receiver α reaches the source at time $t_\alpha + d_\alpha$, and the retransmitted packet reaches receiver β at time $t_\alpha + d_\alpha + d^P_\beta$. Therefore, to ensure that only receiver α sends a NACK back to the sender when both receiver α and β experience a loss, we must have

$$t_\beta > t_\alpha + d_\alpha. \tag{7.1}$$

We say that β fulfills the single-NACK condition *with respect to* α. Given a timeout value t_α for receiver α, we enforce the single-NACK condition be setting

$$t_\beta := t_\alpha + d_\alpha + \epsilon, \tag{7.2}$$

where $\epsilon > 0$ is a small constant to allow for some delay variation without violating (7.1).

Our goal is to set timeouts for receivers in a multicast tree such that any single loss on a link only results in a single NACK coming back to the source. This can be achieved by assigning to each subtree A a *representative receiver* α with round trip delay d_α and timeout t_α such that every receiver $\beta \neq \alpha$ in this subtree A fulfills condition (7.1).

We make the following definitions.

Definition 3 *We call* subtree *A the subtree rooted at node A*[2].

Definition 4 *A* receiver allocation *ϕ for a tree is a mapping of the set τ of nodes onto the set ρ of leaves (receivers), such that $\phi(A)$ is a descendant of A for any $A \in \tau$*[3]*. The receiver $\phi(A)$ is called the* representative receiver *of node A.*

Definition 5 *A* consistent timeout allocation *(in subtree A) associates with each receiver i (in subtree A) a non-negative timeout value t_i, such that for any node B with representative receiver $\alpha = \phi(B)$, all other receivers $\beta \neq \alpha$ in subtree B fulfill the single-NACK condition (7.1) with respect to α.*

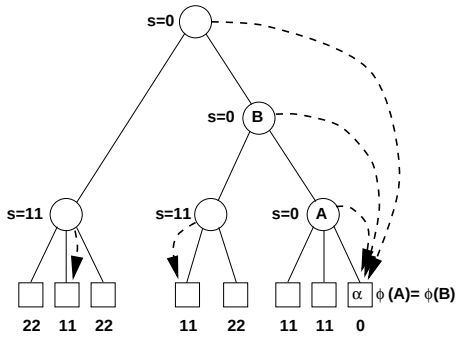

Figure 7.2: An example receiver allocation, and the corresponding consistent timeout allocation.

Figure 7.2 gives an example of a receiver allocation and the corresponding consistent timeout allocation. The round trip delay d_i has been set to 10 for all receivers i, and $\epsilon = 1$.

[2] *We use the classical definition of* subtree rooted at *A, consisting of its root A and* all *of its descendants [18, page 93].*

[3] *Note that if A is a receiver, $\phi(A) = A$.*

A dashed arrow from a node to a receiver means that this receiver is the representative receiver of this node. The numbers below the receivers are the timeout values.

Note the following property that is formalized in the lemma below. Suppose a node A is a descendant of a node B (cf. Fig. 7.2). Furthermore, suppose that B's representative $\alpha = \phi(B)$ lies in subtree A. All receivers in subtree B, and in particular all receivers in subtree A, must fulfill the single-NACK property with respect to α. Therefore, α is also A's representative.

Lemma 5 *If $\phi(B) = \alpha$, then any descendant A of B such that α is in subtree A has $\phi(A) = \alpha$.*

Proof: This follows directly from the previous paragraph. ■

Another way to put this is the following. A receiver α is a representative receiver for a consecutive sequence of ancestors up to the first ancestor that has a different representative receiver. No ancestor further upstream has α as its representative receiver.

We now give the algorithm that determines the consistent timeout allocation in a subtree A, given a receiver allocation ϕ. We need the following definitions.

Definition 6 *The* set of cotrees *of a receiver α is the set of subtrees of A not containing α, that are rooted at children of some node on the path from the sender to α.*

The following definition will be used later in the discussion of lost NACKs.

Definition 7 *The* set of dependent cotrees *of a receiver α for a receiver allocation ϕ contains those cotrees of α whose root's parent has α as its representative receiver.*

Consider Figure 7.3 for an illustration. The idea behind cotrees is the following. Assume that the timeout allocation in each cotree of α is already consistent. Then each cotree is a possible source of *one* unwanted NACK when A is the loss subtree and α is A's representative. By adding a constant offset s to each receiver's timeout within a cotree B, we can fulfill the single-NACK property with respect to α while maintaining the timeout allocation consistent in cotree B. This is the idea behind the algorithm below. Given a receiver allocation ϕ, it recursively computes this offset s to be added to every cotree of the representative receiver of every node in the tree (cf. Figure 7.2.) This results in the consistent timeout allocation in subtree A.

Algorithm 1: proc **comp (A: node, s: non-negative real, ϕ: receiver allocation)**

```
1   α:=φ(A).
2   t_α:=s.
3   for all cotrees B of α,
4       comp (B, t_α + d_α + ε, φ).
5   endfor
```

The first two steps set the timeout of node A's representative receiver $\phi(A)$ such that it respects the single-NACK condition with respect to representative receivers of A's ancestors that lie outside subtree A. The remaining steps recursively compute the consistent timeout allocations in $\phi(A)$'s cotrees, with an offset such that all the receivers in these cotrees fulfill the single-NACK condition with respect to $\phi(A)$. Therefore, given a receiver allocation ϕ, **comp (sender, 0, ϕ)** computes the consistent timeout allocation for the entire tree.

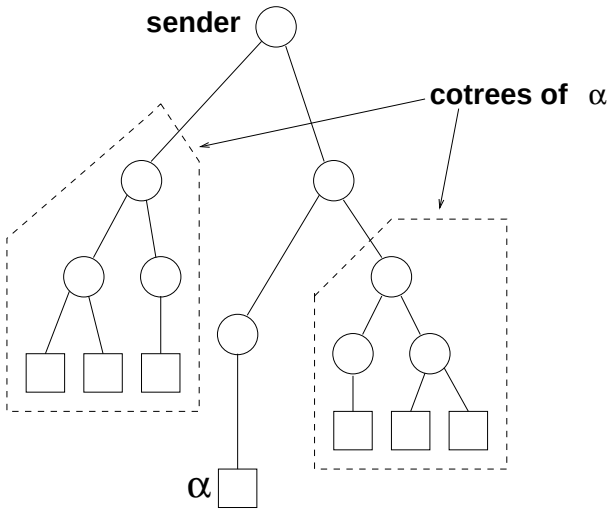

Figure 7.3: The cotrees of receiver α.

7.4.2 Optimal Timeouts

Now that we are able to compute the consistent timeout allocation given the receiver allocation ϕ, i.e., given a representative receiver for each node, we need to establish a sensible cost function to compare receiver allocations. For this, let us consider for a moment how a reliable end-to-end transport protocol based on acknowledgments (negative or positive) works. The goal of the protocol is to deliver data to the application error-free and in order. This means that the transport protocol layer buffers all data following a loss, until the lost data has been retransmitted. Only then can the data be handed over to the application. Now, in our framework, observe what happens to a receiver α in the worst case (the case where nobody else's NACK arrives at the sender earlier than α's) between the moment when the receiver realizes a packet has been lost, and the moment when this packet arrives at this receiver. The receiver runs its timer for time t_α, and then sends a NACK, which takes time d_α^N to arrive at the sender. The retransmitted packet then takes d_α^P to arrive at α. The total *response time* is $t_\alpha + d_\alpha$.

It makes sense to minimize the largest response time $t_\alpha + d_\alpha$, both to minimize the buffering necessary in the transport protocol layer, and to minimize the delay seen by the application. Furthermore, the transport layer of the sender also has to perform buffering, to allow for the retransmission of lost packets. The sender can be sure that all receivers have received a packet if it has not received a NACK after $\max\{t_\alpha + d_\alpha\}$, where we are maximizing over all the receivers in the multicast tree[4].

We therefore want to minimize

$$T = \max_{\alpha \in \rho} \{t_\alpha + d_\alpha\} \tag{7.3}$$

where ρ is the set of all receivers in the multicast tree.

[4]provided the NACK and the retransmitted packet are not lost themselves.

7.4.3 Computing an Optimal Receiver Allocation

In the previous section, we have established a cost function for receiver allocations. We now turn to the core of the problem, which is to find an efficient algorithm to compute the optimal receiver allocation, i.e., the one minimizing the cost function (7.3). This is a combinatorial problem, where the number of possible receiver allocations is potentially very large. In this section, we show that an efficient algorithm does indeed exist. It is based on the fact that only a small number of receiver allocations within any subtree can correspond to the entire tree's optimal receiver allocation. Therefore, this problem allows for a *divide-and-conquer* approach, where the possible receiver allocations within a subtree A can be computed from the possible allocations of all the subtrees rooted at A's children.

Consider a subtree A with n links and n children $A_1, \ldots, A_n$ (cf. Figure 7.4). Assume

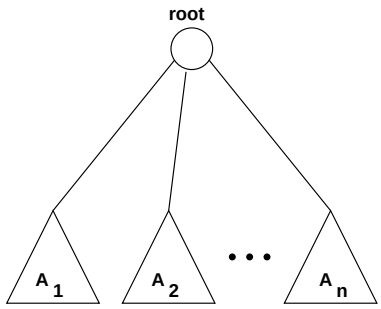

Figure 7.4: A subtree, consisting of a root, n links and n children.

that the receiver allocation *inside* the children are fixed, and that the timeout allocation inside the children are consistent, as given by **comp** $(A_i, \mathbf{0}, \phi)$.

Lemma 5 says that our only degree of freedom in choosing an allocation for A is choosing one of its children's representative as A's representative. If we choose the representative α_i of A_i as the representative of A, then we have to add an offset $s_{A_i}[\phi]$ to all the timeouts of receivers in the other children $\{A_1, \ldots, A_n\} \setminus \{A_i\}$, because these children are cotrees of α_i. The timeouts of the receivers in A_i remain unchanged.

We now make the following definition:

Definition 8 *For a node A and a receiver allocation ϕ, the triple $(s_A[\phi], T_A[\phi], \phi)$ is given by*

$$\begin{aligned} s_A[\phi] &= d_{\phi(A)} + \epsilon \\ T_A[\phi] &= \max_{\alpha \in \rho_A} \{t_\alpha + d_\alpha\} \end{aligned} \tag{7.4}$$

where ρ_A is the set of receivers of subtree A and where the t_α represent a consistent timeout allocation in subtree A.

For a subtree A, $T_A[\phi]$ is the maximum response time (i.e. the cost) of subtree A, and $s_A[\phi]$ is the offset that we have to add to A's siblings if we choose A's representative as A's parent's representative. We call the set of all triples $(s_A[\phi], T_A[\phi], \phi)$ for a subtree A the *characteristic set* of A. Note from (7.4) that $s_A[\phi]$ and $T_A[\phi]$ only depend on the

receiver allocation in subtree A. In other words, two different receiver allocations that are equal within subtree A do not give rise to two elements in the characteristic set. For ease of notation, we do not make this explicit. As an example, the following tables gives the

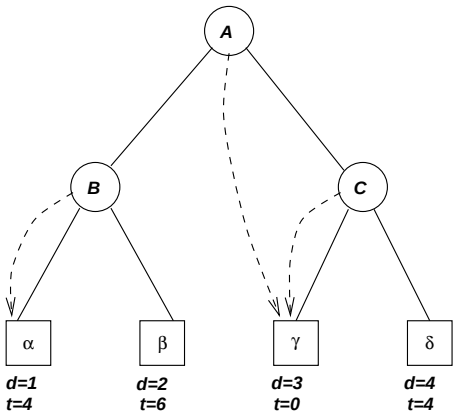

Figure 7.5: An example tree; an optimal allocation and the resulting timeouts with $\epsilon = 1$ are shown.

characteristic sets of nodes A, B and C of the tree in Figure 7.5, as well as the corresponding consistent timeout allocation. The optimal allocation shown in Figure 7.5 is framed.

$s_B[\phi]$	$T_B[\phi]$	ϕ	t_α	t_β
2	4	$B \mapsto \alpha$	0	2
3	4	$B \mapsto \beta$	3	0

$s_C[\phi]$	$T_C[\phi]$	ϕ	t_γ	t_δ
4	8	$C \mapsto \gamma$	0	4
5	8	$C \mapsto \delta$	5	0

$s_A[\phi]$	$T_A[\phi]$	ϕ	t_α	t_β	t_γ	t_δ
2	10	$A \mapsto \alpha$, $B \mapsto \alpha$, $C \mapsto \gamma$	0	2	2	6
2	10	$A \mapsto \alpha$, $B \mapsto \alpha$, $C \mapsto \delta$	0	2	7	2
3	11	$A \mapsto \beta$, $B \mapsto \beta$, $C \mapsto \gamma$	3	0	3	7
3	11	$A \mapsto \beta$, $B \mapsto \beta$, $C \mapsto \delta$	3	0	8	3
4	8	$A \mapsto \gamma$, $B \mapsto \alpha$, $C \mapsto \gamma$	4	6	0	4
4	8	$A \mapsto \gamma$, $B \mapsto \beta$, $C \mapsto \gamma$	7	4	0	4
5	9	$A \mapsto \delta$, $B \mapsto \alpha$, $C \mapsto \delta$	5	7	5	0
5	9	$A \mapsto \delta$, $B \mapsto \beta$, $C \mapsto \delta$	8	5	5	0

We now show that the characteristic set of each node can be computed from the characteristic sets of its children, which means that the optimal allocation ϕ_{OPT} can be computed by divide-and-conquer. If $\phi(A_l) = \phi(A)$, in other words, if A_l's representative is also A's

representative, then it follows from Definition 8 and Algorithm 1 that

$$\begin{aligned} s_A[\phi] &= s_{A_l}[\phi] \\ T_A[\phi] &= \max(T_{A_l}[\phi], \max_{i \in \{1,\ldots,n\} \setminus \{l\}} \{s_{A_l}[\phi] + T_{A_i}[\phi]\}) \end{aligned} \tag{7.5}$$

For a receiver α, the characteristic set contains only the following element.

$$\begin{aligned} s_A[\phi] &= d_\alpha + \epsilon \\ T_A[\phi] &= d_\alpha \\ \phi(A) &= \{A \mapsto \alpha\} \end{aligned} \tag{7.6}$$

This follows readily from Definition 8.

The previous observations can now be used to devise a recursive algorithm that finds an optimal allocation ϕ_{OPT} that minimizes $T_A[\phi]$. Any non-leaf node (switch or sender) in the tree can compute its characteristic set based solely on the characteristic sets of its children using (7.5). The optimal allocation ϕ_{OPT} is then simply the one minimizing $T_S[.]$, where S is the sender. To find each receiver's timeout, Algorithm 1 can be applied.

Let us look at the size of A's characteristic set as a function of the sizes of the children's characteristic sets. Let m_i be the size of A_i's characteristic set. From (7.5) we see that we have n choices for placing A's representative in one of A's children. Furthermore, in each child A_i, we can choose among m_i allocations. Thus, the size of A's characteristic set is

$$m = n \cdot \prod_{i=1}^{n} m_i. \tag{7.7}$$

This algorithm suffers from the combinatorial explosion of the size of the characteristic set, as expressed by (7.7). In fact, this algorithm does nothing else than explicitly enumerate all possible receiver allocations for the entire tree and compute the cost for each of these allocations. This, of course, becomes rapidly impractical as the tree size increases. In the remainder of this section, we show that only a small subset of the characteristic set, which we refer to as the *constrained characteristic set* (CCS), is necessary for each node in the computation of ϕ_{OPT}, and we find a tight upper bound on the size of this set.

Lemma 6 *Consider two elements* $(s_A[\phi], T_A[\phi], \phi)$ *and* $(s_A[\psi], T_A[\psi], \psi)$ *in the characteristic set of a node* A*, where* ϕ *and* ψ *differ in subtree* A*. Assume that* $s_A[\phi] \geq s_A[\psi]$ *and* $T_A[\phi] \geq T_A[\psi]$*. Then an optimal receiver allocation* ϕ_{OPT} *can be found without considering* $(s_A[\phi], T_A[\phi], \phi)$ *in the computation of the characteristic set of* A*'s parent*[5].

Proof: Assume that ϕ and ψ are identical outside subtree A, i.e. that for any X not a descendant of A, we have either $\phi(X) = \psi(X) \notin \rho_A$ (the set of subtree A's receivers), or both $\phi(X) \in A$ and $\psi(X) \in A$ (but possibly different). This poses no restriction on the proof, as the elements of A's characteristic set only depend on the allocation within subtree A. Let B be A's parent.

Each element in a node's characteristic set corresponds to a receiver allocation in the subtree rooted at this node. Call $(s_B[\phi], T_B[\phi], \phi)$ and $(s_B[\psi], T_B[\psi], \psi)$ the elements of B's characteristic set resulting from (7.5).

[5] *In the special case of two equalities, we need to consider one of the two elements, but not both.*

Consider the two cases where (a) B's representative is in subtree A, and (b) where it is not. In case (a), it follows from Definition 8 and the assumption in Lemma 6 that

$$s_B[\phi] = s_A[\phi] \geq s_A[\psi] = s_B[\psi]. \tag{7.8}$$

Thus,

$$\begin{aligned} T_B[\phi] &= \max(T_A[\phi], \max_{X \in c(B) \setminus \{A\}} \{s_A[\phi] + T_X[\phi]\}) \\ &\geq \max(T_A[\psi], \max_{X \in c(B) \setminus \{A\}} \{s_A[\psi] + T_X[\psi]\}) \\ &= T_B[\psi]. \end{aligned} \tag{7.9}$$

because $T_A[\phi] \geq T_A[\psi]$ by assumption and $T_X[\phi] = T_X[\psi]$ because X is outside subtree A (i.e. X is not a descendant of A), and where $c(B)$ represents the set of B's children.

In case (b), let Y be B's child such that subtree Y contains B's representative receiver $\phi(B) = \psi(B)$. Then

$$s_B[\phi] = s_Y[\phi] = s_Y[\psi] = s_B[\psi]. \tag{7.10}$$

The fact that X and Y are outside subtree A implies that

$$\begin{aligned} T_B[\phi] &= \max(T_Y[\phi], \max_{X \in c(B) \setminus \{Y,A\}} \{s_Y[\phi] + T_X[\phi]\}, \\ &\quad s_Y[\phi] + T_A[\phi]) \\ &\geq \max(T_Y[\psi], \max_{X \in c(B) \setminus \{Y,A\}} \{s_Y[\psi] + T_X[\psi]\}, \\ &\quad s_Y[\psi] + T_A[\psi]) \\ &= T_B[\psi]. \end{aligned} \tag{7.11}$$

It follows that the same property that holds for $(s_A[\phi], T_A[\phi], \phi)$ and $(s_A[\psi], T_A[\psi], \psi)$ holds for the two elements of B's characteristic set as well, namely

$$\begin{aligned} s_B[\phi] &\geq s_B[\psi] \\ T_B[\phi] &\geq T_B[\psi] \end{aligned}$$

By induction, it holds for two elements of the characteristic set of sender S. As we need to find the element $(s_S[\phi], T_S[\phi], \phi)$ that minimizes $T_S[\phi]$, it follows that there cannot be a single optimal allocation ϕ as there always exists another allocation ψ that is at least as good, i.e. has an equal or lower T_S. Therefore, to find *some* ϕ_{OPT} through (7.5), it is not necessary to consider $(s_A[\phi], T_A[\phi], \phi)$. This completes the proof. ■

The next lemma states that there exists a global optimum such that if B is any node and A is a child of B, then the receiver allocation in subtree A is locally optimal (i.e. it minimizes $T_A[\phi]$) if B's representative $\phi(B)$ is *not* in subtree A.

Lemma 7 *Assume a node A with n children $A_1, \ldots, A_n$, and assume that A_i's representative $\phi(A_i)$ is selected as A's representative. Then in order to find a globally optimal allocation, it is only necessary to consider the elements $(s_{A_j}[\phi], T_{A_j}[\phi], \phi)$ of the characteristic set of A_j, $j \in \{1, \ldots, n\} \setminus \{i\}$ that minimize $T_{A_j}[\phi]$.*

Proof: As A's representative is not in subtree A_j, $s_A[\phi] = s_{A_i}[\phi]$ is independent of the

choice of receiver allocation in subtree A_j. Also,

$$T_A[\phi] = \max(T_{A_i}[\phi], \max_{j \in \{1,\ldots,n\} \setminus \{i\}} \{s_{A_i}[\phi] + T_{A_j}[\phi]\}) \tag{7.12}$$

shows that selecting a receiver allocation ϕ that does not minimize $T_{A_j}[\phi]$ can only increase $T_A[\phi]$. Thus, selecting a ϕ that does not minimize $T_{A_j}[\phi]$ results in elements in A's characteristic set that need not be considered by virtue of Lemma 6. Therefore, if A's representative is not in subtree A_j, we only have to consider elements of A_j's characteristic set that has minimal $T_{A_j}[\phi]$. ■

Lemma 6 and Lemma 7 increase the efficiency of the recursive algorithm to find ϕ_{OPT}. Lemma 6 limits the size of the characteristic set of a node, and Lemma 7 makes the computation of a node's characteristic set from its children's characteristic sets more efficient, by limiting the number of combinations that have to be considered. We call the subset of the characteristic set that needs to be taken into account at each node the *constrained characteristic set (CCS)*.

We now prove a theorem that gives a tight upper bound on the size of the constrained characteristic set if in the recursive computation, elements are deleted at each node according to Lemma 6.

Theorem 6 *Let h denote the height of a node (and of the subtree rooted at this node), defined as the number of links on the longest path from this node to some receiver in the subtree rooted at this node*[6]*. If elements of the characteristic set of a node are deleted according to Lemma 6, then the size of this set is at most* $\max(h, 1)$.

Proof: The proof makes use of the following two lemmas.

Lemma 8 *A trivial node of height* 0 *(i.e. a receiver) has a constrained characteristic set of size* 1.

Proof: This follows readily from the fact that there is only one possible receiver allocation for a single node, as noted previously (cf. Eqn. (7.6)). ■

Lemma 9 *A tree of height* 1 *has a constrained characteristic set of size* 1.

Proof: A subtree A of height 1 consists of a node and n receivers $\alpha_1, \ldots, \alpha_n$ directly attached to this node. We can assume w.l.g. that $d_{\alpha_1} < d_{\alpha_2} < \ldots < d_{\alpha_n}$. If two round-trip delays were equal, allocating one or the other is equivalent, and it is therefore enough to consider one of them.

There are n elements in the characteristic set of this tree, corresponding to selecting one of the n receivers as the representative. If receiver α_i is chosen as representative, then the element of the characteristic set, denoted by $(s_A[\phi_i], T_A[\phi_i], \phi_i)$ can be computed from (7.6) and (7.5).

$$\begin{aligned} s_A[\phi_i] &= d_{\alpha_i} + \epsilon \\ T_A[\phi_i] &= \max(T_A[\phi_i], \max_{j \in \{1,\ldots,n\} \setminus \{i\}} \{s_A[\phi_i] + T_A[\phi_j]\}) \\ &= d_{\alpha_i} + \epsilon + \max_{j \in \{1,\ldots,n\} \setminus \{i\}} \{d_{\alpha_j}\} \end{aligned} \tag{7.13}$$

[6] *For example, the tree shown in Figure 7.2 is of height 3.*

Therefore, $s_A[\phi_1] < s_A[\phi_2] < \ldots < s_A[\phi_n]$. Also, as $\max_{j \in \{1,\ldots,n\} \setminus \{i\}}\{d_{\alpha_j}\} = d_{\alpha_n}$ whenever $i < n$, it follows that $T_A[\phi_1] < T_A[\phi_2] < \ldots < T_A[\phi_{n-1}]$. As for $T_A[\phi_n]$,

$$T_A[\phi_n] = d_{\alpha_n} + \epsilon + d_{\alpha_{n-1}} \geq T_A[\phi_1] = d_{\alpha_1} + \epsilon + d_{\alpha_1} \tag{7.14}$$

In particular, equality only results for $n = 2$. Therefore, there is only the element corresponding to allocating receiver α_1 with smallest delay as A's representative in CCS_A. ■

We have proved the assertion for the cases where $h = 0$ and $h = 1$. We are now ready to complete the proof for any h. For this, consider a subtree A and denote the receiver in A with the smallest round-trip time with $\alpha_{\min}$ (assuming it is unique.) Denote the roots of the cotrees of $\alpha_{\min}$ in subtree A with $A_1, \ldots, A_n$, as shown in Figure 7.6. We

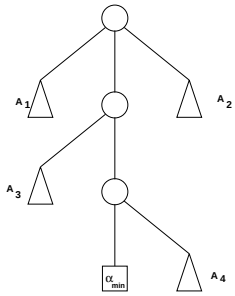

Figure 7.6: The cotrees of receiver $\alpha_{\min}$ with the lowest round-trip delay in the entire subtree.

prove the theorem in two steps. First, we show that there is exactly one element in CCS_A corresponding to allocating $\alpha_{\min}$ as A's representative. Second, we show that all other possible elements in CCS_A correspond to selecting A's representative in one particular cotree of $\alpha_{\min}$. This results in the desired bound on the size of CCS_A.

To show that there is exactly one element in CCS_A corresponding to selecting $\alpha_{\min}$ as A's representative, note that this choice corresponds to an element $(s_A[\phi], T_A[\phi], \phi)$ of CCS_A that minimizes $s_A[\phi] = d_{\alpha_{\min}} + \epsilon$, by definition of $\alpha_{\min}$. Among all allocations that have $\alpha_{\min}$ as A's representative, by virtue of Lemma 7, only one allocation minimizing $T_A[\phi]$ corresponds to an element in CCS_A.

To bound the number of other elements in CCS_A, let $T_{A_i}^{OPT}$ be the minimal $T_{A_i}[.]$ in cotree A_i. Let $T_{DC}^{OPT} = \max_{i=1,\ldots,n}\{T_{A_i}^{OPT}\}$, and the corresponding cotree the *dominant cotree* DC (assuming, for the moment, that it is unique). This is the cotree with the highest cost for its local optimum.

Lemma 7 states that if $\alpha_{\min}$ is A's representative, then a necessary condition for an allocation to be in CCS_A is that it corresponds to the locally optimal receiver allocation in each cotree of $\alpha_{\min}$. Therefore, $T_A[\phi] = s_A[\phi] + T_{DC}^{OPT}$.

Let ψ be another allocation such that A's representative $\alpha = \psi(A) \neq \alpha_{\min}$ is not in the dominant cotree DC. It follows that DC is either a cotree of α, or it is a subtree of a cotree of α. For the single-NACK property to hold with respect to α, the offset $s_A[\psi] = d_\alpha + \epsilon$ has to be added to all the timeouts in α's cotree, and in particular, to the timeouts in DC. Then $T_A[\psi]$ must be at least $s_A[\psi] + T_{DC}^{OPT}$, as by definition T_{DC}^{OPT} minimizes $T_{DC}[.]$. As $s_A[\psi] > s_A[\phi]$ and therefore $T_A[\psi] > T_A[\phi]$, ψ is not in A's CCS (Lemma 6).

Thus, A's CCS has always exactly one element corresponding to $\alpha_{\min}$ as A's representative, and possibly other elements corresponding to allocations that place A's representative α in $\alpha_{\min}$'s dominant cotree DC. For each such choice of α in DC, the same argument as before can be used to show that there is only one allocation ψ possible in A's CCS: α uniquely determines $s_A[\psi]$, and thus only the element minimizing $T_A[\psi]$ survives.

It follows that $\|\mathrm{CCS}_A\| \leq \|\mathrm{CCS}_{DC}\| + 1$. Also, height $(A) \geq$ height (dominant cotree)+1. Therefore, by induction anchored on Lemma 9, the assertion is proved.

If $\alpha_{\min}$'s dominant cotree DC is not unique, i.e. $T^{OPT}_{DC_1} = T^{OPT}_{DC_2} = T^{OPT}_{DC}$, then A's CCS has only one element, corresponding to $\phi(A) = \alpha_{\min}$. To see this, assume there are two dominant cotrees, DC_1 and DC_2. Let ψ be an allocation such that $\psi(A)$ is in DC_1. Due to DC_2, $T_A[\psi]$ must be at least $s_A[\psi] + T^{OPT}_{DC}$. Therefore, there is no element in A's CCS corresponding to ψ. This is easily extended to more than two dominant cotrees. This completes the proof. ■

7.5 Distributed Optimal Timeout Computation

This section explains how the computation of each node's CCS, and therefore the computation of the optimal receiver allocation and of the corresponding consistent timeout allocation, can be distributed. The key result from the previous section, namely the bound on the size of a node's CCS, limits the size of the messages that have to be exchanged in the distributed computation. We also show that only little state has to be maintained in nodes.

Each element of a node's CCS corresponds to a "candidate receiver allocation" in the subtree rooted at this node, i.e. an allocation that *might* be globally optimal. Suppose that a node that has computed its own CCS from its children's CCSs does not know the complete allocation function ϕ corresponding to each element of its CCS, but only (a) in what child subtree lies its own representative receiver, and (b) which among the possible candidate allocations should be chosen in each child. Then it can be seen that any node's representative receiver for some receiver allocation in this node's CCS can be determined in the following way: start at this node and note which is the candidate receiver allocation in the child containing the representative receiver. Go to this child node, which knows in which of its children is its representative receiver for this candidate application. Do this recursively until you reach the representative receiver.

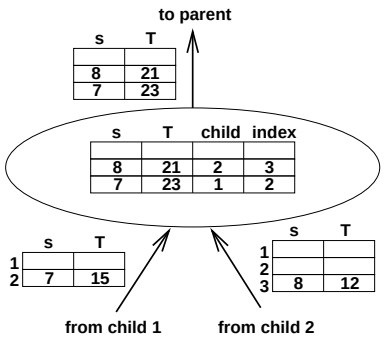

Figure 7.7: The topology information associated with the allocations in a node's CCS can be distributed.

Furthermore, Lemma 7 states that we need to know (b) above actually only for the child subtree containing the representative receiver, because the other children's allocation is locally optimal. Thus, instead of the full allocation function ϕ, we only need to associate with each CCS element (a) a pointer to the child containing the representative receiver, and (b) an index into that child's CCS. Figure 7.7 illustrates this: the node receives two CCSs from its two children, and computes its own CCS. Internally, it maintains for each element of its CCS the pointer to the child subtree containing the representative receiver, and an index into that child's CCS. It sends its own CCS to the parent. In the end, the root knows its own CCS. It can then pick the element (s_{OPT}, T_{OPT}) that minimizes the cost function T.

The downstream part of the algorithm computes the actual timeouts. Each node receives an index into its CCS and an offset from its parent. Call this offset S and the selected CCS element $(s, T, child, index)$. Each node now sends to its representative child *child* the offset S and *index*. It sends to all of its other children an offset $S + s$, and as index the one corresponding to the child's local optimum (cf. Lemma 7). If the node is a receiver, then the offset S it receives from its parent is its timeout value.

Node	child CCS	own CCS
B	$\text{CCS}_\alpha = \{(2,1)\}$ $\text{CCS}_\beta = \{(3,2)\}$	$\{(2,4,\alpha,1)\}$
C	$\text{CCS}_\gamma = \{(4,3)\}$ $\text{CCS}_\delta = \{(5,4)\}$	$\{(4,8,\gamma,1)\}$
A	$\text{CCS}_B = \{(2,4)\}$ $\text{CCS}_C = \{(4,8)\}$	$\{(2,10,B,1), \underline{(4,8,C,1)}\}$

Table 7.1: The upstream part of the distributed timeout computation.

Node	child	index	offset
A	B	1	4
	C	1	0
B	α	1	4
	β	1	6
C	γ	1	0
	δ	1	4

Table 7.2: The downstream part of the computation.

The following example illustrates the distributed algorithm. The topology of the network is depicted in Figure 7.5. Table 7.1 shows the computation each node performs in the upstream part of the protocol. The element chosen in the root's CCS as the best is underlined. Compare this to A's full characteristic set in Section 7.4.3.

Table 7.2 shows the downstream part of the protocol. The resulting timeouts are reported in Figure 7.5. It is interesting to note that the receiver with the smallest round-trip delay is not the tree's representative in the optimal allocation.

7.6 Discussion

The algorithm shown in the previous section computes an optimal set of timeouts. Theorem 6 bounds the size of the upstream messages by the height of a node. In a balanced tree with n receivers and outdegree m, this height is $O(\log_m(n))$. For example, assume that a multicast group has 4096 participants, and that each switch only has two children ($m = 2$). Then the constrained characteristic set of the root is at most of length $\log_2(4096) = 12$. Note that no node in the tree has to know the entire topology, and that communication is local: each node only communicates with its parent and its children. For general trees, the number of elements in the upstream messages can never exceed the diameter of the underlying network topology.

In this section, we discuss other aspects of scalability and robustness. We address the issues of lost NACKs and delay variation. We then examine the impact of the tree topology on the timeout values and on the response times as computed by DTRM. Finally, we discuss how to handle dynamic group membership changes.

7.6.1 Lost NACKs and Delay Variation

Let us briefly consider what happens if the unique NACK gets lost. In this case, every dependent cotree (cf. Definition 7) of the receiver having produced the NACK sends an additional NACK. This number is bounded above by $h_l(m-1)$, where h_l is the height of the loss tree and m is the largest outdegree of any node in the loss tree. In the example above with 4096 receivers, this number would be 12 if the loss tree is the entire tree. This illustrates that a NACK loss results in a reasonable number of additional NACKs, placing only a small burden on the sender.

While the underlying assumption of DTRM is that delay variation remains bounded and can be absorbed by the parameter ϵ, it is interesting to observe that the same property of graceful degradation holds for delay variation on a link (cf. Fig. 7.8).

Let us assume that on some link AB, the delay of the original out-of-sequence packet is varied by Δ_P and that of the retransmitted packet by Δ_R. Also, if the NACK originates in subtree B, let its delay be varied by Δ_N. Let L denote the root of the loss subtree. We have to distinguish three cases.

1. The loss occurs on or below AB: In this case, if $\Delta_N + \Delta_R > \epsilon$, one additional NACK is produced by each dependent cotree of $\phi(L)$. The situation is equivalent to a lost NACK.

2. The loss occurs above AB and $\phi(L)$ lies in subtree B: Let us first look at additional NACKs from within subtree B. If $\Delta_N + \Delta_R > \epsilon$, then each cotree of $\phi(L)$ in subtree B produces an additional NACK (as $\phi(L)$ is B's representative, all of these cotrees are dependent). Next, let us look at additional NACKs from outside subtree B. If $\Delta_P + \Delta_N > \epsilon$, then each dependent cotree of $\phi(L)$ outside subtree B produces an additional NACK. Thus, if both conditions are true, then there is an additional NACK from each dependent cotree of $\phi(L)$. Again, this is equivalent to a lost NACK.

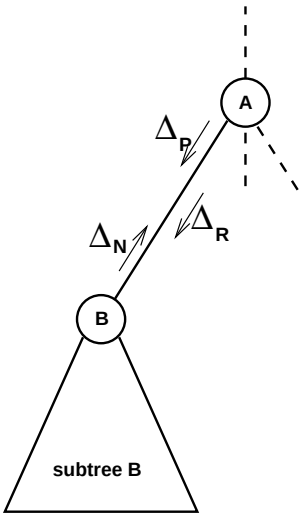

Figure 7.8: Delay variation on a link (A, B): Δ_P is the delay variation on this link of the out-of-sequence packet, Δ_N is the delay variation of the NACK packet, and Δ_R is the delay variation of the retransmitted packet.

3. The loss occurs above AB and $\phi(L)$ is not in subtree B: In this case, no additional NACKs can be produced outside subtree B. If $-\Delta_P + \Delta_R > \epsilon$, then subtree B produces one additional NACK.

Therefore, delay variation at a single link can at most produce as many additional NACKs as a lost NACK. The situation where delay variations occur on all links is hard to analyze. The number of NACKs produced per loss will likely depend on many factors, such as the distribution of delay variation, and its temporal and spatial correlation. We do not propose DTRM for such an environment of unpredictable, unbounded delay variations. It is likely that only adaptive schemes that compute timeouts based on measurements of the network condition will be sufficiently robust. It is a reasonable goal for such heuristic, dynamic schemes to get as close as possible to the performance of a provably optimal scheme in idealized conditions. Thus, we advocate the use of DTRM as a performance evaluation baseline of such heuristics. On the other hand, if the network can make some delay jitter guarantees, which is a quality of service parameter that future networks, such as ATM, are expected to support, then the desirable properties of DTRM (timeout optimality, graceful degradation) can be fully exploited.

7.6.2 Timeouts and Response Times

We study the timeouts and response times computed by DTRM in two different scenarios. First, we take a typical MBone session, with participants in the US and in Europe, and apply DTRM in order to get a feel for the timeout values obtained in a realistic setting. Second, we give a bound on the maximum response time, and we compute the average response times for balanced trees where all the receivers have equal round-trip delays.

The MBone session membership we have used is reported in [119] (cf. Table 7.3). We assume that the IP host `pax.inria.fr` at INRIA is the sender and the other 11 hosts are the receivers. We have used `ping` to measure round-trip delays between `pax` and all the receivers. We have then used `traceroute` to determine the tree topology. Note that

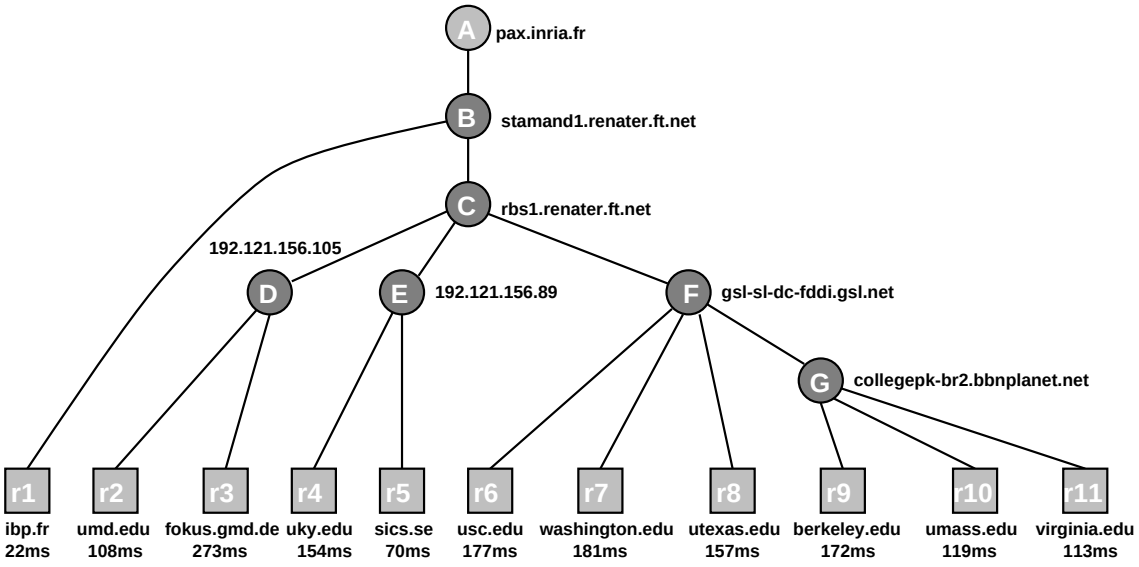

Figure 7.9: The tree topology and the round-trip delays for a typical MBone multicast session.

Host	RTD	Optimal timeout
`ibp.fr`	22ms	118ms
`sics.se`	70ms	118ms
`umd.edu`	108ms	0ms
`virginia.edu`	113ms	118ms
`umass.edu`	119ms	241ms
`uky.edu`	154ms	198ms
`utexas.edu`	157ms	241ms
`berkeley.edu`	172ms	241ms
`usc.edu`	177ms	241ms
`washington.edu`	181ms	241ms
`fokus.gmd.edu`	273ms	118ms

Table 7.3: The receivers of the MBone session, with the RTDs from `pax.inria.fr`.

`traceroute` provides us not with the MBone topology, but with the tree topology that would result if all the IP routers were multicast-capable. Given that the MBone is an overlay network to deal with old (i.e., non multicast-capable) IP routers, this approach yields more realistic topologies than simply using the current MBone tree.

We can make several interesting observations from the timeout values obtained in this setting (with $\epsilon = 10$ms). First, the largest timeout (241ms) in the tree is smaller than the largest round-trip delay[7] of 273ms. The cost of this timeout allocation, i.e. the maximum

[7]Note that he largest RTD is between INRIA, France, and the geographically second-closest site, GMD Fokus in Germany.

response time, is 422ms, for receiver `washington.edu` (181ms + 241ms). Thus, the timeout values obtained in this example group are quite reasonable. A maximum response time of less than twice the largest round-trip delay in a multicast session spanning such a large geographical area is reassuring.

However, the above group, of course, is quite small (11 receivers). We therefore finish this subsection with a discussion of upper bounds of the maximum response time, and with an expression for the average response time for balanced trees. We first obtain an upper bound on the largest timeout value t_i occurring in the tree. For this, assume an arbitrary allocation. As mentioned earlier, the timeout of each receiver is the sum of all $d_i + \epsilon$ for all receivers i that are representatives of ancestors of i. It is straightforward to observe that in a tree of height h, this sum cannot be larger than $h(d_{\max} + \epsilon)$, where $d_{\max}$ is the largest delay of any receiver in the tree. Thus, the maximum response time is bounded from above by

$$h(d_{\max} + \epsilon) + d_{\max}. \tag{7.15}$$

Again, if the tree is balanced, then $h = O(\log_m(n))$.

We now discuss the average response time, which we define as follows. Assume that a single loss occurs with equal probability on any link of the tree[8]. Then the average response time is the expected value of $d_i + t_i$, i.e. the response time of receiver i, where i is the representative receiver of the loss subtree. For the sake of exposition, assume the following balanced tree. Each switch in the tree has outdegree m. The tree is of height h and has exactly m^h receivers. Furthermore, all receivers have identical round-trip delays equal to d. Using this tree has the advantage of making all allocations equivalent in terms of the resulting timeouts. We denote the average response time by $\overline{t}_h$.

The average response time $\overline{t}_h$ can be found through a simple recursion. For a tree of height h, the average response time can be expressed as

$$\overline{t}_h = \frac{T_h}{n_h}, \tag{7.16}$$

where

$$T_h = \sum_{A \in \tau} d_{\phi(A)} + t_{\phi(A)} \tag{7.17}$$

is the sum of the response time of the representative receivers of all tree nodes, and where

$$n_h = \frac{m^{h+1} - 1}{m - 1} \tag{7.18}$$

is the number of nodes in the tree. Note that T_{h+1} can be written in the following way

$$T_{h+1} = mT_h + (m-1)n_h(d + \epsilon) + d. \tag{7.19}$$

The term mT_h is due to the fact that a tree of height $h+1$ contains m subtrees of height h. The term $(m-1)n_h(d+\epsilon)$ is due to the offset $d+\epsilon$ that we have to add to each receiver's timeout in all but one of these subtrees in order to fulfill the single-NACK condition with respect to the tree's representative. Finally, the term d is due to the fact that the tree's

[8]Including a "virtual" link feeding the root, as we do not exclude the case where the loss subtree is the entire tree.

representative (which has $t = 0$) has a response time of d. This recursion yields

$$\bar{t}_h = \frac{m^h(m-1)}{m^{h+1}-1}[d + h(d+\epsilon)] - \frac{m^h - 1}{m^{h+1}-1}\epsilon. \tag{7.20}$$

In summary, both the maximum and the average response time scale like $O(\log_m(n))$, which ensures scalability even to very large groups. Furthermore, we have observed that these bounds are quite pessimistic for heterogeneous topologies and RTD distributions. For example, note that for the tree topology depicted in Fig. 7.9, the actual maximum response time is 422ms, while $h = 5$ and $d_{\max} = 273$ms results in a bound of 1688ms. The actual average response time is 194.5ms.

7.6.3 Dynamic Membership Changes

Another concern for scalability are group membership changes. Because it must be expected that in large groups, the rate of receiver joins and leaves is high [9], their associated overhead must remain small (e.g. in terms of signalling, resource allocation, updating internal state, but also timeout computation.) We outline a solution to this problem. For this, we have to assume that after a timeout computation, all on-tree switches must maintain the following state from the last timeout computation:

1. The offset S they have received from their parent.
2. The child subtree A in which lies their representative receiver.
3. The variable s from the entry in CCS_A that was selected in A.

First, consider the case when a new receiver joins. The new receiver is attached to the existing tree at a node X. Our solution consists in assuming that this new receiver is not a representative (except for itself). This means that X can simply instruct the new receiver to use the timeout value $S + s$ to insure the single-NACK condition.

Second, consider the case when a receiver leaves. If the receiver is not a representative of any subtree, then clearly the receiver can go away without harming the single-NACK property. However, if the receiver *is* a representative of one or several ancestors, then the situation is different. Denote by C the highest ancestor for which the receiver in question is the representative receiver. Then, if we just let the receiver go away, all the cotrees of that receiver within C would produce a NACK if C were the loss subtree - the situation is equivalent to C's NACK being lost.

In other words, before we release the receiver, we need to find a new representative for C. This can be achieved by applying the distributed timeout computation locally within subtree C, while pretending that the receiver wishing to leave did not exist. Once a new set of timeouts has been computed within C, the receiver can leave.

There is no guarantee that the join and leave operations described above conserve the global optimality of the receiver allocation. However, note that their impact tends to be very locally constrained. For example, in a tree with outdegree m, the probability that a randomly picked receiver is not a representative is equal to $1 - m^{-1}$, and the probability that it is only a local representative (i.e. of its parent only) is equal to $m^{-1}(1 - m^{-1})$, etc. In other words, in practice the impact of join and leave operations has a high probability of remaining very locally constrained.

When the cost of the tree moves too far away from the optimum, it becomes necessary to recompute all the timeouts. This can be done in two ways, namely on a periodic basis, or on a demand-driven basis. In the latter case, the source could distribute a threshold cost in the downstream part of the protocol. Any receiver in the tree whose timeout exceeds this threshold would spawn the recomputation.

In conclusion, we observe that dynamic membership changes can be accommodated efficiently in DTRM. The probability that a membership change can be treated locally is high, which is an important requirement for scalable group management. The tradeoffs involved in the recomputation of the timeouts will be the subject of further research.

7.7 Conclusion and Future Work

Timer-based approaches to avoid feedback implosion in reliable multicast protocols have the advantage of requiring no network-level processing of feedback. We have presented a distributed algorithm to compute deterministic timeouts. This algorithm has several advantages.

First, the computation of the timeouts can be entirely distributed. No node in the tree (sender, switches, receivers) needs to maintain global information, and communication to compute timeouts is local between neighbors in the tree. The messages remain small even for very large multicast groups. Second, the single-NACK property is deterministic, i.e. we are guaranteed that only one NACK results from a single loss, provided that the NACK and the retransmitted packet are not lost themselves. If they are lost, then the number of additional NACKs remains acceptably small. Third, this algorithm computes a set of timeouts that is optimal with regard to the maximum response time experienced by the sender and the receivers. This maximizes the efficiency of the multicast service by minimizing the buffering requirements in the transport layer, and by minimizing the delays seen by the application. Fourth, both the maximum and the average response time scale as $O(\log_m(n))$. Dynamic group membership changes can be efficiently handled.

The usefulness of timer-based feedback implosion control is not limited to reliable multicast, of course. Any multicast protocol that relies on feedback from receivers about events occurring on links inside the multicast tree could take advantage of it. An important example is multicast congestion control, where feedback is used to reduce the source rate in case of congestion on a link.

We expect DTRM to prove useful also as a benchmarking baseline for heuristics for timeout computation, such as those designed to operate in networks without delay guarantees. The potential of these heuristics can be estimated from a comparison with this optimal algorithm.

Chapter 8

Conclusion and Outlook

In this chapter, we first summarize briefly the contributions of this thesis. In Section 8.2, we discuss extensions of this work that will provide interesting challenges for future research. We also give a more long-term vision of where we believe research will be able to make significant contributions in the field of networking and communication.

8.1 Summary of Contributions

We have addressed the problem of traffic modeling and control time-scales, motivated by the recent interest in the observation of long-range dependence (LRD) in network traffic. We have shown that the loss probability of finite buffers, contrary to the overflow probability of infinite buffers studied for mathematical tractability, is not sensitive to the presence of LRD in the traffic. This has important implications for the choice of traffic models: even if LRD is observed, traditional Markovian models may often be sufficient for prediction and decision making.

From a control point of view, a finite buffer is not a suitable control mechanism to obtain statistical multiplexing gain (SMG) from traffic exhibiting fluctuations on many time-scales. We have therefore proposed an alternative service model, which relies on a renegotiation mechanism for controlling long-term fluctuations. We have shown that RCBR can extract most of the SMG available in compressed video flows, without the need for large buffers and complex scheduling disciplines in the network. In my current work, I am extending the RCBR service model to multicast communication. Multicasting, especially in combination with hierarchical coding, is important for the scalable transmission of multimedia flows in a heterogeneous environment.

As network bandwidth is statistically shared among RCBR flows, renegotiation blocking can occur with a certain probability. In order to limit this probability, flow admission control must be exercised. Traditional approaches to the admission control problem require an a-priori traffic specification from the user or application; this is undesirable. Measurement-based admission control (MBAC) solves this problem by shifting the burden of traffic specification from the user to the network. Since traffic characteristics are now estimated from existing traffic, they are subject to estimation errors. In collaboration with D. Tse, I have analytically studied measurement-based admission control with the goal of understanding the impact of this estimation error on the quality of service, and have been able to obtain useful engineering guidelines for the design of robust MBAC schemes [49].

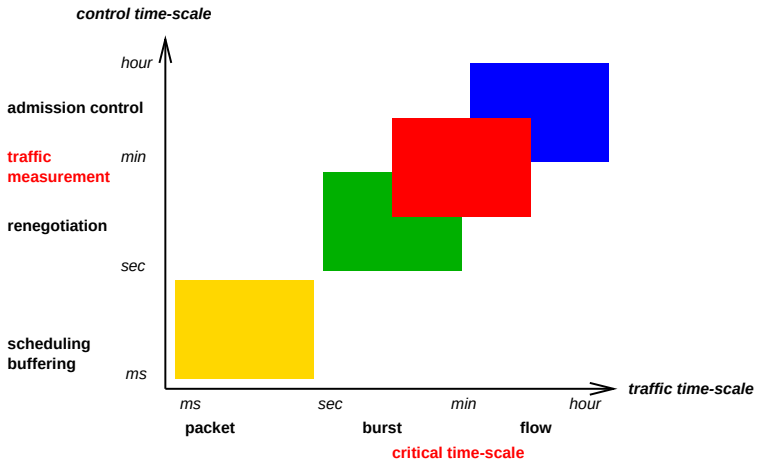

Figure 8.1: The time-scales we have discussed.

Figure 8.1 illustrates how these pieces fit together. It shows the mechanisms we have discussed and the time-scales for which they are active. Comparing this figure with Fig. 2.2 in the Introduction, we have shown the necessity of having a mechanism to accommodate fluctuations longer than the correlation horizon, for which buffering is not effective. The specific mechanism we have proposed here is renegotiation. In Chapter 5, we have discussed the advantages of measurement-based admission controland we have identified the *critical time-scale* as the optimal time-scale over which to perform traffic measurements. The burst, measurement, and flow time-scales overlap as they are interdependent: the performance of the system cannot be fully understood if we make an a-priori time-scale separation assumption between them.

In Chapter 6, we have extended the renegotiation service model to a multicast/multilayer setting. We have argued that it is beneficial to put additional flow dependence state into the network in order to avoid external resource waste.

Finally, we have proposed a solution to the feedback implosion problem in multicast feedback protocols. The timer-based approach we describe has the advantage of requiring no processing of feedback information inside the network, which is consistent with an end-to-end design philosophy.

8.2 Future Directions

There are several promising and important extensions to the measurement-based admission control problem. We will continue our approach of gaining insight using a combination of analytical models and extensive simulation, both model-driven and trace-driven. First, in the current model, we assume that the admission controller has per-flow bandwidth information available to it. We will study MBAC in the same framework when only *aggregate* flow bandwidth information is available. This is important because a measurement device for the aggregate bandwidth is easier to implement, as it does not have to inspect packet or cell tags (source/destination address, flow id, or virtual channel identifier), and because relying only on aggregate information allows us to relax the assumptions about the flow

bandwidth processes (independence, identical distribution), and is therefore more generally applicable. Second, our model does not take into account *application adaptivity*. We will study the MBAC problem for adaptive applications in a utility framework, in which a utility function characterizes a relative measure of quality, such as the perceptual quality of a video signal, as a function of the connection bandwidth. The goal of this extension is to quantify how much more aggressive the admission controller can be if applications are tolerant to changes in connection characteristics. Third, the conjecture that the MBAC model under study is insensitive to long-range dependence still requires a proof.

The future work outlined above is research for which we have a fairly well developed sense of the models, of the research methodology, and of the potential outcomes. In the following, we identify a few trends that will shape my longer-term research agenda. Our goal here is not to precisely define the problems we intend to work on, but to describe our vision of where the practical issues that motivate our research will lie in the future.

We believe that **complexity** will be the main challenge for networking in the coming years. Application and equipment developers face an environment of harsh competition and a great diversity of protocols, vendors, service providers, and applications. The researcher's perspective should take this into account by focusing less on global solutions that assume global control over the infrastructure; rather, the design space should be restricted to solutions that are viable in this dynamic, heterogeneous marketplace.

One very important lesson that we have learned from the impressive success of the Internet is the power of the simplicity of its service model. Offering only connectivity as a largest common denominator of possible services has allowed the Internet to spread rapidly. Monolithic solutions proposing more elaborate service models had more trouble becoming de-facto standards. Research should recognize this by focusing less on complete solutions that are hard to deploy in practice, simply because a global networking infrastructure is not, and probably never will be, a fixed, homogeneous structure under centralized control.

This heterogeneity and continuous change stresses the importance of **network management**, i.e., the feedback control of a large-scale network, where the feedback loop may or may not include one/several humans. Decentralized and flexible reactive control mechanisms acting on all time-scales can ensure that resources are allocated effectively in the absence of a homogeneous, centrally controlled infrastructure. Network monitoring and performance measurement, fault identification and localization, planning and resource provisioning, and configuration management are therefore both important and challenging future research topics, encompassing both architectural and performance issues. Next we identify two specific topics we intend to address.

- It is difficult to obtain accurate state information in a large network with decentralized control such as the Internet, as there is no ubiquitous measurement machinery in place. Fortunately, end-to-end measurements that provide partial observations of network state can be obtained easily. To make the most of these observations, *statistical methods for large-scale end-to-end inference of network state* are needed. These methods will be useful as input for provisioning and pricing systems, for fault discovery, or for obtaining information about a network without administrative access privileges.

- In a large-scale, loosely coupled system such as a communications network, a single fault in the network can generate a potentially large number of alarms from several subsystems. In turn, each of these alarms can have a large number of possible causes; explicitly verifying each of them is costly and slow. The fault prediction can be

improved if the whole set of alarms is taken into account, which is crucial in order to maintain high network availability and to minimize the cost of equipment troubleshooting. Therefore, the development of scalable and distributed methods for fault identification, including machine-learning of system models and their application to finding causes of faults in large-scale systems, is an important and rich research topic.

Recognizing heterogeneity and continuous change in the infrastructure as innate, rather than accidental, also suggests a new approach to modeling and system design. The traditional reductionist approach of studying small, well-defined subproblems, resulting in a top-down design methodology, often leads to designs that are unsuitable in such an ever-evolving environment. An alternative approach views the communication infrastructure as a collection of inherently adaptive entities. Adaptation happens everywhere, on all time-scales, and without implicit coordination, due to application and user behavior, user preferences, long-term network planning and provisioning, competition between providers, technological advances, and content design. The ultimate goal of this viewpoint is a more realistic design methodology for tomorrow's networking environment.

For example, there is currently considerable interest in forward error correction (FEC) techniques, i.e., adding redundancy for fault tolerance, as a way of reaping better performance from the Internet. These mechanisms are typically designed with the network viewed as a *fixed channel* that one wishes to exploit. However, it must be expected that over time, more and more applications will switch to this technique - a type of long-term adaptivity. It is far from clear if the *generalized* use of FEC is beneficial or detrimental, as the *properties of the channel itself* are affected by its widespread deployment.

There is also a potential for explaining, and avoiding, some of the *emergent* behavioral patterns that we already observe in today's Internet, such as transient, large-scale congestion ("storms"), and the instability of well-known routing protocols in large networks. It is crucial to understand such phenomena in order to devise a design methodology that can avoid them.

Research addressing similar issues is already under way in other scientific fields such as social sciences, economics, and evolutionary biology, where the heterogeneity and size of the systems under study is even more striking (for example, at the Santa Fe Institute, or the new Future Systems Group at British Telecom). There is therefore a promising potential for long-term interdisciplinary research.

Bibliography

[1] ATM Forum Traffic Management Specification Version 4.0. ATM Forum Specification /af-tm-0056.000, ATM Forum, April 1996.

[2] P. Abry and D. Veitch. Wavelet Analysis of Long-Range Dependent Traffic. Research Report, INRIA Sophia Antipolis, July 1996.

[3] A. Adas. Supporting Real Time VBR Video Using Dynamic Reservation Based on Linear Prediction. In *Proc. IEEE INFOCOM '96*, San Fransisco, CA, March 1996.

[4] R. Addie, M. Zukerman, and T. Neame. Fractal Traffic: Measurements, Modelling and Performance Evaluation. In *Proc. IEEE INFOCOM '95*, pages 977–984, Boston, Mass., April 1995.

[5] Mostafa H. Ammar. Probabilistic Multicast: Generalizing the Multicast Paradigm to Improve Scalability. In *Proc. IEEE INFOCOM '94*, Toronto, Canada, June 1994.

[6] J. Beran. Statistical Methods for Data with Long Range Dependence. *Statistical Science*, 7(4):404–427, 1992.

[7] J. Beran, R. Sherman, and W. Willinger. Long Range Dependence in Variable Bit Rate Video Traffic. *IEEE Trans. on Communications*, 43(3):1566–1579, February 1995.

[8] R. N. Bhattacharya, V. K. Gupta, and E. Waymire. The Hurst Effect under Trends. *Journal of Applied Probability*, 20:649–662, 1983.

[9] T. Billhartz, J. B. Cain, E. Farrey-Goudreau, D. Fieg, and S. Batsell. Performance and Resource Cost Comparisons for the CBT and PIM Multicast Routing Protocols in DIS Environments. In *Proc. IEEE INFOCOM '96*, pages 85–93, San Francisco, Calif., March 1996.

[10] P. Billingsley. *Probability and Measure (3rd Ed.)*. Wiley, 1995.

[11] J-C. Bolot, T. Turletti, and I. Wakeman. Scalable Feedback Control for Multicast Video Distribution in the Internet. In *Proc. ACM SIGCOMM '94*, London, UK, September 1994.

[12] P.E. Boyer and D.P. Tranchier. A reservation principle with applications to the ATM traffic. *Computer Networks and ISDN Systems*, 24:321–334, 1992.

[13] F. Brichet, J. Roberts, A. Simonian, and D. Veitch. Heavy Traffic Analysis of a Storage Model with Long-Range Dependent On/Off Sources. Technical Report, CNET, 1995.

[14] S. Y. Cheung, M. Ammar, and X. Li. On the Use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution. In *IEEE INFOCOM '96*, San Francisco, Calif., USA, March 1996.

[15] S. Chong, S.Q. Li, and J. Ghosh. Predictive Dynamic Bandwidth Allocation for Efficient Transport of Real-Time VBR Video over ATM. *IEEE Journal on Selected Areas of Communications*, 13:12–23, January 1995.

[16] D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proc. ACM SIGCOMM '92*, pages 14–26, 1992.

[17] D. D. Clark. The design philosophy of the DARPA Internet protocols. In *Proc. ACM SIGCOMM '88*, pages 106–114, Stanford, CA, August 1988.

[18] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill, New York, NY, 1990.

[19] Costas Courcoubetis et al. Admission Control and Routing in ATM Networks using Inferences from Measured Buffer Occupancy. In *ORSA/TIMS special interest meeting*, Monterey, CA, January 1991.

[20] M. Crovella and A. Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. In *Proc. ACM Sigmetrics '96*, pages 160–169, Philadelphia, PA, May 1996.

[21] J. Crowcroft and K. Paliwoda. A Multicast Transport Protocol. In *Proc. ACM SIGCOMM '88*, pages 247–256, August 1988.

[22] Rene L. Cruz. A Calculus for Network Delay, Part I: Network Elements in Isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.

[23] Rene L. Cruz. A Calculus for Network Delay, Part II: Network Analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, January 1991.

[24] J. Cuzick. Boundary Crossing Probabilities for Stationary Gaussian Processes and Brownian Motion. *Transactions of the American Mathematical Society*, pages 469–492, February 1981.

[25] G. de Veciana and J. Walrand. Traffic Shaping for ATM Networks: Asymptotic Analysis and Simulations. *submitted to IEEE/ACM Transactions on Networking*, 1992.

[26] S. E. Deering. Multicast Routing in Internetworks and Extended LANs. In *Proc. ACM SIGCOMM '88*, Stanford, Calif., August 1988.

[27] B. Doshi and S. Dravida. Congestion Controls for Bursty Data Traffic in Wide Area High Speed Networks: In-Call Negotiations. *Proc. ITC Specialist Seminar 7, Morristown, NJ*, 1990.

[28] N. G. Duffield, J. T. Lewis, N. O'Connell, R. Russell, and F. Toomey. Predicting Quality of Service for Traffic with Long-Range Dependence. In *Proc. IEEE ICC'95*, pages 473–477, Seattle, WA, September 1995.

[29] D. Duffy, A. McIntosh, M. Rosenstein, and D. Wilson. Statistical Analysis of CCSN/SS7 Traffic Data from Working CCS Subnetworks. *JSAC*, 12(3):544–551, April 1994.

[30] A. Eleftheriadis and D. Anastassiou. Meeting Arbitrary QoS Constraints Using Dynamic Rate Shaping of Coded Digital Video. *Proc. 5th Workshop on Networking and Operating System Support for Digital Audio and Video*, pages 95–106, April 1995.

[31] A. Elwalid, D. Heyman, T. V. Lakshman, D. Mitra, and A. Weiss. Fundamental Bounds and Approximations for ATM Multiplexers with Applications to Video Teleconferencing. *IEEE J. Sel. Areas in Communication*, August 1995.

[32] A. Elwalid and D. Mitra. Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High-Speed Networks. *IEEE/ACM Transactions on Networking*, 1:329–343, June 1993.

[33] A. Erramilli, O. Narayan, and W. Willinger. Experimental Queueing Analysis with Long-Range Dependent Packet Traffic. *IEEE/ACM Trans. on Networking*, 4(2):209–223, April 1996.

[34] A. Erramilli, R. P. Singh, and P. Pruthi. An Application of Deterministic Chaotic Maps to Model Packet Traffic. *Queueing Systems: Theory and Applications*, to appear, 1995.

[35] V. Firoiu and D. Towsley. Call Admission and Resource Reservation for Multicast Sessions. In *Proc. IEEE INFOCOM '96*, San Francisco, California, March 1996.

[36] Sally Floyd, Van Jacobson, Steven McCanne, Ching-Gung Liu, and Lixia Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. In *Proc. ACM SIGCOMM '95*, August 1995.

[37] M. W. Garrett. *Contributions Toward Real-Time Services on Packet Switched Networks*. PhD thesis, Columbia University, 1993. Chapter IV.

[38] M. W. Garrett and Walter Willinger. Analysis, Modeling and Generation of Self-Similar VBR Video Traffic. In *Proc. ACM SIGCOMM '94*, pages 269–280, London, UK, August 1994.

[39] R.J. Gibbens and P.J. Hunt. Effective Bandwidths for the Multi-type UAS Channel. *Queueing Systems*, 9:17–27, 1991.

[40] R.J. Gibbens, F.P. Kelly, and P.B. Key. A Decision-theoretic Approach to Call Admission Control in ATM Networks. *IEEE JSAC, Special issue on Advances in the Fundamentals of Networking*, August 1995.

[41] M. Grasse, M. R. Frater, and J. F. Arnold. On the Non-Stationarity of MPEG-2 Video Traffic. Technical Report COST 242, University of New South Wales, 1995.

[42] M. Grossglauser. Optimal Deterministic Timeouts for Reliable Scalable Multicast. In *Proc. IEEE INFOCOM '96*, San Francisco, California, March 1996.

[43] M. Grossglauser. Optimal Deterministic Timeouts for Reliable Scalable Multicast. *IEEE Journal on Selected Areas of Communications*, 15(3):422–433, April 1997.

[44] M. Grossglauser and J-C. Bolot. On the Relevance of Long-Range Dependence in Network Traffic. In *Proc. ACM SIGCOMM '96*, San Francisco, Calif., August 1996.

[45] M. Grossglauser and S. Keshav. On CBR Service. In *Proc. IEEE INFOCOM '96*, San Francisco, California, March 1996.

[46] M. Grossglauser, S. Keshav, and D. Tse. RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic. In *Proc. ACM SIGCOMM '95*, pages 219–230, Boston, Mass., August 1995.

[47] M. Grossglauser, S. Keshav, and D. Tse. The Case Against Variable Bit Rate Service. In *Proc. 5th Intl. Workshop on Network and Operating System Support (NOSSDAV '95)*, pages 307–310, Durham, New Hampshire, April 1995.

[48] M. Grossglauser, S. Keshav, and D. Tse. RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic. *IEEE/ACM Transactions on Networking*, December 1997.

[49] M. Grossglauser and D. Tse. A Framework for Robust Measurement-Based Admission Control. In *Proc. ACM SIGCOMM '97*, Cannes, France, September 1997.

[50] H. U. Bräker. High boundary excursions of locally stationary Gaussian processes. In *Proc. of the Conference on Extreme Value Theory and Applications*, Gaithersburg, Maryland, USA, May 1993.

[51] H. U. Bräker. *High boundary excursions of locally stationary Gaussian processes.* PhD thesis, Universität Bern, Switzerland, 1993.

[52] B. Hajek and L. He. On variations of queue responses for inputs with identical mean and autocorrelation fucntions. In *Conference on Information Sciences and Systems*, Princeton University, 1996.

[53] D. P. Heyman and T. V. Lakshman. What are the Implications of Long-Range Dependence for VBR Video Traffic Engineering? *IEEE/ACM Trans. on Networking*, 4(3), June 1996.

[54] B. Hochwald and K. Zeger. Tradeoff Between Source and Channel Coding. *IEEE Trans. on Information Theory*, 43(5), September 1997.

[55] H. W. Holbrook, S. K. Singhal, and D. R. Cheriton. Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation. In *Proc. ACM SIGCOMM '95*, pages 328–341, Cambridge, Mass., September 1995.

[56] I. Hsu and J. Walrand. Quick Detection of Changes in Traffic Statistics: Application to Variable Rate Compression. In *Proceedings of the 32nd Allerton Conference on Communications, Control and Computing, Monticello, IL*, 1993.

[57] I. Hsu and J. Walrand. Dynamic Bandwidth Allocation for ATM Switches. *Journal of Applied Probability*, September 1996.

[58] J.Y. Hui. Resource Allocation for Broadband Networks. *IEEE Journal on Selected Areas in Communications*, 6(9), December 1988.

[59] H. E. Hurst. Long-Term Storage Capacity of Reservoirs. *Proc. American Society of Civil Eng.*, 76(11), 1950.

[60] C. L. Hwang and S. Q. Li. On Input State Space Reduction and Buffer Noneffective Region. In *Proc. IEEE INFOCOM '94*, Toronto, Canada, April 1994.

[61] ITU. *Q.2931 Digital Subscriber Signalling System No. 2 (DSS 2) User-Network Interface (UNI) layer 3 specification for basic call/connection control*, February 1995. available from http://www.itu.ch.

[62] ITU. *Q.2971 Broadband integrated services digital network (B-ISDN) Digital subscriber signalling system No.2 (DSS 2) User-network interface layer 3 specification for point-to-multipoint call/connection control*, October 1995. available from http://www.itu.ch.

[63] V. Jacobson. Congestion avoidance and control. In *Proc. ACM SIGCOMM '88*, pages 314–329, Stanford, Calif., USA, August 1988.

[64] S. Jamin, P. B. Danzig, S. Shenker, and L. Zhang. A Measurement-Based Admission Control Algorithm for Integrated Services Packet Networks. In *Proc. ACM SIGCOMM '95*, 1995.

[65] H. Kanakia, P.P. Mishra, and A. Reibman. An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport. *Proc. ACM SigComm*, 1993.

[66] G. Karlsson and M. Vetterli. Packet Video and Its Integration into the Network Architecture. *IEEE Journal on Selected Areas of Communications*, 7(5), June 1989.

[67] G. Kesidis, J. Walrand, and C.S. Chang. Effective Bandwidths for Multiclass Markov Fluids and Other ATM Sources. *IEEE/ACM Transactions on Networking*, 1(4):424–428, August 1993.

[68] Leonard Kleinrock. *Queueing Systems, Volume 1: Theory.* John Wiley & Sons, New York, NY, 1975.

[69] V. Klemes. The Hurst Phenomenon: A Puzzle? *Water Resources Research*, 10(4):675–688, 1974.

[70] S. Klivansky, A. Mukherjee, and C. Song. On Long-Range Dependence in NSFNET Traffic. Technical Report, Georgia Institute of Technology, 1995.

[71] E. Knightly and H. Zhang. Traffic Characterization and Switch Utilization using a Deterministic Bounding Interval Dependent Traffic Model. In *Proc. IEEE INFOCOM '95*, Boston, Mass., April 1995.

[72] T. V. Lakshman and U. Madhow. Performance Analysis of Window-based Flow Control using TCP/IP: Effect of High Bandwidth-Delay Product and Random Loss. In *Proc. 5th IFIP Conference on High Performance Networking (HPN) '94*, Grenoble, France, June 1994.

[73] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE/ACM Trans. on Networking*, 2(1):1–15, February 1994.

[74] S. Q. Li and C. L. Hwang. Queue Response to Input Correlation Function: Continuous Spectral Analysis. *IEEE/ACM Trans. Networking*, 1(3):678–692, 1993.

[75] X. Li, S. Paul, and M. Ammar. Layered Video Multicast with Retransmissions (LVMR): Evaluation of Hierarchical Rate Control. In *Proc. IEEE INFOCOM '98*, San Francisco, Calif., USA, March 1998.

[76] John C. Lin and Sanjoy Paul. RMTP: A Reliable Multicast Transport Protocol. In *Proc. IEEE INFOCOM '96*, pages 1414–1424, San Francisco, Calif., March 1996.

[77] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven Layered Multicast. In *Proc. ACM SIGCOMM'96*, pages 117–130, Stanford, CA, Sept. 1996.

[78] S. McCanne, M. Vetterli, and V. Jacobson. Low-complexity Video Coding for Receiver-driven Layered Multicast. *IEEE Journal on Selected Areas of Communications*, 16(6):983–1001, August 1997.

[79] J. M. McManus and K. W. Ross. Video-on-Demand Over ATM: Constant-Rate Transmission and Transport. *IEEE Journal on Selected Areas of Communications*, pages 1087–1098, August 1996.

[80] K. Meier-Hellstern, P. Wirth, Y-L. Yang, and D. Hoeflin. Traffic Models for ISDN Data Users: Office Automation Application. In *Proc. ITC'13*, pages 167–172, Copenhagen, June 1991.

[81] M. Nomura, T. Fujii, and N. Ohta. Basic Characteristics of Variable Rate Video Coding in ATM Environment. *IEEE Journal on Selected Areas of Communications*, 7(5), June 1989.

[82] I. Norros. A Buffer with Self-Similar Input. *Queueing Systems: Theory and Applications*, 16(2):382–396, 1994.

[83] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing.* Prentice-Hall, Englewood Cliffs, 1989.

[84] C. Papadopoulos and G. Parulkar. Implosion Control for Multipoint Applications. In *10th IEEE Workshop on Computer Communications*, Seattle, September 1995.

[85] A. K. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to flow control in Integrated Services Networks - The Single Node Case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[86] A. K. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to flow control in Integrated Services Networks - The Multiple Node Case. *IEEE/ACM Transactions on Networking*, 2(1):137–150, April 1994.

[87] M. Parulekar and A. Makowski. Buffer Overflow Probabilities for a Multiplexer with Self-Similar Traffic. In *Proc. IEEE INFOCOM '96*, San Fransisco, CA, March 1996.

[88] V. Paxson. End-to-End Internet Packet Dynamics. In *Proc. ACM SIGCOMM '97*, Cannes, France, September 1997.

[89] V. Paxson and S. Floyd. Wide Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Trans. on Networking*, 3(3):226–244, June 1995.

[90] M. Podolsky, C. Romer, and S. McCanne. Simulation of FEC-Based Error Control for Packet Audio on the Internet. In *Proc. IEEE INFOCOM '98*, San Francisco, Calif., USA, March 1998.

[91] E. P. Rathgeb. Modeling and Performance Comparison of Policing Mechanisms for ATM Network. *IEEE Journal on Selected Areas in Communications*, 9(3):325–334, April 1991.

[92] E. P. Rathgeb. Policing of Realistic VBR Video Traffic in an ATM Network. *International Journal of Digital and Analog Communications Systems*, 6:213–226, 1993.

[93] D. J. Reininger, D. Raychaudhuri, and J. Y. Hui. Bandwidth Renegotiation for VBR Video Over ATM Networks. *IEEE Journal on Selected Areas of Communications*, 14(6), August 1996.

[94] Jennifer Rexford, Subhabrata Sen, Jayanta Dey, Wu chi Feng, James Kurose, John Stankovic, and Don Towsley. Online Smoothing of Live, Variable-Bit-Rate Video. In *International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '97)*, pages 249–257, May 1997.

[95] S. Robert and J-Y. Le Boudec. Can self-similar traffic be modelled by Markovian processes? In *Proc. 1996 International Zurich Seminar on Digital Communications (IZS'96)*, ETH Zentrum, Zurich, Switzerland, February 1996.

[96] B. K. Ryu and S. Lowen. Point Process Approaches to Modeling and Analysis of Self-Similar Traffic. Part I: Model Construction. In *Proc. IEEE INFOCOM '96*, San Fransisco, CA, March 1996.

[97] Bong K. Ryu and Anwar Elwalid. The Importance of Long-Range Dependence of VBR Video Traffic in ATM Traffic Engineering: Myths and Realities. In *Proc. ACM SIGCOMM '96*, San Francisco, Calif., August 1996.

[98] R. Safranek, C. Kalmanek, and R. Garg. Methods for Matching Compressed Video to ATM Networks. *Proc. of IEEE IT Workshop on Information Theory, Multiple Access and Queueing Theory, St. Louis*, page 6, April 1995.

[99] H. Saito and K. Shiomoto. Dynamic Call Admission Control in ATM Networks. *IEEE Journal on Selected Areas of Communications*, 9:982–989, 1991.

[100] J. Salehi, Z. Zhang, J. Kurose, and D. Towsley. Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements through Optimal Smoothing. *Proc. ACM SIGMETRICS, Philadelphia, PA*, May 1996.

[101] P. Sen, B. Maglaris, N. Rikli, and D. Anastassiou. Models for Packet Switching of Variable-Bit-Rate Video Sources. *IEEE Journal on Selected Areas of Communications*, 7(5), June 1989.

[102] S. Shenker. Fundamental Design Issues for the Future Internet. *IEEE Journal on Selected Areas of Communications*, 13(7), 1995.

[103] S. Shenker and L. Zhang. Some Observations on the Dynamics of a Congestion Control Algorithm. *ACM SIGCOMM Computer Communication Review*, 20(5), October 1990.

[104] P. Skelly, M. Schwartz, and S. Dixit. A Histogram-Based Model for Video Traffic Behavior in an ATM Multiplexer. *IEEE/ACM Trans. on Networking*, 1(4), August 1993.

[105] W. T. Strayer, B. J. Dempsey, and A. C. Weaver. *XTP: The Express Transfer Protocol*. Addison-Wesley (Reading, Mass.), 1992.

[106] F. Toutain and O. Huber. A General Preemption-Based Admission Policy Using a Smart Market Approach. In *Proc. IEEE INFOCOM '97*, Kobe, Japan, April 1997.

[107] D. Towsley, J. Kurose, and S. Pingali. A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols. *IEEE Journal on Selected Areas of Communications*, 15(3), April 1997.

[108] D. Tse, R. Gallager, and J. Tsitsiklis. Statistical Multiplexing of Multiple Time-Scale Markov Streams. *IEEE JSAC, Special issue on Advances in the Fundamentals of Networking*, August 1995.

[109] D. Tse and M. Grossglauser. Measurement-Based Call Admission Control: Analysis and Simulation. In *Proc. IEEE INFOCOM '97*, Kobe, Japan, April 1997.

[110] J.S. Turner. Managing Bandwidth in ATM Networks with Bursty Traffic. *IEEE Network Magazine*, September 1992.

[111] D. Veitch. Novel Models of Broadband Traffic. In *Proc. Globecom '93*, pages 362–368, Houston, TX, December 1993.

[112] W. Verbiest and L. Pinnoo. A variable bit rate codec for ATM networks. *IEEE Journal on Selected Areas of Communications*, 7(5):761–770, June 1989.

[113] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like congestion control for layered multicast data transfer. In *Proc. IEEE INFOCOM '98*, San Francisco, Calif., USA, March 1998.

[114] Lorenzo Vicisano, Luigi Rizzo, and Jon Crowcroft. TCP-like Congestion Control for Layered Multicast Data Transfer. In *Proc. IEEE INFOCOM '98*, San Francisco, Calif., USA, March 1998.

[115] A.J. Viterbi and J.K. Omura. *Principles of Digital Communication and Coding*. McGraw-Hill, 1979.

[116] A. Weiss. A New Technique for Analyzing Large Traffic Systems. *Advances in Applied Probability*, 18:506–532, 1986.

[117] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson. Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level. In *Proc. ACM SIGCOMM '95*, pages 100–113, Cambridge, Mass., August 1995.

[118] L.C. Wolf, L. Delgrossi, R. Steinmetz, S. Schaller, and H. Wuttig. Issues in Reserving Resources in Advance. *Proc. 5th Workshop on Networking and Operating System Support for Digital Audio and Video*, pages 27–37, April 1995.

[119] M. Yajnik, J. Kurose, and D. Towsley. Packet Loss Correlation in the MBone Multicast Network. In *IEEE Global Internet Conference*, London, November 1996.

[120] H. Zhang and E.W. Knightly. A New Approach to Support Delay-Sensitive VBR Video in Packet-Switched Networks. *Proc. 5th Workshop on Networking and Operating System Support for Digit al Audio and Video*, pages 275–286, April 1995.

[121] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, September 1993.