Chapter 1: Introduction

1.1. Historical perspective

Computer networks form an essential substrate for a variety of distributed applications, but they are expensive to build and operate. This makes it important to optimize their performance so that users can derive the most benefit at the least cost. Though most networks perform well when lightly used, problems can appear when the network load increases. Loosely speaking, congestion refers to a loss of network performance when a network is heavily loaded. Since congestive phenomena can cause data loss, large delays in data transmission, and a large variance in these delays, controlling or avoiding congestion is a critical problem in network management and design. This dissertation presents some approaches for congestion control in wide-area computer networks.

Historically, the first wide-area networks (WANs) were circuit-switched telephone networks. Since these networks carry traffic of a single type, and the traffic behavior is well known, it is possible to avoid congestion simply by reserving enough resources at the start of each call. By limiting the total number of users, each admitted call can be guaranteed to have enough resources to achieve its performance target, and so there is no congestion. However, resources can be severely underutilized, since the resources blocked by a call, even if idle, are not available to other calls.

Early research in computer data networking led to the development of reservationless store-and-forward data networks (132). These networks are prone to congestion since neither the number of users nor their workload are regulated. Essentially, the efficiency gained by statistical multiplexing of network resources is traded off with the possibility of congestion. This problem was recognized quite early (22), and a number of congestion control schemes were proposed; references (44, 110) provide a detailed review of these.

In the past three years, there has been a renewed interest in congestion control, as a glance at the Bibliography will indicate. We feel that at least three factors have been responsible. First, the spread of networks such as ARPANET, NSFNET, CSNET and BITNET, and their interconnection, has created a very large Internet whose size has made it unmanageable. The large number of users and a complete decentralization of network management made it inevitable that congestion would pose problems sooner or later. Matters came to a crisis around mid-1987, and prompted two pioneering research efforts. Jain, Ramakrishnan and Chiu at Digital Equipment Corporation developed the DECbit congestion control scheme (117). Simultaneously, Jacobson at Lawrence Berkeley Laboratories and Karels at UC Berkeley modified the well-known Transmission Control Protocol (TCP) (21, 108) to intelligently react to congestion, and to recover from it (63). The success of these efforts brought congestion control into focus as a major research area in the Internet community.

The second factor is the development of optical fiber technology. Optical fiber trunks offer data bandwidths that are a factor of 10,000 larger than earlier circuits (a ratio of 600 Mbps to 56kbps). A number of researchers have recognized the critical role of congestion control in high-speed WANs, since the bandwidth-delay product of a single circuit in such networks can be as large as 30 Mbits (600 Mbps x 50 ms round-trip delay across the USA). With such large products, a *single* source could introduce a transient load large enough to swamp buffers at the switches, leading to packet losses and excessive end-to-end delays for *all* the hosts on the network.

The third factor is social, rather than technological. The networking community has long been divided into two camps: the computer data networking community, and the telecommunications community. However, in recent years, the telecom community has realized the benefits of packet switching, resulting in the Asynchronous Transmission Mode (ATM) proposal from CCITT. Similarly, data networking researchers have realized that they need to provide real-time bounds on data transfer for services such CD-quality audio and interactive video (35, 154). This growing together of the two communities has led to a cross-fertilization of ideas about congestion management. Indeed, some of the most exciting proposals for future networks are arising from this interaction.

The present time appears to be critical for the design of future high speed networks, and in particular, their congestion control mechanisms. In this dissertation, we propose a number of ideas that we believe are useful for high speed networks. We hope that our work will contribute to the ongoing debate about congestion control.

This chapter is laid out as follows. Section 1.2 presents the environment of discourse. Section 1.3 defines congestion, and section 1.4 defines and discusses congestion control. Section 1.5 describes the fundamental requirements of a congestion control scheme, and the next section presents the fundamental assumptions made in this thesis. Section 1.7 reviews previous work, and the scope of this thesis is presented in section 1.8.

1.2. Environment of discourse

We survey congestion control techniques in two types of wide-area networks (though the dissertation is limited to techniques suitable for the first type). In both networks, data is sent from *sources* of data to *sinks* through intermediate store-and-forward switching nodes. Sources of data could be human users, transferring characters in a remote login session, or transferring files. For our purposes, we will refer to processes at OSI layer five and above as data sources. Sinks are the ultimate destinations of the data. They are the peer processes of the sources that receive and consume the received data, and they are typically assumed to acknowledge the receipt of each packet. *Switches* route and schedule incoming packets on outgoing lines, placing data in *output buffers* when the arrival rate exceeds the service rate. The simplex stream of packets between a source of data and its sink is called a *conversation*. Usually, a conversation corresponds to a pair of transport level endpoints, for example, two BSD sockets (21,23).

The first type of network under consideration, called a *reservationless network*, is an abstract model for networks such as the Internet. In such a network, while intermediate switches may reserve buffers (which does not reduce statistical multiplexing of the bandwidth), they may not reserve bandwidth (which does). Hosts on reservationless networks are assumed to be connected directly to switches, that in turn connect to other switches or hosts. A switch could be a piece of software that resides in a host, or could be a separate piece of hardware.

The other type of networks are those where switches reserve both bandwidth and buffers on behalf of *Virtual Circuits* (VCs) (such as in Datakit (41)). We call these *reservation-oriented networks*. We assume that these networks carry two types of traffic: performance-oriented traffic, which usually needs some form of real-time delay, bandwidth and jitter guarantees, and best-effort data traffic, which does not make such demands (35). Since no bandwidth is reserved on behalf of best-effort traffic (68), the best-effort component of a reservation-oriented network can be modeled as a reservationless network. Hence, schemes that are designed for reservationless networks can be transferred, with appropriate modifications, to reservation-oriented networks.

We believe that most future generation networks will tend to be reservation-oriented. Nevertheless, there are still some valid reasons to study congestion control in reservationless networks. First, reservationless networks will always be able to use bandwidth more efficiently than reservation-oriented networks due to the gain from statistical multiplexing. So, network providers who want to optimize cost will continue to build reservationless networks. Second, the techniques that are developed for congestion control can be applied to control best-effort traffic in reservation-oriented networks. Thus, the results of this work will apply even in those networks. Third, reservationless networks are currently the most common type of computer network. We believe that because of inertia, and a desire to stay with known and proven technology, they will continue to exist in the future.

1.3. What is congestion?

Despite the large and rapidly proliferating literature on congestion control, we are not aware of a satisfactory definition of congestion. We now discuss some common definitions, point out their flaws, and then propose a new definition that we consider to be superior.

Some common definitions of congestion

Since congestion occurs at high network loads, definitions of congestion focus on some aspect of network behavior under high load. We first discuss a scenario that leads to network congestion in reservationless networks, and then motivate some definitions.

Consider a reservationless network, where, due to some reason, the short term packet arrival rate at some switch exceeds its service rate. (The service rate is determined by the processing time per packet and the bandwidth of the output line. Thus, the bottleneck could be either the switch's CPU or the outgoing line: in either case, there is congestion.) At this point, packets are buffered, leading to delays. The additional delay can cause sources to time out and retransmit, increasing the load on the bottleneck (101). This feedback leads to a rapidly deteriorating situation where retransmissions dominate the traffic, and effective throughput rapidly diminishes (63, 114). Further, if there is switch to switch flow control (as in ARPANET, Tymnet, Datapac, Sirpent, etc.), new packets may not be allowed to enter the switch, and so packets might be delayed at a preceding switch as well. This can lead to deadlock, where all traffic comes to a standstill (132).

Note that three things happen simultaneously. First, the queueing delay of the data packets increases. Second, there may be packet losses. Finally, in the congested state, the traffic is dominated by retransmissions, so that the effective data rate *decreases*. The standard definitions of congestion are thus of the form: ``A network is congested if, due to overload, condition X occurs'', where X is excessive queueing delay, packet loss or decrease in effective throughput. The first definition is used in references (66, 117), the second in reference (63), and the third in reference (85).

These definitions are not satisfactory for several reasons. First, delays and losses are indices of performance that are being improperly used as indices of congestion, since the change in the indices may be due to symptoms of phenomena other than congestion. Second, the definitions do not specify the exact point at which the network can be said to be congested (except in a deterministic network, where the knee of the load-delay curve, and hence congestion, is well defined, but that is the trivial case). For example, while a network that has mean queueing delays in each switch of the order of 1 to 10 service times is certainly not congested, it is not clear whether a network that has a queueing delay of 1000 service times is congested or not. It does not seem possible to come up with any reasonable threshold value to determine congestion!

Third, a network that is congested from the perspective of one user is not necessarily congested from the perspective of another. For example, if user A can tolerate a packet loss rate of 1 in 1000, and user B can tolerate a packet loss rate of 1 in 100, and the actual loss rate is 1 in 500, then A will claim that the network is congested, whereas B will not. A network should be called uncongested only if all the users agree that it is.

New definition

From the discussion above, it is clear that network congestion depends on a user's perspective. A user who demands little from the network can tolerate a loss in performance much better than a more demanding user. For example, a user who uses a network only to send and receive electronic mail will be happy with a delivery delay of a day, while this performance is unacceptable for a user who uses a network for real-time audio communication. The key point is the notion of the *utility* that a user gets from the network, and how this utility degrades with network loading.

The concept of 'utility' used here is borrowed from economic theory. It is used to refer to a user's preference for a resource, or a set of resources (often called a resource bundle). Strictly speaking, the utility of a user is a number that represents the relative preference of that user for a resource (or performance) bundle, so that, if a user prefers bundle A to bundle B, the utility of A is greater than the utility of B. For example, if A is {end-to-end delay of 1 second, average throughput 200 pkts/second}, and B is {end-to-end delay of 100 seconds, average throughput 20000 pkts/second}, a user may prefer A to B, and we would assign a utility to A that is greater than the utility of B, while another user may do the opposite. In classic microeconomic theory, utilities are represented by a function over the resources (140). Since utilities express only a preference ordering, utility functions are insensitive to monotonic translations, and the utilities of two users cannot be compared; the function can only be used to relatively rank two resource bundles from the point of view of a single user.

An example of a utility function is $\alpha T - (1-\alpha)RTT$, where α is a weighting constant, T is the average throughput over some interval, and RTT is the average round-trip-time delay over the same interval. As the throughput increases, the utility increases, and as delays increase, the utility decreases. The choice of α determines the relative weight a user gives to throughput and delay. A delay-sensitive user will choose $\alpha \to 0$, whereas a delay-insensitive user's $\alpha \to 1$.

In practice, a utility function may depend on a threshold. For example, a user may state that he or she is indifferent to delay, as long as it is less than 0.1 seconds. Thus, if the user gets a delay of 0.05 seconds during some interval of time, and 0.06 seconds in a later period, as far as the user is concerned, there has been no loss of utility. However, if some user's utility *does* decrease as a result of an increase in the network load, that user will perceive the network to be congested. This motivates our definition.

Definition

A network is said to be congested from the perspective of user *i* if the utility of *i* decreases due to an increase in network load.

Remarks:

1. A network can be congested from the perspective of one user, and uncongested from the perspective of another.
2. A network can be said to be strictly uncongested if no user perceives it to be congested.
3. A user's utility may decrease due to something other than network load, but the user may not be able to tell the difference. The onus on the user is to determine the cause of the loss of utility, and to take appropriate corrective action.

This definition is better than existing definitions since it avoids the three problems raised earlier. First, we make a clear distinction between a performance index and a congestion index. It is possible for a performance metric to decrease (for example, for RTT to increase), without a change in the congestion index (for example, if $\alpha = 1$). Second, the definition makes it clear that congestion occurs from the point of view of each individual user. Finally, the point of congestion is precisely the one where the user detects a loss of utility. No further precision is necessary, since, if the users are not dissatisfied with the available service, then the network performance, no matter how poor it is in absolute terms, is satisfactory.

Our definition places congestion control in a new light. A network that controls congestion, by our definition, must be responsive to the utility function of the users, and must be able to manage its resources so that there is no loss of utility as the load increases. Thus, the network *must* be able to differentiate between conversations, and prioritize conversations depending on the stringency of their owner's utility. A naive approach that ignores the user's quality-of-service requirements is automatically ruled out by this definition.

1.4. Congestion control

The previous section presented a new definition of congestion; this section describes congestion control. Two styles of control, proactive and reactive control, are presented. It is shown that congestion control must happen at several different time scales.

1.4.1. Proactive and reactive control

Congestion is the loss of utility to a user due to an increase in the network load. Hence, congestion control is defined to be the set of mechanisms that prevent or reduce such a deterioration. Practically speaking, a network can be said to control congestion if it provides each user with mechanisms to specify and obtain utility from the network. For example, if some user desires low queueing delays, then the system should provide a mechanism that allows the user to achieve this objective. If the network is unable to prevent a loss of utility to a user, then it should try to limit the loss to the extent possible, and, further, it should try to be fair to all the affected parties. Thus, in reservationless networks, where a loss of utility at high loads is unavoidable, we are concerned not only with the extent to which utility is lost, but also the degree to which the loss of utility is fairly distributed to the affected users.

A network can provide utility in one of two ways. First, it can request that each user specify a performance requirement, and can reserve resources so that this level of performance is always available to the user. This is proactive or reservation-oriented congestion control. Alternatively, users can be allowed to send data without reserving resources, but with the possibility that, if the network is heavily loaded, they may receive low utility from the network. The second method is applicable in reservationless networks. In this case, users must adapt to changes in the network state, and congestion control refers to ways in which a network can allow users to detect changes in network state, and corresponding mechanisms that adapt the user's flow to changes in this state.

In a strict proactive scheme, the congestion control mechanism is to make reservations of network resources so that resource availability is deterministically guaranteed to admitted conversations. In a reactive scheme, the owners of conversations need to monitor and react to changes in network state to avert congestion. Both styles of control have their advantages and disadvantages. With proactive control, users can be guaranteed that they will never experience loss of utility. On the other hand, to be able to make this guarantee, the number of users has to be restricted, and this could lead to underutilization of the network. Reactive control allows much more flexibility in the allocation of resources. Since users are typically not guaranteed a level of utility by the network, resources can be statistically multiplexed. However, there is always a chance that correlated traffic bursts will overload the network, causing performance degradation, and hence, congestion.

It is important to realize that proactive and reactive control are not mutually exclusive. Hybrid schemes can combine aspects of both approaches. One such hybrid scheme is for the network to provide statistical guarantees (37,48). For example, a user could be guaranteed an end to end delay of less than 10 seconds with 0.9 probability. Such statistical guarantees allow a network administrator to overbook resources in a controlled manner. Thus, statistical multiplexing gains are achieved, but without completely giving up performance guarantees.

Another hybrid scheme is for the network to support two types of users: guaranteed service users and best-effort users (35,68). Guaranteed service (GS) users are given a guarantee of quality of service, and resources are reserved for them. Best-effort (BE) users are not given guarantees and they use up whatever resources are left unutilized by GS users.

Finally, a server may reserve some minimum amount of resources for each user. Since every user has some reservation, some minimum utility is guaranteed. At times of heavy load, users compete for resources kept in a common pool (62). Assuming some degree of independence of traffic, statistical multiplexing can be achieved without the possibility of a complete loss of utility.

1.4.2. Time scales of control

Congestion is a high-load phenomenon. The key to congestion control lies in determining the time scale over which the network is overloaded, and taking control actions on that time scale. This is explained below.

Consider the average load on a single point-to-point link. Note that the 'average load' is an *interval-based metric*. In other words, it is meaningless without also specifying the time interval over which the average is measured. If the average load is high over a small averaging interval, then the congestion control mechanism (for example, the reservation mechanism) has to deal with resource scheduling over the same small time scale. If the average load is high over a longer time scale, the congestion control mechanism needs to deal with the situation over the longer time scale *as well as* on shorter time scales.

An example should clarify this point. Consider a conversation on a unit capacity link. If the conversation is bursty, then it could generate a high load over, say, a 1ms time scale, though the average load over a 1 hour time scale could be much smaller than 1. In this case, if the conversation is delay-sensitive, then the congestion control scheme must take steps to satisfy the user delay requirement on the 1ms time scale. Over longer time scales, since the average demand is small, there is no need for congestion control.

On the other hand, if the conversation has a high average demand on the 1 hour time scale as well as the 1ms time scale, then congestion control has to be active on both time scales. For example, it may do admission control (which works over the 1 hour time scale) to make sure that network resources are available for the conversation. Simultaneously, it may also make scheduling decisions (which work on the 1ms time scale) to meet the delay requirements.

This example illustrates three points. First, congestion control must act on several different time scales simultaneously. Second, the mechanisms at each level must cooperate with each other. Scheduling policies without admission control will not ensure delay guarantees. At the same time, the admission control policy must be aware of the nature of the scheduling policy to decide whether or not to admit a conversation into the network. Third, the time scale is the time period over which a user sees changes in the network state. A congestion control mechanism that is sensitive to network state must operate on the same time scale.

We now discuss five times scales of control: those of months, one day, one session, multiple round trip times (RTTs), and less than one RTT. We believe that the design of congestion control mechanisms for each time scale should be based on sound theoretical arguments. This has the obvious advantages over an *ad hoc* approach: general applicability, ease of understanding, and formal provability of correctness. At each time scale of control, a different theoretical basis is most appropriate, and this is discussed below.

1.4.2.1. Months

Some changes to networks happen over a period of months: for example, an increase in the number of connected sites, or additional communication loads due to a new collaboration between geographically distributed sites. If these changes cause trunk lines to be substantially overloaded over long periods of time, then the only way to provide acceptable service may be to increase trunk bandwidths. Otherwise, no matter how clever the schemes at faster time scales, the network will not be able to accommodate the additional demand.

The typical response to long term overload is to lay additional bandwidth: the gradual replacement of 9600 baud serial lines with 10 Mbps Ethernets and 56 kbps trunks, and now with 100 Mbps FDDI rings and T1 trunks, is a graphic illustration of this process. Over this time scale, the formal problem corresponding to congestion control is that of capacity planning. This problem has been studied in the operations research literature. Basically, a traffic matrix, representing the volume of traffic between all pairs of sites, is constructed. A cost function describing the cost of trunk bandwidth and the utility from the network is then optimized using standard linear programming techniques to obtain an optimal capacity allocation along each path.

The problem with this approach lies in the determination of the traffic matrix. It is hard to collect all the required information, and, besides, the matrix can be quite large. Furthermore, there is no guarantee that the system workload during the next time period would be similar, so the computed solution may solve the wrong problem. Nevertheless, the use of a formal capacity planning approach would be an advance over the current *ad hoc* approach.

1.4.2.2. One day

Traffic measurements have shown that network usage exhibits cyclical behavior, with the time period of a day (13, 105). Networks are typically lightly loaded or idle at night, but are busy from 9 am to 5 pm, reflecting the work day. A congestion control scheme that spreads the load more uniformly throughout the day will ensure that the workload is less bursty, leading to better utilization of capacity. This can be done by introducing a pricing scheme for data transfer (20, 155), and charging more for data transmitted during peak hours than during off-peak hours (7, 111, 130). Peak-load pricing schemes used by electric and telephone utilities have much the same purpose. We believe that using ideas from economics to shape the workload on the order of a day (or several hours) is a useful form of congestion management at this time scale.

While some economic approaches have been proposed in the recent past (32, 146), we feel that the area needs much more study. Current approaches have numerous problems, such as:

- a user's assumed utility function, or, in some approaches, demand curve, is overly simple
- the system takes a long time to reach equilibrium. For example, the SPAWN system (146) postulates multiple rounds of bidding each time a new user enters the system, and each round could take many seconds. Since new users could join every few seconds, it is not clear that the system can ever reach equilibrium.
- the approaches assume that the number of users is fixedl; in fact, given that the systems are slow to reach equilibrium, this number is like to have changed by the time equilibrium is reached
- all the users are assumed to be rational; in reality, some users may make unsophisticated, and hence irrational, decisions
- a single administrative authority is assumed
- it is time-consuming for users to bid for every resource they need
- bursty users, whose peak and average resource requirements differ considerably, are not considered.

However, the economic approach is promising, since it gives deep insights into network pricing and usage accounting that are otherwise unavailable.

1.4.2.3. Session

In connection-oriented networks, a session is the period of time between a call set-up and a call teardown. Admission control in connection oriented-networks is essentially congestion control on the time scale of a session: if the admission of a new conversation could degrade the quality of service of other conversations in the network, then the new conversation should not be admitted. This is yet another form of congestion control.

At the time a conversation is admitted to the network, the network should ensure that the resources requested by the conversation are what it really needs. Thus, the admission control scheme should give incentives to users to declare their resource needs accurately (20). This can be designed using a variant of the standard Clark-Groves direct truth revelation mechanism (since naive users in a system with partial information would choose a dominant strategy equilibrium over more sophisticated Bayesian or sequential equilibria) (80). The theoretical basis of this work is the area of game-theory known as 'mechanism design'. An admission control system based on such principles has been extensively studied by Sanders (121-123).

The mechanism design approach has all the problems of the economic approach, and more. Mechanism design makes strong assumptions about the information structure in the system: for example, each switch is assumed to know the form of the utility function of each user. Further, the dynamics of the system over time are ignored. However, in the same way as economics, it can provide insights into the design of admission control.

The choice of a route at session establishment time can also be used to control congestion. If the routing establishment algorithm keeps track of network utilization along the links, it can route a new circuit along lightly loaded paths. Some proposals have exploited this idea (29,77,92). Integrating feedback flow control with adaptive routing is complex, and the dynamics of their interaction are not well known. Some simplified models for the problem have been studied (29), but the applicability of these studies to real networks is unclear. In this thesis, we assume static routing, so this form of congestion control is specifically ignored.

1.4.2.4. Multiple round trip times

One round trip time (RTT) is the fundamental time constant for feedback flow control. It is the minimum time that is needed for a source of data to determine the effect of its sending rate on the network (117). Congestion control schemes that probe network state and do some kind of filtering on the probes operate on this time scale. Examples are various window adjustment schemes (54,63,96,117).

The theoretical bases for these approaches lie in queueing theory and control theory. The queueing theory approach is well studied (44), but requires strong assumptions about the network such as: Poisson arrivals from all sources, exponential service time distribution at all servers, and independence of traffic. Since observations of real networks have shown that none of these assumptions are satisfied in practice (51,52,104), the results obtained from a stochastic queueing approach are not entirely convincing. We believe that a naive deterministic queueing approach has several benefits, though, and this is discussed in Chapter 4.

The control-theoretic approach has also been studied in the literature, though not as thoroughly (1,38,66,79,133,137,138). The drawback with existing studies is that either they are informal, and thus do not provide any formal proofs (63,117), or they make strong assumptions about traffic behavior, service rates and so on, that do not hold in practice (126,137,138). Chapter 5 presents an alternative approach that overcomes some of these problems.

Assuming that each packet is acknowledged, multiple acknowledgements can be received each RTT. If information about the state of the network is extracted from each acknowledgement, then the congestion control scheme can react to changes even faster than once per RTT. Such a scheme is described in Chapter 5, and is also made possible by a control theoretic modeling of the network.

Previous work in the area of congestion control schemes that work at the multiple RTT time scale is examined in greater detail in section 1.7.

1.4.2.5. Less than one RTT

On a scale of less than once per RTT, congestion control can be considered to be identical to scheduling data at the output queues of switches. The goal of a scheduling policy is to decide which data unit is the next to be delivered on a trunk. This choice determines the bandwidth, delay, and jitter received by each conversation, and hence the choice of the scheduling discipline is critical. A scheduling discipline that does *not* vary its allocations as a function of the network load is hence a congestion control mechanism. Examples are the Virtual Clock scheme (154), Delay-EDD (37) and Stop-and-Go queueing (47).

If the trunk bandwidth is considered as a resource, then the server implementing the scheduling discipline is essentially a resource manager that allocates bandwidths, delays and delay-jitters to each conversation. Then, it is in the best interest of a conversation to send data in such a way that it obtains the best possible utility from the server. Given that all the conversations have this objective, and the gain of one could be the loss of the other, it is clear that this framework can be cast in the form of a non-cooperative game (125). Then, each conversation must choose a strategy (in this case, a sending rate) such that its utility is maximized. Such a model has been considered by Bharath-Kumar and Jaffe (6), Sanders (121-123), Douligeris and Majumdar (25-27) and Lazar *et al* (59). Unfortunately, the game theoretic formulation of the problem requires utilities to be defined for each player, as well as strong information and rationality assumptions, as in the economic approach. Thus, we do not feel that it is viable in practice.

1.4.3. Need for congestion control in future networks

Congestion is a severe problem in current reservationless networks. However, in future networks the available bandwidths and switching speeds will be several orders of magnitude larger (84). Why should congestion arise in such networks? There are several reasons:

- **Large bandwidth-delay product** : The service rate of a circuit multiplied by the round trip time determines the amount of data that a conversation must have outstanding in order to fully utilize the network. The round trip time is bounded from below by the speed-of-light propagation delay through the network. Hence, as the raw trunk bandwidth and the service rate of a conversation increases, so does the amount of outstanding data per conversation. For example, a conversation that is served at 6 Mbps and has a round trip time delay of 60 ms can have as much as 45 kbytes of data outstanding. If the service rate suddenly drops to half of its previous value, or 3 Mbps, due to cross traffic, then it takes 60 ms to react to this change, and 45/2 = 22.5 Mbytes of data can accumulate at the slowest server along the path. Since this far exceeds most buffer sizes, data loss is likely, leading to a loss of utility. Thus, a large bandwidth-delay product causes problems for reactive control and can lead to congestion.
- **Speed Mismatch** : If a switch connects a high speed line to a slower line, then a bursty conversation can, when sending data at the peak rate, fill up its buffer share, and subsequently lose packets at the switch. This creates congestion for loss-sensitive conversations. This source of congestion will persist in high-speed networks, in fact, it is probably more likely in such networks.
- **Topology** : If several input lines simultaneously send data through a switch to a single outgoing line, the outgoing line can be overloaded, leading to large queueing delays, and possible congestion for delay-sensitive traffic. This is a special case of the speed mismatch problem noted earlier.
- **Increased Usage** : Memory sizes have increased exponentially during the last decade. Yet, the demand for memory has remained, since larger memory sizes have made it feasible to develop applications that require them. Drawing a parallel to this trend, we postulate that as bandwidth increases, new applications (such as real time video) will demand these enormous bandwidths. As the available bandwidth gets saturated, the network will be operated in the high-load zone, and congestive problems are likely to reappear.
- **Misbehavior** : Congestion can be induced by misbehaving sources (such as broken sources that send a stream of back-to-back packets). Future networks must protect themselves and other sources from such misbehavior, which will continue to exist.
- **Dynamics** : As network speeds increase, the dynamics of the network also changes. Since queues can build up faster, congestive phenomena can be expected to occur much more rapidly, and perhaps have catastrophic effects (101, 132).

From these observations, we conclude that even though future networks will have larger trunk bandwidths and faster switches, congestion will not disappear.

1.5. Fundamental requirements of a congestion control scheme

We would like a congestion control scheme to have a number of properties. These are:

- Efficiency
- Ability to deal with heterogeneity
- Ability to deal with ill-behaved sources
- Stability
- Scalability
- Simplicity
- Fairness

We discuss these in turn. In the discussion, we will use the term `the control scheme' to mean an integrated control scheme that simultaneously operates over all the time scales of control.

1.5.1. Efficiency

There are two aspects to efficiency. First, how much overhead does the congestion control scheme impose upon the network? As an extreme example, control packets such as the Source Quench packets in TCP/IP (112) themselves overload the congested network. We would like a control scheme to impose a minimal additional burden upon the network. This is not necessarily a binding condition: with the rapid increase in communication bandwidth, the extra load may not significantly affect network efficiency.

Second, does the control scheme lead to underutilization of critical resources in the network? Inefficient control schemes may throttle down sources even when there is no danger of congestion, leading to underutilization of resources. We would not like to operate the network suboptimally. (This raises the issue of determining what is meant by optimal utilization, but we defer the discussion to Chapter 4.)

1.5.2. Heterogeneity

As networks increase in scale and coverage, they span a range of hardware and software architectures. A control scheme that assumes a single packet size, a single transport layer protocol, or a single type of service cannot be successful in such an environment. Thus, we want the control scheme to be implementable on a wide range of network architectures.

1.5.3. Ability to deal with misbehaving sources

Note that in current networks, a network administrator does not have administrative control over the sources of messages. Thus, sources (such as users of personal workstations) are free to manipulate the network protocols to maximize the utility that they get from the network. When a switch informs a source that it should reduce its sending rate, a well-behaved source should do so. However, it is possible that an ill-behaved source may choose to ignore these signals, in the hope that this may enable it to send more data. A congestion control scheme should not fail in the presence of such hosts. In other words, badly behaving sources should not adversely affect the performance of well-behaved sources. This may require punishment of a badly behaved source, or threats of punishment that will give all sources an incentive to behave well.

1.5.4. Stability

Congestion control can be viewed as a classic negative-feedback control problem. One added complexity is that the control signals are delayed. That is, there is a finite delay between the detection of congestion and the receipt of a signal by a source. Further, the system is noisy, since observations of the system's parameters may be corrupted by transients. These complexities may introduce instabilities into the network. Thus, we would like the control scheme to be robust, and if possible, provably stable.

1.5.5. Scalability

It is the nature of a distributed system to grow with time, and we have seen an explosive growth in network sizes in the last decade. The key to success in such an environment is scalability. We would like a congestion control scheme to scale along two orthogonal axes: bandwidth and network size.

The development of high bandwidth fiber-optic media has made it possible to build networks that have three orders of magnitude more bandwidth than networks of the recent past (45000 kbps to 56 kbps) (69). Further, hundreds of thousands of local area networks have been interconnected in the last five years into an enormous internetwork that spans the globe. These developments make it imperative that any proposed congestion control scheme scale well on both axes.

1.5.6. Simplicity

Simplicity (unlike stupidity) is always an asset. Simple protocols are easier to implement, perhaps in hardware, and can handle increases in bandwidth. Also, simple protocols are more likely to be accepted as international standards. Thus, we would like an ideal congestion control mechanism to be simple to implement.

1.5.7. Fairness

It may be necessary for some sources to reduce their network load to control congestion. The choice of which source to throttle (either by requesting it to do so, or by dropping its packets) determines how fairly the network allocates its resources (23). For example, if an excessive demand by some source A causes the throttling of another source B, then clearly the network is treating B unfairly. We do not want a congestion control scheme that results in unfair network resource allocation.

There are two tricky aspects of fairness: definition and implementation. The problem of defining fairness has worried network designers and welfare economists alike. Numerous fairness criteria have been proposed in the literature (43, 55, 106, 107, 115, 139, 149). Each criterion has some problems, and there seems to be no absolute guide to deciding which is the best criterion to adopt (44). Nevertheless, it is necessary to pick some justifiable criterion, since this is much better than none at all.

The second aspect is that of implementation. Implementing a fairness criterion brings in issues that lie beyond what is traditionally considered to be the scope of congestion control. For example, we may decide that it is fair to give preference to sources that are willing to pay for it. If this criterion is to be implemented, a switch needs to determine a pricing schedule and incorporate a contract negotiation and enforcement mechanism (20, 155). This leads naturally to issues in contract theory, mechanism design and incentive compatibility. Such issues have been ignored by existing congestion control schemes.

To summarize, there are a number of issues that are affected by the choice of a congestion control scheme. In this thesis, we present the design of a set of congestion control mechanisms that substantially meet the requirements posed in this section.

1.6. Fundamental assumptions

Underlying any congestion control scheme are some implicit assumptions about the network environment. These unstated assumptions largely determine the nature of the control scheme and its performance limits. We consider some of these assumptions in this section.

1.6.1. Administrative control

Can we, as designers of congestion control mechanisms, assume administrative control over the behavior of sources (for example, by dictating that only one version of a transport protocol is to be used in the network)? Or, can we assume administrative control only over the behavior of switches? Some schemes assume that we can control sources but not switches e.g. (63). Others assume that we can control both the sources and the switches (117). Still others assume complete control over the switches, and the ability to monitor source traffic, but no control over source traffic (54).

This assumption should be constrained by reality. In our work, we assume that we have administrative control over switches. However, source behavior is assumed to be outside our direct control (though it can be monitored, if needed). The implication is that the network must take steps to protect itself and others from malicious or misbehaving users. (Of course, users that abuse the network because of a hardware failure, such as a jammed ethernet controller, are always a threat, even when the sources can be controlled administratively.)

1.6.2. Source complexity

How complex should one assume sources to be? Since we do not have administrative control over sources, we assume that sources will perform actions that will maximize their own utility from the network. If the congestion control scheme allows intelligent users to manipulate the scheme for their own benefit, they will do so. On the other hand, users may not have the capability to respond to complex directives from the network, so we cannot assume that all users will act intelligently. In other words, while a congestion control scheme should not assume sophisticated behavior on the part of the users, at the same time, it should not be open to attack from such users.

1.6.3. Gateway complexity

Some authors assume that switches can be made complex enough to set bits on packet headers, or even determine user utility functions, whereas others assume that switches simply route packets. Ignoring monetary considerations, since we have administrative control, we can make switch control algorithms as complex as we wish, constrained only by speed requirements. It has been claimed that for high speed operation, switches should be dumb and fast. We believe that speed does not preclude complexity. What we need is a switch that is fast *and* intelligent. This can be achieved by

- having hardware support for rapid switching (100)
- optimizing for the average case (18)
- removing signaling information from the data path (41)
- choice of scheduling algorithm (156)
- an efficient call processing architecture (131).

Thus, we will assume that a switch can make fairly intelligent decisions, provided that this can be done at high speeds.

1.6.4. Bargaining power

The ultimate authority in a computer network lies in the ability to drop packets (or delay them). Since this authority lies with the switches, they ultimately have all the bargaining power. In other words, they can always coerce sources to do what they want them to do (unless this is so ridiculous that a source would rather not send any data). Any scheme that overlooks this fact loses a useful mechanism to control source behavior. Thus, schemes that treat switches and sources as peer entities (for example, (117)) are fundamentally flawed: they need to posit cooperative sources precisely because they ignore the authority that is automatically vested in switches.

1.6.5. Responsibility for congestion control

Either the sources or the switches could be made responsible for congestion control. If sources are responsible, they must detect congestion and avert it. If switches are responsible, they must take steps to ensure that sources reduce their traffic when congestion occurs, or allocate resources to avert congestion.

We believe that congestion control is a network function. If we leave the responsibility for it to sources that are not under our administrative control, then we are endangering the network. Further, the congestion detection and management functionality has to be duplicated at each of the (many) sources. In contrast, it is natural to make the fewer, controllable switches responsible for congestion control.

Note that responsibility is not the same as functionality. In other words, having responsibility for congestion does not mean that the switches have to actually perform all the actions necessary for congestion control. Switches can enforce rules that make it incentive compatible for sources to help in containing congestion. For example, a Fair Queueing (23) switch has the responsibility for congestion control, but it does congestion control by forcing *sources* to behave correctly during congestion (this is discussed in Chapter 2).

1.6.6. Bandwidth-delay product

The physical characteristics of the network play a crucial role in determining how effective a scheme will be. A congestion control scheme has to operate in some bandwidth and delay regime. When the bandwidth times round-trip-time product is small in comparison to the window size, feedback from switches to sources is feasible. However, if this product is large, buffering full windows for all users may not be feasible.

Many current schemes operate on the scale of multiple round-trip-time delays (for example, taking flow control decisions only once every two round-trip times (66)). They become infeasible when the bandwidth-delay product is large. This large product is precisely why congestion control schemes for high speed networks are so difficult, and so necessary. In this thesis, we assume that the network operates in the large bandwidth-delay product regime.

1.6.7. Traffic model

The choice of a traffic model influences the design of a congestion control scheme, since the scheme is evaluated with respect to this model. There are hidden dangers here: for example, schemes that assume Poisson sources may not robust, if, in practice, traffic does not obey this distribution. It is best to design schemes that are insensitive to the choice of the traffic model. This is achieved if a scheme does not make assumptions about the arrival distribution of packets at the switches.

What should be the traffic model? We do not have much data at our disposal, since there are no high-speed WANs yet available to measure. However, there are three trends that point to a reasonable model. First, the move towards integration of data, telephony and video services indicates that some number of sources in our environment will be phone and video sources. These can generate high bandwidth traffic over periods of time spanning minutes or hours.

Second, existing studies have shown that data traffic is very bursty (52, 104). This tendency will certainly be exaggerated by increases in line speeds. Finally, we note that current applications are mostly of two sorts - low bandwidth interactive conversations, and high bandwidth off-line bulk data transfer (12). At higher speeds, the bulk data transfers that last several seconds today will collapse into bursts. This reinforces our belief that future traffic will basically be bursty.

To sum up, we expect that the traffic will be generated by two kinds of sources: one that demands a sustained high bandwidth, and the other that generates bursts of traffic at random intervals of time. We call these sources 'FTP' and 'Telnet' in this thesis (these terms are probably outdated, but we use them for convenience). This model is fairly simple, and does not involve any assumptions about packet arrival distributions. Thus, schemes that work for this model will probably work for a large variety of parametrically constrained models as well (e.g. for traffic where the bursts are exponentially or uniformly distributed). Since the traffic characteristics for future networks are still unknown, this model is speculative. However, we think that it is as reasonable as any that have so far been studied.

Summary of our assumptions

- We assume that we have administrative control over switches, but not over sources.
- Some sources will be intelligent enough to exploit any flaws in the design, but not all the sources will be able to implement complex congestion and flow control strategies.
- Switches should be fast, but they should carry out intelligently designed congestion control strategies.
- The switches should not be treated as peer entities of sources, their inherent capacity to delay or drop packets should be used to coerce sources to behave in a socially acceptable manner.
- The responsibility for congestion control should lie with the switches, sources cannot be trusted with this job.
- The scheme should operate in a regime where the delay bandwidth product is large as compared to current window sizes.
- Traffic consists of two types of sources, ones that generate sustained high bandwidth traffic, and others that generate intermittent bursts.

1.7. Previous work

This section surveys previous work in congestion control at the session, multiple RTT and less than one RTT time scales. Research in these areas of congestion control has mushroomed in the recent past. Consequently, this survey cannot claim to be complete. We have tried to mention all the research efforts that we are aware of, but it is almost certain that some others have been overlooked.

While our work stresses the need for a network to be sensitive to a user's utility, previous work on congestion control has concentrated mainly on mechanisms for avoiding packet losses and reducing queueing delay. The efforts have been strongly oriented towards either reservationless networks or reservation-oriented networks. For reservation-oriented networks, the work had been at the session and scheduling time scales, and for reservationless networks exclusively at the multiple RTT time scale. We now discuss previous work for each class of network.

1.7.1. Reservationless networks

In reservationless networks, control has to be reactive. A reactive congestion control scheme is implemented at two locations: at the switches, where congestion occurs, and at the sources, which control the net inflow of packets into the network. Typically, a switch uses some metric (such as overflow of buffers) to determine the onset of congestion, and implicitly or explicitly communicates this problem to the sources, which reduce their input traffic. Even in this simplified picture, a few problems are apparent.

How is congestion to be detected? (Congestion Detection)

How is the problem signaled to sources? (Communication)

What actions do the switches take? (Decongestion)

Which sources are held responsible for congestion? (Selection)

What actions must the sources take? (Flow Control)

What if some sources ignore these signals? (Enforcement)

These questions must be answered by every reactive congestion control scheme. Depending upon the choices made in answering each question, a variety of schemes have been proposed.

Comprehensive surveys by Gerla and Kleinrock (44) and Pouzin (110) discuss numerous congestion control schemes. However, the taxonomies they develop are oriented towards classifying flow control techniques, and are not appropriate in our work. Instead, we will base our survey on the questions raised earlier.

1.7.1.1. Congestion detection

How is a switch or source to detect congestion? There are several alternatives.

- The most common one is to notice that the output buffers at a switch are full, and there is no space for incoming packets. If the switch wishes to avoid packet loss, congestion avoidance steps can be taken when some fraction of the buffers are full (such as in the FQbit scheme discussed in Chapter 2). A time average of buffer occupancy can help smooth transient spikes in queue occupancy (116, 117).
- A switch may monitor output line usage. It has been found that congestion occurs when trunk usage goes over a threshold (typically 90%) and so this metric can be used as a signal of impending congestion (as in CIGALE or Cyclades) (61, 88). The problem with this metric is that congestion avoidance could keep the output line underutilized, leading to possible inefficiency.
- A source may monitor round-trip delays. An increase in these delays signals an increase in queue sizes, and possible congestion (66).
- A source may probe the network state using some probing scheme (for example, the packet-pair method described in Chapter 4).
- A source can keep a timer that sets off an alarm when a packet is not acknowledged 'in time' (108). When the alarm goes off, congestion is suspected (153).

1.7.1.2. Communication

Communication of congestion information from the congested switch to a source can be implicit or explicit. When communication is explicit, the switch sends information in packet headers (115) or in control packets such as Source Quench packets (112), choke packets (88), state-exchange packets (79, 120), rate-control messages (148), or throttle packets (70) to the source. Implicit communication occurs when a source uses probe values (75), retransmission timers (153), throughput monitoring (147), or delay monitoring (66) to indicate the (sometimes only suspected) occurrence of congestion.

Explicit communication imposes an extra burden on the network, since the network needs to transmit more packets than usual, and this may lead to a loss in efficiency. On the other hand, with implicit communication, a source may not be able to distinguish between congestion and other performance problems, such as a hardware problem (153). Thus, the communication channel is quite noisy, and a cause of potential instability.

1.7.1.3. Switch action

An overloaded switch can signal impending congestion to the sources, and, at worst, can drop packets. In virtual circuit network layers, hop level flow control can throttle upstream virtual circuits. In any case, the problem is to select either the source to throttle, or whose packets to drop. The fairness of the congestion control scheme depends on this choice made by the switches.

Schemes that rely on line usage as a congestion metric throttle all sources sending packets on an overloaded link (14, 110). This scheme is unfair in the sense that sources that use a small fraction of the congested link are punished as severely as large users. The Loss-load curve scheme (148) and the selective DECbit scheme (115) seek to overcome this problem by computing the share of the load due to each user, and selectively dropping packets from that source.

If buffer usage is a congestion metric, switches drop packets or throttle sources when a source exceeds its share of buffers. This share is determined by the buffer allocation strategy, and the rate at which the buffers are emptied depends upon the service discipline. Thus, the buffer allocation strategy and the service discipline jointly determine which sources are affected (with the three exceptions noted below). We now discuss these two aspects of switch behavior.

The optimal buffer allocation scheme has received a lot of attention in the literature, and is surveyed in (44). Overall, the conclusion in that survey is that a scheme where each output line is guaranteed a minimum number of buffers, and is not allowed to exceed a maximum, is fair and efficient. Other schemes, such as Input Buffer Limit (81), Channel Queue Limit (70), Buffer Class (45) and Matsumoto's X.75 proposal (91) discriminate against some sources, and are thus unfair.

Queue service disciplines are implemented in the servers on the output queues of packet switches (100). The choice of service discipline influences the kind of performance guarantees that can be made to network clients in reservation-oriented networks (156). We defer a discussion of the choice of service discipline in reservationless networks to Chapter 2.

We claimed earlier that the choice of which source or conversation is affected by congestion is implicitly determined by the service mechanism and the buffer allocation policy. Some exceptions are in the DEC scheme (115), the Loss-load curve approach (148) and the Random dropping scheme (90). These schemes explicitly try to be fair in allocating responsibility for congestion to sources. In the DECbit and Loss-load schemes, a switch maintains state information about the demand for bandwidth by each source and its fair share. When the demand exceeds the fair share, and the buffers are getting filled, that source is penalized. This is fair by design. Random dropping is an ad hoc technique to achieve fairness. The essential idea is that, when a buffer is full, the packet to drop is selected at random. Hence, users who use more bandwidth, and will probably occupy a larger fraction of the buffer, have a higher chance of packet loss. However, work by Mankin has shown that, by itself, this scheme does not reduce congestion (90).

1.7.1.4. Flow control

A number of congestion control schemes have been proposed that operate at the sources. These schemes use the loss of a packet (or the receipt of choke information) to reduce the source sending rate in some way. The two main types of schemes are choke schemes and rate-control schemes.

In a choke scheme, a source shuts down when it detects congestion. After some time, the source is allowed to start up again (63,88). Choking is not efficient, since the reaction of the sources is too abrupt. The stability of the choke scheme has not been analyzed, but the simple example given in §1.5.4 seems to indicate that the scheme is prone to oscillations.

In a rate-control scheme, when a source detects congestion it reduces the rate at which it sends out packets, either using a window adjustment scheme (117) or a rate adjustment scheme (17,75). The latter is particularly suitable for sources that do rate based flow control. The advantage of rate control schemes over choke schemes is that rate control allows a gradual transition between sending no packets at all to sending full blast. Rate control seems to be an attractive alternative for congestion control. Detailed analysis of the stability and efficiency of rate control are presented in Chapters 5 and 6.

1.7.1.5. Enforcement

Congestion is a social problem and, unless the performance perceived by each source is decoupled from the performance perceived by other sources, every source must cooperate in solving it. However, in the large non-trustable, non-cooperative networks of the future, such cooperation can no longer be assumed (118). Thus, solutions that are predicated upon cooperative sources are not satisfactory.

Many current schemes do not study enforcement. For example, in the DECbit scheme (117), if a source chooses to ignore the congestion avoidance bits set by a switch, then the other cooperative sources will automatically give up bandwidth to the ill-behaved source. The same is true for the Jacobson-Karels TCP scheme (63). This problem is avoided partially by the Random-drop scheme (90), and completely by the Loss-load curve method (148).

There is a need for solutions that will work in the presence of ill-behaved hosts. That is, a congestion control decision must not only be communicated to the sources, it must also be enforced. Of all existing proposals, only the ones based on the round-robin (such as Fair Queueing, or Earliest Due Date) service discipline have this property. This is discussed in Chapter 2.

1.7.2. Reservation-oriented networks

In reservation-oriented networks, network resources can be allocated at the start of each session. Then, the network can guarantee a performance level to a conversation by performing admission control. This can guarantee congestion control, but perhaps at the cost of underutilization of network resources.

One of the earliest schemes to prevent packet losses was developed for the Datakit network (41). Here, the network places a limit on the size of the flow control window of each conversation, and the connection establishment packet reserves a full window's worth of buffer space at each intermediate switch. Thus, every virtual circuit, once established, is guaranteed to find enough buffers for each outstanding packet, and packet loss is avoided.

The complementary scheme is to reserve bandwidth instead of buffers. This is the approach taken by Zhang in the Flow Network (154), by Ferrari *et al* in their real-time channel establishment scheme (37), by Topolcic *et al* in the ST-II proposal (134) and by Cidon *et al* for the PARIS network (16). In a hybrid scheme described in reference (24) a source makes a reservation for buffers at the beginning of a call, and a reservation for bandwidth before the start of each burst. This allows bandwidth to be efficiently shared, but each burst experiences a round trip time delay.

There are four major problems with any naive reservation scheme: scaling, queueing delay, underutilization and enforcement.

1.7.2.1. Scaling

For large and high speed networks, each switch needs a considerable amount of memory. For example, if the RTT is 70ms over a 1Gbps fiber, one may need as much as 70Mbits of buffering per conversation per switch! The two-window scheme proposed by Hahne *et al* (54) greatly reduces this memory demand, and makes buffer reservation quite attractive. Work by Mitra *et al* has shown that, theoretically, this requirement can be reduced even further (96). Another way to reduce buffer requirements is to require the user to obey some traffic profile (34).

1.7.2.2. Delay

In a proactive scheme, at overload, a full window could be buffered at the bottleneck. When this happens, the queueing delay could be unacceptable. Delays can be bounded by computing the worst-case delay at the time of call set-up, and doing admission control (37).

1.7.2.3. Underutilization

The major problem with reservations is that the network could be underutilized: an over-zealous admission control scheme could prevent congestion by allowing only a few conversations to enter. This is not acceptable. The crux of the problem lies in determining how many conversations can be admitted into the network without reducing the performance guarantees made to the existing conversations. This has been studied by Ferrari et al (37). One solution to the problem is to define statistical guarantees, where some degree of performance loss can be tolerated (35). Conversations with statistical guarantees can be statistically multiplexed. Other hybrid schemes have been described in section 1.4.1.

1.7.2.4. Enforcement

How can we ensure that conversations actually send data at their reserved rate? There are two types of enforcement mechanisms: those that act at the network access point, and those that act at each switch.

Some researchers, especially from the telecom community, have postulated the existence of Network Access Points (NAPs) where users (subscribers) can access a homogeneous switching fabric. The NAPs can monitor and shape user traffic. If some users violate their stated traffic envelope, their data can be dropped or violation-tagged (5). An efficient way to monitor, and, if necessary, reshape user traffic behavior to make it less bursty, is the leaky-bucket scheme (87, 128, 136, 141).

Other schemes do enforcement at each switch. This is through some form of round-robin-like queueing discipline at each switch. In the Flow network, the Virtual Clock mechanism is used (154), and in the real-time channel establishment proposal, the discipline is Earliest-Due-Date with distributed flow control (37). Ferrari and Verma (36) and Kanakia (68) have proposed mechanisms based on calendar queues (11) that have a similar effect.

1.7.3. Quality of service

One can view congestion control as being able to guarantee quality of service at high loads. There has been some previous work in guaranteeing quality of service in networks.

Postel made an early suggestion for reservationless networks, though this was not studied in any depth (113). Golestani's Stop-and-go queueing provides conversations with bandwidth, delay and jitter bounds, and is similar in spirit to Kalmanek *et al*'s Hierarchical Round Robin scheme (46, 47, 68). Ferrari *et al* have proposed schemes that deliver deterministic and statistical guarantees for delay, bandwidth, packet loss and jitter (37, 142). Other related work is that of Lazar, who has proposed a network that offers classes of service, where each class has a performance guarantee (83). Note that having only a small number of classes of service, instead of letting each user define its own performance requirements, makes implementation easier, but does not adequately satisfy our definition of congestion control. Work by Estrin and Clark to ensure that a conversation can determine a route that best satisfies its quality of service requirement (policy routing) is also relevant (19, 31).

1.8. Scope of the thesis

As our survey indicates, the study of congestion control is a large and rapidly expanding area. Since it is impossible to study the entire area in any depth in a single dissertation, we limit our scope to congestion control in reservationless networks and only at the multiple-RTT and faster than one RTT time scales. The techniques developed in this work can be used for the 'best-effort' data traffic in reservation-oriented networks.

We initially turn our attention to the problem of protecting well-behaved users from ill-behaved ones by allocating all users a fair share of the network bandwidth (Chapter 2). We show that this Fair Queueing discipline not only ensures that each user gets a fair share of the network bandwidth, but also enables users to probe the network state. After describing techniques for the efficient implementation of Fair Queueing (Chapter 3), we present a novel state probing technique that a source can use in networks of Fair Queueing servers (Chapter 4). A flow control algorithm based on control theoretic principles that uses the probe values to do predictive control is then described in Chapter 5. Chapter 6 provides simulation studies to back up our claims, and Chapter 7 presents our conclusions.

Our original contributions are in the theoretical modeling of the congestion and congestion control problems, deriving theoretically sound solutions, and using these to develop practical algorithms. These algorithms are tested using a network simulator (73) on a set of eight test scenarios.

(132). This problem was recognized quite early (22), references (44,110) provide a detailed review of these. control scheme (117). Control Protocol (TCP) (21,108) and to recover from it (63). and interactive video (35,154). endpoints, for example, two BSD sockets (21,23). *Virtual Circuits* (VCs) (such as in Datakit (41)). not make such demands (35). (68), increasing the load on the bottleneck (101). (63,114). standstill (132). The first definition is used in references (66,117), the second in reference (63), and the third in reference (85). over the resources (140). (37,48). (35,68). in a common pool (62). a day (13,105). data transfer (20,155), peak hours than during off-peak hours (7,111,130). the recent past (32,146), For example, the SPAWN system (146) users to declare their resource needs accurately (20). (80). been extensively studied by Sanders (121-123). (29,77,92). (29), determine the effect of its sending rate on the network (117). (54,63,96,117). (44), (51,52,104), (1,38,66,79,133,137,138). (63,117), (126,137,138). Examples are the Virtual Clock scheme (154), Delay-EDD (37) and Stop-and-Go queueing (47). game (125). (6), Sanders (121-123), Douligeris and Majumdar (25-27) and Lazar *et al* (59). several orders of magnitude larger (84). and perhaps have catastrophic effects (101,132). (112) themselves overload the congested network. of the recent past (45000 kbps to 56 kbps) (69). network allocates its resources (23). (43,55,106,107,115,139,149). criterion to adopt (44). Nevertheless, it is necessary to negotiation and enforcement mechanism (20,155). e.g. (63). the switches (117). Still others assume complete control but no control over source traffic (54). having hardware support for rapid switching (100) optimizing for the average case (18) removing signaling information from the data path (41) choice of scheduling algorithm (156) an efficient call processing architecture (131). peer entities (for example, (117)) For example, a Fair Queueing (23) switch has the responsibility for (66)). very bursty (52,104). and high bandwidth off-line bulk data transfer (12). Gerla and Kleinrock (44) and Pouzin (110) transient spikes in queue occupancy (116,117). (as in CIGALE or Cyclades) (61,88). and possible congestion (66). not acknowledged 'in time' (108). When the alarm goes off, congestion is suspected (153). the switch sends information in packet headers (115) or Source Quench packets (112), choke packets (88), state-exchange packets (79,120), rate-control messages (148), or throttle packets (70) probe values (75), retransmission timers (153), throughput monitoring (147), or delay monitoring (66) such as a hardware problem (153). (14,110). The Loss-load curve scheme (148) and the selective DECbit scheme (115) seek to attention in the literature, and is surveyed in (44). Other schemes, such as Input Buffer Limit (81), Channel Queue Limit (70), Buffer Class (45) and Matsumoto's X.75 proposal (91) packet switches (100). reservation-oriented networks (156). Some exceptions are in the DEC scheme (115), the Loss-load curve approach (148) and the Random dropping scheme (90). scheme does not reduce congestion (90). (63,88). either using a window adjustment scheme (117) or a rate adjustment scheme (17,75). of the future, such cooperation can no longer be assumed (118). For example, in the DECbit scheme (117), The same is true for the Jacobson-Karels TCP scheme (63). (90), and completely by the Loss-load curve method (148). developed for the Datakit network (41). This is the approach taken by Zhang in the Flow Network (154), (37), by Topolcic *et al* in the ST-II proposal (134) and by Cidon *et al* for the PARIS network (16). In a hybrid scheme described in reference (24) The two-window scheme proposed by Hahne *et al* (54) this requirement can be reduced even further (96). user to obey some traffic profile (34). at the time of call set-up, and doing admission control (37). (37). (35). their data can be dropped or violation-tagged (5). (87,128,136,141). In the Flow network, the Virtual Clock mechanism is used (154), distributed flow control (37). Ferrari and Verma (36) and Kanakia (68) have proposed mechanisms based on calendar queues (11) that have a similar effect. though this was not studied in any depth (113). (46,47,68). (37,142). performance guarantee (83). is also relevant (19,31). simulator (73). congestion (44). even if a universal standard such as ISO (12) Nagle (101) proposed a *fair queueing* (FQ) fairly (55,72). Recently Morgan (98) has examined the role such queueing and has been analyzed extensively (42,86). Similar work has been presented by Zhang (154); the fair queueing algorithm presented here (156). The max-min fairness criterion (43,55,115) (115). (this emulation discrepancy bound is proved in reference (50)). The IP protocol (109) does have a field for type-of-service, Recent work (57) raises questions about the { 1~+~{1 over {N}} ({k sup 2} ~+~ k(2 epsilon -1)) right }~~for~~{P sub F} left { 1~+~{1 over {N}} ({k sup 2} ~+~ k(2 epsilon -1)) right } found in reference (23). There are few analytical results here, but Hahne (55) is crucial in the design of switch algorithms (101). queueing algorithms to do so (125). Prue and Postel (113) (94,95) the packet to be dropped is chosen at random (90). (39,56,124,154). (154). at 1.2 Gbps (69). (50,57,89,93,125,126). such as NetBlt (17), and the technology for this is well known (65). (93). since N_{ac} is strictly non-increasing in (t_{chk}, t). has been proposed (57). (58,78). traffic in the sense of (57), TCP protocol (108) with the modifications made by Jacobson (63). we used the REAL network simulator (73) Along with the work done by McKenney (93) and Heybey et al (57), this work provides the practitioner Queueing, such as the Virtual Clock algorithm (154). `i_min[k]` is the position in `max` of the swap(min(a), min(b)); swap(i_max(i_min(a)), i_max(i_min(b))); swap(i_min(a), i_min(b)); pointer consistency, i.e. that `i_min[i_max[a]] = a` and `i_max[i_min[a]] = a`. `i_min[n] = i_max[n] = n` . min(ptr) = data; for (; (ptr/2 >= 1) &&(min(ptr) < min (ptr/2)); ptr /=2) min(1) = INFINITY; save = i_min(1); max(save) = -1; if (min(a) < min(a/2)) smaller = a/2; smaller_son = (min(lson(a)) < min(rson(a))) ? lson(a) : rson(a); if (min(smaller_son) < min(a)) smaller = smaller_son; attendant inefficiencies (50). of Bessel functions (135); of data traffic in current networks (13,51), (129). (144,145). analyzed in (129). (143,145). flow control scheme (17). scheme modified an idea described by Clark et al. for NETBLT (17), and sent to the source using a state exchange protocol (120). (64,97). so, unlike TCP (63,153), (64). (T_0, T_1) if in every time interval $(t_1, t_2) \in (T_0, T_1)$, $(T_0 + 2R(0), T_1)$. conditions are trivially satisfied in $(T_0 + 2R(0), T_1)$. For example, the Jacobson-Karels version of TCP (63) and so will lose bandwidth when V increases (64). Jain's delay based scheme (66) will respond (129). In recent work (156), we have shown that the FQ discipline is similar to the Virtual Clock (154) and Delay-EDD (37) service disciplines. though not using packet pairs, is described in reference (53), (63,108) or Jain's delay-based approach (66). As mentioned in (2), Separation Theorem (3). In recent work, Mitra et al (96,97) Processor Sharing (50), (13,51). (B/2 mi nb (k) mi sk pl mu (k) RTT(k) pl mu (k+1) RTT (k + 1)) (B/2 mi n hat sub b (k) mi sk pl mu hat (k) RTT(k) pl mu hat (k+1) RTT hat (k + 1)) (B/2 mi n hat sub b (k) mi sk pl 2 mu hat (k) RTT(k)) (B/2 mi n hat sub b (k) mi sk pl 2 mu hat (k) RTT(k)) (B/2 mi n hat sub b (k) mi sk pl 2 mu hat (k) RTT(k)) Similar studies are mentioned in (30). chosen to lie inside the range (0,B), such as the second method of Liapunov ((102) page 558). known and robust technique (49). bold "x" eq left (pile { nb ~ above mu ~ above mu sub -1 ~} right) $\mathbf{G} = \begin{bmatrix} 1 & -Rtt & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ $\mathbf{H} = \begin{bmatrix} Rtt \\ 0 \\ 0 \end{bmatrix}$ $\nu_1 = \begin{bmatrix} 0 \\ \omega \\ 0 \end{bmatrix}$ $\mathbf{C} = (\ 0\ 0\ 1\ )$ $E((\hat{x}(t) - x(t))^T(\hat{x}(t) - x(t)))$, to yield many other types of filters (49). right) (nu sub 1 sup T (theta) nu sub 2 (theta)) ~ right } eq left (pile { { Q ~ S~ } above { S sup T ~ R~ } } right) ~delta $\mathbf{Q} = E \begin{bmatrix} 0 & 0 & 0 \\ 0 & \omega^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ $\mathbf{S} = E \begin{bmatrix} 0 \\ \omega\nu_2 \\ 0 \end{bmatrix}$ system. We follow the derivation in (49) (pg 249). bold "K" (k) eq (bold "G" SIGMA (k) bold "C" sup T pl bold " S ") (bold (82,152,158). (151), found in reference (76). We also make the fluid approximation (1), This approximation is commonly used (8,133), and both analysis (97) and simulations show time interval (t, t+J), but in fact can work together (67). scheme proposed by Hahne *et al* (54) is attractive. (6,9,10,27,126). work by Jain (116) and Jacobson (63), In recent work, Ko *et al* (79) described originally by Agnew (1), and since extended by Filipiak (38) and Tipper *et al* (133). Vakil, Hsiao and Lazar (138) Vakil and Lazar (137) have considered Robertazzi and Lazar (119) and Hsiao and Lazar (60) in the benchmark suite described in reference (23) is comparable to that of the DECbit scheme (117), (63,117)), messy and only partially understood (157). experimental high speed network testbed (69). lk1 eq 1 over { RTT(k)} (B/2 mi nhat-1 mi sk pl mu RTT(k) pl mu RTT (k)) lk1 eq 2 mu mi lk pl 1 over { RTT(k)} (B/2 mi nhat-1) lambda (2) eq 2 mu mi mu pl 1 over { RTT (2)} (B/2 mi B/2) lambda (3) eq 2 mu mi mu pl 1 over { RTT(3)} (B/2 - B/2) We follow the description of LQG presented in (102) (pg. 835). bold "Q" eq left (pile { { 1 ~0 ~0 ~ } above { 0 ~0 ~0 ~ } above { 0 ~0 ~0 ~ } } right) (99) (93,117,154), (33). as in reference (12). traffic (13,51,105). (33,103), (4,28) (73,74). transport flow control protocol (108,150), (63,127,154,157), the DECbit scheme (114-117). as implemented in XNS's SPP (150) or in TCP (before 4.3 Tahoe BSD) (108), Jacobson and Karels' (JK) modified TCP (63) (15,115-117), (63,71)). protocols such as NETBLT (17) congestion avoidance scheme (66). (115,148), (see reference (115) and the simulation results below), on the description in references (115,116). of 40 and 570 bytes in the Internet (12). Floyd and Jacobson in (39,40)). (39,40). (54). (though global tradeoffs are still possible (6)). and has also been studied exhaustively in references (39,40). (101)). by Prof. S. Singh and Prof. A. Agrawala (129)). laws is a well known problem (49). thesis by D. Verma (141).