Adaptive Admission Congestion Control

Zygmunt Haas
AT&T Bell Laboratories
Room 4F-501
Crawfords Corner Rd
Holmdel, NJ 07733
(201)-949-7319

Abstract

In this paper, we describe a novel congestion control scheme for high-speed networks. The scheme is based on periodic transmission through the network of time-stamped sampling packets that sense the congestion status of the network. Upon reception, the sampling packet delays are calculated, averaged, and used to determine the state of the network. The information on the state of the network is then used to drive the network adaptive admission control. The major advantage of the proposed scheme over conventional congestion control techniques is that it copes with traffic surges that are shorter than the network round-trip delay. This is achieved by controlling traffic admission with continuous estimate of the network state. The scheme is targeted towards networks that carry aggregated traffic, and can be applied to ATM-based networks.

1 Introduction and Motivation

We define network congestion as:

A state of a network, in which some network resource is oversubscribed/overdemanded, **and in which the availability of that resource decreases because of the oversubscription/overdemand.**

In other words, network congestion results in a real loss of the overutilized resource. In most cases, the resource utilization measure is assumed to be the *goodput*[1].

We treat network congestion as an *end-to-end* issue. In other words, we distinguish between congestion and *contention*. Contention is a state in which some network resource is oversubscribed/overdemanded. However, in contrast to congestion, contention does not necessarily and **directly** result in loss of the resource. Thus contention within a switch is not (necessarily) network congestion. Contention can, however, lead to congestion, especially if it lasts long enough.

Figure 1 demonstrates the phenomenon of loss of resources due to congestion (the "congested" curve) and the behavior of an ideally congestion-controlled network (the "ideal curve"). In practice, relatively minor loss of the resource due to congestion[2], as shown by the "controlled" curve, is

[1]The *goodput* is the rate of useful data delivered by the network, whereas *throughput* is the total rate of data delivered by the network and includes duplicated, erroneous, and misdelivered packets.

[2]In fact, some loss of the resources is inevitable in networks relying on detection of loss of the resource to control congestion; i.e., the *reactive* congestion-control methods.

satisfactory. The idea behind congestion control is to get as close as possible to the ideal curve **given the knowledge or ignorance of the traffic pattern that the network is expected to experience.** This is rather a crucial observation. If the traffic pattern is a very predictable one, the congestion control problem is simple to solve. Unfortunately, in most cases the traffic is either completely unknown or insufficiently specified. Thus the congestion control scheme needs to be robust enough to perform relatively well in "nearly all" cases. Therefore, a single scheme may not be adequate to solve congestion control problems and a combination of methods is needed, each emphasizing different aspects of congestion control.

As noted, oversubscription of resources alone is not enough to create network congestion. There must exist some mechanism that results in loss of the resource. In other words, in a token ring network with nodes with "infinite queues" and no time-out mechanism, there is no loss of resources as the offered traffic grows above the network capacity, since the unsent packets simply queue at the network entrances but the network continues to carry traffic in the amount of network capacity. On the other hand, in "Aloha-type" networks, as the offered traffic increases above 18% ([1]), the network throughput rapidly decreases.

The proliferation of congestion control schemes comes in an era of considerable research on very high speed (hundreds-of-Mbps and Gbps) networks. This correlation is not mere coincidence. It is believed that congestion-control in high-speed networks is a much more difficult problem than in low-speed networks, due to lower coupling between the transmitting and the receiving ends in high-speed networks. In other words, it is more difficult to control a system with long propagation delay.[3] The scheme presented in this work increases the synchronization between the transmitting and the receiving ends, thus eliminating some of the effect of long propagation delay.

Congestion-control schemes may be classified into two categories: *pro-active* and *reactive* methods. Reactive methods monitor the network performance (either directly, by monitoring the state of the resource, or indirectly, by monitoring some other correlated parameter), and upon detection of developing congestion, take some action that drives the network out of this state. Reactive methods rely heavily on feedback from the network; thus, these methods can successfully cope with congestion in the network only when the congestion develops with a time constant on the order of at least the round trip delay. Among the reactive methods are dynamic windowing ([2,3]) and schemes that drop excessive (possibly prioritized) traffic ([4]).

Pro-active methods, as opposed to the reactive schemes, constantly activate some mechanism that reduces the possibility of the network getting into a congested state.[4] Some commonly known pro-active methods are bandwidth reservation, traffic shaping/smoothing ([5]), and admission control (leaky bucket [6], for example).

The main disadvantage of reactive methods is the relatively slow reaction time to building congestion, since these methods rely on some indication from the network on increasing congestion. This indication is usually provided only after the actual transmission of traffic takes place. Pro-active methods, in general, do not suffer from this disadvantage, since these schemes are continuously applied. However, most pro-active methods that rely on an open-loop control scheme are not robust enough and cannot, therefore, guard the network against all possible traffic patterns.

[3] Propagation delay is measured in units of packet transmission time. Thus, even though the propagation delay in seconds remains constant, the propagation delay in packet transmission time increases linearly with transmission rate for fixed packet length.

[4] One can draw an analogy from the medical field: reactive methods are to pro-active schemes as curative medicine is to preventive medicine.

In the next sections, we present a novel approach to congestion control. This scheme relies on continuous feedback from the network to drive the network admission control. Thus the scheme combines the advantage of closed-loop control (usually used in reactive methods) with the continuous activation feature (usually present in pro-active methods).

2 Network Model

Our network model is shown in Figure 2. We concentrate on high-speed communication networks with a large number of users. Each user may have access to some small amount of the total link capacity. In other words, we anticipate that the link capacity far exceeds the sourcing/sinking capability of an individual user. Consequently, a single user is not able to **significantly** change the (congestion) status of the network. For example, consider 2.5 Gbps links and 155 Mbps single user access. A single user may utilize only about 6% of the total link bandwidth. Consequently, because of the large number of users, the resulting aggregated link traffic is rather **slowly** varying.[5] Moreover, we assume that the users are connected to the network by a network interface. What we actually mean by "network interface" can be a variety of devices: multiplexers, routers, or gateways. Furthermore, we assume that the congestion control mechanism is implemented in network interfaces that are considered part of "the network." Consequently, a user has no access to the congestion control scheme and cannot gain advantage by disobeying the scheme's rules.

Our model assumes some connection-oriented services. However, the scheme can be modified to accommodate a connectionless environment.

3 Adaptive Admission Congestion Control

The Adaptive Admission Congestion Control (AACC) scheme is an adaptively adjustable rate-based algorithm, controlled by the changes in the averaged measured delay through the network. The delay is acquired by sampling packets that are periodically inserted into the network. Time stamping of the sampling packets and subsequent calculation of the changes in the delay enables computation of the control parameter without transmitter and receiver clocks synchronization.

The basic idea behind the proposed scheme is to maintain a single parameter per each connection/path[6] within the network that estimates the level of congestion of this connection/path. The parameter is continuously updated by *periodic sampling packets* that are sent between the source and the destination[7] and that sense the congestion status of the network. The scheme uses the value of this parameter to adjust the transmission rate from the source to the destination, or to encourage/defer initiation of new data transfers on the path.

In our scheme, shown in Figure 3, the source and the destination ends maintain constant synchronization for the purpose of congestion control. This synchronization is achieved by the periodic exchange of sample packets. These sample packets that carry time stamps indicating the

[5] Especially for links "deep" in the network. What we mean by "slowly varying" is, roughly, a less than 10% change in one round trip delay. Also, we assumed no correlation between users' traffic. This may be incorrect in the case of a global event; e.g., major network deadlock.

[6] The path could be the ATM's Virtual Path.

[7] Periodic exchange of states was proposed in [7] to improve the performance of transport protocols. Here we use the idea of periodic sampling packets, in which the packets themselves sample the network to provide estimation of the network status. Both ideas can, however, be combined, so that only one kind of periodic exchange is implemented.

time the packet was entered into the network, are inserted into the same queues and follow the same path as regular packets. Upon arrival of a sample packet at its destination, the packet's time stamp is extracted from the packet format, and the packet *virtual delay* is calculated. *Virtual delay* is the difference between the transmitter and the receiver clocks. It is not a true delay, since the transmitter and the receiver clocks are unsynchronized.[8] A *virtual delay* is piggy-backed on a sampling packet going in the reverse direction. As the reverse-directed sample packet is received, the *virtual delay* is compared with the *virtual delay* of the previous sampling packet and the difference in *virtual delays* is calculated. These differences in *virtual delays* are averaged over several packets and the information is used, along with the information on average sample packets' arrival rates, to estimate the level of network congestion.[9] The estimate of the network congestion is expressed by a single parameter, β, with $\beta \approx 1$ indicating low congestion and $\beta \approx 0$ high congestion. The parameter β is used to control the network admission control.

Thus the proposed scheme is composed of three components:

1. Admission rate-control through a constantly adjusted congestion parameter,
2. Periodic exchange of sampling packets to acquire an estimation of the current congestion status of the network, and
3. Updating of the congestion parameter, based on the congestion status.

We emphasize that the AACC scheme is a framework, rather than a design. Thus, several implementations of the AACC scheme are possible, with advantages of one implementation over another depending on the specific performance requirements.

Adaptive admission rate control

The admission in the AACC scheme is based on a rate-control model and is adaptively controlled by the value of β; i.e., the time interval between packet insertion into the network is controlled by β. The larger the β, the shorter the inter-packet gap is; i.e, the larger the value of β, the more traffic is admitted into the network.[10] In the example in section 4, a packet is admitted every clock tick (clearly, a clock tick period is longer than a packet length). The clock ticks are controlled by the value of β and are also referred to as inter-packet gaps. An example of the inter-packet gap as a function of β is given by Equation 2. Immediately upon β adaptation, the inter-packet gap is also adjusted.

This handling of admission has the advantage of stabilizing the traffic arrival process (for instance, batch arrivals patterns are broken down), and the disadvantage of slightly increased waiting time at the network interface. This traffic stabilization of the admission scheme (which has some flavor of traffic shaping) may play a central role in the design of congestion-control schemes, since it reduces the dependency of the design on the traffic arrival process.

In a variation of admission control, β is used as a dynamic token allocation parameter in the *leaky bucket*-like[11] scheme ([6]). The amount of token is based on the changes in the value of β and on the actual number of transmitted packets during the last update interval, *quota*. Thus, if in the

[8] *Virtual delay* has no meaning, except when used to determine changes in the delay.

[9] By using the delay differences rather than absolute values, the need to synchronize network clocks is eliminated.

[10] Sampling packets are not subject to admission control.

[11] The leaky bucket technique is an implementation of admission control, in which some number of tokens is allocated during each update interval. Each transmitted packet removes a token from the pool. When no more tokens are available, no more packets are inserted into the network.

n^{th} update interval there were x transmitted packets, $\beta_n = 0.5$ and $\beta_{n+1} = 0.55$, the number of tokens for the $(n+1)$ interval will be $\min(x \cdot (1+\epsilon), quota)$, where ϵ is the token increment that corresponds to the increase of β from 0.5 to 0.55. To reduce the traffic burstiness, the token arrivals should be equally spaced in time, similarly to the clock-ticks described above.

Periodic exchange of sampling packets

Sampling packets are periodically exchanged between any two network interfaces with open connections. These connections may correspond to the Virtual Path Identifiers in ATM networks, which aggregate many Virtual Circuits, and thus amortize the scheme's overhead over many connections. Both the transmitter and the receiver have local clocks that are assumed to have the same ticking period, yet they may not be synchronized. The sampling packets are inserted into the network by the network interface, which stamps the packets with a time-stamp. The packets then travel through the network and are undistinguished from regular data packets. The network interface at the receiving end of the network removes the packets from the data stream and calculates the *virtual delay,* which is the difference between the time stamp and the value of the local clock. This *virtual delay* is piggy-backed onto the sampling packets directed in the opposite direction. The transmitter pairs two consecutively arriving packets and calculates the differences in the *virtual delays*, which equals to the differences in the actual delay[12]. To show this, assume that the transmitter and the receiver clocks are labeled c_t and c_r respectively, and that packets n and $n+1$ admitted at times t_n and t_{n+1} experience delays d_n and d_{n+1} through the network, respectively. Packet n gets time stamp $c_t(t_n)$, while packet $n+1$ gets time stamp $c_t(t_{n+1})$. At the receiving end, the two packets arrive at receiver clock $c_r(t_n + d_n)$ and $c_r(t_{n+1} + d_{n+1})$. The receiver calculates the difference between the time-stamps and the value of the receiver clock at the time of arrival, the *virtual delay*, which equals to $c_r(t_n + d_n) - c_t(t_n)$ and $c_r(t_{n+1} + d_{n+1}) - c_t(t_{n+1})$, for the two packets. When these *virtual delays* are received at the transmitter, their difference is calculated, which equals to the difference between the actual delays of the two packets:

$$c_r(t_{n+1} + d_{n+1}) - c_r(t_n + d_n) - c_t(t_{n+1}) + c_t(t_n) = \delta + (d_{n+1} - d_n) - \delta = d_{n+1} - d_n, \qquad (1)$$

where δ is the sampling packet period interval, i.e., $t_{n+1} - t_n = \delta$. The last equation follows from the fact that both clocks have the same ticking period; i.e., $c_t(x+\epsilon) - c_t(x) = c_r(y+\epsilon) - c_r(y) = \epsilon$. The differences in the actual delays (between any two consecutive packets) are averaged for the purpose of reduction of noise level and reduction of the effect of lost samples. By employing this procedure, each end knows the congestion status of the network in the correct direction.

It should be pointed out that sampling packets exchange is only one possible way to implement the AACC scheme. In particular, another possibility is to build time-stamping information into the regular packets structure.[13] However, by using sampling packets, congestion control is implemented *on top* of a network; i.e., requiring no modification to the existing subnet. Moreover, this type of congestion control can be implemented on a subset of a subnet; i.e., if part of the high speed network is dedicated and utilized by a group of users, they may decide to use this congestion control scheme without coordinating with the rest of the subnet. For example, a private network may be created in a public ATM SONET-based network that uses its own end-to-end congestion control scheme

[12] For example, the difference in *virtual delays* is calculated for packets 1 and 2, packets 3 and 4, packets 5 and 6, etc.

[13] Using regular data packets to perform the sampling operation has the advantage of reducing the AACC scheme overhead. Of course, if there are no regular packets to be sent, sampling packets must still be generated. However, the overhead in this case is created when the actual utilization is low.

requiring no changes or special features from the public network. This is an interesting possibility for the future public high-speed networks.

Adaptation of the congestion parameter

The congestion parameter, β, is adjusted based on the delay of the sampling packets. Since the delay of the sampling packet is a rather noisy variable, some kind of filtration is necessary. Also, since β depends on both the utilization and the delay, there is no direct $\beta(delay)$ function. However, a function of increments of β as a function of increments in delay can be composed. A simple example of such a function is shown in Figure 5. More elaborate functions may be required to achieve different levels of performance. In particular, a more practical function would be one with increments in β that increase with an increase in the changes in delay; i.e., a larger $\Delta(\beta)$ for a larger $\Delta(delay)$.

Discussion

The proposed scheme has several advantages, over conventional congestion control schemes. First, because of the periodic exchange of sample packets, the estimation of congestion is performed continuously and not just at the time of an actual transmission.[14] This feature eliminates the problem that occurs with transmission of data that is shorter than the round trip delay[15] and with initialization of new data flows[16] (referred to as the "cold start" phenomenon). Second, the periodic exchange of sample packets decouples the congestion control mechanism from the actual data transfer (acknowledgement mechanism, for instance), reducing the effect of positive feedback on the congestion control mechanism. When congestion occurs, the sample packets are also subject to excessive delay, which serves as a direct indication of congestion. In contrast, in most feedback-based schemes when congestion occurs, the information about the congestion is also delayed, increasing the congestion even more,[17] and the congestion-control scheme is activated only upon time-out. Third, by using the delay of the path within the network as a direct indication of congestion, the actual congestion event is detected and treated. Some schemes use other indirect indications (such as missing packets or negative acknowledgements), leading to improper or delayed congestion detection. Fourth, the admission control shapes the input traffic, reducing the dependency of the scheme's performance on the actual traffic arrival model. Fifth, the proposed scheme can be combined with other protocols that rely on periodic exchange of information ([7]), thus reducing the bandwidth and processing overhead associated with the periodic exchange of samples. Moreover, the rate at which the sample packets are generated should be such that the total overhead of the scheme is minor.[18] Finally, the scheme is an end-to-end scheme and does not require changes in the subnet itself;[19] it is easy to implement and can be integrated with other congestion control means, leading to overall superior performance.

---

[14] Some schemes, for example, adaptive windowing, stop operating when the channel idles.

[15] Most congestion control methods perform well in an environment where the changes of congestion are with time constant on the order of round trip delay. When, however, a short duration traffic is presented to the network, these congestion control schemes have no means to control the source since the congestion is detected after the transmission is actually completed.

[16] Virtual circuits, for example.

[17] The reason for the difference is the means by which the congestion information is acquired: sampling packets, in our case, acknowledgements/negative acknowledgements coupled with a time-out mechanism, for example, in other schemes.

[18] The scheme does, however, use bandwidth, which is expected to be an excessive resource in future networks, to solve a network control problem.

[19] This may be of crucial importance, since high-speed networks may be built before the actual traffic mix is known or can be predicted.

4 An example of Adaptive Admission Congestion Control

In order to demonstrate the capability and performance of the AACC scheme, we have analyzed a simple one-queue network example shown in Figure 4. The single queue in the network is fed from two queued sources, whose arrival is Poisson with parameters λ_1 and λ_2. The outputs of these queues are rate-controlled by parameter β_1 and β_2, respectively. This rate-control is performed by adjusting the inter-packet gap according to the following function:

$$\text{inter} - \text{packet gap} = \min(\frac{1}{\beta}, quota) - 1, \tag{2}$$

where *quota* is the interval, in packet transmission time units,[20] at which the value of β is updated. In other words, when $\beta = 1$, the inter-packet gap is zero, and when $\beta = 0$, the inter-packet gap is maximal and equals to *quota*.

The periodic exchange of the sampling packets (the second attribute of the AACC scheme) is performed as follows: The update packets are inserted directly into the network at rate r_{update}, bypassing the admission control mechanism, and are collected at the network exit; i.e., the output of the single queue.

The third attribute of AACC, the update of β, is performed according to the following double-threshold algorithm, shown in Figure 5. When the current average change in the delay[21] (i.e., $\delta(delay)$), is less then θ_1, β is increased by Δ_p. Similarly, if the average change in delay is larger than θ_2, β is decreased by Δ_n. If the average change in delay is in the range between θ_1 and θ_2, β is not changed. The parameters, θ_1, θ_2, Δ_p, and Δ_n control the performance of the scheme. More specifically, the Δ-s control the speed with which β adapts to changes in the traffic arrival rates (the λ-s). In other words, the larger the Δ-s, the faster the scheme adjusts itself to large changes in traffic patterns, but the more coarse the adjustment is as well. Adjustment of the parameters is strongly dependent on the actual network topology. We will present here only simulation results for the example in Figure 4.

An additional parameter used in the simulation below is the *time-limit*. It is the maximum delay above which a packet is declared lost. It can correspond to the *build-out* delay ([8]) in real-time traffic.

4.1 Dynamic and static performance

Dynamic performance is a time trace of a network behavior as a function of time, when some sample input is applied. We will use dynamic behavior to assist us with the explanation of how the scheme works. The operational conditions of the scheme for this example are as follows: $\theta_1 = 2$, $\theta_2 = 2$, $\Delta_n = 0.05$, $\Delta_p = 0.005$.

A sample of the two inputs (λ_1 and λ_2) as a function of time are shown in Figures 6 and Figure 7. The traces of the inputs were generated by Markov Modulated Poisson Process (MMPP)[9] as shown in Figure 8. The parameters of the MMPP were: $n = 333$, state no. 1 corresponds to $\lambda = 0.03$ and state no. n to $\lambda = 1$. The transition time between states is exponentially distributed with mean time equal to 3 units. The trace of β_1 and of the network queue size is shown in Figure 9 and

[20] All delays are normalized to packet transmission time. Thus packet transmission time is assumed to be 1.

[21] The average change in the delay is computed by subtracting the *virtual delays* of two consecutive sampling packets and is averaged over the last *quota* interval.

10. As long as the sum of λ_1 and λ_2 is less than 1, $\beta_1 = 1$[22]. As the sum of the inputs increases (congestion starts building up), the network delay increases and β_1 decreases accordingly to allow less and less traffic into the network. Finally, as the input traffic goes down again, β_1 returns to value 1. The size of the network queue shows how the scheme prevents queue build-up. For comparison, a trace of an uncontrolled network (subject to the same input pattern) is shown in Figure 11. The uncontrolled scheme performs well untill the point $\sum \lambda_i \approx 1$, where the network queue builds-up rapidly. Clearly, the uncontrolled case results in an excessive packet loss.

Static one-dimensional behavior is measured by fixing the arrival rates ($\lambda_1 = \lambda_2 = \lambda$) and measuring the *goodput* of the system. This is done for a broad range of arrival rates. In our example, goodput is throughput with a delay shorter than the *time-limit*. The simulated performance of the AACC scheme is shown in Figure 12. Additional parameter values used are: *time-limit*=10, $quota = 100$, and $r_{update} = 0.1$[23]. For comparison, the uncontrolled case[24] is also shown in Figure 12. The uncontrolled case has a sharp goodput degradation point that starts around $\rho = 0.9$[25]. The AACC case does not experience such degradation of performance. Instead, the goodput stabilizes at about 0.75[26] [27]. The AACC scheme performs worse than the uncontrolled scheme around $\rho = 0.8$. This is due to the sampling packets overhead in the AACC scheme. Of course, considering the performance degradation that occurs if no congestion control is exercised, this penalty is more than justified. Also, as discussed later, the penalty can be reduced by properly adjusting the parameter r_{update}.

4.2 Packet loss

As defined in section 1, network congestion translates into the loss of a real resource: the *goodput*, in our case. For real-time traffic, the decrease in goodput is created by packet loss,[28] since delay in a congested network increases and some packets are "late" for their scheduled replay time. Figure 13 demonstrates the dramatic improvement of packet loss probability in the AACC scheme. In this simulation, the "build-out" delay was chosen in such a way that the AACC scheme would result in a maximum of 5% packet loss ([8]). As can be observed from the Figure, the packet loss remains stable above $\rho \approx 0.8$. On the other hand, the packet loss for the uncontrolled case increases to nearly 100% when offered traffic approaches 1. Note that the difference is an important one. One might have thought that congestion in the controlled and in the uncontrolled cases corresponds to the same penalty, since in the uncontrolled case the packets are lost in the network, while in the controlled case the packets are never admitted to the network, thus lost at the network interface. This is incorrect. The network investment in packets that are admitted and later lost in the network is wasted. Lost packets not only do not contribute to the goodput, but further reduce the goodput

[22] Except the initial period of time when β_1 increases towards 1.

[23] r_{update} of 0.1 means that a sampling packet is sent from each source to the destination every 10 time-units.

[24] In the uncontrolled case, there is no congestion control; the packets are queued at the arriving queues and forwarded into the network at the maximum rate.

[25] ρ is the total packets departure rate from the network.

[26] Obviously, the network "loses" packet in the range where $\sum \lambda > \rho$. However, the goodput does not decrease as more traffic is offered to the network; i.e., the congestion is controlled.

[27] There is a 20% overhead associated with the periodic sampling packets. Thus the actual network "throughput" is around 0.95. It is assumed that the sampling packets do not contribute to the goodput.

[28] A packet is declared lost if its delay is longer than some "build-out" delay that the real-time system was designed for. Since longer "build-out" delay decreases the probability of packet loss, systems are designed for maximal "build-out" delay that does not cause loss of real-traffic interactivity.

by increasing the delay of other packets – this is the positive feedback phenomenon of network congestion.

The performance of the AACC scheme is worse than the uncontrolled case in the area near $\rho = 0.8$ in Figure 13. This is caused by the overhead of the sampling packets: 20% in the case shown in the Figure. Reducing this frequency of the sampling packets exchange results in reduction of the packet loss probability in this region.

4.3 Effect of Propagation Delay

Reactive congestion control schemes do not usually perform well in networks with long propagation delay due to the slow feedback in these networks. Since the scheme proposed here also relies on feedback, its performance is limited by the propagation delay. However, the proposed scheme decreases the effect of propagation delay with two mechanisms: first, the admission control "staggers" the packets in time, thus reducing burstness and avoiding clustering; second, the continuous feedback increases the synchronization between the two communicating entities, thus reducing the effect of rapidly changing traffic patterns (new connections, for instance). Figure 14 shows the effect of changing the network propagation delay, while other parameters are fixed.[29] The propagation delay has little effect on the *goodput* for low network utilization. However, for high utilization, an increase in propagation delay decreases the goodput. The peculiar behavior of the goodput for long propagation delays – peaking and then declining – is due to the fact that the scheme cannot fully control the admission mechanism when the propagation delay is too long. Thus there are short incidences of congestion resulting in delays longer than the *time-limit*. This phenomenon can be "corrected" by reducing the threshold variables, on the expense of the *goodput* reduction. Note, nevertheless, that above $\rho = 1.0$, the *goodput* stabilizes, and there is no further decrease.

4.4 Effect of threshold parameters

There are four threshold variables, Δ_n, Δ_p, θ_1, and θ_2, whose values are based upon the required performance of the scheme and on the traffic statistics.[30] In the case of the example in question, the performance is determined by the maximum tolerable delay, *time-limit*. In principle, the more bursty the network traffic is, the larger the distance between the two threshold values θ_1 and θ_2 should be. Also, the more strict the performance requirements are (a smaller percent of lost packets, for instance), the lower the absolute values of the thresholds should be. However, too low thresholds have the undesirable effect of lowering the *goodput*. This is illustrated in Figure 15. In this simulation, θ_2 was set constant at 2 and θ_1 was varied. The too low value of θ_1 results in a loss of *goodput*, since β is kept conservatively low and not enough traffic is admitted to the network. On the other hand, high values of θ_1 result in too much traffic admission, which leads to increased delay and excessive traffic loss.

The Δ-s regulate the speed and the stability of the adaptive adjustment of β. A small Δ_p increases the stability of the adjustment process, but also results in a slow increase in network

[29]The parameter in this graph is the propagation delay measured in packet transmission time. Thus, 200 units means that the round-trip delay through the network is 200 packet transmission times. Other parameters: *quota* = 100, $r_{update} = 0.02$, $\Delta_n = 0.005$, $\Delta_p = 0.05$, $\theta_1 = 4$, $\theta_2 = 8$, and *time-limit=10*.

[30]The dependency on traffic statistics is reduced by the rate-controlled admissions used in the AACC scheme.

traffic.[31] Δ_n controls the severity of the scheme response in the case of violation of the performance measure. Thus, for example, if the penalty for packet loss is large, Δ_n should be large accordingly. However, if some percentage of packet loss is permissible, Δ_n may assume smaller values. The lower the Δ_n, the more stable the network traffic is, but too low of a Δ_n may result in too slow of a response to increasing congestion, and thus loss of resources.

An interesting approach that can be incorporated into the proposed scheme is to adaptively change the threshold parameters. With this approach, the system "learns" the slow variations in the traffic patterns, and adjusts the system parameters to the traffic statistics. The use of neural networks for congestion-control ([10]) is an extension of this idea.

4.5 Effect of frequency of sampling packet exchange

The bandwidth required for the sampling packets is the overhead of the AACC scheme; as shown in Figure 16,[32] the penalty in *goodput* decreases "nearly" proportional with r_{update}, the frequency of sampling packet exchange. Thus the frequency of the sampling packet exchange should be minimized. On the one hand, the frequency should not be smaller than $1/quota$, the frequency of the adaptive β update algorithm. (*quota* should be minimized, and is determined by the complexity of the adaptive algorithm, the processing power of the hardware that executes the algorithm, and the maximum number of simultaneous connections.) On the other hand, the number of samples per update interval *quota* should be large enough to allow for the statistical variations in the network status. Thus the decision on the actual values for r_{update} and *quota* depends on the network topology and the network traffic.

5 Summary and concluding remarks

In this paper, we have proposed a new congestion-control scheme based on periodic exchange of sampling packets and on adaptive admission control. The scheme is designed for an end-to-end distributed operation; i.e., it is meant to work in the network interfaces. The benefit of this design is that no changes or additions are required within the subnet. An important feature of the proposed scheme is that it can cope with traffic surges that are short compared to the round trip delay. This feature is clearly lacking in most of the current schemes found in the literature. Our scheme has the advantage of both, *pro-active* congestion control schemes (since the scheme is based on *continuous* monitoring of the status of the network) and *reactive* schemes (since it relays on the feedback from the network). The scheme can be used in ATM-types of networks that carry aggregated traffic. We have demonstrated the scheme performance for a simple one-queue network.

References

[1] N. Abramson, "The ALOHA System - Another Alternative for Computer Communications," *Proc. FJCC*, pp. 281-285, 1970.

[31] This is especially evident at the start of network operation (see Figure 9). This phenomenon is sometimes called "slow start." It has the beneficial effect of stabilizing the network behavior.

[32] The simulation was done using the following parameters: $\Delta_p = 0.005$, $\Delta_n = 0.05$, $\theta_1 = 4$, $\theta_2 = 8$, and *time-limit*=10. In the graph, 2% overhead means 1 sampling packet per 100 data packets for each one of the two connections.

[2] Raj Jain, "A Timeout-based Congestion Control Scheme for Window Flow-controlled Networks," *IEEE Journal on Selected Areas in Communications,* SAC-4(7), October 1986.

[3] V. Jacobson, "Congestion Avoidance and Control," in Proceedings of *Sigcomm'88,* Stanford, CA, August 16-19, 1988.

[4] A. E. Eckberg, Jr., D. T. Luan, and D. M. Lucantoni, "Meeting the Challenge: Congestion and Flow Control Strategies for Broadband Information Transport," *Globecom'90,* Dallas, TX, November 27-30, 1989.

[5] M. Murata, Y. Oie, T. Suda, and H. Miyahara, "Analysis of a Discrete-Time Single-Server Queue with Bursty Inputs for Traffic Control in ATM Networks," *IEEE Journal on Selected Areas in Communications,* vol.8, no.3, April 1990.

[6] Moshe Sidi, Wen-Zu Liu, Israel Cidon, and Inder Gopal, "Congestion Control through Input Rate Regulation," *Globecom'90,* Dallas, TX, November 27-30, 1989.

[7] K. Sabnani, M. H. Nguyen, and C. D. Tsao, "High Speed Network Protocols," 6^{th} *IEEE Int. Workshop on Microelectronics and Photonics in Communications,* New Seabury, MA, June 6-9, 1989.

[8] Z. Haas and R. D. Gitlin, "On the Packet Size in Broadband Integrated Networks," Proceedings of *Infocom'91,* Bal-Harbour, FL, April 9-11, 1991.

[9] H. Heffes, "A Class of Data Traffic Processes – Covariance Function Characterization and Related Queuing Results," *Bell System Technical Journal,* vol.59, no.6, July-August 1980.

[10] Atsushi Hiramatsu, "ATM Communication Network Control by Neural Networks," *IEEE Transactions on Neural Networks,* vol.1, no.1, March 1990.

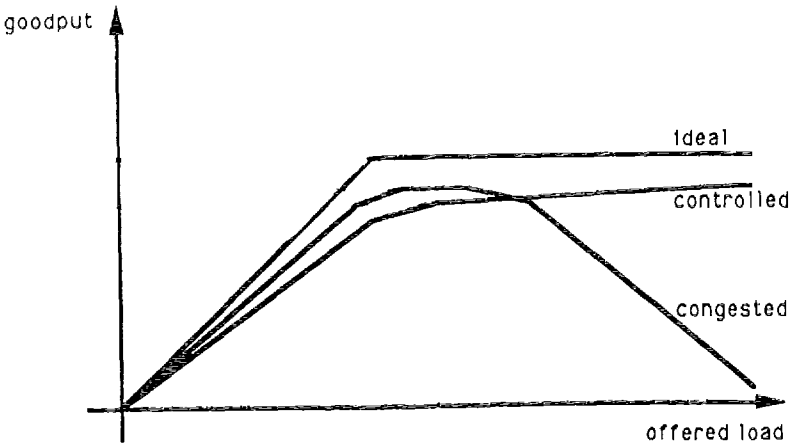

Figure 1: Congestion and congestion control

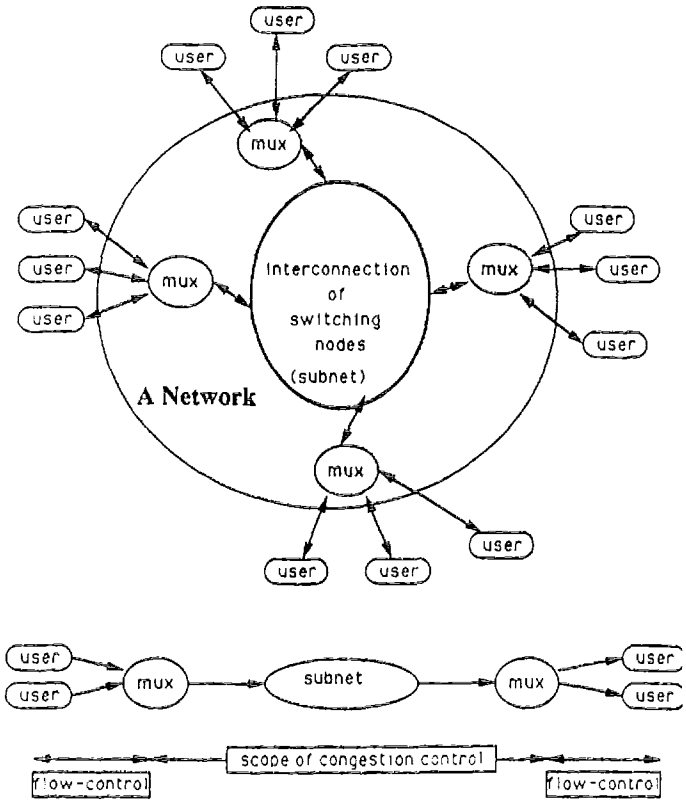

Figure 2: The network model

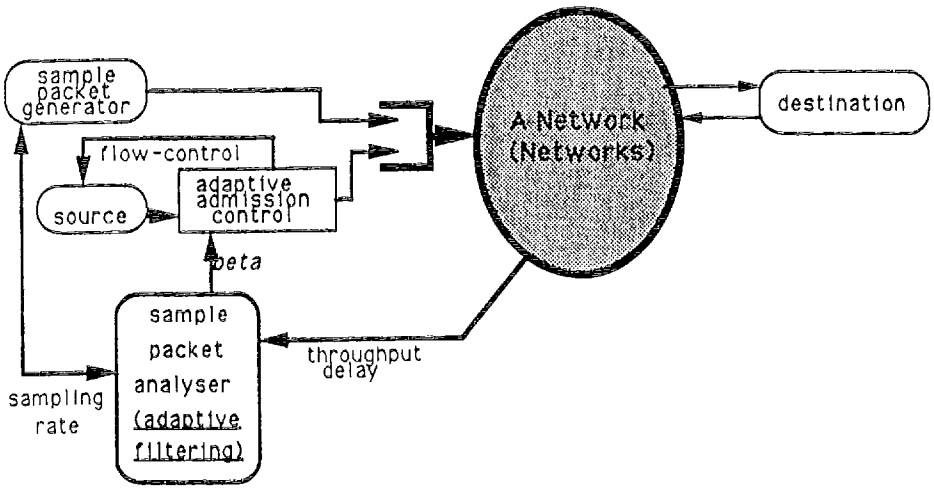

Figure 3: The adaptive admission control scheme

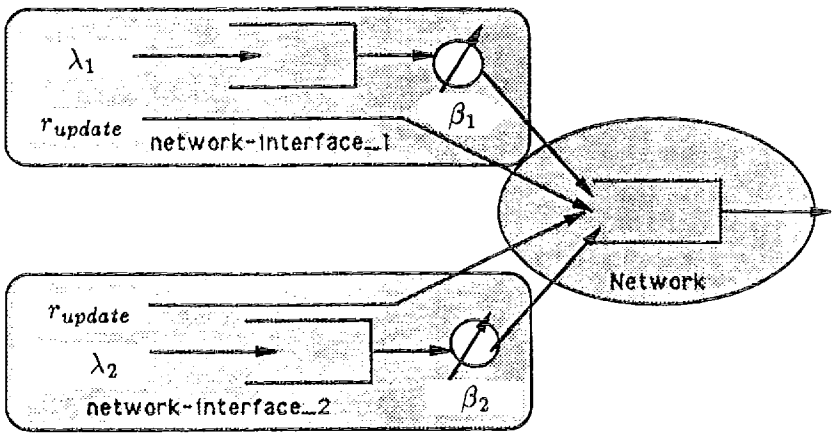

Figure 4: Simple one-queue network example

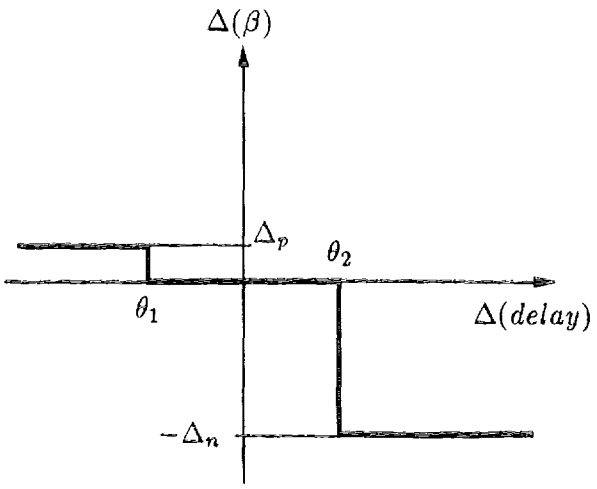

Figure 5: The double-threshold β update algorithm

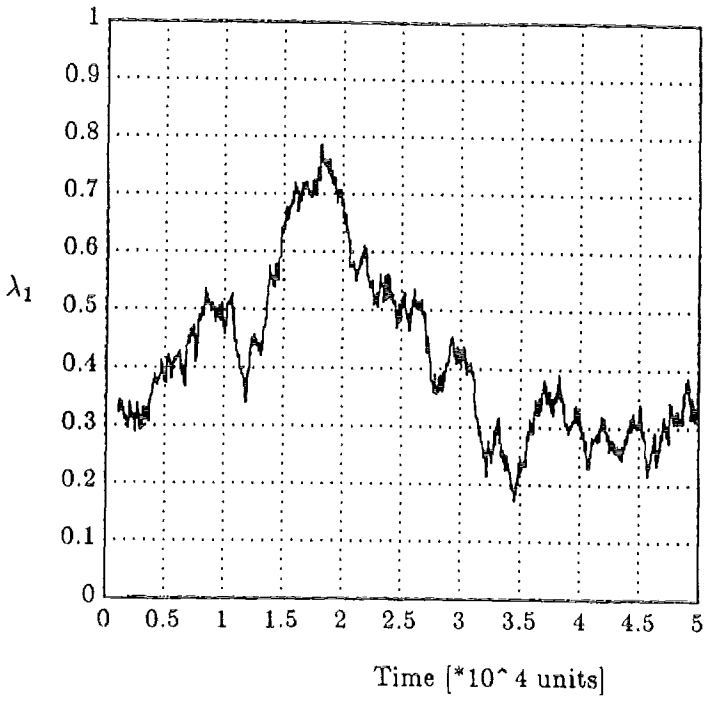

Figure 6: Dynamic trace: trace of λ_1

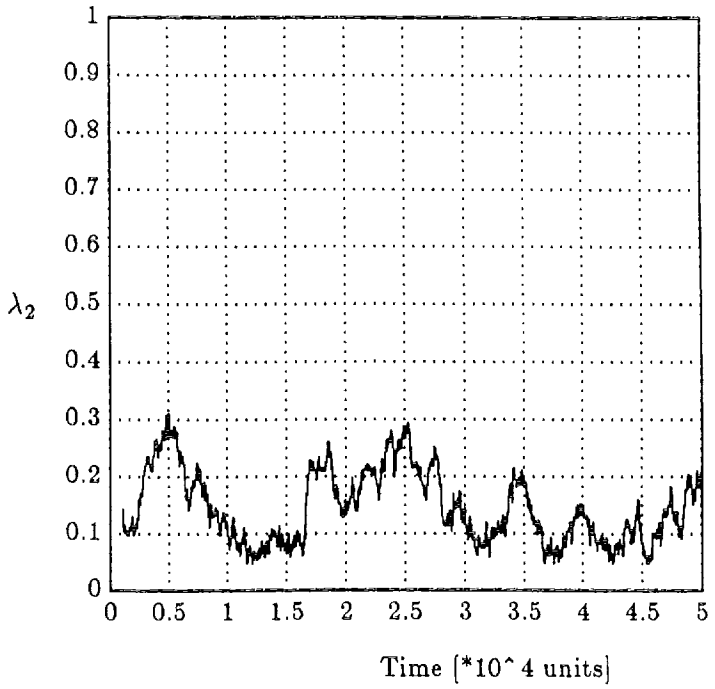

Figure 7: Dynamic trace: trace of λ_2

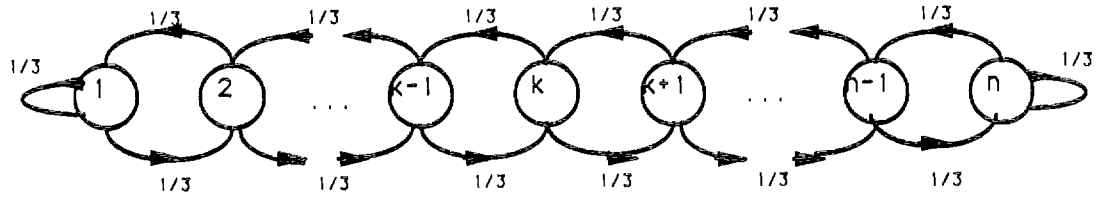

Figure 8: MMPP used in generating the input for dynamic trace

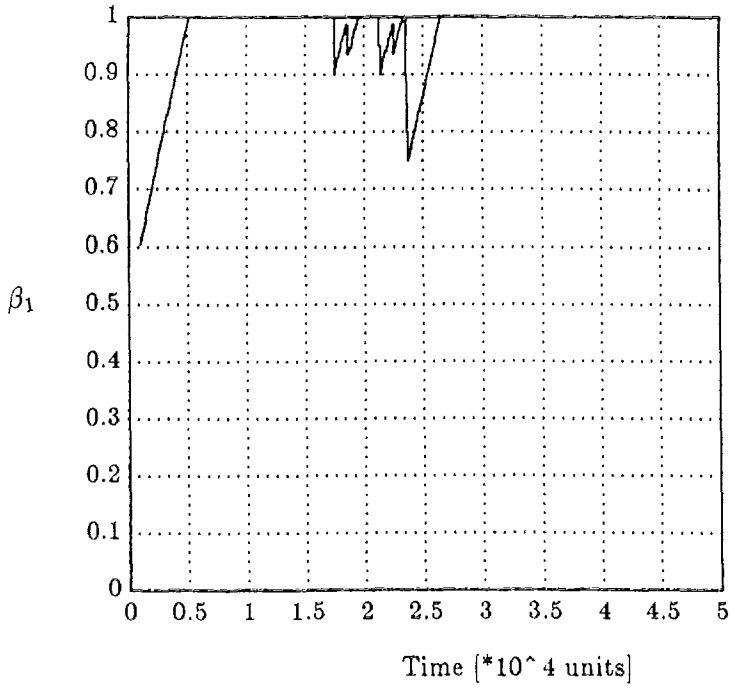

Figure 9: Dynamic trace: trace of β_1

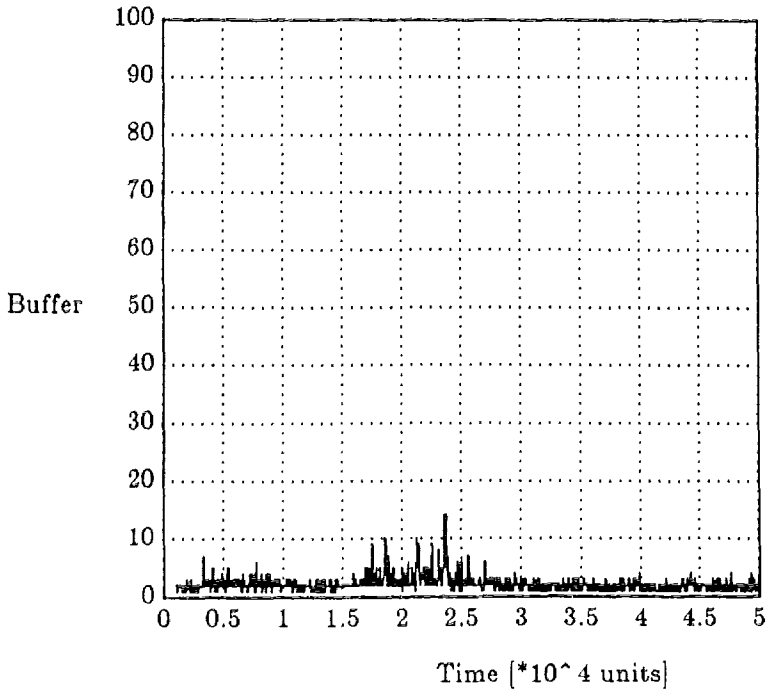

Figure 10: Dynamic trace: trace of the network queue

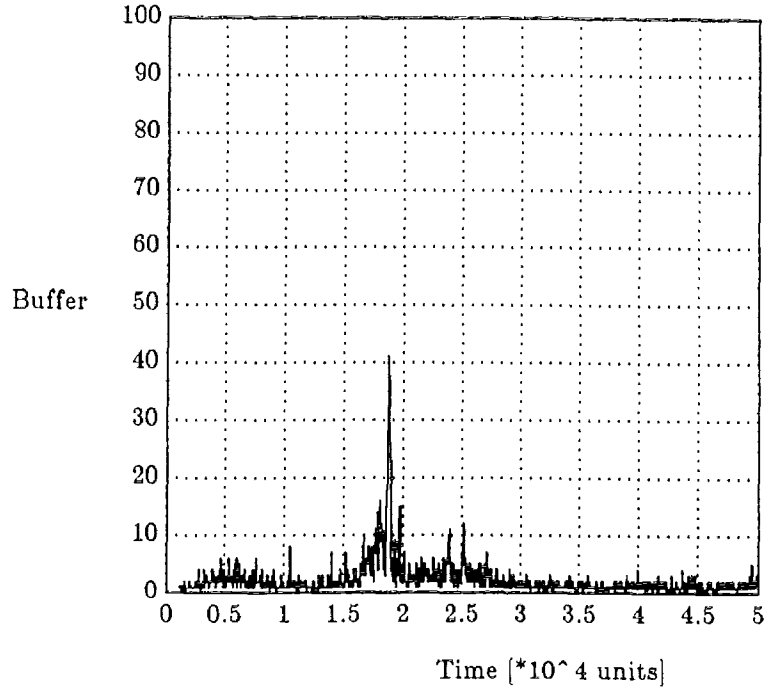

Figure 11: Uncontrolled dynamic trace: trace of the network queue

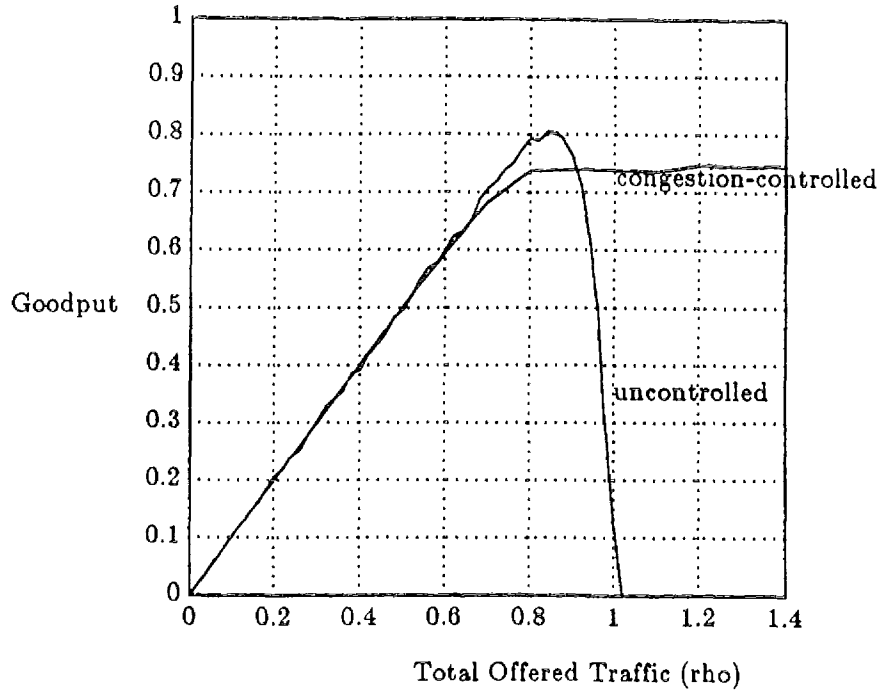

Figure 12: The AACC scheme and the uncontrolled case static performance

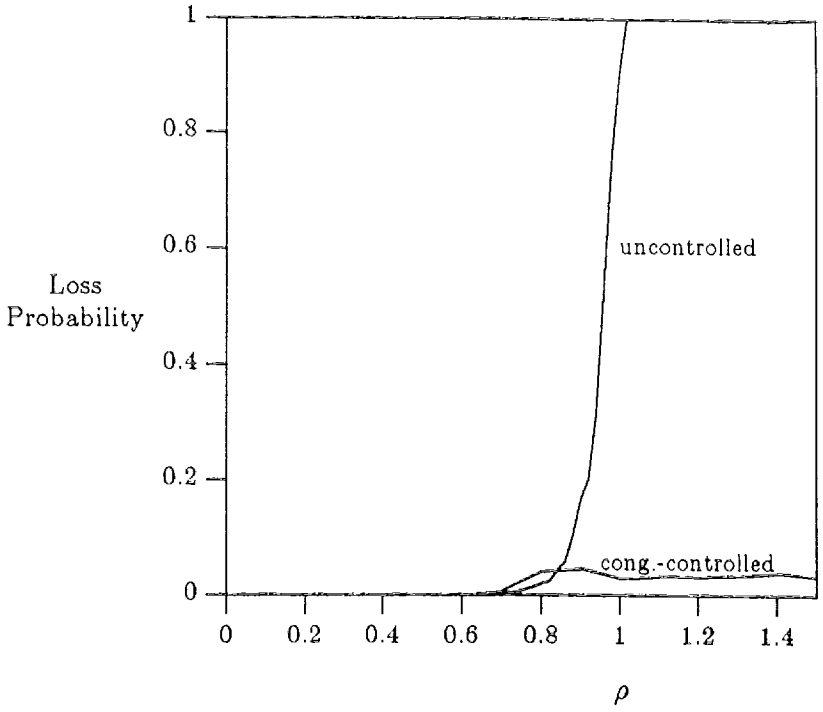

Figure 13: Packet loss probability vs. offered traffic

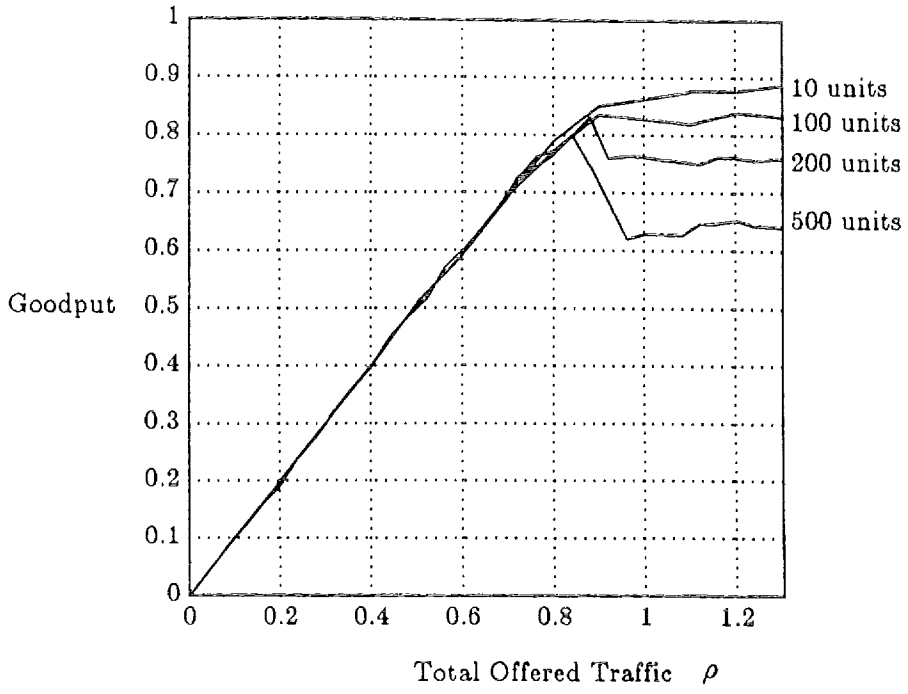

Figure 14: Effect of propagation delay on the AACC scheme performance

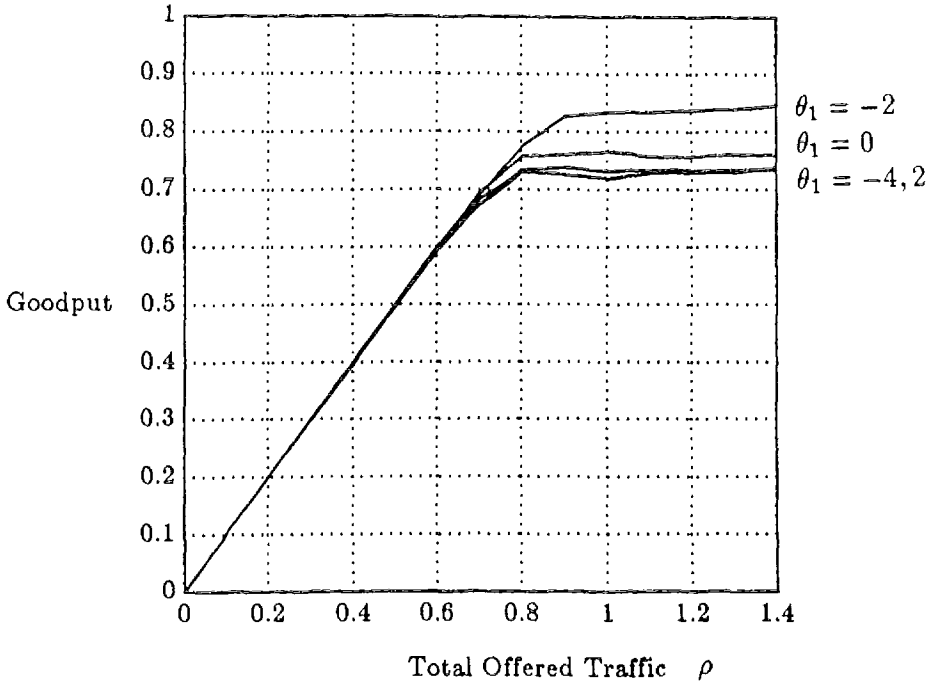

Figure 15: Effect of changing θ_1 on the *goodput*

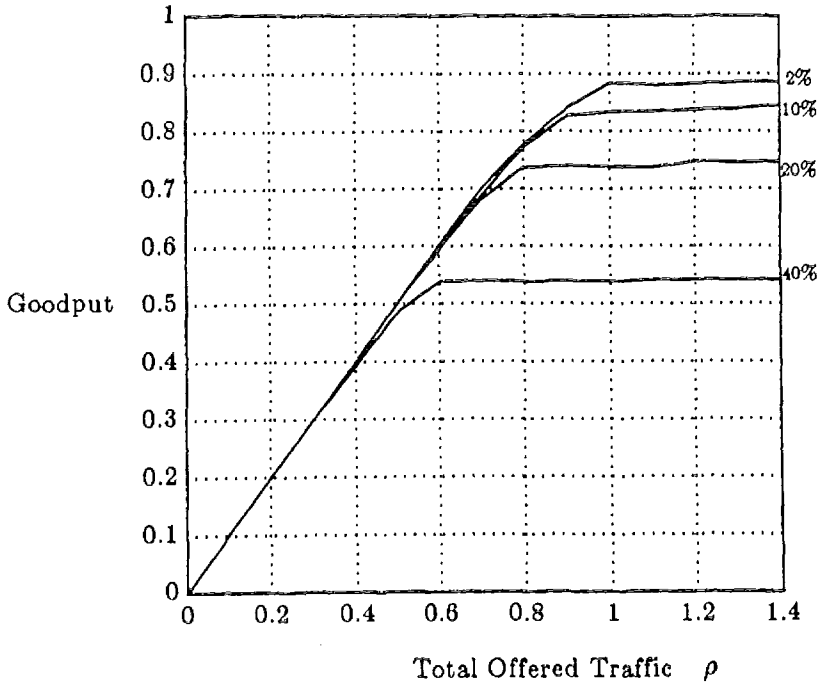

Figure 16: Effect of sampling packet overhead on performance