Bandwidth Management and Congestion Control in plaNET

The plaNET 1-Gb/s wide-area network holds lessons that apply to any high-speed integrated network — including ATM-based systems.

by Israel Cidon, Inder Gopal and Roch Guérin

Packet-switched networks have changed considerably in recent years. One factor has been the dramatic increase in the capacity of communication links to greater than 1 Gb/s, representing a significant increase over typical links in today's packet-switched networks [1]. A second factor is the altered nature of traffic transmitted through these networks. Both of these factors have a significant impact on the design of network protocols and control procedures. It is now accepted that packet-switched networks, or such variants of packet switching as the Asynchronous Transfer Mode (ATM) [2-4], will form the basis for multimedia high-speed networks that will transmit voice, data, and video through a common set of backbone nodes and links.

In this article, we elaborate on the protocols and mechanisms necessary for network bandwidth management and congestion control. We draw heavily on the lessons learned from the design and implementation of the plaNET network. However, we believe that most of the conclusions are general and can be applied to other high-speed networks, including ATM-based systems.

The overall plaNET architecture is described in a variety of other papers, e.g., [5, 6]. The pla NET architecture is based on the previous PARIS system [1]. A plaNET network will be deployed in several different Gigabit networking testbeds, including the AURORA testbed sponsored by the National Science Foundation/ Defense Advanced Research Projects Agency (NSF/DARPA) [7, 8], and the IBM/BellSouth joint study [9]. Thus, the ideas presented in this article will be experimentally tested in real environments in the near future.

Briefly, plaNET is a high-speed packet-switching system for integrated voice, video, and data communications. In order to achieve low packet delay and high nodal throughput, plaNET implements the packet-handling functions in dedicated high-speed hardware. Network control functions (which are mostly implemented in software) make use of the fast packet-handling services in order to facilitate their operations.

ISRAEL CIDON is a faculty member at the Technion, Haifa, Israel

INDER GOPAL is Manager of the Broadband Networking Department at the IBM Thomas J. Watson Research Center.

ROCH GUÉRIN is a member of the IBM Thomas J. Watson Research Center.

The plaNET architecture embodies the key notion of *transparency* [5]. This means that it is possible to transmit information through the network in a variety of different formats, including PARIS-style [1, 10] variable- size packets and ATM-style [2-4] fixed-size cells. Packets can be routed using the Automatic Network Routing (ANR) format (a self-routing scheme, described later), the label-swapping format used in ATM, or a multicast format. The idea is to use the routing format most closely matched to the incoming traffic stream. It is argued that the transparent capability of the network offers significant advantages in terms of flexibility, performance, and function over pure ATM systems [5].

Since the various mechanisms are closely coupled, it is often impossible to adequately motivate any single component without having a general understanding of the overall system. Thus, we initially provide a general overview without going into too much detail on any single component. In subsequent sections, we elaborate on the individual components the system comprises.

Overview

The design of the congestion control mechanism is driven mainly by the nature of the carried traffic. We expect to support a wide variety of different traffic types over plaNET, with different requirements in terms of bandwidth, packet loss, delay, number of connections, etc. The network will have to provide adequate levels of service to all these traffic types. Since the network has finite capacity, this means some ability to allocate resources (link bandwidth, buffer space, switch capacity, processing, etc.) among the contending traffic streams in a "fair" manner. In the rest of this article, we attempt to make these notions clearer.

Conventional mechanisms for controlling congestion within the network are usually based on window mechanisms [11]. Such mechanisms typically rely on the end-to-end exchange of control messages in order to regulate traffic flow. The control messages, sometimes with additional congestion information added by the intermediate nodes, are used as feedback by the source node to regulate

0163-6804/91/$01.00 1991© IEEE

its traffic. In high-speed networks, the propagation delays across the network typically dominate switching and buffering delays. Thus, the feedback from the network is usually outdated, and any action the source takes is too late to resolve buffering or switching congestion. This argues for mechanisms that do not rely so heavily on network feedback. It is also important that the congestion control mechanism operates at the speed of the link. For this reason, computationally intensive control schemes are less desirable than simple schemes that can be easily implemented in high-speed hardware.

The nature of traffic also affects the design of the congestion control. While today's data traffic can usually be slowed down in order to cope with network congestion, it is likely that the real-time nature of the traffic in broadband networks will require some level of bandwidth guarantee. Real-time traffic (e.g., voice, video, and image) has an intrinsic rate determined by external factors outside the control of the network. Typically, this rate can be estimated by the network prior to the establishment of the connection. The ability to slow down such sources is usually very limited. Note, however, that the packet arrival process is stochastic, so there is no guarantee that over short periods the source will keep to the specified average rate.

The above factors suggest a congestion control mechanism that does not react too dynamically to network conditions. Instead, it uses knowledge of the extrinsic parameters associated with the connection, together with knowledge of the available capacity in the network, in order to ensure that a connection accepted into the network can be accommodated. Once the call is accepted into the network, controls at the source force it to conform to these prespecified parameters (and attempt to change them if they are no longer appropriate). Buffer management and scheduling functions ensure that traffic flow within the network is regulated in conformance with the prespecified classes of service.

The congestion control mechanisms described in this article operate at two different levels. One set of controls (termed "connection control") operates at the connection, or call, level (we use the terms interchangeably). The other set of controls (termed "steady-state control") operates at the packet level. In the next few paragraphs, we briefly sketch out the operations and interrelationship between connection and steady-state controls.

Connection Control

Connection control comprises the functions related to the setup of a call. These include the call characterization function, which categorizes and quantifies the network resources required by the call; the path selection and admission function, which makes a decision on whether or not to permit a new call access to the network and, if it is admitted, determines the path over which it is to be routed; the call setup function of establishing the connection and updating associated tables and state information; and directory and other application-level functions, which perform important services such as the location of remote network resources.

Our approach to connection control is based on the use of a routing topology database similar to the one in ARPANET [12]. Basically, each node maintains a routing topology database with link weights reflecting the traffic over each link. When link weights change substantially, updates flow to every node using a fast broadcast algorithm [13]. At the call setup time, the source node attempts to characterize the new call. It determines characteristics that define the type of call, the destination, whether or not a bandwidth guarantee is required, and, if so, an "equivalent capacity" measure that captures the traffic loading that the call will generate. Taken together, these characteristics capture the call's requirements from the network. The source node then uses the information in its local topology database to compute a path (we use path and route interchangeably) capable of carrying the traffic and providing the level of service required by that traffic type. If no suitable path can be found between the source and destination, the call will be blocked.

Once a path is computed, a call setup procedure is typically initiated. As part of the procedure, an end-to-end call setup packet flows over the path and is copied by the intermediate nodes along the path. The call setup procedure actually "locks" up the capacity, ensuring that nodes on the computed path are aware that some capacity on their links is now being used by the new connection. These nodes then update their picture of the bandwidth usage on their adjacent links based on the bandwidth information in the call setup packets. This updated information may change the link weights and trigger a new broadcast. In plaNET, we employ an efficient and fast method for performing the link utilization update, which uses a fast tree broadcast function. Together with the speed of the network, this algorithm reduces the problem of transient differences between the network image of the topology databases and the actual network state.

Details on the distributed aspects of maintaining a topology database in each node can be found in [13]. For the purpose of this article, we assume that every node has a complete updated view of the entire network, including accurate information on the loading of all the links. In the section "plaNET Connection Control," we discuss the details of connection control, focusing on three key aspects: call characterization, path computation and call admission, and call setup.

Steady-State Control

Steady-state control consists of the mechanisms that are applied after connections have been accepted into the plaNET network. These mechanisms monitor, control, and react to the steady-state flow of traffic. They ensure that the traffic generated into the network behaves as assumed by the connection control procedures and enforce the different classes of service that are supported within a plaNET network. These controls are applied both at the access points to the network as well as within the network. The controls at the network access points consist of a rate control mechanism and a traffic estimation module. Within the network, intermediate nodes implement different scheduling and buffering policies to enforce the plaNET classes of service.

Connections are regulated at the input to conform to prespecified parameters. This input control is performed through a variant of the leaky-bucket mechanism [14]. However, unlike the mechanism

Real-time traffic has an intrinsic rate determined by external factors outside the control of the network.

The long-lived mode is most appropriate for predictable and steady sources.

in [14], we permit packets to queue at the input and introduce a spacer to ensure that excessive queuing does not take place at bottleneck links. Details on the scheme are provided later. The parameters of the leaky bucket can be adjusted in order to conform as much as possible to the characteristics of the call. Therefore, by regulating the admissions and routing of new sources into the network, it is possible to provide a performance guarantee to all network traffic.

In some cases, it is not possible to accurately predict, *a priori*, the parameters associated with a source. In such cases, it is important to have the ability to estimate the traffic flow and react dynamically to changes in characteristics of the connection. We incorporate this dynamic traffic estimation ability into plaNET. The mechanism to renegotiate the connection parameters with the network is an extension of the call setup/takedown procedure and is described in that section.

In some instances, it may not be desirable to ensure that all traffic passes through an input throttle. One example is datagram traffic, which is difficult to predict. Another example is a variable-rate video coder, where the rate varies by orders of magnitude between still-picture transmission and rapid motion. In such cases, the strategy adopted in plaNET is to permit traffic to bypass the input throttle, but to identify this traffic explicitly. The treatment of this traffic is different in the intermediate nodes, as it is not guaranteed the same level of performance as reserved traffic.

The buffer management in the intermediate nodes ensures that the various traffic types get the appropriate levels of service. The scheme divides the buffer into several distinct First-In First-Out (FIFO) queues, which are served according to a non-preemptive fixed priority order. The priority reflects different delay and loss sensitivities between different connections. Our scheme also allows discrimination between different types of traffic within the same connection (and hence, within the same FIFO buffer), using a buffer threshold policy to be described later.

In the section on "Steady-State Controls," we present details, focusing on three key aspects: rate control, traffic estimation, and intermediate buffer management.

plaNET Connection Control

As mentioned earlier, in this section, we focus on three aspects of connection control: call characterization, path computation and call admission, and call setup.

Call Characterization

When a connection request is made, the connection is first assigned to a particular service class based on its requirement and characteristics. If the service class requires bandwidth reservation, the amount of bandwidth required by the connection is determined as a function of its traffic characteristics. We refer to this process of classifying and quantifying the connection request as call characterization.

Our model for the user population of a future high-speed network is a heterogeneous one. We will continue to see traditional telecommunication network users, who will generate relatively predictable and steady traffic streams (albeit at much higher rates than today). At the other end of the spectrum, we will see interactive bursty data users whose peak-to-average traffic ratio may be several orders of magnitude. We will also see traffic that falls somewhere between these two extreme ends of the spectrum, generated by new high-speed multimedia applications that will combine data, video, and image.

In order to accommodate these different types of users, we provide three modes of bandwidth reservation. The first mode is the "long-lived" mode, which reserves bandwidth for the duration of the call. The second mode, the "on-demand" mode, adopts a burst mode of reservation, while the third, "non-reserved," mode does not perform any reservation. Based on the nature of the traffic, the user can choose (and mix) the three options.

As mentioned, the long-lived mode involves reservation of bandwidth for the duration of the call. This option is most appropriate for predictable and steady sources, though it is possible to use it for bursty sources. The basic issue here is to quantify the amount of bandwidth that needs to be reserved for the connection. For steady streams, this quantification is relatively simple. However, for more bursty sources, the problem becomes nontrivial. Reservation at the peak rate is too costly, and reservation at the average rate cannot guarantee acceptable operation during periods where the source transmits at the peak rate. The notion of "equivalent bandwidth," introduced in the next section, permits us to compute the amount of bandwidth required in order to permit adequate levels of service. In some cases, users may only require a long-lived guarantee for the minimal amount of bandwidth required for the continuation of the connection at an acceptable quality. Additional traffic may be sent using one of the other two modes.

The on-demand category represents the next level of reservation in terms of bandwidth guarantee. It offers less of a guarantee than does the long-lived mode, but offers advantages in terms of bandwidth usage efficiency.

The essential idea is that the user reserves bandwidth only when it needs to transmit. When the user has a burst of information to send, it initiates a procedure that reserves bandwidth equal to the peak rate at which it expects to transmit during that burst. It uses the bandwidth as long as needed and then returns it to the bandwidth pool. The assumption is that the path of the call is precomputed, and additional bandwidth reservations and releases are performed on the given path according to the actual behavior of the call.

This class of traffic still requires the exercise of admission control at the time of connection setup. Calls are blocked if the network has insufficient resources to handle them. Thus, here too it is important that a call's bandwidth demand be adequately quantified through its equivalent capacity. The primary advantage of this mode of operation is that the equivalent capacity can be much closer to the average bandwidth than the peak bandwidth. Thus, more calls can be admitted into the network.

The primary disadvantage of this mode of operation is that the bandwidth may not be available when requested. Once an on-demand request is granted, it is assured that the bandwidth will be available as long as it is not released. Therefore, the task can be completed once started. On the other hand, during times of congestion some users may be either denied the requested bandwidth or given a smaller amount as specified by the setup message. If this happens too frequently, it might be reasonable to assume that the current route is too congested, and a rerouting procedure might be invoked.

Another disadvantage is that the reservation phase needed before each activity period (burst) may slow its start-up time. In plaNET, special care was given to the acceleration and efficiency of the setup/takedown bandwidth processes. However, no system can overcome the propagation delays that dominate in wide-area networks. In some cases, we take an optimistic approach and assume that the request is likely to be granted. The traffic is sent prior to the reservation confirmation as "red" traffic under the assumption that the request will be granted. If the reservation is eventually denied, the source will stop transmitting.

In the non-reserved mode, no bandwidth is allocated to the user. The user does not obtain any guarantee of service and simply fits its traffic into any free bandwidth that happens to be available. This traffic may get "bumped" when no bandwidth is available. Since such users do not reserve any bandwidth, they may transmit even at times of congestion, and therefore may violate the designed network maximum operational point. Therefore, it is crucial to protect reserved services from the impact of non-reserved ones. This protection is implemented at the level of buffer management, where service to non-reserved packets should not be at the expense of reserved traffic even under extreme load situations.

There are two types of traffic that fit into the non-reserved category. The first traffic type represents excess traffic from a connection with reserved capacity or traffic that arrives before the confirmation of a reservation is received. Rather than discarding such traffic at the entrance to the network, it is often preferable to transmit it through the network with no explicit reservation. This traffic will be sent as "red," and consequently will not degrade the performance of reserved ("green") traffic, while at the same time being kept in sequence with it.

The second type of non-reserved traffic is that without stringent delay requirements and with activity periods that are short in comparison to the setup period. The most common example is datagram-type traffic such as electronic mail. Skipping the reservation phase completely for connections of short durations can both increase the amount of statistical multiplexing and decrease the load on the route selection and bandwidth reservation services. This traffic can use the gaps in the bandwidth usage of reserved connections. Similar to the "red" traffic case, it must be guaranteed that the existence of excess non-reserved traffic does not degrade the quality of service given to reserved traffic. On the other hand, no sequence ordering needs to be maintained between the reserved and non-reserved packets. As described later, reserved traffic is given a non-preemptive priority over non-reserved traffic.

We have thus defined three ways of handling bandwidth reservation requirements: long-lived, on-demand and non-reserved. Many users cannot be mapped into a single reservation class. In real high-speed networks, it is likely that many users (such as multimedia applications and variable-rate services) will be better served by a combination. Such applications will require some bandwidth with a long-lived guarantee, some on-demand, and some non-reserved. This combination can be provided to such users with no violation of packet sequencing.

In addition, there is some blurring of the boundaries between the three categories. We have the ability to estimate the traffic generated by a long-lived connection and modify the initial estimate of its characteristics determined at connection setup time. This modified estimate may cause a new setup that reflects the new parameters. We expect this modification to be much slower in its variation than the on-demand reservation. Consequently, we classify this estimation/modification process in terms of service guarantee, bandwidth efficiency, and reservation parameters as a slowly varying long-lived connection.

Equivalent Bandwidth

In the case of long-lived and on-demand connections, the network must ensure that the bandwidth needed to carry the traffic generated by the connection remains available over the connection's lifetime. The key problem here is to parameterize the requirement from the network in terms of the characteristics of the source. In plaNET, we use a notion called "equivalent bandwidth" of the connection [15-19]. This bandwidth represents the equivalent amount of link capacity that is consumed by connections on a link. It is a function of both the characteristics of individual connections and their interaction within a link. The desired network Grade Of Service (GOS) is met only if, at all links, the aggregate equivalent bandwidth of connections remains below link capacity.

Verifying that the aggregate equivalent bandwidth remains below the link capacity is typically a very complex task, since it must account not only for differences in connection characteristics but also for their interactions within the network. In addition, it must be performed "on-line," i.e., track changes in link loads as connections are added and removed from the network. The approach implemented in plaNET relies on a simple yet reasonably accurate approximation, which captures the effect of connections, characteristics and interactions while allowing on-line monitoring and evaluation of link loads. The key features of the approach are now reviewed; details can be found in [18].

We focus here on equivalent bandwidth for long-lived reservations. The first step in determining the amount of bandwidth required by a connection is the specification of its characteristics. In plaNET, a long-lived connection is characterized by three parameters that capture the essential features of traffic sources. Specifically, connection *i* is represented by a call metric vector $(R_{\text{peak}}^{(i)}, \rho_i, b_i)$, where $R_{\text{peak}}^{(i)}$ is the peak rate at which the source can generate data, ρ_i is the utilization or fraction of time it is active and source transmitting at $R_{\text{peak}}^{(i)}$ (the mean bit rate is and its variance $\sigma_i^2 = m_i(R_{\text{peak}}^{(i)} - m_i)$), and b_i is the average duration of an active period.

From the above call metric vector, it remains to determine the amount of bandwidth that needs to be allocated to a connection. This is obtained from

It is crucial to protect reserved services from the impact of non-reserved ones. This protection is implemented at the level of buffer management.

the combination of two approximations. The first one considers a connection in isolation and determines its bandwidth requirements as a function of its traffic parameters. The second focuses on the interaction of connections within the network and captures the effect of statistical multiplexing on bandwidth requirements. As shown in [18], these two approximations yield reasonably accurate bandwidth estimates over complementary ranges of connection characteristics. Therefore, together they give adequate and computationally efficient estimates for the bandwidth requirements of connections and the associated link loads.

Recalling the results of [18], the bandwidth $\hat{c}$ required by an individual connection with call metric vector $(R_{\text{peak}}, \rho, b)$ can be estimated using a simple fluid-flow model, and is given by:

$$\hat{c} \quad \frac{\alpha b(1-\rho)R_{\text{peak}} - x + \sqrt{[\alpha b(1-\rho)R_{\text{peak}} - x]^2 + 4x\alpha b\rho(1-\rho)R_{\text{peak}}}}{2\alpha b(1-\rho)} \qquad (1)$$

where x represents the available buffer space, and $\alpha = \ln(1/\varepsilon)$ with ε the desired loss probability (GOS) in the network. Conversely, the amount of bandwidth $\hat{B}$ required by N connections multiplexed on the same link can be approximated by:

$$\hat{B} = m + \alpha'\sigma, \quad \text{with} \quad \alpha' = \sqrt{-2\ln(\varepsilon) - \ln(2\pi)}, \qquad (2)$$

where m is the mean aggregate bit rate $(m = \sum_{i=1}^{N} m_i)$ and σ is the standard deviation of the aggregate bit rate $(\sigma^2 = \sum_{i=1}^{N} \sigma_i^2)$. The final estimate of the amount of link capacity $\hat{C}$ required by a set of N connections is then obtained from a simple combination of the above approximations

$$\hat{C} = \min\left\{ m + \alpha'\sigma \ , \ \sum_{i=1}^{N} \hat{c}_i \right\}, \qquad (3)$$

where the quantities $\hat{c}_i$ are computed from equation (1). The expression provides a computationally simple and reasonably accurate estimate of load levels on network links [18]. This information is key to controlling the usage of network resources and ensuring performance guarantees to connections.

Path Selection and Admission

Another important function is the path selection and admission procedure. Its goal is to maximize the chance of selecting a path capable of adequately supporting a new connection while distributing the traffic to utilize network resources as efficiently as possible. For this purpose, paths are selected dynamically by the origin node of a connection, as a function of both the requirements of the new connection and the current network state. This requires that each node be aware of the current state, i.e., link loads of all other network nodes.

Storage and Distribution of Network States. In plaNET, load information is maintained for all links in the form of a triplet or link metric vector, which is stored in the previously mentioned topology database kept by each node. The structure of this vector reflects the equivalent bandwidth computation described earlier. The exact form of the link metric vector for link j is as follows:

$$\mathrm{L}_j = \left(m = \sum_{i=1}^{N} m_i, \sigma^2 = \sum_{i=1}^{N} \sigma_i^2, \hat{C}_{(N)} = \sum_{i=1}^{N} \hat{c}_i \right) \qquad (4)$$

where N is the number of connections currently multiplexed on link j, m and σ^2 are the mean and variance of the aggregate bit rate, and $\hat{C}_{(N)}$ is the sum of the N individual equivalent connection bandwidths computed from equation (1). Note that the equivalent capacity allocated on a link, as given by equation (3), is easily obtained from the vector $\mathbf{L}_j$.

An important feature of equation (4) is that it allows for incremental updates of link metric vectors as connections are added or removed. Specifically, a request for connection establishment or removal with call metric vector $(R_{\text{peak}}^{(i)}, \rho_i, b_i)$ is used to compute a connection request vector $\mathbf{r}_i$ of the form:

$$\mathrm{r}_i = (m_i, \sigma_i^2, \hat{c}_i) \qquad (5)$$

The new link metric vector $\mathbf{L}'_j$ after adding (removing) a connection with request vector $\mathbf{r}_i$ is simply given by component-wise addition (subtraction) of $\mathbf{L}_j$ and $\mathbf{r}_i$.

The link metric vectors are continuously updated as connections are added and removed. However, a network-wide broadcast is triggered only after a significant enough change in the load level of a link. There is clearly a trade-off between the accuracy of the link load information maintained in the databases and the frequency at which updates are generated. plaNET implements an update algorithm [20] that attempts to bound the frequency at which updates are generated while allowing rapid distribution of significant changes in link loads. This ensures adequate accuracy of the link load information with relatively small communication and processing overhead.

Path Computation and Admission

The primary objective of the plaNET path selection strategy is to maximize the long-term network throughput; the secondary objective is to provide the lowest possible end-to-end delay. Because of the inherent complexity of identifying an "optimal" dynamic routing policy, plaNET relies on a heuristic approach that attempts to both balance the load in the network and favor the shortest possible paths. We briefly describe and motivate the algorithm, and the reader is referred to [21] for details.

Load balancing is intuitively desirable since it avoids early saturation of links, which forces connections onto longer and more costly alternate paths. This is similar to policies of current circuit switched networks, which favor links with the largest number of idle trunks [22, 23]. Another advantage of load balancing is that, in a packet-switched network, it also minimizes the end-to-end delay seen by connections.

The intuition behind favoring short paths is to minimize the amount of network resources required to route a given connection, which yields higher connection utilization. This is again similar to heuristics used in circuit-switched networks [24]. Note that the selection of the shortest path often contradicts load balancing (a heavily loaded two-hop path is selected over a lightly loaded three-hop one).

The algorithm used in plaNET attempts to reconcile the potentially contradictory criteria mentioned above. It is based on a modified shortest-path algorithm, where link lengths are increasing functions of the load (favoring lightly loaded links to promote load balancing) of the form:

$$d_j = \frac{C_j}{(C_j - \hat{C}_j^{(1)})\,(C_j - \hat{C}_j^{(2)})}\ , \qquad (6)$$

where C_j is the reservable[1] capacity on link j, $\hat{C}_j^{(1)}$ is the allocated equivalent capacity on link j prior to adding the new connection request, and $\hat{C}_j^{(2)}$ is the allocated equivalent capacity on link j after adding the new connection request. Both quantities can be easily derived from the stored link metric vector $\mathbf{L}_j$ and the request vector $\mathbf{r}_i$.

The main modification to a traditional shortest-path algorithm is in constraining the shortest path to also have the minimum possible hop count. This favors paths that require fewer network resources. Specifically, upon receiving a new connection request, the algorithm generates the minimum hop-count path that can accommodate[2] the new connection with the smallest total length. The algorithm also allows connections to specify a maximum length threshold P_T, beyond which a path is deemed unacceptable. Such a length threshold is useful in preventing paths from using too many slow-speed links.

In parallel to the above criteria, the path selection in plaNET defines the notion of primary and secondary links for a given connection and Origin-Destination (OD) pair. Primary links belong to the most efficient paths available to the connection, while secondary links are part of longer alternate paths used by the connection only when the more direct paths are saturated. The access of connections to such alternate paths must, however, be controlled to both protect the "direct" traffic and preserve network throughput. Therefore, a load threshold is imposed on secondary links to prevent connections from being routed on alternate paths when at least one secondary link is loaded beyond the specified threshold. This scheme is similar in spirit to the trunk reservation approach used in circuit-switched networks [25].

In order to provide additional flexibility in managing and allocating bandwidth, path selection in plaNET also allows for the specification of connection priorities. These priorities should not be confused with delay or loss priorities, which define the quality of service received by a connection. Rather, these priorities reflect the "importance" of a particular connection. In particular, plaNET allows connections to preempt lower-priority connections when no paths are available. New connections specify both a setup priority and a holding priority, with the holding priority always higher than the setup priority. The holding priority is kept by connections once established, while the setup priority is used to preempt other connections if no available path is found. The use of two such priorities allows the network to tailor the service provided to connections, i.e., some connections may tolerate being blocked when the network is congested but do not want to be preempted once established.

Connection priorities are reflected in the topology database through distinct link metric vectors, and path computation always initially uses the link metric vector that reflects the load induced by all connection priorities. If no available path is found, a new path computation is initiated, using a link metric vector that only accounts for connections with higher priority than the incoming one. If a path is generated, the required connection preemptions take place as the new connection is established. The description of this setup phase is the topic of the next section.

Call Setup

The bandwidth setup and release mechanisms used in plaNET make use of the underlying routing mechanisms and built-in hardware components. This is done in order to minimize the latency in their operations and permit a high rate of bandwidth setup and release operations. While plaNET supports a variety of routing modes, including ATM-style label-swapping, the routing mechanisms used by the setup/release procedure are the following [5, 6, 13].

The bandwidth setup and release mechanisms used in plaNET make use of the underlying routing mechanisms and built-in hardware components.

Setup and release messages are sent using the ANR routing mode, which is basically a hardware built-in source- or self-routing technique. The routing field carries a sequence of link identifiers (which must be unique at a given node), where the identifier of the next link in the path is at the beginning of the list. This first link identifier is used at a node to route the packet to the corresponding outgoing link and is then stripped off. The next identifier is used at the next node. Once a path is calculated at the source, no additional path setup is required at the intermediate nodes. This makes the ANR routing mode very attractive for setup functions, since it self-routes packets through the precalculated path.

In the selective copy routing mechanism, each ANR identifier also carries a bit that indicates if the packet should be copied in this node, in addition to being forwarded over the corresponding link. An additional field in the packet header specifies the type of function, e.g., bandwidth management, to which the packet should be copied. The function itself may be placed in various locations in the plaNET node. In the current plaNET implementation, the bandwidth management function is handled by a link processor. Therefore, the existence of the bandwidth management copy indication will cause the packet to be copied to the processor of the link through which the packet is transferred.

Reverse path accumulation takes into account that in order for the bandwidth management processors to respond to a bandwidth reservation, they must know the path back to the source. A special function in plaNET, which can be selectively enabled for a particular packet, allows the list of links that were traversed along the way to accumulate after the routing field. This gives the recipient of the packet the exact return path to the sender.

Each plaNET link adapter has its own processor, which tracks the amount of bandwidth in use over that particular link. The distribution of bandwidth management to multiple link processors allows the efficiency of parallel

The plaNET rate-control mechanism operates at the source and regulates the flow of packets into the network.

processing. In the plaNET network, we use the setup/release procedures as the mechanism to inform the intermediate nodes about the amount of bandwidth that is allocated to the calls over their local links.

Another task of the setup procedure is to reconfirm the availability of the reserved bandwidth for the new call. This task is necessary because of the potential latency in the operation of the bandwidth reservation. Calls that are simultaneously initiated from different sources may allocate capacity from the same link without being aware of each other. Typically, this will cause no harm if the call bandwidths are small compared to the residual available capacity. However, for congested links or high-bandwidth calls, such as high-quality video calls, this might cause overutilization and, hence, excessive packet loss. Other tasks of the setup/release procedure and detailed description of its operation can be found in [13, 26].

The bandwidth setup procedure is composed of two complementary phases. In the first phase, the source of the call notifies the destination and the intermediate nodes along the path of the new call and its characteristics. This phase is accomplished by the source sending a direct message to the destination, which is copied by the intermediate nodes (using the selective copy mechanism) while accumulating the reverse path. The setup message carries the maximum and minimum bandwidth required and a unique call ID.

The second phase includes a call confirmation process in which a confirmation message is transferred from the intermediate nodes back to the source. Each node checks and returns the amount of reserved capacity available. If none can be reserved, the confirmation message is converted into an abort message. The confirmation phase is optional, in the sense that in most cases the source does not wait for the confirmation message before end-to-end communication is enabled. However, the reception of an abort message will cause the session to be immediately terminated. Upon reception of a confirmation message, the source node may then resend a setup message to correct the actual amount of bandwidth reserved according to the minimal amount it got from all intermediate nodes.

Call termination is very similar to call setup, without the confirmation phase. However, since the call might be terminated by external events such as failures along the path, we must have other ways to terminate the call and release the reserved capacity in such events. These mechanisms are discussed in [13, 26].

While bandwidth reservation is currently handled by the link processors, future implementations of plaNET will have a full hardware implementation of the call/bandwidth setup/takedown procedure. Such an implementation will further simplify the task of fast reservation and release of bandwidth. The idea is that the setup message will pass through special pipelined hardware on each link adapter, which will, on-the-fly, check the available capacity against the maximum (preferred) and minimum requested values. (The packet will also carry the amount already reserved for the connection, so no fast table look-up is required.) The hardware will then record the amount of capacity reserved and insert this value in the maximum requested bandwidth field in the setup packet to reflect the actual value reserved.

This creates a sequence of increasing reserved capacities in the message. The lowest capacity reserved will always be at the front. If the amount of capacity available is less than the minimum, the inserted maximum field will be set to zero and the message will not be forwarded any further. Once the setup message reaches its final node in the path (the destination or a blocked link), it will be sent back to the source using the accumulated reverse path, and will travel through the same hardware system. The hardware at each node will then adjust its bandwidth reservation for the connection, as a function of the final value of the reserved bandwidth.

Since the setup message will travel at a high priority level, it is possible to operate such fast reservation at a latency very close to the round-trip propagation delay. The amount of time wasted is therefore minimized. Such hardware implementation allows for fast setup/release and tracking of bandwidth usage. There is no need for any table look-up, and thus, the design matches the basic philosophy of simplicity that governs the pla NET hardware design.

Another hardware implementation of bandwidth reservations is reported in [27-29] for an ATM-based network. The main difference between the two techniques is that in ATM, a virtual path must exist prior to the reservation/release process, since the messages must use an existing VPI/VCI connection. This is not the case in plaNET since the ANR routing mode is used. In addition, our scheme provides a complete negotiation with the network in order to reserve the actual bandwidth available. In addition, no table look-up is needed for a typical operation. (For exceptions for large-bandwidth calls, see [13].)

Steady State Controls

In this section, we review the control mechanisms that are applied after connections have been accepted into the plaNET network. These mechanisms monitor, control, and react to the steady-state flow of traffic. We focus on three key aspects: rate control, traffic estimation, and intermediate buffer management

Rate Control

The plaNET rate-control mechanism operates at the source and regulates the flow of packets into the network. Typically, there is one instance of rate control per connection. The mechanism consists of a buffered leaky bucket plus spacer [30-34]. The buffered leaky bucket is a well known scheme to control the average rate of a connection, while allowing for some amount of burstiness. Its two main components are a token pool and a data buffer. Tokens, termed "green" tokens in the rest of this section, are generated at a constant rate γ into the token pool, which can store a maximum of M tokens. Each token represents a transmission credit of q bits for the source, i.e., the transmission of an n-bit packet[3] into the network requires $\overline{n/q}$ tokens, where $\overline{x}$ is the largest integer greater than or equal to x. The rate γ at which tokens are generated determines the long-term average bandwidth available to the connection, while the token pool size M limits the size of bursts that the connection can generate into the network. The data buffer is used to queue packets, which cannot immediately enter the network upon their arrival, i.e., there are not enough tokens available in the token pool.

The spacer provides a "gap insertion" function that limits the peak rate at which a connection is allowed to generate traffic into the network. This is needed to both protect the network from misbehaving users, i.e., transmission of a full token pool at the network rate, and avoid reliance on the knowledge of the attaching physical interfaces, i.e., a low-speed user may be connected through a high-speed channel. The peak rate limitation is achieved by requiring a minimum spacing between consecutive packets. The value of this interpacket gap is a function of both the size of the previous packet transmitted and the spacing rate β of the connection. Specifically, the spacer enforces a gap of n/β after transmission of a packet of size n. This is achieved by entering the tokens used by a packet into a spacer pool, which is emptied at a constant rate β. Transmission of the next packet can only take place when the spacer pool is empty.

The spacing rate typically verifies $\beta = R_{\text{peak}}$, so that packets are only delayed at the access point because of the unavailability of enough tokens. However, because of either the presence of a slow-speed link $(C_j < R_{\text{peak}})$ on a connection's path or the need to smooth the incoming traffic, the spacing rate may be chosen to verify $\beta < R_{\text{peak}}$. In particular, as indicated by equation (1), reducing the peak rate of a connection may significantly reduce the equivalent bandwidth it requires. (For additional details and discussions, see [20].)

The smoothing of incoming traffic can also be achieved by appropriately adjusting the leaky bucket parameters so that more packets are delayed. This is typically achieved by reducing the size M of the token pool available to a connection. For a given token generation rate γ, a lower M results in more delayed packets, which then enter the network at the token rate γ instead of the higher source peak rate R_{peak}. Quantitative examples of the trade-off between increased access delay and smoother traffic can be found in [30, 31], while [20] describes a simple approach to size the token pool needed to ensure a given delay probability. However, any decrease in the token pool size must be compensated for by a corresponding increase in the buffer size to keep the access loss probability constant. In fact, it is well known that the probability of losing packets at the access point is only a function of the combined token pool plus buffer size [20, 30, 35]. Decreasing the token pool size while increasing the buffer size in proportion will, therefore, increase the probability of the packets' delay without increasing their access loss probability.

The problem of setting the leaky bucket parameters is, however, typically more involved than simply selecting the value M yielding the desired delay probability. The goal of the "ideal" access control mechanism is to remain transparent to well-behaved users, while protecting the network from the excesses of misbehaving ones. These two objectives are obviously difficult to conciliate, and the approach taken in plaNET is to provide as transparent an access as possible while ensuring network protection. It is now briefly reviewed, and the reader is referred to [20] for details.

The first step is to relate the leaky bucket parameters to the bandwidth allocation procedure described earlier. In particular, the token generation rate γ should never exceed the equivalent bandwidth allocated to a connection on any of its links. Violating this condition could result in network congestion in cases of users saturating their access queues. Such misbehaving users generate a constant bit stream into the network at rate γ, and it is therefore necessary that $\sum \gamma^{(i)} < C_j$ for any set of connections multiplexed on link j. Once γ has been set so that this condition is satisfied, the token pool size M can be computed to ensure a given desired access delay probability (i.e., transparency) [20]. As mentioned above, the desired M can be obtained from a simple model that relies on fluid-flow representation of the leaky bucket system [20, 36]. The token pool size needed to ensure an access delay probability below ξ is then given by:

$$M = \frac{b(1-\rho)\gamma(R_{\text{peak}} - \gamma)}{\gamma - \rho R_{\text{peak}}} \times \ln\left[\frac{(\gamma - \rho R_{\text{peak}}) + \rho\xi(R_{\text{peak}} - \gamma)}{\xi\gamma(1-\rho)}\right], \quad (7)$$

where $(R_{\text{peak}}, \rho, b)$ are the previously defined source characteristics, and γ is the token generation rate allocated to the connection.

The selected access delay probability ξ is typically of the order of a few percent or lower, depending on the resulting M value. In particular, the size of the token pool should not exceed some fraction, e.g., one-fourth, of the buffer size available at intermediate network nodes. This is necessary to ensure that the worst-case bursts generated by misbehaving users do not impact network performance.

While the value of ξ determines the probability that an incoming user packet will find an insufficient number of tokens in the pool upon its arrival, plaNET offers users an alternative to delaying access to the network.

In the absence of enough tokens, a connection can select to have its packets sent into the network marked as "red" [33]. In case of congestion at intermediate nodes, "red" packets are discarded first by the network. This allows packets to immediately enter the network without any further delay[4], at the cost of a potentially higher loss probability in the network. Alternatively, the user can select to send "red" packets only when the access queue exceeds a given threshold. This can be used to prevent excessive access delay or to allow packets that would otherwise have been discarded because of a full access queue to take their chance in the network. The actual discard policy and intermediate-node buffer structure are described later. In addition, the access control limits the volume of "red" traffic a connection is allowed to generate into the network. This is achieved by means of a "red" traffic limiter, which prevents connections from sending "red" traffic in excess of a certain fraction, e.g., one-tenth, of their negotiated maximum average rate γ.

The smoothing of incoming traffic can be achieved by appropriately adjusting the leaky bucket parameters so that more packets are delayed.

Detection of significant changes in the characteristics of a connection are key in providing users with the type of performance they expect.

The use of "red" traffic can be further tailored, for example, to ensure that all segments of the same data unit are sent with the same "color." These and other options are made available in plaNET through the flexibility of an efficient software implementation of the rate-control mechanism.

This implementation uses a high-speed processor together with some simple hardware assistance, and is capable of supporting a large number of simultaneous connections. Its structure, rather than emulating the leaky bucket operation, computes for each packet reaching the head of its queue the appropriate transmission time. Therefore, instead of continuously updating the leaky bucket state (token pool count) and deciding *if* a new packet can be sent, the software actually computes ahead of time *when* the packet should be sent and updates the leaky bucket state accordingly. In addition to its flexibility and the fact that it can support many connections, the advantage of this approach is that it also avoids the problems of rate- and clock-matching, which are being faced by hardware-based implementations.

Traffic Estimation

The detection of significant and long-term changes in the characteristics of a connection are key in providing users with the type of performance they expect while allowing for efficient usage of network resources. In particular, many users initially may not be able to exactly specify their traffic characteristics according to the call metric described earlier. The network will typically assign such connections to a "default" connection type. This mapping, however, will often be inaccurate and will have to be tuned to avoid either insufficient or excessive bandwidth allocation. This requires that the network monitor the traffic generated by such a source in order to appropriately adjust the bandwidth reservation and leaky bucket parameters. The monitoring of a source is also often useful in detecting misbehaving users and taking additional corrective actions. In this section, we discuss primarily long-lived connections.

In plaNET, this monitoring is done by an estimation module collocated with the rate-control mechanism. The quantities that need to be estimated are mainly the source utilization ρ and the mean burst period b. The peak rate R_{peak} can often be assumed to remain constant as it is typically imposed by external constraints, which are unlikely to change. The source utilization is easily estimated from the total received byte count over the estimation period T. The average burst size over the period T can be estimated either directly, by counting the number of source activity periods, or indirectly, from the access delay probability $\hat{\xi}$. As shown in [20], the estimate of the average burst duration $\hat{b}$ is related to the estimated access delay probability $\hat{\xi}$ and source utilization $\hat{\rho}$ through the following expression:

$$\hat{b}=\left[\frac{(1-\hat{\rho})\gamma(R_{peak}-\gamma)}{M(\gamma-\hat{\rho}R_{peak})}\times\ln\left(\frac{(\gamma-\hat{\rho}R_{peak})+\hat{\rho}\hat{\xi}(R_{peak}-\gamma)}{\hat{\xi}\gamma(1-\hat{\rho})}\right)\right]^{-1}, \quad (8)$$

where γ and M are the current leaky bucket parameters. The advantage of equation (8) over a direct measurement of $\hat{b}$ is that it also captures the impact of burst period distribution and even correlation. The above estimate $\hat{b}$ corresponds to the average burst period of an "equivalent" On-Off source with the same bandwidth requirement as the actual connection itself [20].

From the estimated values $\hat{\rho}$ and $\hat{b}$ for the source utilization and mean burst period, an estimate for the connection metric estimate $(R_{peak},\hat{\rho},\hat{b})$ can then be obtained. This estimated connection metric provides the necessary information to determine, using the procedures outlined in the previous sections, the appropriate bandwidth reservation and leaky bucket parameters for the source. These quantities can now be adjusted to respond to changes in connection characteristics. It is desirable that these adjustments be both robust to short-term changes and responsive to large sudden jumps. In particular, the system should not trigger a change in reserved bandwidth because of a short-term overload but should quickly respond to a sudden increase (or decrease) in source activity. These two objectives are obviously contradictory, and some trade-offs are unavoidable.

The first component in providing a robust yet responsive adaptation mechanism is the estimation procedure itself. plaNET will investigate two estimation algorithms and their performance for various types of traffic. The first algorithm is a simple exponential smoothing, which has the advantage of tremendous simplicity while allowing some flexibility in trading responsiveness for robustness, i.e., by adjusting the coefficient of the exponential filter [37]. The second algorithm is based on a bimodal adaptive filter, which has been shown to perform well for a wide variety of traffic types while being only slightly more complex than an exponential filter [38]. In both cases, the estimation module generates a connection metric estimate $(R_{peak},\hat{\rho},\hat{b})$ every T time units. It then remains to decide when to actually modify the bandwidth reservation and leaky bucket parameters of a connection.

At the end of each monitoring interval T, the estimated connection metric is used to compute a new estimate for the equivalent bandwidth required by the connection—see equation (1). This new value is then compared to the current one, and an adjustment is triggered[5] if a significant enough difference exists. Such a change in bandwidth reservation is then followed by the modification of the leaky bucket parameters. Note that for bandwidth increases, the change only takes effect after successful completion of the setup process required to secure the additional bandwidth on the connection's path. If the additional bandwidth is not available, a new path may optionally be computed. This adaptive allocation of bandwidth allows the network to both use its resources efficiently, i.e., idle connections release any unused bandwidth, and provide an adequate GOS to connections with either unknown or varying traffic characteristics.

Intermediate-Node Buffer Management

The conceptual model of the nodal buffering in plaNET is based on output queuing, which reflects the fact that the current switching mechanism is a high-speed shared media [10]. Buffers exist at both the incoming and outgoing link sides. However, it can easily be proven that the input side of the switch has no loss with a small number of maximal-size packet buffers. Therefore, the main queuing point is the outgoing link section, after packets have traversed the switching system.

In plaNET, in order to allow for simple and efficient implementation it was decided not to dedicate buffers to specific connections. In contrast to such proposals [39, 40], the complete buffer pool is shared among all connections. Distinctions in service policy are accomplished by stamping individual packets (at their source) with different GOS indicators. Therefore, the same connection may send packets at different service classes.

Packets in plaNET can belong to different delay priority classes. The packet buffer at the outgoing link is partitioned into several buffer pools, which are served in a fixed non-preemptive priority order. The number of buffer pools at certain links can be smaller than the number of delay priority classes defined. This is due to the fact that, in some high-speed links, the impact of queuing delay on the total packet delay is negligible. Therefore, in some links several delay priority classes can be grouped together in the same buffer pool. Note that there is no guarantee of FIFO ordering among the different delay priorities.

In addition to delay priority, plaNET has a notion of loss priority. Loss priority (we use the term "color" to denote this) can be thought of as a subdivision of the delay priority classes [33]. Essentially, loss priority or color reflects the "importance" of a packet. Packets of lower loss priority are discarded first whenever there is congestion within a particular delay priority class. Note, however, that FIFO ordering within a delay priority class is guaranteed. Therefore, loss priority is only used to determine which packets to discard and never interferes with packet service order.

We use a simple threshold policy within each priority class to accommodate different colors. Packets of a certain color are associated with a threshold value. When such a packet arrives to the outgoing link section, whether the number of bytes occupying the buffer pool of its priority class is below this threshold is checked. If affirmative, the packet is admitted; otherwise, it is dropped. As mentioned above, this threshold policy is simple to implement. It is not an exact implementation of loss priority, since it operates only at the input to a queue and does not remove packets of low loss priority already within the queue.

The different buffer policies are mapped into the GOS used by the various connections. As previously described, we have two types of reserved connections as well as non-reserved connections. These three connection types are mapped into the three delay priorities defined in plaNET. Non-reserved traffic (which is distinct from excess traffic of a reserved connection) is mapped into the lowest delay priority class. Reserved traffic is mapped into the two higher delay priority classes according to whether the traffic is real-time or non-real-time.

Within each delay priority class, two colors are currently supported (red and green) and are mapped to different thresholds. The green-colored traffic is associated with the normal reserved traffic and is allowed to access the full buffer pool. The red-colored traffic is considered to be traffic in excess of reservation and is given a considerably lower threshold value. (See [33] for a more detailed analysis). Under lightly loaded conditions, green and red packets will share the network resources in an identical manner. The relative ordering of green and red packets is preserved (i.e., green packets never overtake red packets generated from the same source). This fact is important in simplifying the resequencing at the end points.

In plaNET, in order to allow for simple and efficient implementation the complete buffer pool is shared among all connections.

Conclusion

This article outlines some of the key ideas in the plaNET bandwidth management and congestion control. The references have a wealth of additional detail. As noted, we plan to deploy plaNET networks in a series of field trials in the near future. One field trial (AURORA) is described in detail in [7, 8]. The plaNET switch deployed in AURORA will operate at an aggregate speed of 6 Gb/s and will interface to SONET OC-12 links operating at 622 Mb/s. We will use a 1 Gb/s Local Area Network (LAN) (ORBIT) to attach end users into the plaNET switch [41, 42]. The control and management functions described in this article will be implemented in RISC processors on the switch adaptors and on an IBM RS/6000 machine attached into the ORBIT LAN. These field trials will enable us to validate and refine the ideas presented in this article.

Acknowledgments

The authors would like to acknowledge the many people who contributed to different aspects of the work described in this article, in particular, H. Ahmadi, J. Chen, P. Chimento, J. Derby, J. Drake, P. M. Gopal, L. Gün, J. Janniello, M. Kaplan, S. Kutten, M. Naghshineh, K. Potter, K. Sohraby, and M. Sidi.

References

[l] I. Cidon and I. Gopal, "PARIS: An Approach to Integrated High-Speed Private Networks," *Int'l. J. Digital and Analog Cabled Sys.*, vol. 1, no. 2, pp. 77-86, Apr.-June 1988.

[2} Special Issue on Asynchronous Transfer Mode, *Int'l. J. Digital and Analog Cabled Sys.*, vol. 1, no. 4, Sept.-Oct. 1988.

[3] "Draft — General B-ISDN Aspects," CCITT Study Group XVIII, Report R 34, June 1990.

[4] J.-Y. Le Boudec, "The Asynchronous Transfer Mode: A Tutorial," IBM Research Report no. RZ 2133, May 1991.

[5] I. S. Gopal and R. A. Guérin, "Network Transparency: The plaNET Approach," IBM Research Report, 1991 (submitted to INFOCOM '92).

[6] "plaNET: An Architecture for Transparent Fast Packet-Switched Networks," IBM Research Division, 1991 (in preparation).

[7] Special Report, "Gigabit Network Testbeds," *IEEE Comp. Mag.*, vol. 23, no. 9, pp. 77-80, 1990.

[8] D. Clark et al.,"The AURORA Gigabit Testbed," *Comp. Networks and ISDN*, to appear, 1991.

[9] Press Release, *Wall St. J.*, Jan. 29, 1991.

[10] I. Gdon, I. Gopal, G. Grover, and M. Sidi, "Real-Time Packet Switching: A Perforrnance Analysis," *IEEE J. Sel. Areas Commun.*, vol. JSAC-6, no. 9, pp. 1,576-1,586, Dec. 1988.

[11] M. Gerla and L. Kleinrock, "Flow Control: A Comparative Survey," *IEEE Trans. Commun.*, vol. COM-28, no. 4, pp. 553-574, Apr. 1980.

[12] J. M. McQuillan, 1. Richer, and E. C. Rosen, "The New Routing Algorithrn for the ARPANET," *IEEE Trans. Commun.*, vol. COM-28, no. 5, pp. 711-719, May 1980.

[13] I. Cidon, I. Gopal, M. Kaplan, and S. Kutten, "Distributed Control for PARIS," *Proc. 9th Annual ACM Symp. on Principles of Distributed Comp.*, pp. 145-160, 1990.

[14] J. S. Turner, "New Directions in Cornrnunications (or Which Way to the Information Age)," *IEEE Commun. Mag.*, vol. 24, no. 10, pp. 8-15, Oct. 1986.

[15] G. Gallassi, G. Rigolio, and L. Fratta, "ATM: Bandwidth Assignment and Bandwidth Enforcement Policies," *Proc. GLOBECOM '89*, pp. 1,788-1,793, 1989.

[16] M. Dècina and T. Toniatti, "On Bandwidth Allocation to Bursty Virtual Connections in ATM Networks," *Proc. ICC '90*, pp. 844-851, 1990.

[17] M. Dècina, T. Toniatti, P. Vaccari, and L. Verri, "Bandwidth Assignment and Virtual Call Blocking in ATM Networks," *Proc. INFOCOM '90*, pp. 881-888, 1990.

[18] R. Guérin, H. Ahmadi, and M. Naghshineh, "Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed

Networks," *IEEE J. Sel. Areas Commun.*, no. JSAC-7, Sept. 1991.

[19] J. A. S. Monteiro, M. Gerla, and L. Fratta, "Statistical Multiplexing in ATM Networks," *Proc. 4th Int'l. Conf. on Data Commun. and Their Performance*, Barcelona, Spain, pp. 148-162, 1990.

[20] R. A. Guérin and L. Gün, "A Unified Approach to Bandwidth Allocation in Fast Packet-Switched Networks," IBM Research Report, 1991 (submitted to INFOCOM '92).

[21] H. Ahmadi, J. S.-C. Chen, and R. Guérin, "Dynamic Routing and Call Control in High-Speed Integrated Networks," *Proc. Workshop on Sys. Eng. and Traffic Eng., 13th Int'l. Teletraffic Cong.*, Copenhagen, Denmark, June 19-26, 1991.

[22] G. R. Ash, "Use of a Trunk Status Map for Real-Tlme DNHR," *Proc. of 11th Int'l. Teletraffc Cong.*, M. Akıyama, ed., pp. 4.4A-4-1-4.4A-4.7, Kyoto, Japan, Elsevier Science Publishers B. V. (North-Holland), Sept. 1985.

[23] E. W. M. Wong and T.-S. Yum, "Maximum Free Circuit Routing in Clrcuit-Switched Networks," *Proc. INFOCOM '90*, pp. 934-937, San Francisco, CA, June 1990.

[24] B. R. Hurley, C. J. R. Seidl, and W. F. Sewell, "A Survey of Dynamic Routing Methods for Circuit-Switched Networks," *IEEE Commun. Mag.*, vol. 25, no. 9, pp. 13-21, Sept. 1987.

[25] R. S. Krupp, "Stabilization of Alternate Routing Networks," *Proc. ICC' 82*, pp. 31.2.1-31.2.5, Philadelphia, PA, June 1982.

[26] I. Cidon, I. Gopal, and A. Segall, "Fast Connection Establishment in High-Speed Networks," *Proc. SIGCOMM '90*, pp. 287-296, Philadelphia, PA, Sept. 1990.

[27] P. E. Boyer, "A Congestion Control for the ATM," *Proc. 7th ITC Sem.*, Morristown, NJ, Oct. 1990.

[28] P. E. Boyer, J.-R. Louvion, and D. P. Tranchier, "Intelligent Multiplexing in ATM Based Networks," *Proc. IEEE MULTIMEDIA '90*, Bordeaux, France, Nov. 1990.

[29] P. E. Boyer and D. P. Tranchier, "A Reservation Principle with Applications to ATM Traffic Control," private communication, 1991.

[30] M. Sidi, W.-Z. Liu, I. Cidon, and I. Gopal, "Congestion Control Through Input Rate Regulation," *Proc. Globecom '89*, pp. 49.2.1-49.2.5, Dallas, TX, 1989.

[31] H. Ahmadi, R Guérin, and K. Sohraby, "Analysis of a Rate-Based Access Control Mechanism for High-Speed Networks," IBM Research Report no. RC 15831, 1990.

[32] H. Ahmadi, R. Guérin, and K. Sohraby, "Analysis of Leaky Bucket Access Control Mechanism with Batch Arrival Process," *Proc. GLOBECOM '90*, 1990.

[33] K. Bala, I. Cidon, and K. Sohraby, "Congestion Control for High Speed Packet Switched Networks," *Proc. INFOCOM '90*, pp. 520-S26, 1990.

[34] K. Sohraby and M. Sidi, "On the Perforrnance of Bursty and Correlated Sources Subject to Leaky Bucket Rate-Based Access Control Schemes," *Proc. LNFOCOM '91*, 1991.

[35] A. W. Berger, "Performance Analysis of a Rate Control Throttle Where Tokens and Jobs Queue," *IEEE J. Sel. Areas Commun.*, vol. JSAC-9, no. 2, pp. 165-170, Feb. 1991.

[36] A. I. Elwalid and D. Mitra, "Analysis and Design of Rate-Based Congestion Control of High-Speed Networks, I: Stochastic Fluid Models, Access Regulation," *QUESTA*, vol. 9, 1991.

[37] R. G. Brown, *Smoothing, Forecasting and Prediction of Discrete Time Series*, Englewood Cliffs, NJ: Prentice-Hall, 1962.

[38] H. Ahmadi and P. Kerrnani, "Real-Time Network Load Estimation in Packet-Switched Networks," G. Pujolle and R. Puigjaner, eds., *Data Commun. Sys. and Their Perf.*, pp. 325-340, North-Holland, 1991.

[39] L. Zhang, "A New Architecture for Packet Switching Network Protocols," Ph.D. thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, 1989.

[40] B. Mackruki, "A Study of Source Traffic Management and Buffer Allocation in ATM Networks," *Proc. 7th ITC Specialist Sem.*, NJ, Oct. 1990.

[41] I. Cidon and Y. Ofek, "Distributed Fairness Algorithms for Local Area Networks with Concurrent Transmission," *Proc. 3rd Int'l. Workshop on Dist. Algorithms*, 1989.

[42] I. Cidon and Y. Ofek, "Metaring — A FullDuplex Ring with Fairness and Spatial Reuse," *Proc. INFOCOM '90*, pp. 969-981, 1990.

Footnotes

1. Typically 85% of the raw link bandwidth.
2. All hnks on the path verify $\hat{c} < c$.
3. Recall that plaNET allows for the transmission of variable-size packets.
4. Assuming the spacer pool is empty.
5. This feature may not be applicable to all types of connections, i.e., certain connections may have negotiated a fixed bandwidth irrespective of their actual traffic.

Biography

ISRAEL CIDON received the B.Sc. and D.Sc. degrees in electrical engineering from the Technion - Israel Institute of Technology, Haifa, Israel, in 1980 and 1984, respectively. In 1985, he joined IBM at the Thomas J. Watson Research Center, where he was a Research Staff Member and Manager of the Network Architectures and Algorithms Group. At the beginning of the 1991-1992 academic year, he returned to the Department of Electrical Engineering at the Technion, where he was a faculty member from 1984 to 1985. His areas of interest are high speed local and wide area networks, distributed network algorithms and packet radio networks.

INDER GOPAL received the B.A. degree in engineering science from Oxford University, Oxford, England, in 1977, and the M.S. and Ph.D. degrees in electrical engineering from Columbia University, New York, in 1978 and 1982, respectively. He has been at the IBM Thomas J. Watson Research Center, serving in various technical and management positions, since 1982. Currently he is Manager of the Broadband Networking Department, involved in the NSF/DARPA-sponsored AURORA Gigabit testbed, and several other research projects in the area of high-speed networking. His other research interests are distributed algorithms, communication protocols, network security, high-speed packet switches and multimedia communications. He published extensively in these areas. He received an Outstanding Innovation Award from IBM for his work on the PARIS high-speed network.

ROCH GUERIN received the Diplome d'Ingenieur from the Ecole Nationale Superieure des Telecommunications, Paris, France, in 1983, and the M.S. and Ph.D. from the California Institute of Technology, both in electrical engineering, in 1984 and 1986, respectively. He has been at the IBM Thomas J. Watson Research Center since August 1986, where he is now a member of the Broadband Networking Department, working on the architecture, implementation and field trial testing of high-speed networks. His other research interests are in the area of performance analysis and modeling of communications systems, in particular, congestion control, bandwidth management, dynamic routing and their interactions in high-speed networks.