Designing a New Architecture for Packet Switching Communication Networks

Lixia Zhang

Motivated by the long-outstanding network performance problems, in this paper the obstacles in achieving good performance are identified; it is concluded that the existing network architectures, namely the datagram and virtual circuit, lack the components for an effective performance control. How to design a new architecture for packet switching networks to overcome this defect is discussed.

Introduction

Packet switching technology has achieved a great success in its first two decades. Many packet switching networks are now running around the world, delivering billions of data packets every day. We have learned not only how to build large-scale packet switching networks, but also how to interconnect disparate networks.

We are yet to learn the principles of performance control in packet switching; we have not understood the issues well. A noticeable example is the network traffic congestion problem, which, despite years of research effort, has never been satisfactorily solved.

Problems with Existing Network Architectures

Packet switching networks aim at offering data transmission services to multiple applications. In this paper we define the terms *service request* as the data transmission requirements from network users, and *network performance* as the quality of the service provided by the network.

In the hierarchy of network protocols (see Fig. 1.), the network layer plays an important role. Its functionality is to deliver user data bits with a satisfiable performance. Above it is the end-to-end transport layer residing in users' hosts. It is the network layer, together with the ones below, that physically carries out data transmissions. Its performance determines how good the network services will be.

Offering a good performance is a great challenge to a packet switching network, for its dynamic resource sharing feature opens a whole new field of system control and resource management. In addition, the network must meet widely different service requirements from different applications (remote login applications desire a low transmission delay, while file transfers expect a high throughput, for example), and must connect a broad range of end machines, which often differ by orders of magnitude in capacities.

Traffic congestion has been one of the major obstacles in achieving good network performance. Data from all users are statistically multiplexed on shared network resources; the moment-to-moment bandwidth requirements of individual users often vary dramatically. In addition, switch nodes in many cases adjoin communication channels with largely mismatched bandwidths. The network is designed to tolerate modest traffic fluctuations, but not without limit. Whenever a packet surge exceeds the limit, it will congest one or more network switches and bring about disastrous consequences: creating long transmission delays, causing data losses, or even blocking up the network, breaking down end-user connections.

The difficulty of congestion prevention is due to the fact that packet traffic is characterized as bursty and unpredictable[10]. In theoretical studies, it is difficult to first find a tractable model to describe the dynamics of the network load. In practice, neither of the datagram and virtual circuit (VC) networks can effectively prevent traffic congestion [15]. Datagram networks deliver individual packets as independent entities; such a traffic

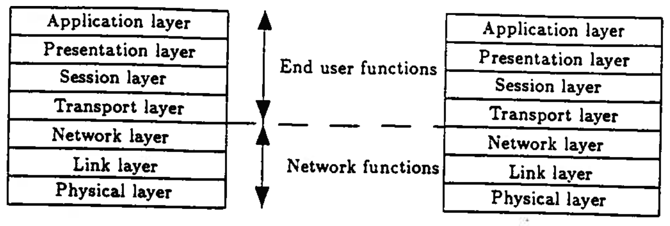

Fig. 1. Network protocol layer hierarchy

model gives a high service flexibility but makes traffic control difficult, for the network cannot effectively manage resources or control traffic on a per packet basis. Current VC networks emphasize the reliability of data delivery, and avoid packet losses due to congestion by buffer allocations along connection paths; however buffer management do not effectively prevent congestion [14]. As a result, in most of today's packet networks, the performance is largely up to the offered load. It has been a common experience that the network performance degrades rapidly whenever traffic fluctuations reach a moderate level.

Another performance problem with today's packet switching networks is lack of support for the type-of-service (TOS) transmissions. The need for the TOS was long recognized, but the network architecture design has fallen behind in providing the TOS supporting mechanisms. Some network protocols, such as IP [12] and SNA [5], defined a TOS selection, but the TOS specifications are in qualitative terms, for example IP specifies the transmission throughput or delay as high or low, rather than using quantitative values. A qualitative specification may imply different values to different network entities. For instance, a user may request a high throughput connection for a one Mbps data transmission, while the network may understand the request as being of 50 Kbps order, or vice versa. The TOS supporting mechanisms, if they exist at all, are usually routing or transmission priority selections, instead of ensuring a specific grade of service. Such TOS supports are likely to result in a best effort service, with the actual transmission performance varying as the network load fluctuates. Since today's packet networks have not been able to effectively control traffic volume, an adequate TOS support is yet to be achieved.

Adequate network resources are a necessary condition for good performance, but the resources alone do not automatically ensure a satisfactory performance. The poor performance we see today is more due to lack of traffic control than due to resource limitations. Theoretically, a packet switching network can allow any single user an unlimited share of the network resources. Without control, however, this service flexibility can easily be abused. An example of such problems is reported in [11]: the megabits per second bandwidth of Ethernets is luxury for today's distributed system applications. However, it often happens that one or more hosts send large numbers of garbage packets, and effectively disable the network usage of other users; worse yet, if these garbages are broadcast packets, not only does the net become congested, but all the hosts on the net are affected. Similar phenomena have been observed in many other places.

At the mean time, more and more new applications are being introduced into packet switching networks, such as packet voice, image data transmission, and teleconferencing, to name a few. These new applications require that the network provide truly guaranteed TOS services that meet diverse and stringent performance requirements. Remote login users have patiently tolerated frequent long network delays; packet voice, on the other hand, will not work if the specific delay limit is not ensured. New applications bring another impetus to the research for effective network performance control. Their successes, or the further success of packet switching, will depend on how well the network can meet appplications diverse service requirements.

To conclude the section, we consider that the performance problems seen in today's packet switching networks reflect an architectural defect, that is, the effective traffic control components were missing from the design. Both datagram and virtual circuit architectures emerged in the early stage of packet switching era, when the understanding of the new technology was being gained through experimentation; the performance control issues of packet switching, as in usual cases with any new technology, exposed only after the early success made pioneering packet networks grow to an enormous size.

In the rest of this paper we discuss how to design a new network architecture to achieve good network performance.

Design Principles and Issues

We set a simple goal for the new network architecture: to provide network users a data transmission service with TOS selections and an ensured performance, which implies eliminating congestion. We also assume a simple network model to start with, in order to focus on the traffic control issues. We consider a single network that consists of a number of switch nodes and point-to-point duplex links, and three layers in the OSI reference model: link, network, and transport layers.

Our focus is on the network layer protocol. We will state the basic functionalities of the link layer protocol in a later section, but will not specify design details. The transport layer is the highest layer considered in this model, which, together with the layers above, represents the *network users*. The word *user* in the rest of the paper generally means an entity, which may be a host, a process, or even a human user, that requests data transmission services.

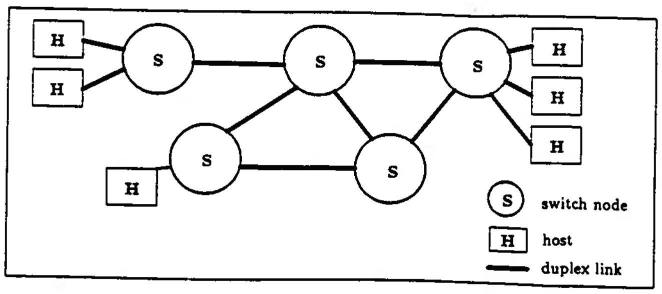

Fig. 2. A simple network model

The major theme threaded through our design effort is the traffic control and network resource management. Two decisions were made early. First, for a system with statistic load, the primary performance control is to control the system utilization, therefore the network must have a control handle on user data transmission. Second, the diversity in service requirements suggests that a quantitative service specification be submitted to the network together with data transmission tasks; with this knowledge, the network can then manage resources accordingly. This specification should also serve as an estimate of the user's transmission behavior, to facilitate the traffic control. Facing random traffic, at this moment, we see nothing better than a description from users as a proper estimate on their own transmission behavior.

The first step is to define a basic data transmission entity, that can be used by the host to express a data transmission request, and by the network to allocate resources for it, and apply control actions to it. In datagram networks, this entity is the packet; in virtual circuit networks, the virtual connection; and in the new architecture, we define a new entity, named *flow*. A number of questions must be answered during the process of designing the new architecture:

1. What characterize the flow?
2. What network resources should be managed?
3. How does the network control statistic packet flow?
4. How should applications specify their service requests?

The rest of this section discusses these questions; the next section outlines the new architecture.

Flow: the Basic Transmission Entity

A flow is defined as composing of a stream of packets that travel through the same route from the source to the destination, and that require the same grade of transmission service. A flow should meet the following requirements:

1. It is identifiable.
2. It has a specific service request.
3. It is controllable, that is, the data transmission which it represents can be adjusted or stopped.

A flow differs from packets in datagram networks in that it is treated as one sequence of packets, rather than as an aggregation of independent entities, so that the network can allocate resources to a flow and can monitor its behavior. A flow differs from a virtual connection in VC networks in that it is not concerned with the data transmission semantics, such as the connection open, close, or the state information about data; instead, a flow is characterized by its transmission requirements, and concerned with the allocation and deallocation of network data forwarding resources that are required to deliver data of that flow within the specified performance bounds.

As such, the network routing decision is required to be made on a per flow basis, and data transmissions are required to follow three *logical* steps: setting up a flow, transmitting data, and terminating the flow. Whenever having data to transmit, a user sends a flow setup request message containing the service requirements of the intended transmission, which can be a file shipment of several mega-byte data, a remote login session, or a long lived session for collecting data from remote sensors. The request message travels along a chosen route; as it passes through each switch node, network resources will be allocated according to the service requested; the resources will be deallocated at the flow termination.

There exist cases, namely transaction-oriented applications that exchange only one or a few packets each time (we call them *short transfers*), that seem infeasible to require a flow setup before data transmissions. The impact and handling of short transfers in a flow network will be discussed later.

Network Resource Management

What network resources should be controlled? Processing power, communication channels, and buffer space are the three major components in packet switching networks, with the first two as the driving forces in accomplishing data transmissions. We call the processing power and communication channels together as the network *data forwarding resources*, which exclude buffer space. The asynchronous resource-sharing feature of packet switching requires data buffering inside the network, to resolve temporary contentions on the shared forwarding resources. Buffer space itself, however, does not directly contribute to data forwarding tasks.

Along what dimension should the data forwarding resources be measured? The switch node CPU processes a certain number of packets per second, and the communication channels drain out a certain number of data bits per second. The dynamic resource sharing feature distinguishes packet switching from traditional circuit switching technology, but does not change the nature of the packet switching network as being a transportation system, where the capacity should be measured in rate, and the sharing among users should also be controlled by rate.

Furthermore, to provide a ensured performance as circuit switching networks do, the packet switching

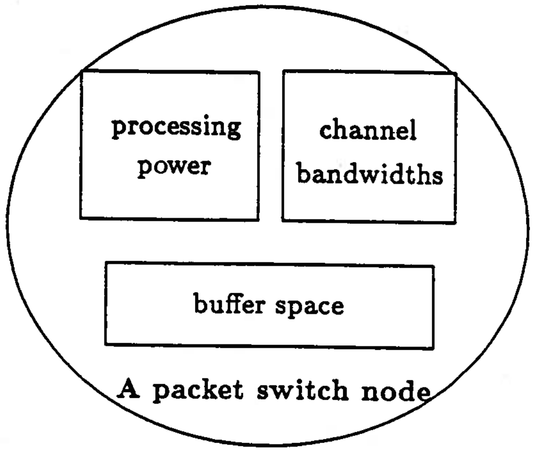

Fig. 3. Network resources are managed by the switch node: processing power, communication channels, and buffers.

network must also allocate resources to individual data transmissions. Here we make a distinction between *resource allocation* and *statistical multiplexing:* the former means reserving a certain *amount* of resources for individual data flows, while the latter refers to the way the shared resources are utilized. No pieces are dedicated to any specific user; rather, they are used upon the presence of data.

The concept of controlling packet networks by rate has been previously considered. In discussing network flow control, Cerf pointed out that, "It is generally the case that flow control is enforced through the allocation of permits to send packets and the reservation of buffers to receive them. In fact, flow control should really be dealt with by metering the rate of flow of the packets into the network bound for given destinations. But for asynchronous systems, the measurement and control of rate of flow is very difficult to implement. This is still very much a research topic, and the problem arises at the levels of protocol beyond (outside) the basic pocket-switching network." [3]

Cerf raised two problems, one is how to implement rate control in an asynchronous system; the other is the impact of rate control on network applications, for a rate-based traffic control implies rate-based data generations. We discuss these two problems next.

Control on Statistical Multiplexing

Packets from all sources are statistically multiplexed on shared network resources. The model of Poisson packet arrivals with exponentially distributed packet lengths has been widely adopted as a packet traffic approximation in the network performance analysis, such as estimating the average transmission delay [9]. Leaving alone the debate of whether this model is justifiable (there are a number of measurements that do not agree with the Poisson assumption [1, 6, 8]), more specific information about the traffic is needed to perform the traffic control. To control is to regulate; to apply control actions requires to identify individual data sources, to monitor their usage of the network resources, and to readjust their transmissions whenever necessary.

A close look at the network traffic shows that the traffic is not totally haphazard. Individual data transmissions, in most cases if not all, either are well under control, or behave with certain predictable or identifiable characteristics. An example of the former case is bulk data shipments: when a user requests a transmission, it is the transport protocol that determines how the shipment, which may consist of hundreds or thousands of packets, is carried out. An example of the latter is packet voice: a conversation generates packets randomly, due to random talk spurts, but with well known average and upper bound rates.

Furthermore, unlike the rigid requirements of human communications, where long delays or bandwidth variations are intolerable, computer generated data traffic has, at least in most cases, the flexibility of tolerating delays and channel bandwidth adjustment. Some applications can even accept postponement of transmission services for a short time period with no negative effect on performance; network mail and background file transfers are such examples.

We consider that both application designers and network builders, whenever possible, should explore such regularities of network applications; and design and implement protocols to make data sources produce predictable or controllable packet flows. By given these regularities and controllabilities, the network can then coordinate its traffic control with data sources. The better the network knows the demand, the easier and more effective the control will be, the higher the achievable performance, and the higher the achievable network resource utilization. On the other hand, the network will not be able to make any performance promise if it does not have any knowledge of the traffic.

Given that the network resources are managed on a rate basis, we further consider that the packet switching network should control traffic on an average rate basis. Given an aggregate of multiple statistical flows, the network can meet individual users throughput requirements by allocating data forwarding resources according to the required average throughput value, and can meet the delay requirements by system utilization control, and by proper packet processing ordering.

It should be pointed out that merely blocking traffic at the network entrance is not an acceptable control mechanism. Currently, the network problems, such as congestion, are mirrored back to users indirectly through long delays or high packet losses, which end users find particulary difficult to handle correctly [13]. Generally speaking, computers perform well only when being explicitly informed of what to do; they do poorly in coping with unpredicted abnormal conditions, such as long network delays caused by the interface blocking. Therefore the network should adjust traffic volume by regulating data generation rates at sources, i.e. exchanging specific control parameters with users. An explicit transmission rate control can help users perform an educated self-restraint on data generation. For instance, remote login applications may normally send one packet for each character (as is feasible in a local environment with plenty channel bandwidths), and compact several characters into one packet when receiving packet rate limitations. Advanced coding techniques are available today that can generate varying bit rates for real time voice, to adapt to network transmission limitations [2].

Service Specification

Flows need a systematic way to express their demands so that the network can allocate resources accordingly. A set of service attributes need be defined first. The attributes should be chosen independently both from applications and from the underlying network hardware technology, so that they can be used to describe requirements of both today's and future unknown applications, and be supported by both current and future advanced technologies.

Functionally, the network offers a data transportation service. A physical transportation goes in a two-dimension space: volume (throughput) and time (delay). A third dimension is needed to describe the transmission quality, namely reliability. Since our architecture design aims at network resource management and traffic control, three basic attributes, throughput, delay, and error rate, should give adequate description of a transport service in terms of resource allocations[1].

Physical systems usually are not perfect, data transmission reliability beyond what is provided by the physical system has to be achieved either by forward-error-correction coding or by automatic retransmissions or both. The former converts reduced effective throughput to higher reliability, the latter, both reduced throughput and longer transmission delays. To keep this paper in size, we temporarily ignore the transmission error issue, and focus on how to provide secured delay and throughput in data transmission services.

The attribute values should be expressed by statistical parameters. This is because, generally speaking, most data sources do not generate constant information flows, and because packet switching utilizes resources only upon actual data traversing the network, thereby packet flows mirror back the randomness in data generations. In addition, statistical resource sharing implies possible queueing delays, thus the transmission delays behave statistically as well. The attribute parameters should also express different *degrees* on service constraints, which vary among applications. For instance, although both packet voice and remote login applications expect short transmission delays, the delivery limit on voice packets are more rigid: any overdued packets will be discarded; while a short delay average is satisfactory for remote login users, since a longer delay generally does not make the packet obsolete.

What and how many statistical parameters are needed for each attribute? Among many possible choices, we should try a good balance between the expressive power of the parameter set and the complexity of the protocol. System simplicity favors using the fewest possible parameters, on the other hand, a sufficient set of parameters is desirable so that all applications can adequately express their requirements, even though not every single application would need all. Users can leave uninterested parameters unspecified, which will then take a default value assigned by the network.

We look at transmission delays first. We choose two parameters: the expected transmission delay (ED) and tolerable delay limit (DL). Being a statistically shared system, the packet switching netork in general cannot guarantee a random data source the delivery of every packet within a tight delay limit, therefore a specific delay limit value must also be associated with a tolerable loss ratio (LR).

Packet voice cares about the limit of transmission delays; all packets delivered over the limit are equivalent to losses, which are tolerable if within a few percentage. To express this requirement, the values of the DL and LR should be specified, but the ED can be left out. A bulk data transmission, when carried out in real time, will probably concern with the mean delay but can tolerate delay variations, hence the values of the DL need not be specified. Non real-time applications may leave out both delay parameters to the network default values, which may also indicate to the network that their transmission requests can be postponed in the case of severe resource shortages.

In general, reliable protocols require a loose upper-limit on the packet life inside the network. This value can be passed to the network by stamping a Time-to-Live (TTL) value on packets; the network will drop a packet when its TTL expires.

The parameters for throughput is somehow difficult to choose, for packet transmissions are considered being bursty and random, how does a user determine the throughput value of its flows? The previous discussion suggests controlling on the average transmission rate. Given that there is no generally valid model for computer data generations, we are to experiment with a new approach. We propose to use a pair of values, an average interval (AI), and an average rate (AR), to specify the average transmission rate, i.e. a flow will transmit at about a rate of AR packets(bytes), averaging over each AI time period. We also choose a third parameter, throughput tolerable variance ratio (TVR, in percentage), to indicate the throughput adjustability of a flow. A TVR with the value of 0% marks a rigid flow; with TVR = 50%, a flow that can tolerate up to one half reduction in transmission rate. Unlike the delay parameter case, users should offer a throughput estimate whenever possible, to help the network control its utilization.

Users may specify no service requirement, if it is impossible, or infeasible, to do so. The network takes a "best effort" approach to fill all unspecified service parameters. For flows with no throughput parameters, it will monitor the flow volume; for those with no delay parameters, it will assign them a low order in packet processing. Whenever the network falls short in resources, these flows will be subject to readjustment on throughput or postponement of service.

To end the discussion on transmission service specification, we point out that the role of the service specification is twofold. On the one hand, it enables the network to check whether adequate resources are available before accepting a new data transmission request. On the other hand, it also serves as a measure that the user's own transmission behavior can be compared against; slight variations from the specified values shall probably be tolerated; significant departures, however, should be notified of the network first, or as soon as possible.

Building Blocks in the New Network Architecture Design

Stated in a simple way, a system in general consists of three basic types of components: the resources, the demand, and the control mechanisms that regulate the usage of resources. Correspondingly, the network architecture under discussion consists of switch nodes that represent resources, flows as the demand, and network control mechanisms. The architecture defines the functions of each component, and the relations between them.

[1] Here we only discuss how to provide a good service to flows in the network; in fact, being able to control the traffic makes the service availability controllable, but due to space limit the availability issue is not discussed here. Also because the main concern is the network resource management, other issues, such as transmission security, are not discussed either.

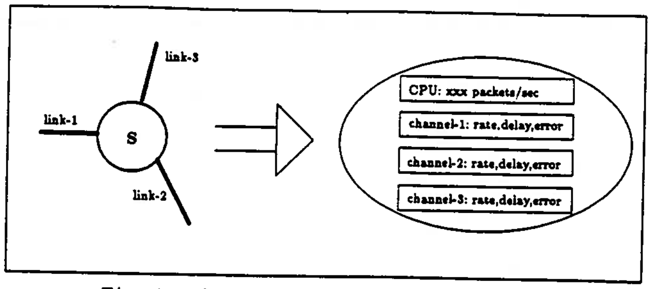
link-3
link-1
S
link-2
CPU: xxx packets/sec
channel-1: rate,delay,error
channel-2: rate,delay,error
channel-3: rate,delay,error

Fig. 4. A network switch node model

In this section, we discuss the major issues of the network components, and the design problems to solve. The final design of the new architecture is still in progress, and will be reported in future papers.

Network Switch Node

Switch nodes represent and manage the distributed data forwarding resources of the packet network. A duplex communication channel between two nodes is modeled as a pair of simplex ones, each belonging to the sending node it attaches to. The channels may be of different physical media. We ignore those details and mark the communication channels with different transmission error rates, bandwidths, and delays. All switches are equal in functionality, each can be either a net interface or a transit node, or both; but they may have different processing capacities. We also assume that each node has a sufficiently large packet buffer space.

A switch node knows it capacity of data forwarding resources, which include the CPU and communication channels. It allocates, and monitors the usage of, its resources among flows. This requires the node keep the utilization information for each piece of its resources, as well as the transmission information of each of the flows passing through it. In other words, the node keeps the *state* of the resources and traffic. There is no state synchronization between nodes, however; each node makes independent commitment to carrying flows.

A switch node may fail. When this happens, its resources and state of the resources are lost together. Nodes fail independently; one's failure can be detected by neighbors. Because nodes do not share the state information, one's failure does not interrupt others data forwarding functions. This is the so called "non-fate-sharing" principle [4].

Service requests come from the application layer at the top. The supporting mechanisms, on the other hand, must be built bottom-up. To fulfill the performance commitment to users, the network layer in turn has two requirements to the link layer. First, the link layer must pass packets between nodes with a known delay, for the network delay is an additive function of all link delays along the route, the net must be able to control the delay at every hop. This means no automatic packet retransmission hidden from the network layer, as it is often the case in today's link layer protocols (HDLC, for example). Second, the link layer must pass all received packets to the network layer, including those with detected bit errors, for there exist applications that prefer packets with bit errors to losses or retransmissions [7], while for others, such as flow setup requests, reliability is required. In short, the basic function of the link layer is to pass packets between adjacent switches, without imposing any constraints other than physical channel limitations, upon the network performance. Decisions on error packets should be left to the network layer, where more information about individuals requirements is available.

Flow Protocol

We define a network data delivery protocol, named Flow Protocol (FP), to carry user data from sources to destinations. A flow sets up a duplex channel between the two ends, which is identified by two things: a flow ID, and a service request. The flow ID should be independent from network switches, and be unique within the network. We also define an auxiliary protocol, named Flow Control Protocol (FCP), to carry out communications between the network and users, such as the flow setup and termination messages, or other network control messages to users and user inquiry messages to the network.

To perform the resource allocation, the user specifies its required service in a flow setup message; the services required in both directions can be estimated from application-specific knowledge. For example, if a flow is a one-way data transmission, acknowledgments will compose of the reverse traffic with a predictable throughput; if a flow carries a voice conversation, it will require symmetric service in both directions.

The flow setup request, R, travels through a chosen route. As it passes each switch node, the node makes a decision on whether to accept the request. If accepted, it forwards R to the next hop; if it can accommodate the new flow only with a smaller throughput, it changes the throughput value (within the TVR region) in R before forwarding it; otherwise it puts a reject mark on R and returns it to the sender. When R successfully returns, the flow is set ready. Flow termination is of the "abortive" type only: either end of a flow may initiate a flow termination message, which then travels through the path. Upon receiving a termination message, a node erases the corresponding flow state immediately and releases the resources.

To optimize network performance, a flow may start data transmission immediately following the flow setup request message. This fast-setup can have one of the three possible outcomes:

1. Most of the time the request will be granted with no adjustment, and the flow setup delay is avoided with no extra cost. When this is not the case,
2. the flow may be asked to adjust the throughput. The setup message will get returned to the user, marked with the service grade that can be supported. The user should then compute the excessive amount of data that have been sent and hold on the transmission for a time period equivalent to that needed to drain out the excessive amount of data;
3. if the flow setup is rejected, all switches preceding the rejection point, when seeing the rejected request message, may simply drop off the data, if any has already entered the network.

As mentioned earlier, there are short transfers that do not match the flow block favorably. Their short durations make it infeasible to do individual flow setups. Transmissions without a setup phase implies an uncontrolled part of the network input, i.e. they will not be rejected even when there is a resource shortage.

Individual short transfers are not much concerned with throughput, but do expect a low transit delay in general. The low delay can be met by controlling the network utilization, which in turn requires controlling the network input. One way to handle short transfers is to treat them together as a special flow, monitor their total volume, and preserve the resources accordingly. This strategy will work well when short transfers compose of a small portion of the total traffic, and the aggregation remains stable. If they become a significant portion of the traffic, the network may lose control. The network must first ensure the committed performance for other established flows, and may have to delay, or even discard, packets of short transfers at high tides of traffic fluctuation, similar to the situation in today's datagram networks.

Traffic Control

The network applies traffic control to each individual flow. There are four major control functions:

1. making decisions on accepting new flow requests;
2. monitoring established flows, i.e. to check how well each flow follows its specified manner;
3. ensuring the performance of individual flows; and
4. dynamically readjusting flows with the users when necessary.

All control actions are based on the state information at each switch node, and are carried out by each node independently.

There are several reasons that the service requests of established flows may need to be dynamically readjusted now and then:

1. The original parameters may not be a good estimate of the actual flow behavior.
2. A user may want to change the behavior during a flow running.
3. The network may encounter a temporary resource shortage, e.g. due to a component failure, or due to a sudden load increase.

Switch nodes or end users send FCP messages to the other party requesting for changes. Such dynamic adjustment necessarily raises an interesting question of the system stability. Both the traffic control mechanisms and the system stability issues are currently being studied.

Summary

The poor performance observed in today's packet switching networks motivated our effort to design a new network architecture, based on the following considerations:

1. A network can achieve a satisfactory performance only by building effective performance control mechanisms into the architecture.
2. A system that supports multiple diverse applications must define service specification mechanisms so that users can declare their specific service requirements.
3. The network performance control mechanism consists of two parts: the service specifications from users, and the performance supports from the network architecture.

A new network entity, flow, is defined to carry data transmissions. A flow threads together the pieces of distributed network resources to carry out its data transmission task. The service is robust because it is the flow that claims resources along the way as it passes switches; there will be no network state resynchronization needed during flow recoveries from switch failures.

This paper presents some preliminary thought on the new architecture design. The design will be experimented through simulation tests, to verify our current understanding of the performance control issues in packet switching networks.

Acknowledgment

The Distributed System Group at MIT-LCS provides a warm research environment; in particular, I would like to thank Dave Clark for many illuminating discussions.

References

[1] P. Amer, et. al., *Local Area Broadcast Network Measurement: Traffic Characterization*. Technical Report 86-12, Department of Computer and Information Science, University of Delaware, 1986.

[2] T. Bially, B. Gold, and S. Seneff. "A Technique for Adaptive Voice Flow Control in Integrated Packet Networks." *IEEE Transactions on Communications* COM-28(3):325-333, March, 1980.

[3] V. Cerf. "Packet Communication Technology." *Protocols and Techniques for Data Communication Networks*. Prentice-Hall, Inc., 1981, pages 1-34, Chapter 1.

[4] D. Clark. Some Thought on the DARPA Internet Architecture. Paper in preparation.

[5] R. J. Cypser. *Communication Architecture for Distributed Systems*. Addison-Wesley Publishing Company, 1978.

[6] D. Feldmeier. Traffic Measurements on a Token Ring Network. *In Proceedings of Computer Networking Symposium*. November, 1986.

[7] C. Weinstein and J. Forgie. "Experience with Speech Communication In Packet Networks." *IEEE Journal on Selected Areas in Communications* SAC-1(6):963-980, December, 1983.

[8] R. Jain and S. Routhier. "Packet Trains—Measurements and A New Model for Computer Network Traffic." *IEEE Journal on Selected Areas in Communications* SAC-4(6):986-995, September, 1986.

[9] L. Kleinrock. "Queueing Systems." *Volume II: Computer Applications*. John Wiley & Sons, Inc, 1976.

[10] L. Kleinrock. "A Decade of Network Development." *Journal of Telecommunication Networks*, Spring, 1982.

[11] K. Lantz et al. "An Empirical Study of Distributed Application Performance." *IEEE Transactions on Software Engineering* SE-11(10), October, 1985.

[12] J. Postel. "DoD Standard Internet Protocol." Network Information Center RFC-791, SRI International. September, 1981

[13] Lixia Zhang. "Why TCP Timers Don't Work Well." *In*

Proceedings of Symposium on Communication Architectures and Protocols. ACM SIGCOMM, 1986.

[14] Lixia Zhang. "Congestion Control in Packet-Switched Computer Networks." *In The Proceedings of the Second International Conference on Computers and Applications (Beijing, China).* June, 1987.

[15] Lixia Zhang. "Some Thoughts on the Packet Network Architecture." To appear in Computer Communication Review 17(1 & 2):3-17, January/April, 1987.

Lixia Zhang received B.S. degree in Physics from Heilongjiang University, China, in 1976, and M.S. degree in Electrical Engineering from California State University, Los Angeles in 1981. She is currently working for her Ph.D. degree in computer science at Massachusetts Institute of Technology. Her research interests include computer network architectures and protocols, distributed applications, and operating systems. ■