PARIS: AN APPROACH TO INTEGRATED HIGH-SPEED PRIVATE NETWORKS

ISRAEL CIDON AND INDER S. GOPAL
IBM T. J. Watson Research Center, Yorktown Heights, N.Y. 10598, U.S.A.

SUMMARY

This paper describes a design of a high-speed packet switching system for integrated voice, video and data communications. The system makes use of a simplified network architecture in order to achieve the low packet delay and high nodal throughput necessary for the transport of voice and video. A prototype of this system has been implemented and is now being tested under a variety of packet traffic loads. We have demonstrated that this system provides a cost-effective solution for private integrated networks.

KEY WORDS Integrated networks Packet switching Broadband ISDN Private networks

INTRODUCTION

Today's communication systems carry data traffic through packet switching techniques but use circuit switching techniques for voice. Conventional wisdom has argued that the statistical multiplexing offered by packet switching makes it ideally suited to traffic of a bursty nature such as interactive data. For steady streams of traffic, such as voice, the nodal processing overheads necessary for each packet overcome any bandwidth savings that statistical multiplexing may achieve, and circuit switching techniques are more appropriate. The conventional wisdom was extensively studied and tested by several studies in the late seventies.[1-3] Essentially, all of these works were unable to show conclusively that packet switching is suitable for voice because they were attempting to use general packet switching techniques (SNA, ARPA) that were developed for the transport of data.[4, 5] These techniques use a general purpose processor to do the packet switching in software, and consequently are not able to provide the substantial throughputs necessary for voice.

The first work to realize the true potential of packet switched voice was Reference 6, wherein the basic idea of off-loading the packet switching function onto specialized hardware is proposed. The target there is a replacement of the telephone toll switch with a packet switch. Thus, high throughput is the major consideration. The design is fairly complex and requires extensive use of custom VLSI chips. The design requirement calls for a node that supports around a thousand 1·5 Mb/s links with an aggregate throughput of around 6 Gb/s.

Our design requirements are somewhat different. We are interested in designing a *private integrated voice/video/data network*. We envisage that such a network will consist of a high-speed *backbone* network comprised of several switching nodes interconnected by links of speeds of the order of 100 Mb/s. Although lower speed links may exist, we believe that the availability of leased T3 and fibre optic links will make it cost effective to have a few links of high speed rather than many links of lower speed. In addition to the backbone there will exist a *peripheral* network which will essentially provide access into the switching nodes. The peripheral network will be comprised of relatively low-speed links and may not use the same protocols or switching techniques used in the backbone. In addition, the peripheral network will perform the task of multiplexing the relatively slow end users to the high-speed backbone. Thus, the backbone switching nodes will primarily handle high-speed lines. The number of high-speed links entering each switching node will be relatively small (probably less than 20) and the aggregate throughput will be in the 1–4 Gb/s range. In addition to the requirement of low delay (less than a millisecond per node) and high throughput, cost and design complexity are key factors in our environment. The main thrust of this paper is to show that for such an environment, packet switching is a cost effective and viable technology.

Another thrust of the paper is to demonstrate the value of designing the network as a single complete system. In particular, rather than designing networking architecture and nodal hardware structure separately, we tailor the architecture and hardware to work efficiently with each other. There are several instances where the design of the networking architecture with the hardware in mind considerably simplifies the hardware. The consequence of this system design approach is a high-speed packet switching node that can satisfy the throughput and delay requirements and is simple enough to be

0894–3222/88/020077–09$05.00

Received 4 May 1988

readily implemented with off-the-shelf components. We refer to our system as PARIS, which is an acronym for *Packetized Automatic Routing Integrated System*.

An initial prototype of PARIS has been implemented and is currently working in our lab. Each switching node can support up to 10 fibre optic lines each operating at 100 Mb/s. The design can be easily enhanced to support aggregate throughputs of up to 4 Gb/s per node.

The PARIS networking architecture can be decomposed into two major parts:

1. Network transport. These are the functions that are executed in the actual transmission of packets. These functions include flow control, error recovery, intermediate node routing, etc.
2. Network control. This essentially comprises the functions that are related to the set-up of a call. The functions include the call acceptance function that makes decisions on whether or not to permit a new call access to the network, the route computation function that determines the path that a call is to be routed over, as well as directory and other application level functions which perform important services such as the location of remote network resources.

The PARIS approach to network transport is quite different from conventional packet switched networks. The key driving requirements are to reduce the end-to-end delay in order to satisfy real time delivery requirements and to achieve high nodal throughput. In order to accomplish this, the processing in the intermediate nodes is reduced to a minimum. The intermediate node routing function is performed through a technique called automatic network routing (ANR) which requires no table look-up or storage. Congestion control and priorities are implemented through very simple procedures. Most of the transport level functions (such as flow control and error recovery for data, and packetization and reassembly for voice and video) are performed on an end-to-end basis.

The PARIS approach to network control is a decentralized one. For fault tolerance and performance reasons, it is well accepted that for moderate-sized networks decentralized control is preferable to reliance upon one or more central controller(s). Thus, in the PARIS system, every backbone node participates in a set of distributed algorithms which collectively comprise the network control of the system.

Figure 1 is a network picture showing the various components in a PARIS node. A PARIS node consists of three basic components, the network control unit (NCU), the switching subsystem (SS) and the end point processor (EPP). The functions of these components are briefly described below.

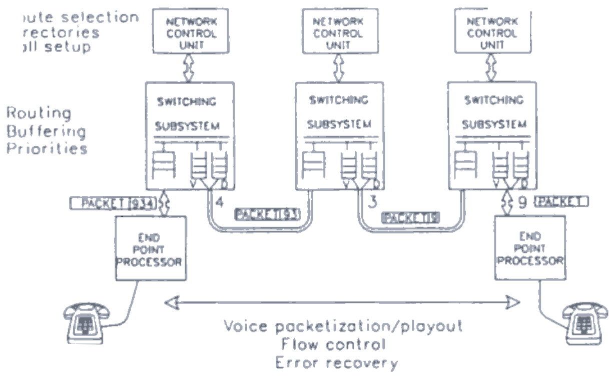

Figure 1. PARIS network

1. The NCU implements all the network control functions. In addition it interacts with the other components in gathering information (e.g. traffic statistics) that is needed in performing these functions. It also interacts with the NCUs in other nodes both in gathering information and in performing the functions themselves.
The SS performs the intermediate node functions that are involved in the transport of packets. These functions include packet framing, packet buffering, determination of the outgoing link, the actual switching function of transferring the packet from the incoming to the outgoing link, congestion control to prevent excessive packet queueing, priority functions. In addition to intermediate node functions, the SS performs statistics gathering to report to the NCU.
3. The EPP is responsible for constructing the packet in the appropriate format and delivering it to the network. This involves inserting the appropriate headers and delimiting flags and performing bit-stuffing and destuffing operations (if needed). In addition, the EPP performs the end-to-end functions of flow control, error recovery and reassmbly/playout etc. These end-to-end EPP functions are sensitive to the nature of the traffic that is being transmitted.

NETWORK TRANSPOR'

The packet structure

Figure 2 describes the structure of a packet. The important points are as follows:

1. Leading and trailing delimiters (flags) for defining packet boundaries.
2. A two-byte control which is largely unused at present except for packet priority level (two bits), a copy bit which indicates whether the packet should be copied to the NCU in addition to being transferring the outgoing link and a broadcast bit which causes the packet to be sent on all outgoing links.
3. The automatic network routing (ANR) field, which is composed of h words, each word being two or more bits in length (h is the

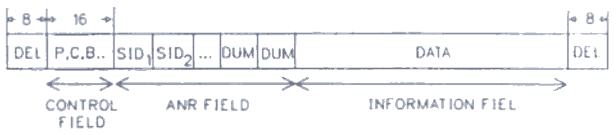

Figure 2. Packet structure

number of hops that the packet has to travel). Each word represents an outgoing link on the packet's path. We shall give more details on this field later on in the section.

4. The information field, which is of variable length. There is a maximum and a minimum size for the information field which depends on the actual implementation. Typically, the maximum is about 4K bytes and the minimum about 8 bytes. The information field also contains headers and trailers that relate to the end-to-end protocols.

Automatic network routing

As the ANR is a key point of the design, we shall elaborate on the ANR field a little further. As mentioned above, the ANR field is composed of h words. The ith word in the ANR field defines the outgoing link label of the ith hop along the packet path. The outgoing link label is essentially the internal switch ID or address (SID) of the outgoing link adaptor. Thus, the packet header contains all the routing information necessary for the routing of the packet within each intermediate node along the path. As the packet progresses through the network the 'used' SIDs are stripped off, so that the first bits in the ANR field always contain the routing information for the current node. This process is depicted in Figure 1. Thus, every node will examine a fixed location in the header without having to know of its position in the path. No external table look-ups or processing are necessary, thereby ensuring minimal nodal delay. The ANR technique is analogous to the self-routing header technique used to route information through a multi-stage switch.[7] The difference is that we are using the header across an entire general topology network rather than within a single node.

The ANR technique has several advantages in a high-speed packet switching system and some disadvantages. The major advantage over conventional packet switching techniques such as the LPID swapping procedure used by TYMNET[8] is the fact that no table look-up or processing is necessary at an intermediate node. This eliminates any potential processing bottle-necks and minimizes intermediate node delays. Most importantly, it removes the need for the intermediate node routing tables to be updated at the time of call set-up while preserving the ability for dynamic route selection. This reduces considerably the load on the NCU of the intermediate nodes during call set-up, thereby eliminating potential throughput bottle-necks and permitting calls to be set up and taken down much faster than in conventional circuit switched systems. Thus, route switching (i.e. changing the path of an ongoing call) without disruption to the end user becomes feasible. In addition, the hardware structure of a switching node can be streamlined as no routing table updates have to be sent from the NCU to the switching subsystem. This permits the adaptor design to be much simpler, potentially permitting a much more compact implementation of the switching node. In addition to reducing the delays in the call set-up phase, the ANR technique also permits packets to be sent from the source to destination *without* a set-up phase. Thus, data applications which benefit from a 'datagram' service, typically those that generate occasional packets but require rapid response times, can be effectively supported by this technique.

However, there are some disadvantages with the ANR technique. First, the inclusion of the entire path in the header wastes communication capacity. Thus, in communication capacity limited systems, such as conventional data packet switched systems, the ANR technique is not suitable. In networks with high capacity links, communication capacity is no longer a bottle-neck and some wastage is acceptable. Moreover, studies on typical networks and on randomly generated large networks[9] have indicated that typical communication paths are almost always less than 7 hops long. Since used SIDs are being stripped off, the average overhead is only half of the path length resulting in relatively short ANR fields. Another disadvantage of ANR is the added burden placed upon the originating node, as it has to know the entire path. Again, this disadvantage is not as significant as it may seem as many conventional packet switched systems[5, 10] already maintain full network topology information at every node even though they perform hop-by-hop routing.

There are two SIDs that are reserved for special use. The reserved SIDs are the all-zeros SID, which is always the SID of the NCU adaptor, and the all-ones SID, which is termed the dummy SID and is always unused. Observe that since the SIDs might be of different length, it is important to enforce a prefix condition within a single node. In other words, it is important to ensure that no SID is a prefix of another SID within the same node. This prefix condition must hold for the NCU SID and the dummy SID as well.

The ANR field is terminated by two successive dummy SIDs (which are stripped off at the end point.) The reason for this is to ensure that bit errors in the ANR field which may cause the packet to be misdirected do not cause packets to travel around the network for a very long time. After a relatively small number of hops, the dummy SIDs will be the used to route the packet to a non-existent link adaptor, causing the packet to be discarded.

For simplicity reasons in our prototype we are using a fixed-length SID of a single byte. Four of these bits are used for addressing the actual outgoing link. The other four bits are used for selective copy and broadcast mechanisms.

The intermediate node functions

We describe here the network transport functions that are implemented in the switching subsystem of each PARIS node.

Figure 3 describes the components of the SS. The key components of the SS are:

1. The switching kernel (SK), which performs the basic switching function of transferring packets from source to destination.
2. The link adaptors, that are each comprised of a receive part and a transmit part. The receive link adaptors receive incoming packets from the link and send them through the switching kernel. The transmit link adaptors receive packets from the SK, buffer them as necessary and transmit them over the outgoing link.
3. The monitor that collects traffic statistics and reports them to the NCU.

There are two basic motivations behind the design:

1. To minimize the transit time for a packet through an intermediate node.
2. To achieve the above goal with minimal hardware complexity.

Optimizing channel bandwidth use, a primary objective of many current packet switching systems, is viewed here as a secondary objective as link capacities are assumed to be very large. In other words, in order to minimize nodal transit time and to reduce nodal complexity, we are prepared to use more channel bandwidth than necessary. There are

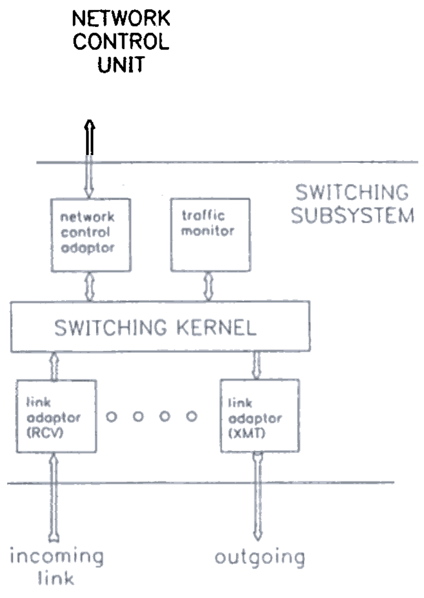

Figure 3. The switching subsystem

some key aspects, unique to our design, that enable us to achieve our goals. We shall elaborate on these aspects further as we follow through a packet transfer from incoming to outgoing link.

A packet arrives at the receive adaptor as a high-speed serial bit stream. The receive adaptor first has the job of recognizing the packet, performing a serial-to-parallel transfer and storing it into its incoming buffers. In addition the SID of this particular adaptor is stamped to the beginning of the packet. This procedure allows the NCU and the monitor (if needed) to identify the source link of this packet. If the link protocol requires a bit-stuffing operation we have adopted an end-to-end bit-stuffing protocol. Thus, rather than performing the bit stuffing and destuffing on every hop, as in conventional link protocols such as HDLC, we perform it once on an end-to-end basis. The stuffing is done only in the information field. The header fields are to be used in the intermediate nodes and therefore cannot be bit-stuffed end-to-end. Thus we enforce a structure on the header, which ensures that this portion of the packet does not contain any flags, and therefore need not be bit-stuffed and can be used by the intermediate nodes directly. The logic at the receiver adaptor is therefore relatively simple. Once the receiver has stored a complete packet, it signals the SK that it wishes to transmit a packet.

The method by which the SK transmits the packet is, of course, dependent on its internal structure. If the SK has a single-path[7] architecture such as a bus, it is important that the contention for the SK be resolved quickly. This is to ensure that queueing delays at the receiver are strictly limited and the system has effectively a single queueing point at the transmit adaptor. This can be achieved if the SK adopts a round-robin approach to serving the adaptors and if the speed of the SK is at least as large as the sum of the speeds of all incoming links. Round-robin operation of the SK is also inherently fair as it guarantees every source equal access. For the single-path SK, contention delay can be reduced still further or eliminated altogether by pipelining the contention resolution with the transmission of the previous packet.

Another important aspect of an implementation on a single-path SK is that once the receiver adaptor (source) has obtained control of the SK, it should retain control for the duration that it takes to send one or more complete packets. This transmission of a packet as a complete entity has the basic advantage of eliminating contention at the transmit adaptor (destination). In other words, a destination will receive only one packet at a time, and does not have to consider the possibility of simultaneously receiving packets from two separate sources. This simplifies considerably the design of the transmit adaptor.

The use of a single-path SK along with a round-robin arbitration policy ensures that the number of buffers at the receive adaptor is bounded as long as the SK throughput is at least large as the sum of all input rates. It has been proved[11] that no more than 3·35 buffers (each of a maximal length packet) are needed at each input of the SK in the worst case.

The destination identified by the first SID in the ANR field copies the packet into its buffers. In order to guarantee low delay for certain processes we provide various priority levels. This is accomplished through a dedicated buffer at each destination, for each priority level. Depending on the priority bits in the header of the packet, the packet is placed into the corresponding buffer. Once a complete packet has been collected, the destination begins reading out the contents of the buffer and transmitting it over the outgoing link. A non-preemptive priority is accomplished, whereby low priority packets are read only if the higher priority buffers do not contain a full packet.

A detailed performance analysis[11] shows that in a typical network at 85 per cent utilization the nodal delay is bounded by 1ms with packet loss better than 10^{-6}. These results also suggest that the usual assumption made that voice must have priority over data is not valid.

Before actually transmitting the packet over the link, the transmission adaptor has to perform a couple of extra tasks. First, in order to ensure that the SID for the next hop is at the right bit position in the packet, the current SID along with the source link SID stamp have to be removed. Secondly, packet start and termination delimiters have to be reconstructed. Finally, a parallel-to-serial conversion has to be performed.

We have completed tracing the packet path through the switching kernel. Control packets that are destined to the NCU travel in a similar fashion, except that the destination address is the network control adaptor address, i.e. the all-zeros SID.

The monitor is the last, but by no means the least important, part of our SS. The dynamic route computation algorithm uses, as a basic input, measurement of link loading. The monitor provides a simple, efficient way to gather such measurements. In each switching subsystem, there exists a single monitor that 'observes' the traffic through the SK.

For each packet, the monitor collects the priority level, the SID bit positions to indicate the outgoing link, and the packet length. In case the monitor is not able to keep up with the speed of the links, it will randomly sample packets, and collect the information from that random sample. A processor then reads the raw information, and periodically performs some computations in order to derive information about the status of all its outbound links, including average link loading, packet queue lengths etc. The monitor informs the NCU of the current status.

The end-point functions

In this selection we discuss the protocols that are implemented in the EPP. As mentioned before, they include flow control and error recovery for data and the packetization/playout protocol for voice.

Data protocols. The data protocols in PARIS are optimized to work with real time data. We adopt a layered structure, as depicted in Figure 4. The lowest level is the input 'throttle', which is a flow control mechanism designed for real time data. It has the property that is guarantees a certain minimum average rate for every session with the ability to send bursts of information at much higher than the minimum average rate. The next level is that of error recovery. This layer retransmits packets that are lost from buffer overflows or from random bit errors. The highest layer is that of end-to-end pacing which ensures speed matching between the receiver and transmitter. This is to avoid the possibility of receiver buffer overflow.

The input 'throttle'

The flow control protocol is based on the simple technique of regulating the input. We first describe the basic scheme, which guarantees a fair use of bandwidth by all users but is relatively conservative in nature. In the set-up of a data call there is, associated with that call, an average bit rate (which we denote by A packets/s) and a burst size parameter (which we denote by N).

The scheme, depicted in Figure 5, operates as follows. A packet leaving the throttle mechanism requires a token. Tokens are produced periodically at a rate of one token every $1/A$ seconds unless the pool has reached a maximum value of N tokens. Thus, if a session has several tokens in the pool it can transmit a burst of packets at the rate at which the link can permit, probably much faster than its average rate. Once it goes beyond this initial burst,

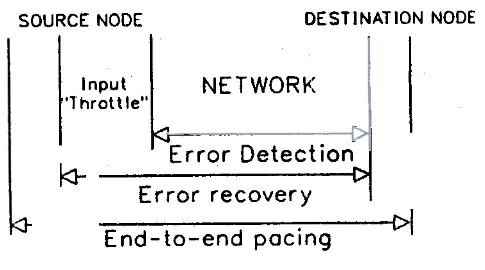

Figure 4. The data protocol

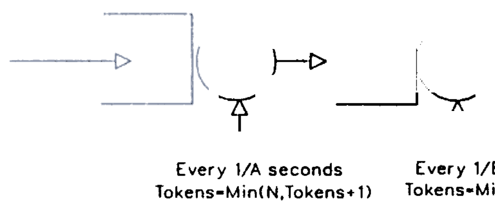

Figure The basic flow control scheme

it is constrained to transmitting one packet every 1/A seconds.

This scheme, while maintaining the average input rate, allows for a temporary increase of the peak transmission rate, often needed in such environments. The justification for allowing for temporarily increments in the traffic rate is the law of large numbers. As many calls share a high capacity link and the different calls are uncorrelated, the probability that many of them are simultaneously in their peak rate is very low. However, in order to place an additional protection upon the network, we place a second throttle in series with the first. This ensures that the speed of packet transmission *never* exceeds the 'bottle-neck' rate of B packets/s. It is particularly necessary if the input into the network is comparable or higher in bandwidth than that of the backbone links.

Essentially the choice of N determines the burstiness of the transmission. A value of 1 ensures a smooth flow of traffic. Typically, the choice of N should match the application being supported.

The basic scheme is fair but relatively conservative. It does not permit data users to temporarily take advantage of capacity that is unused and is being wasted. Thus, we have built into the mechanism some adaptive capability whereby it can dynamically raise the average rate, A, if the network is lightly loaded. In the adaptive scheme the parameter A can vary dynamically based on network conditions. The first decision that needs to be made is the choice of congestion measure. As our scheme is an end-to-end scheme the most desirable measures are those that can be estimated directly by the end node without intermediate node involvement. The measures that fall into this category are round trip packet delay and loss probability. Unfortunately (from that perspective), loss probability is very small (of the order of 10^{-6}) and is therefore very difficult to estimate precisely. Similarly, the variability of the delay which is the indicator of congestion is only a few milliseconds. Thus, we are forced to abandon these measures and instead adopt a simple measure that requires intermediate node involvement.

The measure is intermediate node congestion and we estimate it as follows. Whenever a transmit link adaptor experiences buffer overflow, it stamps the congestion bit of all acknowledgement packets that use that link in the reverse direction (receive link adaptor). As all the routes in PARIS are bidirectional, the destinations of packets in the reverse direction are sources for packets in the forward direction. This ensures that the source of the traffic realizes that there is congestion and takes steps to alleviate the congestion in the forward direction. We use acknowledgements because the number of acknowledgements in the receive direction is proportional to the traffic in the forward direction, therefore eliminating potential problems caused by asymmetric traffic loads.

From the end node perspective the scheme operates as follows. A session always uses its basic assigned rate and will not attempt to change it unless packets begin to build up. If this occurs, and if no congestion has occurred, the node will increase its average rate by steps until packet queues at the input are small. If congestion occurs, the rate is reduced until it reaches the basic rate below which it never goes.

Occasionally, during a data session, a need for a higher long-term average speed arises. This may happen, for example, when the mode of work is changed from an interactive one to a large file transfer. Reliance upon the adaptive mechanism is undesirable as it does not provide any guarantee of higher bandwidth. In such cases, using the fast call set up service of the network, a new call set up procedure is initiated and more capacity is reserved by the networks nodes (re-routing of the call may be needed). When the end user does not need this extra capacity any longer, it initiates a call take-down procedure for releasing that capacity.

Error recovery

Error recovery is also moved to the end nodes and no link-by-link error recovery is used. The motivation behind this is again to reduce intermediate node loading, and the relatively low error rates expected of high-speed links make this approach feasible from a performance viewpoint. The end-to-end recovery protocol is a standard ARQ scheme based on CRC checking at the receiver, an acknowledgement mechanism, together with time-outs and packet retransmission at the transmitter. In order to compute the correct time-outs, the path length and links capacities along the path of the specific call are used. The end-to-end data protocol also prevents buffer overflow at the end nodes due to a rate mismatch between the end users.

Voice protocols. The basic purpose of the voice protocol is to convert a continuous voice bit stream into packets at the transmitter and to reconstruct the continuous bit stream at the receiver from the arriving stream of packets. The difficulty arises because the packets encounter different delays due to the random queuing delays at the intermediate nodes. In addition, packets may be lost in the intermediate nodes requiring interpolation on the part of the receiver in order to preserve the appearance of a continuous voice bit stream. Sometimes, however, missing packets may indicate silence periods in the speech and the receiver is required to recognize and reproduce these silence periods accurately. All this is further complicated by the fact that the transmitter and receiver clocks may not be completely synchronized, resulting in packets being generated and played out at slightly different rates.

The PARIS voice protocol described in Reference

12 performs these functions without requiring the use of sequence numbers or time stamps in the header of each packet.

NETWORK CONTROL

As mentioned before the design philosophy of the PARIS network control is based on decentralization. Thus, each node participates in a collection of distributed algorithms that, together, comprise the network control. The NCU in each node implements the distributed algorithm.

Route selection

The fundamental difference between route selection for packetized data networks and route selection for packetized voice/data integrated networks is that in the latter kind of network it is not possible to slow down the real-time traffic (voice) through flow control mechanisms. Consequently, before admitting a call into the network, some guarantee must be provided that the communication capacity required by the call can be satisfied. If not, the call must be denied access into the network or 'blocked'.

The basic design choice for the PARIS system is a distributed route selection mechanism using a routing topology database similar to the one in Reference 10. Basically, each NCU maintains a routing topology database with link weights reflecting the traffic over each link. When link weights change it substantially updates flow to every node using a broadcast algorithm described in Reference 13. At the call set-up time, the source NCU computes a path from its local routing topology database and computes the ANR field from source to destination and back. This information is sent to the source user. The end-to-end call set-up packet flows over this path. Special care is given to the case where the required call bandwidth is large compared to the capacity of one or more links along the path. In such a case an extra reservation action might be needed during the end-to-end call set-up process. This will be described in the next section.

If no suitable path can be found between source and destination, the call will be blocked. The scheme provides control of the path at the source and obtains relatively efficient paths. However, as the update of link weight takes finite time, there is the possibility of some unnecessary blocking because of temporarily inaccurate information in the routing database. An efficient way of performing the link weight update that uses the fast hardware copy function of PARIS is described in Reference 13. This new algorithm has lower time complexity than conventional algorithms.[5,15] Together with the speed of the network this algorithm reduces the problem of transient inconsistencies of topology databases.

As the full topology is known, the actual process of route computation for a given call is basically centralized. An extensive study of different shortest-path schemes with different cost functions was done.[16] The study results show that schemes that use accurate link utilization give very small improvement in blocking probability compared to schemes that uses only few threshold parameters. Consequently, the load update algorithm should not be triggered very frequently.

In addition to the primary route there is a possibility for certain important sessions that a secondary route is calculated in advance. Once a failure is suspected over the primary route the call is switched to the secondary route with minimal interruption. In order to avoid common failures, the primary and the secondary routes should share a minimum number of nodes.

In the following we describe a way for calculating a secondary route which shares the minimum number of nodes with the primary route. The secondary route is basically a route that is 'most likely' to be active when the primary route fails. It is defined to be the route that has the fewest nodes in common with the primary route. Let the set of nodes that the route r_i traverses be labelled $N(r_i)$. Let us also indicate the primary route by r_p and the secondary route by r_i. The secondary route has the property that $|N(r_s) \cap N(r_p)|$ is as small as possible.

Algorithm for computing secondary routes

1. Select a large number, M, which is larger than the number of links in the network.
2. Assign weights to the links in the network according to the following rules.
   (a) For links which are on the primary route but not adjacent to the end nodes (source or destination) a weight of $2M + 1$ is assigned.
   (b) For links adjacent to a node in the primary route (except the end nodes) a weight of $M + 1$ is assigned.
   (c) For all other links a weight of 1 is assigned.
3. In the above weighted graph calculate the minimal weighted path between the end nodes. This is done using a standard shortest-path algorithm. The computed path is the secondary route.

In the following we prove that this secondary path shares the fewest number of nodes with the primary route. In fact, of all paths that share the fewest number of nodes with the primary route, the path that we compute will have the fewest hops. Each shared node in the second path adds $2M + 1$ to the total path weight. This may easily be explained by adding $M + 1$ to the weight when entering a node of the first path and adding $M + 1$ when leaving such a node. This proves that the weight of the computed path is exactly $2M \times$(the number of shared nodes with the primary path) + the number of links in the secondary path. As M is chosen to be very large and as we calculated a minimum weighted path it follows that the computed path has the above-claimed properties.

The call set-up protocol

The call set-up protocol in PARIS differs substantially from the previous packet switched set-up protocols such as the one for LPID that is used in Reference 10. Unlike LPID routing, there is no need in ANR for the intermediate node to maintain a logical connection number or the mapping from this number to the outgoing link. Consequently, the call set-up protocols need to involve the intermediate node in only a peripheral manner. This involvement is only to update the count of the number of calls in progress over each link. This number will be used to determine whether or not another call will be accepted.

Before the call set-up can be performed the EPP requests its NCU for a complete route to (and from) the destination end user, specifying the average capacity of the connection and a class of service parameter that specify the level of burstiness. Upon the receipt of this request the NCU first locates the physical destination through an optional directory search. Secondly, it computes the best route(s) to that destination (or blocks the call if such a route does not exist). Next the process of the call set-up is initiated from the source via a call set-up message. The basic scheme includes a single end-to-end message (that carries the call ID, parameters and the reverse ANR) that is directly sent to the destination. This message is acknowledged via a similar end-to-end message that is copied by all NCUs along the path. (Here we are using the hardware copy feature implemented by the PARIS switching subsystem). If the bandwidth requested by the call is small compared to the residual bandwidth along the path, the probability of bandwidth overflow due to simultaneous calls that attempt to capture the same residual bandwidth is small and can be ignored. This probability is further reduced by the fast update process used in PARIS.[13] If the call bandwidth requirement is high (for example for a high-quality video call), we provide enhancements to the basic call set-up protocol. In such a case the call set-up message is forced to traverse through some selected NCUs in order to guarantee that no bandwidth overflow occurs. The set of these NCUs that are neighbours of the bottleneck links is provided by the local NCU and is embedded in the call set-up ANR.

The call set-up process is being refreshed periodically in order to hold the guaranteed bandwidth. By default, if no set-up message is received by the NCU after some time-out, the capacity of the call is released just as in the case of a call take-down. The call take-down process is similar to the call set-up.

Link status monitoring

One of the aspects of the PARIS system is that it does not require a general purpose processor to be part of the link adaptor. Consequently, the network controllers themselves are responsible for detecting the link status. In the following we describe a protocol performed by the network controller to detect failures and reconnection of its adjacent links.

The network controller tests each of its adjacent links, whether it is operational or not, by periodically sending a special message called a *BOOMERANG* message. The BOOMERANG message has a special ANR field causing it to be sent over that particular link, to be sent back over the same link and then to be switched back to the network controller. In addition, the BOOMERANG message is copied at the adjacent node by the local network controller. As long as the BOOMERANG messages return the link is considered to be operational. If the controller detects that some (the exact number will be optimized to the network structure) sequential BOOMERANG messages did not make it back, it declares the link as not operational.

CONCLUSION

We have presented an overview of a design for a high-speed packet system for the transport of integrated traffic in a private environment. We have demonstrated that by off-loading most of the functions to the end points the task of building a high-throughput switching node is considerably simplified. The PARIS architecture decouples the control process from the switching process by limiting the involvement of the control software to set-up and take-down of streams of packets rather than handling individual packets.

A prototype PARIS network has been implemented at the IBM T. J. Watson Research Center. The switching sub-system can support a throughput of 1·2Gb/s and uses 100Mb/s private fibre-optic links. The network control unit is implemented using a PC. The hardware design is based exclusively on off-the-shelf components and we believe that it can be easily enhanced to support total throughput around 5Gb/s.

ACKNOWLEDGEMENT

The authors thank their colleagues Alan Baratz, Jeffrey Jaffe, Joong Ma, Yoram Ofek, David Seidman and James Janniello for their numerous comments and suggestions regarding this work.

REFERENCES

1. D. Cohen, 'A protocol for packet switching voice communication', *Computer Networks*, **2**, 320–331 (1978).
2. C. J. Weinstein and J. W. Forgie, 'Experience with speech communication in packet networks', *IEEE J. Selected Areas in Comm.*, SAC-1, 963–980 (1983).

3. D. Conrads and P. Kermani, 'A feasibility study of using store-and-forward data communication networks to transmit digitized speech', *IBM Research Report, RC 9157,* October 1981.
4. J. D. Atkins, 'Path control—the network layer of system network architecture', in Paul E. Green (ed.), *Computer Network Architecture and Protocols,* Plenum, 1982, chap. 11.
5. J. M. McQuilliam and D. C. Walden, 'The ARPA network design decisions', *Computer Networks,* **1**, 243–289 (1977).
6. J. S. Turner and L. F. Wyatt, 'A packet network architecture for integrated services', *GLOBECOM,* 1983, 2.1.1–2.1.6.
7. J. J. Kultzer and W. A. Montgomery, 'Statistical switching architectures for future services', *ISS'84,* Florence, Italy, 1984.
8. M. Schwartz and T. E. Stern, 'Routing protocols', in Paul E. Green (ed.), *Computer Network Architecture and Protocols,* Plenum, 1982, chap. 11.
9. M. S. Chen, 'Shortest paths in random graphs', *IBM Research Report, RC 11835,* April 1986.
10. A. E. Baratz, J. P. Gray, P. E. Green Jr., J. M. Jaffe and D. P. Pozefsky, 'SNA networks of small systems', *IEEE J. Selected Areas in Commun.*, SAC-3, (3) 416–426, (1985).
11. I. Cidon, I. Gopal, G. Grover and M. Sidi, 'Real time packet switching: a performance analysis', to appear in *IEEE J. Selected Areas in Commun.*
12. Joong S. Ma, 'A blind voice packet synchronization strategy', in preparation.
13. I. Cidon, I. S. Gopal and S. Kutten, 'New models and algorithms for future networks', to appear in *PODC'88.*
14. A. E. Baratz and A. Segall. 'Reliable link initialization procedures', *IBM Research Report, RC 10032,* 1984.
15. A. Segall 'Distributed network protocols', *IEEE Trans. Information Theory,* IT-29, (1), 23–35 (1983).
16. C. T. Chou and S. Zaks, 'Issues in route computation', *IBM Research Report*, to appear.

Authors' biographie.

Israel Cidon received the B.Sc (summa cum laude) and the D.Sc. degrees from the Technion-Israel Institute of Technology, Haifa, Israel, in 1980 and 1984, respectively, both in electrical engineering.

From 1980 to 1984 he was a Teaching Assistant and a Teaching Instructor at the Technion. From 1984 to 1985 he was on the faculty with the Department of Electrical Engineering at the Technion. From 1985 he is with IBM T.J. Watson Research Center.

His current research interests are in communication networks and distributed algorithms.

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.

Since 1982, he has been a Research Staff Member in the Computer Sciences Department at the IBM T.J. Watson Research Center, Yorktown Heights. Currently he is manager of the Integrated Networks group. His research interests are in communication networks, high-speed packet switching, and in distributed algorithms.

Dr. Gopal is currently an editor for the *IEEE Transactions on Communications*, guest editor for *Algorithmica*, and guest editor for *IEEE Journal on Selected Areas in Communications*. He was formerly a technical editor for *IEEE Communications Magazine.*