
Burst Switching—An Update

Stanford R. Amstutz

BURST SWITCHING RESEARCH IN DISPERSED control and integrated switching at GTE Laboratories is described here more fully than in previously available material [1-3].

Burst dispersed control has a non-centralized architecture employing many autonomous cooperating microprocessors connected at the edge of the transport network. It is possible to add control capacity without limit, or to compensate for control processor failure, by redistributing the processing workload among the control processors, using the transport network itself as the redistribution switch. The control reconfiguration process can be carried on even while normal call processing continues. This process is described and its timing given. The dispersed control architecture is not specific to burst switching and can work with other self-routing transports such as fast packet.

Burst transport is integrated in that voice and data are switched through the same switching fabric and transmission media. Burst switching is compared to and distinguished from fast packet, fast circuit, and ATM switching. Misunderstandings about burst transport that have appeared in the literature are corrected, to wit: burst does not immediately clip in case of channel contention; burst switches voice and data in the same way; a burst switch interfaces naturally to other types of switches. Round-trip delay performance is calculated to be less than 5 ms. The article concludes with the current status of the burst project.

Burst transport is integrated in that voice and data are switched through the same switching fabric and transmission media.

Burst Control

Dispersed Control

The original objectives for the burst control were three:

- *Unlimited capacity expandability*—We wanted to overcome the capacity limitation of a conventional centralized control, where even a tightly coupled multiprocessor has limits—its number of processors is limited by electrical constraints regarding the number of ports into shared memory and the number of ports onto an interprocessor bus.
- *Non-centralized*—We wanted a control that had no single point of failure.
- *Dispersed*—We wanted a control that could be distributed so that portions of a burst switch network could operate independently of the rest.

These objectives were met by a call processor per link switch, as was suggested in [1]. But this solution was not fully satisfactory. There was no doubt that a microprocessor had the capacity to provide call set-up and feature services for all of the ports of a link switch. The concern was that it would have more capacity (thus cost) than could be used. More important, the sizeable memory required for all the functions and features of a modern switch would have to be replicated at every link switch. Finally, a call processor failure at a link switch, while it would only disable call placement services for the ports of that switch, had no economic backup method.

An improved approach was conceived of in which the call processor was given only a port appearance, so that all of its messages to other control processors would be transferred via the network itself, as control bursts. The network certainly had the capability for this, as it had been designed to transfer data bursts between customer computers. Now there was no longer a linkage between a link switch and a call processor; a call processor could be equipped at any port.

With the new approach, the original objectives were still met, and the disadvantages of the first approach were avoided, as follows. The number of ports to be handled by a call processor became very flexible. As many or as few ports as desired could be assigned to a particular call processor; there was no longer the fixed number at a link switch to be served. The amount of replicated call processor memory in the system would be only as much as was needed to support the demand for call processing services. And most important, each call processor in the system could be backed up by transferring its load to the surviving call processors. One-for-one redundancy and special switchover networks were not required. The switching network itself could be used as the switchover network to replace failed processors with functioning processors.

There are three types of processor in the burst dispersed control:

- *Port Processor (PP)*. Every port circuit includes a PP. Its function is to convert between the external signal form and the internal burst form. Thus, for example, it converts off-hook detection or tones received to control bursts sent. It converts control bursts received to tones sent, or ringing. These are control functions.

While the PP is considered part of the control, it also performs some transport functions, such as creating bursts from a continuous stream of digitized analog samples in an analog port, by running a voice/silence detection algorithm.

The Z80 is used for the PP in the experimental model.

- *Call Processor (CP)*. CPs do most of the decision making in the system and they have the largest program of any of the control processors. They do call set-up and feature implementation. Each CP performs services for a subset of all of the PPs. Thus, when a PP creates an off-hook message, it sends it to its CP.

The M68000 is used for the CP in the experimental model.

- *Administrative Processor (AP).* APs are basically database machines. Translation from directory number to equipment number and accumulation of usage information for billing are done by an AP. An AP performs services for a subset of all the system CPs.

In a smaller system, the administrative processes can run on the CPs, as is the case in the experimental model. But if the system is large enough to require separate APs, at least two would be equipped for failure protection.

Figure 1 shows the three kinds of control processors, each having a port appearance only, on the outside of the transport network.

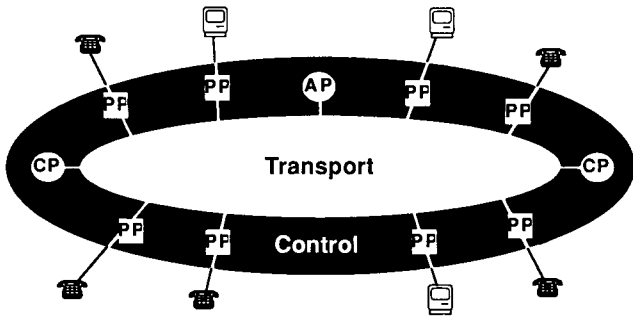


Fig. 1. Control processors outside transport.

Each PP goes to a particular CP for call processing services, and each CP goes to a particular AP for administrative processing services. Thus we have the concept of service sets. A service set includes a processor of one type as server, and processors of another type as clients.

Figure 2 represents six service sets: two AP service sets with CPs as clients, and four CP service sets with PPs as clients. Each client has the network address of its server, so that service requests can be forwarded as control bursts. We expect that a CP will be able to serve 1,000 to 2,000 PP clients, so Figure 2 signifies nothing about service set sizes.

Figure 3 shows the message exchange for an ordinary voice call set-up and take-down. This figure requires little explanation. In Phase 11, each port receives the header that will convey bursts to the other port, thus establishing the virtual connection. In Phase 12, the connection is used to transfer voice bursts. Otherwise, each PP communicates only with its CP.

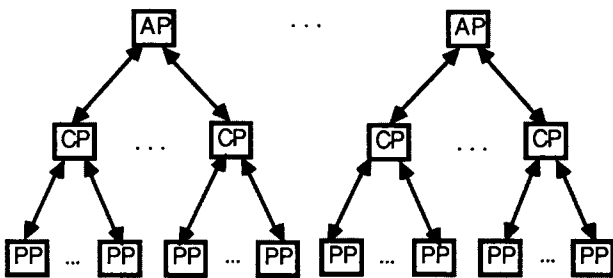


Fig. 2. Six service sets.

Control Reconfiguration

We now consider how the claimed flexibility with respect to the addition of processing capacity or the compensation for processor failure is realized. All such actions may be seen to be redefinitions of service sets. Let us first consider the addition of capacity.

A processor is overloaded if it receives more service requests than it can process with acceptable dispatch. The amount of processing requested will be related to the number

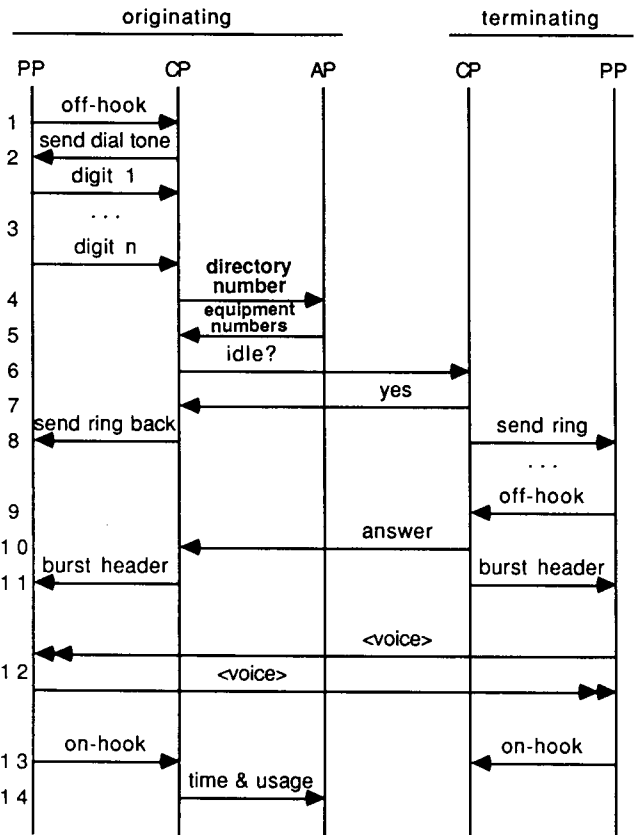


Fig. 3. Voice connection/disconnection.

of clients in its service set, and their individual activity. So the load on a processor can be reduced by transferring clients to other servers. If there are other less loaded servers in the system, clients can be transferred to them. If most or all servers need load relief, new serving processors must be added to the system to assume some of the load.

If a processor fails, its load must be assumed by other processors so that its clients continue to have a designated server.

In all cases, the desired adjustment of loading is accomplished by service set redefinition: the movement of clients from some service sets to others. The service sets of Figure 2 are shown, redefined, in Figure 4. This configuration might result if one of the CPs failed and its ports (PPs) were redistributed to other CPs. Alternatively, one can view Figure 4 as the starting point. The addition of a CP and redistribution of ports to it provides the enhanced processing configuration of Figure 2.

The transfer must be made while normal processing continues; it is not acceptable that the system shut down during the period required for control reconfiguration. Of course, the addition of reconfiguration traffic will be a significant addition to normal call handling traffic, so it is important to accomplish the reconfiguration expeditiously.

We now outline the data movements required to move a PP from one CP service set to another. (The movement of a CP from one AP service set to another is very similar, though simpler.) The movement is entirely accomplished by message exchange among control processors. However, we will not attempt to detail that message exchange; we will simply outline which processors require which kinds of information and where they get it:

- The process starts when an AP, called the initiating AP, decides to move a port, either upon the receipt of an explicit authenticated command from a craft workstation or because it has determined from background test message traffic that a failure has occurred and ports must be moved.

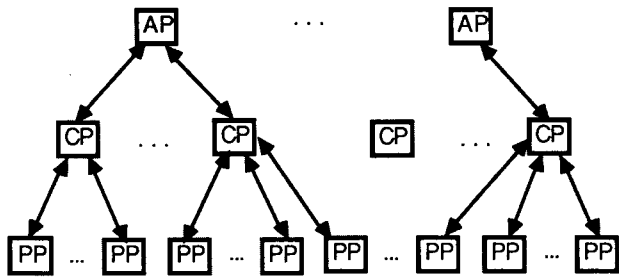


Fig. 4. Five service sets.

- All APs need to know to which service set a moved port is being added. Each receives this information from the initiating AP and updates its tables, defining all system service sets.
- The CP adding a port needs to know the port's static data—the seldom-changed class marks and features the port is entitled to use. The CP acquires this information from its serving AP, where the static data on all ports is stored.
- The CP adding a port desires to know the port's dynamic data—information describing the call status of the port. It acquires this information from the port's old CP when it can. If the old CP has failed, the information is lost and the new CP reconstructs as much of the port's call status as it can from the port's PP itself. We note that connected calls are fully reconstructed from information in the PPs themselves.
- The PP of the port being moved must know the address of its new CP. It acquires this information from the new CP itself.

All of these movements could be accomplished without difficulty were it not for the requirement that normal call handling continue while reconfiguration is under way. Because information must be moved from one processor to another, which takes time, it is possible that data may not be consistent over the whole system. Here is an example of a problem that could ensue unless care is taken.

In Phase 5 of Figure 3, the AP returns the PP address of the called port and the address of its serving CP. This CP address is used by the originating CP for the idle query of Phase 6. Suppose that a port, P, is called while it is in the process of movement from CP A to CP B. In the control reconfiguration process, the translation tables are updated with the new CP address before the new CP is informed of its new ports' identities. Thus, it would be possible for CP B to receive an idle query about port P before it knew it would be the new server for P. Knowing nothing of P, it would make no response.

This problem and others of this type are solved by locking. To continue the example, at the time P's CP entry in the translation table is changed to B, it is locked. This means that a translation request against P's directory number will be delayed (queued) until the lock is released, and the lock is released only after CP B has responded saying it now has full cognizance of and responsibility for P.

Control Reconfiguration Times

A good many messages are required to coordinate the movement of even one port. The amount of time required to move the many ports a CP might serve is an important operating parameter.

The reconfiguration software supports two kinds of reconfiguration: immediate and orderly. An immediate reconfiguration moves ports among processors as quickly as possible and is intended for compensating for processor failure, where it is important to restore service as quickly as possible. In an immediate reconfiguration, established calls are not lost but calls in the process of establishment, in digit receipt for example, will be reset.

An orderly reconfiguration puts the premium on recovery of all system data, even if it takes longer, and is intended for load balancing and the addition of processors. In an orderly reconfiguration, calls in the process of establishment are not interrupted; a port is not moved until it reaches a stable point in its process.

We have calculated how long reconfigurations take, assuming: first, that each processor can compose and send messages at 8,000 characters/s; second, that each processor can receive and process messages at 8,000 characters/s; and third, that reconfiguration traffic takes precedence over normal call handling traffic. The model calculates the total time as the sum of all the message transfer times that must be done in sequence. (The time to transfer messages sent in parallel is the time of the longest.) The times were calculated as a function of the total ports on the system and the maximum number of ports allowed per CP, R . The number of CPs on the system increases with total ports so that no CP serves more than R ports.

Figure 5 shows the times for an immediate delete of a CP: the time to move all the ports from one CP and distribute them uniformly to all the other CPs, so that at the end all the other CPs serve equal numbers of ports.

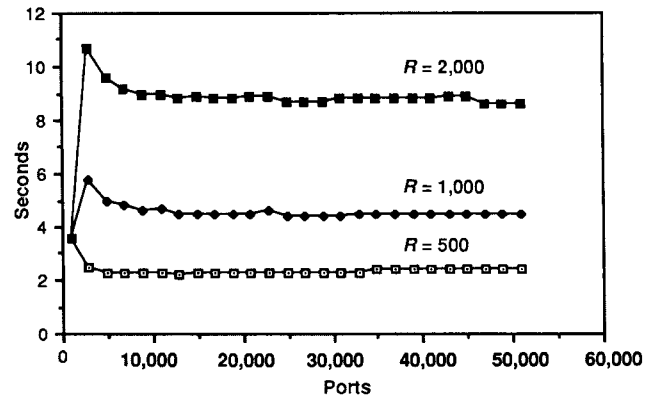


Fig. 5. Immediate delete times.

Figure 6 shows the times for an orderly add of a CP: the time to move some ports from each original CP and distribute them to the added CP, so that at the end all CPs are uniformly loaded.

We note that for both plots, the time is not primarily a function of the number of ports on the system but is proportional to the number of ports per CP. This is to be expected. The time to move ports is proportional to the number of ports moved, and that is determined by R .

The times for orderly adds are significantly longer than for immediate deletes. This is caused by two factors. First, an orderly move takes longer than an immediate move because more messages are sent. A CP acquiring a port in an orderly move will request the port's transient state information from the port's current CP. This is not done in an immediate move, since the immediate move would normally be used in case of CP failure, when a port's assigned CP would not be able to supply the information.

Second, a delete is faster than an add. In a delete, the remaining CPs work in parallel to acquire the deleted processor's ports. In an add, the added CP works alone to acquire ports from each of the other CPs.

Table I gives spot values for all four combinations of immediate/orderly and delete/add for a 50,000-port system. Here it can be seen that immediate is faster than orderly (for either delete or add), and delete is faster than add (for either immediate or orderly).

Table I also gives the lockout times. The lockout time is the interval during which critical tables are being updated and is the maximum time during which an element of a table cannot

Table I. Reconfiguration Times (s)

Ports CP	Delete			Add		
	Lockout	Immediate	Orderly	Lockout	Immediate	Orderly
500	0.3	2.4	4.6	0.2	4.5	6.6
1,000	0.4	4.5	8.9	0.4	8.7	13.0
2,000	0.8	8.8	17.6	0.7	17.1	25.5

be accessed for ordinary call handling during the reconfiguration process. The average lockout time would be half of the maximum shown.

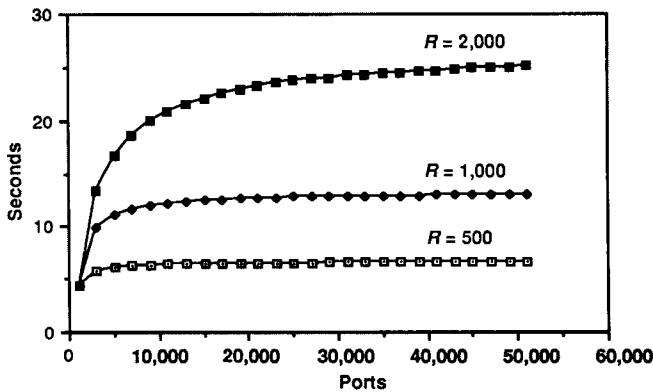


Fig. 6. Orderly add times.

Burst Dispersed Control Summary

Burst dispersed control is seen to offer the following advantages:

- Capacity expansion limited only by the number of system ports
- Ability to back up CPs and APs via the switching network itself using 1:n redundancy
- No centralized control point and thus no single point of failure
- Can operate in parts and continue to provide service even if the network is severed

The dispersed control is not limited to use with burst transport. It is useable with any self-routing transport network that provides for the delivery of information packets based on a destination address contained in the header, such that any two processors of the control can intercommunicate. Certainly a packet or ATM network supplies this capability.

Burst Transport

Architecture

Figure 7 shows that a burst switch comprises link switches and hub switches. The link switches are small switching elements designed to be remotely located from the hub switch.

Thus, the burst switch is itself a network but corresponds in this configuration to an end-office switch. Networks of burst switches and other types of switches can be formed.

Lines and trunks access a burst switch via a link switch. The experimental model link switch serves 24 ports (lines or trunks). Link switches are interconnected by links, T1 links running at 1.544 Mb/s in the experimental model. The link switches are organized into link groups, with 16 link switches per link group.

The link switches are genuine switches. A connection between ports on the same link switch is completed entirely within that link switch; a connection between ports on two link switches in the same link group is completed entirely within that link group.

The hub switch is a high-capacity ring switch designed for connections among 256 link groups. The experimental model hub switch runs at a clock rate sufficient to support 16 link groups.

While the link groups may be configured as trees or rays, the ring interconnection shown in Figure 7 is preferred because it provides alternate routing: every link switch has two ways to get to the hub, and two ways to get to every other link switch in its link group.

Burst Format

A burst consists of a header, an information field of any length, and a termination character signalling the end of the burst.

As shown in Figure 8, the 4-byte header bears the burst type, the address of the destination port, and the header check sum. The destination port address is expressed in terms of link group, link switch within link group, and port within link switch.

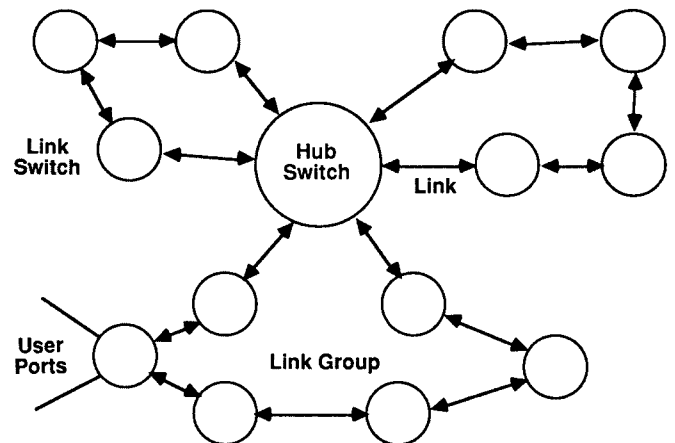


Fig. 7. Burst switch.

The information consists of the digitized samples of a talk spurt, a transmission block of data characters, or the characters of a message between control processors.

The termination character, FF_{hex} in the experimental model, is the means by which processors receiving a burst identify the end of that burst. So that this value can be used as a header or information character, it is preceded by an escape character, 7F_{hex}, when it does not signify end of burst.

Link Switch

Figure 9 shows the link switch in greater detail. Lines and trunks access the link switch through Port Circuits (PCs). The PC transforms the external signal form to the internal burst form. Thus, bursts are created in the PCs. When a PC determines that a burst is beginning, applying the criteria appropriate

		Bit							
		0	1	2	3	4	5	6	7
Byte	0	Burst Type				Link Switch			
	1	Link Group							
	2					Port			
	3	Header Check Sum							

Fig. 8. Burst header format.

ate for the external signal form it handles, it prefixes the header and supplies the characters of the burst to the PC bus.

Each burst passes through a dynamically assigned buffer in Buffer Memory (BM). The Switching Processors (SPs) administer the movement of burst characters in and out of BM. A burst may arrive over a link or from a local port, and be delivered to a link or to a local port. Between bursts, the termination character is used as the fill character.

The input SP acquires a buffer from the free queue to pass the burst's characters through; deposits each character received in the buffer; interprets the destination address in the burst's header to select the appropriate output; and queues the burst on that output.

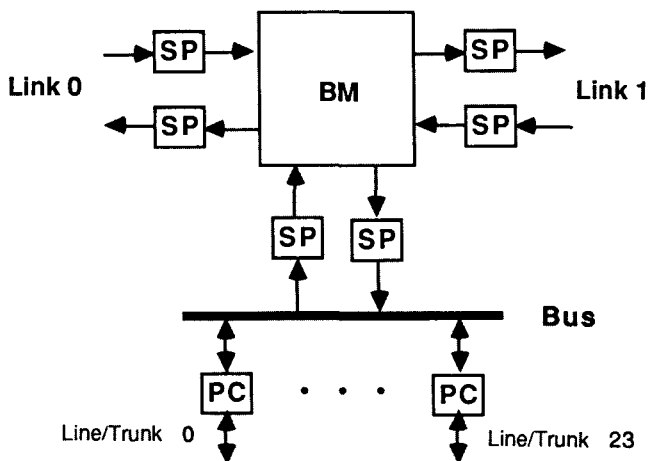


Fig. 9. Link switch detail.

The output SP finds the burst on its queue; reads the burst's characters one by one from the buffer and delivers each to its link (or port); and returns the buffer to the free queue at the end of the burst.

The SPs are specially designed processors that can execute a 64-b instruction every 100 ns. The load on the input SPs is the highest. They must be able to perform all actions associated with storing and interpreting a character each channel time.

Switching Method

Now we consider the flow of bursts through a link switch. Figure 10 shows a simplified case consisting of only one input Switching Processor (SPi) and one output Switching Processor (SPo). The queue for SPo output is represented by Q. Three bursts are in transit through three buffers in memory (B1-B3). Each burst is in one of the three major phases of burst transport.

SPi has received the first character of the first burst in its channel c1 and has stored it in buffer B1. SPi previously acquired B1 from the free queue, a queue just like Q. SPi has retained the address of B1 (b1) in its local memory associated with channel c1 so that subsequent characters of the burst will be stored in B1. On the basis of the information in the first character, SPi has determined that output should be via SPo, therefore it has enqueued the burst on Q. This means that SPi has passed the address b1 to Q.

But output of the first burst has not yet commenced. SPo, when its next idle channel occurs, will refer to Q to acquire the address of any burst awaiting output and get b1. The burst's characters will accumulate in B1 until the idle channel occurs.

SPi is inputting the characters of the second burst to B2 as SPo is outputting them. SPi inputs the characters from its channel c2 and SPo outputs the characters in its channel cX, probably a different channel number. Both processors have the buffer address b2, one for input and one for output. SPo has previously acquired b2 from Q. B2 probably has only one char-

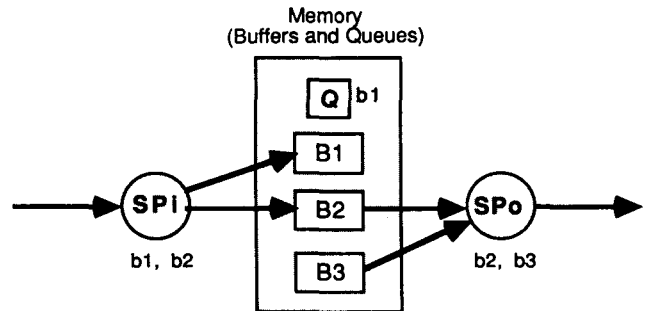


Fig. 10. Burst flow.

acter at a time in it, since the expected case is that an output channel will be acquired within one frame time from the time input begins. Since the input and output rates are equal, there is no need to accumulate characters in B2 before output starts.

SPi has detected the end of the third burst and no longer retains b3. SPo is outputting the last characters from B3. When SPo detects the end of the burst, it will return B3 to the free queue.

We note from this example that there is nearly immediate cut-through; thus, the transit delay through the switch is low. There is no accumulation of characters in a packet sense. Output can begin as soon as enough header characters have been received to permit routing, and as soon thereafter as an idle output channel occurs. In almost all cases, the first character supplies enough information to permit immediate routing. There are only two exceptions: when a burst enters the hub, the second byte must be received to determine the destination link group; and when the burst arrives at the destination link switch, the third byte must be received to determine which port the burst must be queued to (see Figure 8).

Differences in Handling Voice and Data Bursts

The description above applies equally to voice, data, and control bursts. All are switched through exactly the same switching fabric and handled in a nearly identical way. If a burst is queued for output and no output channel is free, the burst must wait until a channel becomes free. (Preemption was not included in our experimental model since the probability that a burst will need to wait more than a frame time to acquire a channel is extremely small.) While waiting, the burst's characters will accumulate in a buffer.

However, there are some differences in how voice and data ought best to be handled. Voice is more sensitive to delay than data, and data is more sensitive to loss than voice. Thus, it makes some sense to give preference to voice bursts in channel assignment (to minimize delay) and to buffer delayed data bursts (to eliminate loss). In addition, it is desirable to keep the system responsive by propagating control bursts quickly, and since they are short and occupy little channel time, they are in turn given preference over voice. Thus, a queue such as represented in Figure 10 is in fact eight queues, one for each burst type and assignment priority. An output SP will assign an idle channel to the first burst in the highest-priority non-empty queue.

We note from Figure 10 that every burst passes through a buffer; thus, a voice burst can be buffered as well. But there is a tradeoff between buffering the burst (thus contributing to delay) or clipping the burst, in the rare circumstance when a voice burst is stymied awaiting an output channel. When a burst is clipped, the accumulated characters are thrown away. Thus, there is front-end character loss but the later characters of the burst are not delayed. Beyond a certain point it is better to clip than to further delay the entire burst.

The choice we have made in the experimental model is to accumulate up to 32 voice characters (4 ms of delay at 64 kb/s)

before clipping. As soon as a channel becomes available, output of the voice burst will begin on it. The whole burst is not lost unless no channel becomes free during the entire duration of the burst. So a voice burst will not be delayed by more than this amount, though it may be clipped on the front.

This ability to buffer portions of voice bursts was not appreciated in [4, p. 654], where burst was characterized as instantly clipping voice bursts if no channel was available at the moment of need. Since each burst flows through its buffer, and buffers can be chained, as much of a burst can be buffered as desired.

So bursts of all types are handled in the same way, each flowing through its buffer, etc. The differences with respect to channel assignment priority and the extent of buffering do not constitute any fundamental difference, as has been elsewhere misunderstood [5, p. 25], but are simply enhancements to improve the performance with respect to control, voice, and data.

Interfacing to Other Systems—Port Circuits

While the burst discipline is used within a burst switch (among link and hub switches), and between burst switches using burst trunks, a burst switch is entirely capable of interfacing to switches of other types. The link switch shown in Figure 9 is where outside access to a burst switch occurs, and the Port Circuit (PC) is where the conversion between the outside signal form and the burst form is done.

Figure 11 shows the essential elements of the PC for an analog telephone. All of these elements do signal conversion of their own particular sort. A burst arriving from the PC bus on the left will enter the PP, character by character. The PP strips the header and termination characters from the burst and supplies the information characters to the codec. Between bursts, the PP supplies characters of "quiet tone." The codec converts the digital sample stream to an analog signal, and the Subscriber Line Interface Circuit (SLIC) converts to 2-wire.

In the reverse direction, the PP runs the voice/silence detection algorithm on the digital sample stream from the codec. Upon detecting a silence-to-voice transition, it begins issuing a burst with header. Upon detecting a voice-to-silence transition, it ends the burst with a termination character.

PCs for other signal types are similar. The PC for an E&M trunk is much the same except that no SLIC is required. The PC for an RS-232 line is basically a PP and a Universal Asynchronous Receiver Transmitter (UART). The PC for a T1 trunk would consist of a port process for each channel, plus a multiplexer/demultiplexer. Here, a number of channel processes could run on a common processor. The PC for an X.25 trunk would consist basically of a PP and packet network interface.

There is no difficulty, conceptual or otherwise, in interfacing a burst switch to switches of other types.

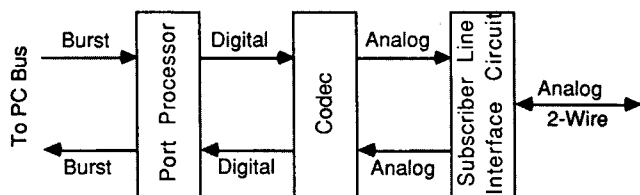


Fig. 11. Analog telephone port circuit.

Is Burst Fast Circuit Switching?

It has sometimes been suggested that burst is a form of fast circuit switching. Burst is not fast circuit switching as defined in the early 1980s [6] [7], when it was envisioned that a circuit switched connection could be broken down and reestablished with rapid signalling between talk spurts. It has rightly been observed that this procedure would be complicated and would

constitute a significant delay even with rapid signalling.

In burst, two ports are "connected" when each has received the network address of the other. The control processors, as described above, supply this address during call set-up to the PP in the form of a burst header bearing the address. When a port determines that a burst has begun, it prefixes the header to the information, and the self-routing properties of the burst transport use the header destination address to route the burst for delivery to the other port in the connection. No switching or transmission facilities are reserved until a burst occurs. And in particular, no retransmission of signalling information is required. Then burst is a form of fast circuit switching as that term is used in [8].

Is Burst Fast Packet Switching?

A burst looks like a packet, with a header bearing information that is used to route the burst to its destination. But a burst is sent in a time division channel on a link between nodes, interleaved with other bursts, whereas a packet is sent using the full link bandwidth, first one packet and then another. This small difference has significant performance consequences, as outlined below.

Delay

In a packet switch, the rate of character transmission on the links between nodes is usually greater than the rate at which the information is generated. Packets are used to convert between generation rate and transmission rate. A packet is accumulated at source rate and transmitted at link rate. The time of accumulation constitutes a delay called packetization delay. In addition, in most packet systems, a packet is fully received from a link before it is forwarded [9]. This reception delay is considerably shorter than the packetization delay, since it occurs at link rates, but it also constitutes a delay. Both of these delays are related to the packet length: the longer the packet, the longer the delay.

These delays are avoided in burst switching. Here, there is no packetization delay because the link transmission rate equals the information generation rate for voice traffic, which is presumed to continue as the dominant form of traffic even when voice compression techniques are used. As soon as a burst is found to be starting, transmission of the burst on the next link can commence. Moreover, there is no reception delay at an intermediate node. Output can begin as soon the routing on the first header byte is done.

The delay performance of burst is attractive because there are neither packetization nor reception delays. Below, we examine the computed delays for a burst switch.

Delay Variation

Traffic will ebb and flow, with the number of active users and with each active user's burst issuance frequency. As the traffic at a node increases, a burst must expect to wait longer for an available outgoing channel. So the delay experienced by one burst may vary by a few milliseconds from that experienced by another in the same connection. This delay variation translates into a variation in the time between bursts. This is of no consequence at all for data. In voice, talk spurts average about 300 ms in length. At 33% speech activity, the silence intervals between talk spurts average about 600 ms in length. If the length of the silence interval as received differs from that sent by a small percentage, it cannot be detected. Therefore, delay variation is not a problem for burst.

The situation for voice packet switching is different. So that packetization and packet reception delays are not too great, the packet length is usually constrained to be considerably less than the average length of a talk spurt, requiring most talk spurts to be sliced into many packets. But the delay experienced by one packet of a talk spurt will normally differ from

the delays experienced by others. A later packet might be less delayed and outrun an earlier packet. A later packet might be more delayed than an earlier packet, creating a gap at the output. Thus, an apparatus is required, of time stamps, sequence numbering, and an additional purposeful delay at the output for the first packet (to permit later more delayed packets to catch up)—all to permit correct reconstitution of a talk spurt.

The burst approach, with an entire talk spurt or data block in one burst, avoids this complexity, and also avoids the additional destination end delay.

Header Efficiency

Each packet requires a header. The shorter the packet, the more packets and the more header characters, which increase the percentage of link bandwidth used for overhead. So a tradeoff is necessary in packet switching: short packets have less delay (desirable) but greater overhead (undesirable).

This tradeoff is avoided in burst. Here, there is no need to packetize; thus, there is no need to establish a packet length. An entire data block or talk spurt is contained in one burst, with only one header required, even for bursts that are endless (e.g., music). Thus the header overhead is significantly less in burst than in packet switching.

Bandwidth Flexibility

The burst approach is channelized. This works very well if all sources are less than or equal to the channel rate. However, multiple channels must be put together for sources that are greater than the channel rate.

In the packet approach, a packet is accumulated at the source rate, whatever that is, and transmitted at the link rate, whatever that is. Thus, a variety of source rates can be handled in a very natural way.

If the system is required to handle a great variety of rates, the packet switching approach is superior to burst. If voice traffic predominates, the burst approach is superior.

How Does Burst Compare to ATM?

Asynchronous Transfer Mode (ATM) is packet switching, so the assertions made above with respect to fast packet switching apply to ATM as well.

The main emphasis in ATM work is on broadband switching, at speeds greater than, say, 50 mb/s. In addition ATM packet sizes may be quite short—16 or 32 bytes of information [10]. Such short packet lengths mean that initial packetization delays are correspondingly short—2 or 4 ms at a voice sampling interval of 125 μ s. The high speeds maintained between switches mean that the tandem packet reception delays are very short: a 36-byte packet (4 header bytes and 32 information bytes assumed) would be received in 5.76 μ s at 50 mb/s. So the combination of high speeds and short packet lengths probably make the delay performance of ATM switches quite acceptable.

As noted above, short packet lengths decrease bandwidth utilization: to $32/36 = 0.89$ in the example. This is relatively high, and the 11% inefficiency is hardly of much importance given the great bandwidths available on fiber.

But if longer packets of 256 bytes or more [11] are used, the initial packetization time increases to 32 ms or more. This could create echo problems unless all phones in the network are 4-wire or have echo cancellation.

This leaves the question of delay variation, which must be compensated for. We have no information on the delay variation and compensation of ATM switches.

Computed Burst Delay

A burst will often pass through many switching nodes on the way to its destination. Even though the delay through one node is small, the cumulative delay through a series of link and hub

switches is an important performance parameter.

To determine the echo performance of a burst switch, we computed the round trip delay a burst would experience if it passed through eight link switches in the origin link group, the hub switch, and eight link switches in the destination link group; and then back again, as shown in Figure 12.

Each link switch along this path of 32 link switch and 2 hub switch transits had 24 ports generating 0.25 Erlang of traffic at 0.33 burst activity toward and from the hub.

This study showed that the burst delay had only two components: fixed delays (e.g. serial to parallel character accumula-

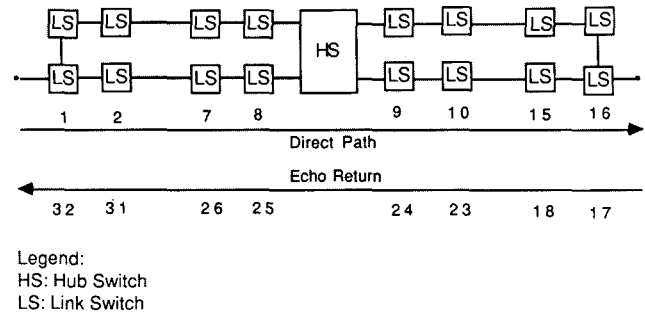


Fig. 12. Echo path.

tion time) and channel slip delays (i.e., the delay between the incoming and outgoing channel). Queuing delays (i.e., the delay a burst experiences because there is no idle channel available within a frame time) occur with negligible probability.

The total delay probability distribution then consists of the convolution of all the individual switch element channel slip delay distributions together with the sum of the individual switch fixed delays. The computed results were that the round trip delay would be < 5 ms at a probability of 0.99999. Thus, the burst delay performance is acceptable even without echo cancellers.

Burst Project Status

Four link switches and a hub switch have been built and operate satisfactorily. These are shown in the Figure 13 photograph.



Fig. 13. Burst project link switches and hub switch.

The dispersed control has been programmed to provide ordinary voice and data call set-up, as well as a few user features (e.g., line and trunk hunting, and group and directed call pick-

up), sufficient to verify that a non-centralized dispersed control can handle the features likely to be required of a commercial or military switch. Finally, on-line reconfiguration, in which CPs can be added or deleted either by command or on the basis of failed background test message traffic concurrently with normal call processing, performs successfully.

References

- [1] S. R. Amstutz, "Burst Switching—An Introduction," *IEEE Commun. Mag.*, Nov. 1983.
- [2] S. R. Amstutz and R. D. Packard, "Dispersed Control Architecture and Specification Compilation of Burst Switching," *Software Engineering for Telecommunication Switching Systems*, Eindhoven, The Netherlands, Apr. 14, 1986.
- [3] E. F. Haselton, S. R. Amstutz, and J. M. Lenart, "Burst-Switching Communications System," United States Patent No. 4,698,803, Oct. 6, 1987.
- [4] P. O'Reilly, "Burst and Fast-Packet Switching: Performance Comparisons," *Infocom '86*.
- [5] T. M. Chen and D. G. Messerschmitt, "Integrated Voice/Data Switching," *IEEE Commun. Mag.*, June 1988.
- [6] E. A. Harrington, "Voice/Data Integration Using Circuit Switched Networks," *IEEE Trans. on Commun.*, vol. COM-28, no. 6, June 1980.

(Continued from page 24)

- [2] G. Hayward, L. Linnell, D. Mahoney, and L. Smoot, "A Broadband ISDN Local Access System Using Emerging-Technology Components," *Proc. of Int'l. Switching Symp. '87*, vol. 3, pp. 597–601, 1987.
- [3] J. S. Turner, "New Directions in Communications," *IEEE Commun. Mag.*, vol. 24, no. 10, pp. 8–15, Oct. 1986.
- [4] J. P. Coudreuse and M. Servel, "Prelude: An Asynchronous Time-Division Switched Network," *Proc. of the IEEE Int'l. Conf. on Commun. '87*, vol. 2, pp. 769–773, 1987.
- [5] C. Day, J. Giacomelli, and J. Hickey, "Applications of Self-Routing Switches to LATA Fiber Optic Networks," *Proc. of Int'l. Switching Symp. '87*, vol. 3, pp. 519–523, 1987.
- [6] D. Spears, "Broadband ISDN—Service Visions and Technological Realities," *Int'l. J. for Analog and Digital Cabled Syst.*, vol. 1, no. 1, pp. 3–18, Jan. 1988.
- [7] S. Minzer, "Broadband User-Network Interfaces to ISDN," *Proc. of the IEEE Int'l. Conf. on Commun. '87*, vol. 1, pp. 364–369, June 1987.
- [8] CCITT Recommendation I.121, "Broadband Aspects of ISDN," Blue Book, Geneva, Switzerland, 1989.
- [9] R. Ballart, Y. C. Ching, "SONET: Now It's the Standard Optical Network," *IEEE Commun. Mag.*, vol. 29, no. 3, pp. 8–15, Mar. 1989.
- [10] CCITT Recommendation G.707, "Synchronous Digital Hierarchy Bit Rates," Blue Book, Geneva, Switzerland, 1989.
- [11] CCITT Recommendation G.708, "Network Node Interface for the Synchronous Digital Hierarchy," Blue Book, Geneva, Switzerland, 1989.
- [12] CCITT Recommendation G.709, "Synchronous Multiplexing Structure," Blue Book, Geneva, Switzerland, 1989.
- [13] T. Kuhn, *The Structure of Scientific Revolutions*, 2nd ed., Chicago, IL: University of Chicago Press, 1970.
- [14] J. Gechter and P. O'Reilly, "Conceptual Issues for ATM," *IEEE Network*, vol. 3, no. 1, pp. 14–16, Jan. 1989.

- [7] P. Chen, "Use Hybrid Switches for Voice and Data," *Computer Design*, Oct. 1983.
- [8] S. D. Personick and W. O. Fleckenstein, "Communications Switching—From Operators to Photonics," *Proc. of the IEEE*, vol. 75, no. 10, Oct. 1987.
- [9] J. S. Turner, "Fast Packet Switch," United States Patent No. 4,491,945, Jan. 1, 1985.
- [10] M. DePrycker, "Definition of Network Options for the Belgian ATM Broadband Experiment," *IEEE J. on Sel. Areas in Commun.*, Dec. 1988.
- [11] K. Y. Eng, M. G. Hluchyj, Yu, and Y. S. Yeh, "A Knockout Switch for Variable-Length Packets," *Proc., IEEE Int'l. Conf. on Commun. '87*, June 1987.
- [12] A. Huang and Scott Knauer, "Starlite: A Wideband Digital Switch," *Proc., IEEE GLOBECOM '84*, Dec. 1984.

Biography

Stanford R. Amstutz received his S.B. in electrical engineering from the Massachusetts Institute of Technology in 1955 and his M.S. in mathematics from Northeastern University in 1965. He is now Manager of the Distributed Systems Architecture Department at GTE Laboratories, Waltham, MA. Previously, he was Principal Investigator for Burst Switching at GTE Laboratories; Director of Software Development for the Nixdorf Computer Corporation; and Vice President of Product Development for Comshare, Inc. He holds eight patents and received the GTE Warner Achievement Award for Burst Switching Technology.

- [15] B. Eklundh, I. Gard, and G. Leijonhufvud, "A Layered Architecture for ATM Networks," *Proc. of IEEE Globecom '88*, pp. 409–413, Nov./Dec. 1988.
- [16] R. M. Newman, Z. L. Budrikis, and J. L. Hullett, "The QPSX MAN," *IEEE Commun. Mag.*, vol. 26, no. 4, pp. 20–28, Apr. 1988.
- [17] D. P. Hsing, F. Vakil, and G. H. Estes, "On Cell Size and Header Error Control of Asynchronous Transfer Mode (ATM)," *Proc. of IEEE Globecom '88*, pp. 394–399, Nov./Dec. 1988.
- [18] T1S1.1/88-390, "Mapping BISDN ATM Cells into SONET STS-3c Payload," AT&T Technologies, Oct. 1988.
- [19] S. E. Minzer and D. R. Spears, "New Directions in Signaling for Broadband ISDN," *IEEE Commun. Mag.*, vol. 27, no. 2, pp. 6–14 and 52, Feb. 1989.

Biography

Steven E. Minzer is a Member of Technical Staff in the Multimedia Communications Research Division of Bellcore, Morristown, New Jersey. He has been involved in Broadband ISDN standards activities since 1985, participating in Committee T1 and CCITT Study Group XVIII meetings. He has also been involved in protocol design for an experimental broadband network at Bellcore. Before joining Bellcore, he was with AT&T Bell Laboratories, designing and implementing software for the DMERT operating system. He also worked in database administration and software development for GTE Automatic Electric. He obtained degrees in political science from Cornell University (B.A.) and Ohio University (M.A.) and in computer science from Northern Illinois University (M.S.). His publications include several papers and a Bellcore Special Report on BISDN and articles in *ISS '87* and *ICC '87* on protocols for the experimental broadband network.