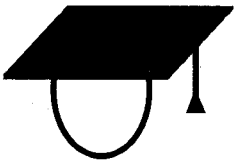


A Literature Survey on Traffic Dispersion

Eva Gustafsson, Royal Institute of Technology
Gunnar Karlsson, Swedish Institute of Computer Science

Abstract

Aggregation of resources is a means to improve performance and efficiency in statistically shared systems in general, and communication networks in particular. One approach to this is traffic dispersion, which means that the traffic from a source is spread over multiple paths and transmitted in parallel through the network. Traffic dispersion may help in utilizing network resources to their full potential, while providing quality-of-service guarantees. It is a topic gaining interest, and much work has recently been done in the field. The results are, however, difficult to find, since the technique appears under many different labels. This article is therefore an attempt to gather and report on the work done on traffic dispersion in communication networks.

Processes that have sporadic need for a resource are most efficiently supported if they may share it statistically. Allotted a fixed quantity of the resource, they would rarely use it, and when needed it might be insufficient. The efficiency can be measured in how well the resource is utilized and how frequently more processes than can be handled attempt to access the resource (some accesses get blocked). Blocking may be reduced by sharing a larger amount of the resource over a correspondingly higher number of processes.

The provision of a resource may, however, be more economical if it is offered unit by unit to meet a growing demand; or the resource may need to be partitioned simply because the required size cannot be provided as a monolithic whole. Consider that resources, such as the processing power of a central processing unit (CPU), storage capacity of a random access memory, or transmission capacity of a link, are available in limited quantities due to physics, production methods, and economics. A resource partitioned into separate units may also be more reliable if failure of one unit does not affect the operation of the others. It is well known that a resource which is split into several physical units is better utilized for a given probability of blocking if the full resource can be shared as a whole rather than if each unit is shared separately by a limited number of processes. This is illustrated in Fig. 1.

In this article, we will look at a specific instance of this general method. The processes are communication processes, and the resource is the link capacity in a packet-switched network. A fundamental characteristic of packet switching is the use of statistical multiplexing, which allows several sources to share the capacity of a link statistically. This means that the demand for capacity may sometimes exceed what is available, and packets may be lost. Such a problem of statistical multiplexing is aggravated when the traffic arrives in bursts. Recent studies [1–3] indicate that data traffic exhibits pronounced correlation with a high variance over long time periods. Keeping the loss probability at a tolerable level for such traffic would require unreasonably low utilization of the network capacity. The problem might be handled at the source by spreading the packets in time (shaping). However, when the traffic is generated at a high rate in long bursts, the delay introduced by the shaping may become unacceptable.

For a reasonably well connected network there would be several paths from a sender to any given destination. It may be necessary, for instance, to provide reliability. The total capacity is thus partitioned spatially over the paths. Traditionally, only one of them would be chosen for the information transfer; usually the shortest one, measured in actual length or number of hops. This procedure corresponds to spatial partitioning, according to Fig. 1b. Consider also that if the sending process should need more capacity than any single path could provide, the communication would be eternally blocked.

Instead of using a single path, the sending process might disperse its traffic over all the paths leading to the desired destination, according to Fig. 1c. A resource sharing close to the optimal would then be possible. We call this generic technique *traffic dispersion* (Fig. 2).

Traffic dispersion makes it possible to alleviate the effects of bursty traffic and equalize the network load without introducing the delay incurred by shaping. The technique applies equally well to datagram as to virtual circuit-switched networks (e.g., based on the Internet Protocol, IP, and asynchronous transfer mode, ATM, respectively). The traffic can be dispersed over multiple paths in the network, multiple links within a path, or multiple physical channels, such as frequency or wavelength channels, within a link (Fig. 3). The important

0890-8044/97/$10.00 © 1997 IEEE

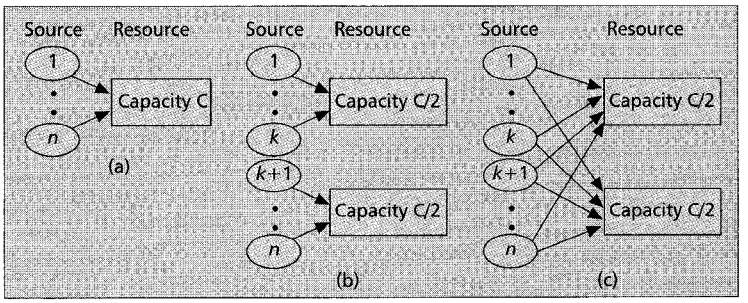
thing in order to yield a gain is that the paths, links, or channels do not share transmission capacity statistically. Traffic is dispersed packet by packet or batch by batch over the chosen set of routes. The packets are put back in order at the receiver if needed. The bigger the difference in lengths of the routes, the greater the chance that packets overtake each other.

Over the last few years, there seems to be a growing interest in traffic dispersion to handle bursty traffic. It is, however, hard to get an overview of the accomplishments to date since they have been published under various guises. This article is an attempt to gather and report the work on spatial traffic dispersion in packet-switched networks. We also mention the issues which, despite all the research that has already been done, remain to be addressed. Most of the work has concentrated on evaluating the gain in statistical sharing for dispersion of various granularity. There are, however, also studies on the use of forward error correction for enhancing reliability and compensating for packet loss, on protocol functions to provide traffic dispersion in ATM networks, on the resequencing problem, and on switch architectures that use dispersion internally to reduce queuing.

Although this survey is restricted to the usage of the dispersion idea to networking, we acknowledge that it has been proposed in other contexts. Rabin suggests dispersion of the information in a file into a number of pieces or locations, while using error-correcting codes to increase security and fault tolerance [4]. Also, the traffic in mobile communications may be dispersed over the full width of the available spectrum within one radio cell in order to make all sources equally affected by poor-quality frequency regions. Apart from better statistical sharing of the spectral resource, such spread-spectrum techniques also make it more difficult to eavesdrop on a source.

Traffic dispersion is akin to alternate-path (multipath) routing, which provides several possible paths from which to choose in case the optimal path for some reason becomes inoperable. Sidhu *et al.*, for example, suggest controlling congestion by alternate-path routing: if a node notices congestion on its primary path, it reroutes the traffic over a precomputed alternate path [5]. Bahk and El Zarki propose a similar scheme [6]. The distinction we make between traffic dispersion and alternate-path routing is that the latter is done on the time scale of a session, while dispersion is done on packets or batches of packets within a session. In most instances dispersion is preventive, while alternate-path routing is reactive, triggered by some network problem.

The article is organized as follows. The second section presents different algorithms for traffic dispersion in communication networks, and the third section discusses some remaining research areas in the field. The fourth section concludes the article.



■ Figure 1. *a) Sporadic sources should optimally share a common resource statistically; b) the sharing deteriorates when the resource is fragmented into multiple physical systems; c) the sharing can be improved by dispersing the resource requests over all of the physical systems.*

Traffic Dispersion in Communication Networks

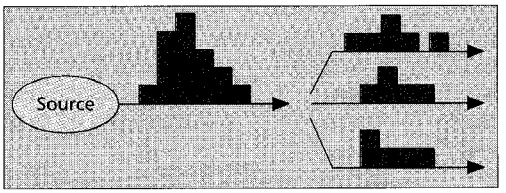
Dispersity Routing

Maxemchuk has, since 1975, advocated spreading the traffic from a source in space rather than in time as a means for load balancing and fault handling in packet-switched networks [7–9]. According to the technique, which he calls *dispersity routing*, a message is divided into a number of submessages, which are transmitted in parallel over disjoint paths in the network (Fig. 4). His work is, as far as we know, the first proposal for traffic dispersion, and we have therefore chosen to present it in detail.

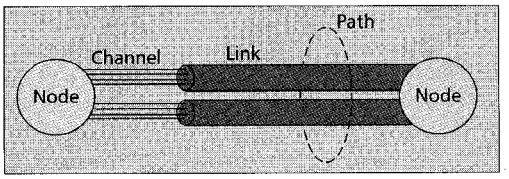
Traffic dispersion requires disjoint paths, meaning that the paths should not share links statistically. This makes a transmission error on one path independent from errors on the other paths, and forward error correcting codes could therefore be successfully used. The paths used for a transmission should also be of roughly equal length. They should traverse the same number of nodes in the network, in order to minimize the delay variations among packets sent on different paths, and a similar distance, to avoid divergence in propagation delay.

Maxemchuk distinguishes between redundant and nonredundant alternatives. With the former, a message is divided into as many submessages as there are paths available for the transmission, and each submessage is transmitted along one of these paths. All the submessages must be received to permit the original message to be reconstructed, and the total delay is thus dependent on the worst path used. Redundant dispersity routing, on the other hand, allows a message to be divided into fewer submessages than there are paths, and additional submessages are constructed as linear combinations of the primary ones. This technique allows error-correcting codes to be applied, and has the advantage that only some of the submessages have to be received to reconstruct the original message. Clearly, this improves the delay and packet loss properties at the expense of sending more data through the network.

Dispersity routing was originally suggested for store-and-forward data networks [7]. The method was shown to be superior to ordinary single-path transmissions by reducing the calculated average delay of a message by about 75 percent when dispersed over two paths, and by 90–95 percent when dispersed over five paths. These results were obtained assuming exponentially distributed message lengths and interarrival times. The variance of the delay increased, since one must wait for multiple submessages to arrive, but the total delay was still lower than with single-path transmission. Moreover, with redundancy, dispersity routing proved to sustain a larger number of transmission errors before requiring a message to be retransmitted, and made it possible to continue operation despite link failures.

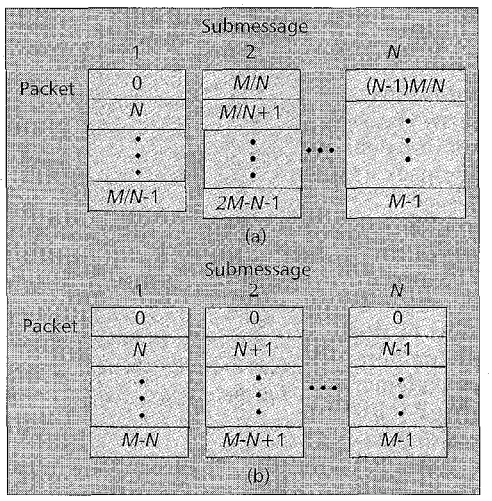


■ Figure 2. *Illustration of spatial traffic dispersion.*



■ Figure 3. *Multiple channels and links between nodes in a network.*

Many of these advantages are shown to be maintained when introducing the routing strategy in other types of networks [8, 9]. In [8], an analytic model is developed to verify the effects of traffic dispersion in virtual circuit networks. The model focuses on an example of transmitting medical images where the message size is exponentially distributed with an average of 50 Mb, and the interarrival time between messages is also exponentially distributed with a mean of 11 s. The channel rate is 150 Mb/s, and 95 percent of the messages are required to be received within 1 s. With this type of traffic, the utilization of a dedicated channel is rather low, only about 3 percent. Therefore, multiple sources should preferably share the channels in order to increase utilization. Traffic dispersion could help to reduce the effects of each burst on a single channel.



■ Figure 4. *Two ways of dividing a message consisting of* M *packets into* N *submessages. A submessage is formed by a) a few consecutive packets of the message or (b) periodically chosen packets of the message.*

In the virtual circuit network the model considers, connection setup is done as follows. A source establishes a connection by placing a call on the relevant paths, but it is not assigned capacity at this time. When the source is ready to transmit, it requests a channel from each switch on the paths. If a channel is not available at a switch the message is blocked, and the source will have to retry at a later occasion. The probability of blocking is assumed to be independent at successive switches along a path, at switches on parallel paths, and for successive retransmissions. In this example, all paths used by a source for transmitting a message are independent and of equal length, measured in switches per path.

With nonredundant dispersion over five paths, with five channels per path and one switch per path, a channel utilization of almost 80 percent is obtained. The general result for nonredundant dispersion and dispersion with one redundant path is that the number of sources which can share a channel and still meet the performance requirements increases with an increasing number of paths. The number of sources also grows with the number of channels per path. When the number of switches per path increases, so does the blocking probability, and hence the number of sources that can share a channel decreases.

Maxemchuk also investigates the case when a source does not behave as expected. In his example, the utilization of one of the sources is increased from 3 to 100 percent. The results reveal that with nonredundant dispersion and five paths used, more than 90 percent of the messages still get through in less than 1 s, and the larger the number of sources sharing a switch, the smaller the effect of a misbehaving source. Redundant dispersion shows an even better capability of handling such a "rogue source."

In [9], the model is slightly modified and adapted to an ATM network. The main differences are that submessages may travel different distances through the network, different traffic classes may have different retry strategies, and the message transmission times are not long compared to the propagation delay. This model considers a random access scheme where the first cell of a submessage demands a channel on the fly and sets up the connection. The last cell of the submessage tears the connection down. If a channel cannot be acquired, the submessage is blocked and has to be retransmitted. This means that the successfully received submessages have to be buffered at the receiver, waiting for the remaining parts to be retransmitted.

When using three paths, nonredundant dispersion doubles the throughput if the ratio of the channel capacity to the source transmission rate is 2, and increases the throughput by a third if the ratio is 10. This indicates that dispersion is more useful when the capacity per path is small compared to the source rate. Adding redundancy decreases the throughput, since the redundant information causes a larger load increase than do the retransmissions.

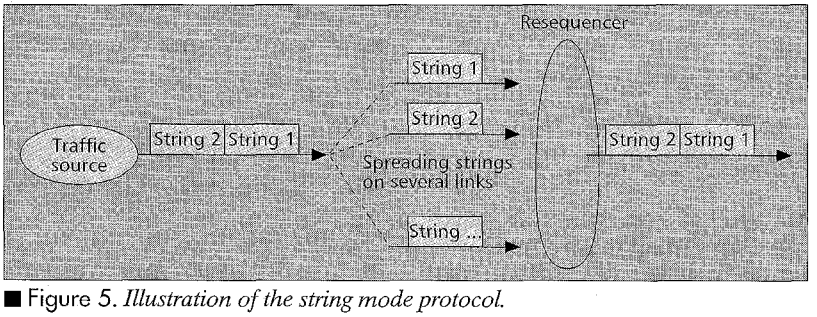
In essence, dispersion is shown to equalize load and increase overall network utilization. With redundancy, it shows a strong capability of dealing with unexpected large traffic sources. Maxemchuk also points out the possibility of reducing the message transmission time, or the capacity needed on each path. The latter point is discussed later.

The String-Mode Protocol

A more recent implementation of the traffic dispersion idea is the string mode protocol, presented by Déjean *et al.* [10, 11] and Jagd *et al.* String mode is a protocol for dividing long bursts into smaller subbursts, called *strings*, and distributing them on a number of parallel links (virtual paths) in an ATM network (Fig. 5).

String mode differs from dispersity routing in that it creates a string from a number of consecutive cells with a guide cell at its beginning. The guide cell contains self-routing information and a string identifier which allows several strings to be multiplexed on the same virtual path. The payload-carrying cells following a guide cell are identified by the same string identifier, which is eventually released by the last cell of the string. String mode may thus be viewed as a protocol layer above the ATM layer in the switches. This added layer routes strings that are larger then cells — the routing format of the ATM layer. It is thereby similar to an IP layer routing packets.

A flow of data handled by the string mode protocol passes through a "chopper," string routers, and a resequencer. The chopper divides the data flow into strings. Routing of strings is determined during connection setup, but the individual ATM virtual paths are not allocated at this point. Instead, they are determined on the fly, so the function of a string router in the network is to select a virtual path for the string



■ Figure 5. *Illustration of the string mode protocol.*

whenever a guide cell arrives. Since all the cells within a string follow the same guide cell, they always arrive in sequence. The strings encounter different delays along different virtual paths and may arrive at the destination overlapping or even completely out of order. They are therefore buffered at the resequencer, which will arrange them in order to reconstruct the original message. Two possible methods of handling the resequencer memory are discussed. One method is to write in the strings as they arrive, and then search the memory area to discern the strings and read them out in sequence. This searching procedure has to be very fast. Another possibility is to write each string into its own memory area, so the strings are resequenced spatially. The latter strategy requires more complicated memory partitioning, but in return the memory does not have to be searched in order to read out the strings.

For bit rates below a certain threshold, the string length is set to infinity, whereby the whole data flow is routed in one and the same string. With increasing bit rates, the string length shortens towards a minimum value, where the addition of a guide cell eventually results in an unacceptable high overhead. It should be noted that the string mode protocol is somewhat sensitive to transmission faults, since the loss of a guide cell results in loss of an entire string.

The effectiveness of string mode is evaluated by simulations. A number of on-off sources, described by two-state Markov chains, generate traffic into a number of first-in first-out (FIFO) queues. Two dispersion strategies are investigated, namely random, according to a uniform distribution, and cyclic spreading of strings. In the system simulated in [10, 11], the deterministic service time of each FIFO queue corresponds to an output rate of 135 Mb/s. The buffer size is fixed at 128 cells, and the peak-to-mean ratio of each traffic source is 5. The results are given in terms of cell loss probability.

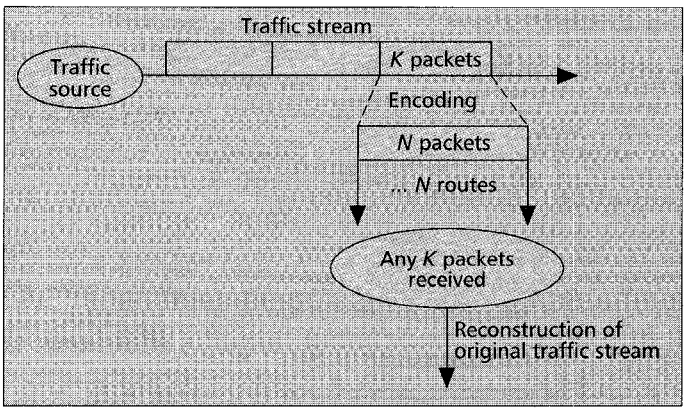
When increasing the number of links used for a transmission, the cell loss probability improves and levels out at about 15 links. It also turns out that minimum cell loss probability is obtained with a string length between 10 and 20 cells. More generally, the string length should be kept at about 10 percent of the buffer capacity to get the best performance. For fairly high probabilities of cell loss, such as 10^{-2}, the load can be increased up to its double value when using the string mode protocol. Throughout the simulations, cyclic distribution of strings gives significantly better results than random distribution.

The string mode protocol defines how the spreading, routing, and resequencing of strings should be organized in an ATM network. The studies define optimal string length, optimal number of links, and preferred dispersion strategy of strings.

The Vector Routing Algorithm

The vector routing algorithm proposed by Lee and Liew [13] is a dispersion scheme with redundancy. As in the string mode protocol, the traffic from a source is partitioned into submessages (called *streams* by the authors), each consisting of K packets. The vector routing algorithm encodes the K packets into $N > K$ packets, which are sent in parallel through the network along N separate routes (Fig. 6). When any K of the N packets are correctly received, the original message can be reconstructed. This makes the scheme similar to redundant dispersity routing with a submessage structure according to Fig. 4b.

Lee and Liew analytically model the burstiness of a traffic stream, and show that it is reduced by dispersion. Similarly, they analyze the error-correcting capability of the algorithm they propose, which uses codes derived from Reed-Solomon codes, and find that if bit error detection is used for every packet, the algorithm is capable of correcting $N - K$ packet errors in each submessage; otherwise, it corrects $(N - K)/2$ packet errors. A third analysis shows that the loss probability for a submessage is reduced for increasing N, given that the code efficiency K/N remains constant and that losses on the routes are independent of one another.



■ Figure 6. *Illustration of the vector routing algorithm.*

The article contains a discussion on the use of the algorithm in the different service classes of ATM. Traffic from class A (circuit emulation) is not bursty in nature, but the vector routing algorithm can be useful when the demand for capacity exceeds the resource of one single link. For classes B and C (variable-bit-rate service and connection-oriented data service) the algorithm is claimed to help handle bursty traffic. If vector routing is to be used for class D (connectionless data service), it has to be set up between each pair of nodes through the network.

With reference to the vector routing algorithm, Ding and Liew [14] give a more general analysis of how traffic dispersion, with and without redundancy, affects the network. The first switch connected to the traffic source disperses the traffic, which is transmitted through the network and eventually decoded, corrected, and resequenced at the receiver. The behavior of each switch along a path, due to dispersion, is assumed to be identical to the behavior of the first switch. The only exception is the last switch, where the dispersed traffic streams meet again; that switch is assumed to always behave as without dispersion. The switches are assumed to be output-buffered, and the mean length of one switch output queue is evaluated for infinite buffers and independent identically distributed sources with a general bulk arrival distribution.

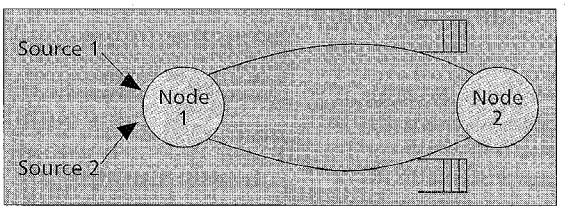
The results show that when the network load is low, dispersion is superior to one-path transmissions, and a large amount of redundancy may be used without aggravating the performance. When the network load becomes heavy, the amount of redundancy must be kept small in order to make dispersion useful. The mean queue sizes with and without dispersion become equal at about $\rho \cdot N/K = 0.9$, where $N \geq K$ and ρ is the network load without dispersion.

Next, the cell loss probability on one of the channels in a 4 x 4-switch is simulated for a certain buffer size. The traffic sources in this case are of the on-off type, each characterized by a two-state Markov chain. It turns out that dispersion with or without redundancy reduces the cell loss probability. When the network load increases, a large amount of redundancy may eventually lead to a loss probability of the same size as without dispersion. More precisely, the effective load $\rho \cdot N/K$ should again be kept smaller than 0.9.

Essentially, this dispersion scheme confirms the earlier results on reducing cell loss probability and traffic burstiness. This is done by investigating the behavior of the switches in the network. The vector routing algorithm specifies a coding scheme, based on the Reed-Solomon code, for use in redundant dispersion.

Resource Allocation on the Cell, Burst, and Connection Levels

Krishnan and Silvester [15–17] claim a necessity of adjusting routing decisions in ATM networks to the needs of each specific type of traffic. They discuss making resource allocation on the cell, burst, or connection level, depending on traffic characteristics, thereby achieving finer control of the network resources.



■ Figure 7. *Single-hop network with two nodes.*

In [15] the authors develop formulas for calculating the queuing delay, the queue blocking probability, and, in the case of dispersion, the resequencing delay. The formulas consider on-off sources, in the shape of two-state Markov chains, sending traffic over a single-hop network, which consists of two nodes according to Fig. 7. These formulas are used to compare the performance of connection-level allocation to dispersion of packets over two paths in a round-robin fashion.

With infinite buffers, the sum of queuing and resequencing delays decreases when the packets are dispersed. The average delay is thus decreased, by 20–30 percent, despite the resequencing. The queue blocking probability is calculated with a buffer size of 10, and is shown to decrease to about 1/5th its original value when dispersion is used. In addition, the blocking probability can be decreased even further when a "join-shortest-queue" strategy is used instead of round-robin dispersion. Such dynamic dispersion will be further discussed later. The total average delay for sources with different average rates, and for links with different capacity, shows similar behavior as above; so does the delay for several sources multiplexed together. In all these results, the gain increases with the squared coefficient of variance of the arrival traffic. This indicates that dispersion becomes more useful as the sources become more bursty.

In [16], the formula is extended to contain a larger number of sources, and is applied to an ATM multiplexer with traffic dispersion. The results from this show that dispersion decreases the average and standard deviation of the queuing delay as the number of multiplexed sources increases. Increasing the number of parallel paths also improves the queuing performance.

The results in [17] again show a decreased loss probability due to dispersion, and also reveal that the number of sources that can be supported at a multiplexer for a given quality of service increases with dispersion.

Krishnan and Silvester claim that unichannel transmission and resource allocation on a connection level suits nonbursty traffic, while dispersion and allocation on a lower level is preferred for bursty traffic sources. These arguments are in some sense similar to those used for the string mode protocol. Déjean *et al.*, however, advocate unichannel transmission through infinite string length on the basis of low source rate, not low burstiness.

Capacity Allocation on a Burst-by-Burst Basis

Suzuki and Tobagi [18] suggest to transfer individual bursts over multiple links and multiple paths to obtain statistical multiplexing gain for high-speed bursty traffic. This is dispersion according to Fig. 4b, where each submessage consists of a traffic burst.

When a traffic source requests the network to set up a route between the source and a destination, information on several possible routes is stored within the nodes, but no capacity is actually reserved. The reservation is performed for each burst, and the peak capacity of the burst is allocated on one of the possible paths. This is done step by step along the path, by choosing an available link between a pair of nodes. If, between any pair of nodes, none of the links has sufficient capacity, another path is chosen. If none of the paths can provide the required capacity, the burst is blocked and has to retry later. When the burst has been transmitted, the capacity is released.

The peak allocation guarantees absence of packet loss and also minimizes queuing delays in the nodes. Since the packet sequence is preserved within each burst, this scheme does not suffer from resequencing problems at the packet level. Complete bursts may arrive out of order, though, and the authors suggest preventing it by not sending a burst before the preceding burst is properly received. This may, however, cause unnecessary delays, and might require substantial buffering capacity at the sender.

To verify the advantages of the scheme, formulas are developed for two-state Markov chain sources, and multiple links and paths that are disjoint and of equal length. Similar to previously discussed results, these formulas show a decrease in average delay and burst blocking probability when introducing dispersion. One example considers a source peak rate of 100 Mb/s, a link capacity of 100 Mb/s, and a link length of 10 hops. By increasing the number of parallel links from 1 to 16, the network utilization can be raised from 1 to 40 percent, while still meeting the requirement of a 1 percent blocking probability.

Essentially, these results confirm those obtained for string mode and by Krishnan and Silvester. What mainly distinguishes this scheme from string mode is the string length and the fact that while string mode resequences the strings, the burst-by-burst allocation scheme avoids resequencing by delaying the bursts.

Capacity Allocation in B-ISDN

The subject of dispersing cells in ATM networks is also treated by Cheng [19]. He considers cyclic and dynamic dispersion of cells. Dispersion requires resequencing in some form, and Cheng claims that the resequencing can be done by buffering for non-real-time services as for conventional datagram networks. The buffering introduces a certain resequencing delay and might not be acceptable for real-time services. Cheng suggests a combination of unichannel and multichannel connections: if cell sequence must be preserved, the user should request a unichannel connection. Such a combination is essentially the basis of the work by Krishnan and Silvester, and is also included in the string mode protocol.

If cells are to be spread cyclically over a number of channels, which also carry unichannel connections, there might be overload, and cells will be lost. When allocating the channels dynamically according to network load, Cheng claims that the traffic distribution can be adjusted so that no individual channel is overloaded. The subject of dynamic dispersion was also briefly mentioned by Krishnan and Silvester in the "join-shortest-queue" strategy. Dynamic allocation will require additional overhead and higher complexity in the system. It also differs from the traffic dispersion idea discussed so far by being reactive rather than preventive in nature.

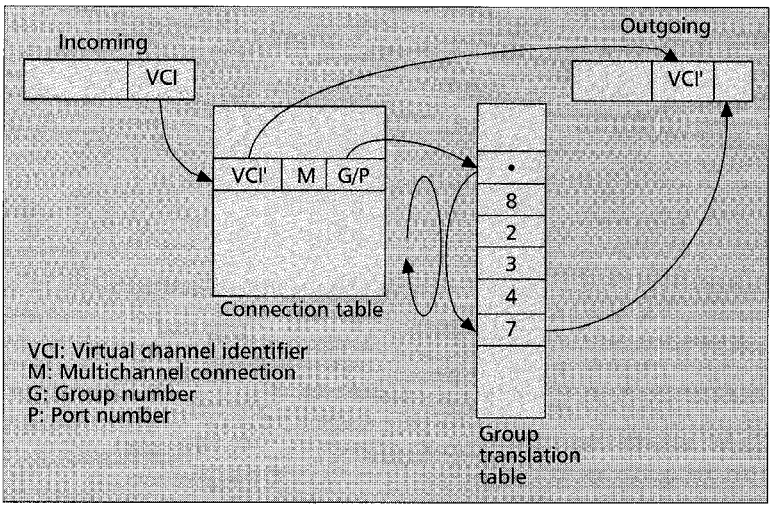
Cheng makes an analytic performance comparison of a statistical multiplexer with combined unichannel and multichannel transmissions. In this model, 320 independent, identically distributed on-off sources generate traffic at peak rate, which is 1/16th of the channel capacity, and traffic is dispersed over four channels. The formulas developed give the cell loss probability for different combinations of unichannel and multichannel transmissions. It turns out that the cell loss probability

is high for unichannel transmissions and decreases with multichannel transmissions. Round-robin dispersion decreases the cell loss probability to about 1/5th its original value, while dynamic dispersion gives a larger decrease.

A call admission control scheme is also suggested. Services are divided into a number of classes, depending on their peak and mean rates. The virtual cell loss is defined as the average amount by which the instantaneous aggregate bit rate exceeds the channel capacity, and a call is admitted when the virtual loss is below a given value.

Finally, Cheng makes a contribution in the traffic dispersion area by discussing the switch architecture for handling cyclic dispersion of cells. Cells coming from the same input virtual channel are dispersed cyclically over the output channels belonging to the destined link group, consisting of the links connecting two nodes. This is achieved by adding an extra stage in the header translation process, where an incoming virtual channel identifier (the virtual path identifier is not mentioned) is mapped not only to an outgoing virtual channel identifier, but also to an output port number to which the cell shall be switched (Fig. 8). The incoming virtual channel identifier is used to access a connection table. Each entry in this table contains three fields: outgoing virtual channel identifier, connection type, and group number/output port number. If the connection type is unichannel, the output port number is found in the third field, and no further mapping is needed. For a multichannel connection, the group number found in the third field is used to access the index group translation table, where the members of a group are stored in consecutive memory locations. A pointer is used to find the appropriate output port number for the cell in question. The pointer advances one step for each arriving cell, thus spreading the cells cyclically. This scheme assumes that dispersion, and thereby also routing decisions, of cells take place within a switch, so extra complexity is required in the switch.

These results confirm previous results by decreasing the loss probability when using dispersion. Cheng also adds to the results from other reports by showing the effects of dynamic dispersion and discussing the switch architecture.



■ Figure 8. *Illustration of the header translation in a switch for round-robin dispersion in multichannel connections.*

Parallel ATM Connections for Gigabit-Speed Applications

The performance of parallel ATM connections for gigabit-per-second applications is also analyzed by Plotkin and Varaiya [20]. Their study is on dispersing messages cyclically over several ATM virtual channels. Each message consists of a fixed number of ATM cells, and the dispersion is actually similar to the string mode strategy.

Plotkin and Varaiya evaluate methods for calculating the delays that messages suffer during transmission and resequencing. The evaluation is performed for ATM virtual channels on links of equal capacity, each containing one switch. First, multiplexing two virtual channels is analyzed. One ATM virtual channel carries Bernoulli traffic; the other carries bursty traffic. The traffic bursts are obtained by dispersing messages, which arrive back to back, cyclically over the available channels. These bursts are hence of constant length and arrive at fixed time intervals. The results reveal that the channel delay increases linearly with the burst length and also with the load. Furthermore, the distribution of cell delay appears to be approximately normally distributed. The cell loss is shown to be log-linear to the ratio of buffer size to burst size.

The authors also show that the expected resequencing delay is insensitive to burst length and small relative to burst duration, while the variance grows linearly with the burst length. This indicates that the mean delay itself is insufficient to describe a requested quality-of-service parameter during connection setup; the variance has to be accounted for as well.

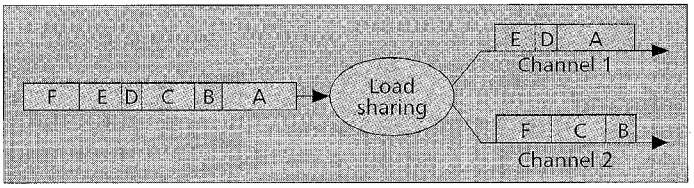
Plotkin's and Varaya's contribution to the traffic dispersion idea is a thorough discussion of resequencing and cell delay, and in particular the normal distribution of cell delay and the variance of resequencing delay.

Dispersion for Fault Management in Realtime Networks

Banerjea [21, 22] uses Maxemchuk's redundant dispersion scheme to increase fault tolerance in real-time networks. By simulations, he investigates fault tolerance for different amounts of redundancy and different demands on the paths to be disjoint.

Redundant traffic dispersion is, in this context, mainly used to handle network failures, but also to deal with transmission errors. Loss due to congestion and buffer overflow in the network nodes is not considered. The dispersion scheme is assumed to work on top of a connection-oriented real-time communication service, which controls congestion through a resource reservation strategy. When a source wants to send, paths are chosen in the network and channels are established on each path. A modified Bellman-Ford routing algorithm is used to find the paths. The paths should preferably be completely disjoint, but if that requirement cannot be met, partially disjoint paths are used. The results consider dispersion where one or two channels may share a path. When two channels are allowed to share a path, the redundancy must be increased in order to handle a network failure, which may result in the simultaneous loss of information on two channels instead of one.

Banerjea distinguishes between "hot" and "cold" standby. All the previously discussed redundant dispersion schemes use "hot" standby, which means that the redundant paths are used to send forward error correction information. With "cold" standby, the redundant paths are used for other best-effort traffic, but are immediately switched to carry information in the case of failure on some of the ordinary paths. The latter strategy requires less capacity than sending redundant information along with the information, but results in delay due to fault detection and change of routes.



■ Figure 9. *Dispersion of variable-sized packets with fair load sharing among the channels.*

Simulation results show that redundant traffic dispersion increases the network error and fault tolerance without violating real-time guarantees on delay and loss. The capability of the network to set up the required number of routes is also investigated. It turns out that the possible number of connections that can be established increases as the number of used paths increases. This is because the capacity required for each channel decreases as the number of paths increases. For the same reason, channels for dispersed traffic are easier to set up than ordinary channels when the network load gets higher. As the number of paths used approaches the total number of parallel paths in the network, the ability to set up connections decreases. In the studied network, the total number of paths that could be set up was five, and the largest number of successfully established connections was obtained for three to four paths. When relaxing the requirements on the paths to be disjoint, the maximum number of successfully established connections may be obtained when using all the available parallel paths in the network. This is obtained at the expense of possibly losing more information in a network failure.

Banerjea also discusses the question of suitable resequencing buffers at the receiver. The buffers in his simulations are fairly small, since the real-time requirements guarantee a low average and variation of delay along each used channel. The delay jitter is thus kept at a low level. In essence, Banerjea contributes to the traffic dispersion area by investigating redundancy amounts and demands on paths to be disjoint with respect to network failures. He also includes a discussion on the implementation issues of dispersion in real-time networks.

Striping

Striping is yet another name for the technique of aggregating multiple resources in order to obtain higher performance [23]. Striping was originally developed for use within disk subsystems, but has also become helpful in network subsystems.

The main advantage pointed out by Traw and Smith in [23] is that striping makes it possible to meet capacity requirements even if they exceed the capacity of a single link. Another possibility is that if, for example, a single network interface or any other hardware component cannot support the required capacity, possibly multiple instances of the component could. The authors also mention the possibility of motivating technological improvements in commercially available network services. It is usually hard to motivate technological improvements, such as increasing network capacity, since the applications that require it will not be fully developed until it is available. By aggregating resources, higher capacity can be obtained and new capacity-usurping applications deployed.

The authors distinguish four levels of striping for the example of sending traffic by Transmission Control Protocol/Internet Protocol (TCP/IP) over ATM/SONET (synchronous optical network). "Byte striping" at the ATM layer means dispersing the bytes of ATM cells over multiple SONET channels; however, the resequencing could be problematic. "Cell striping" at the ATM adaptation layer disperses cells over virtual channels in a round-robin fashion. Sequence numbers could be used for the resequencing. Another possibility mentioned would be to introduce additional ATM cells, which synchronize the timing relationship between the circuits, thereby helping in resequencing. "Packet striping" at the TCP level means that IP packets are dispersed. IP does not guarantee orderly delivery of packets, so there is no need to try to preserve sequence. This type of striping, by the use of multiple paths, is claimed to reduce head-of-line blocking in routers, which occurs when small packets are forced to wait for larger packets to be transmitted.

"TCP striping" over the application layer involves dispersion of application-layer protocol data units (PDUs). This means that loss or errors which might occur during transmission are corrected by the nature of the PDUs before arriving at the destination. There is a discussion on simply dispersing the traffic cyclically without taking into account the different qualities of service on the paths which may result in poor performance.

The technique is further developed by Adiseshu *et al.*, who consider dispersion of variable-size packets [24]. If the packets are dispersed cyclically, the load may not be well balanced among the channels used. Instead, Adiseshu *et al.* use fair queuing algorithms in reverse to get fair load sharing when dispersing packets of variable size in a communication network (Fig. 9). By employing weighted fair load sharing, the load imposed on each channel can also be adapted to the available resources of that specific channel.

By using a fair queuing algorithm derived from the dispersion algorithm used by the sender, the receiver can reconstruct the original packet stream from the arriving channels. This works, provided there are no packet losses in the network during transmission. If packet losses may occur, the sender must periodically resynchronize with the receiver. An obvious method of detecting losses is to number the packets. However, if there is no space in the packets for adding sequence numbers, other strategies must be employed. Adiseshu *et al.* suggest providing synchronization by marker packets, which are sent periodically on each channel. A marker packet contains a number, which is increased every time the sender has finished a cycle of dispersion. The more frequently the marker packets are sent, the fewer packets are delivered out of order, at the expense of increased overhead in the network. The strategy was implemented in a system dispersing IP packets over a combination of ATM and Ethernet links, and its properties were verified.

Traffic Dispersion

The most recent contributions are by the authors themselves, and the strategy studied is nonredundant dispersion according to Fig. 4b [25–28]. The work in [26] gives an introduction to the method and investigates how dispersion affects the queuing behavior in a one-stage multiplexer. Calculations and simulations show that cyclic dispersion of ATM cells, generated by a two-state Markov chain, reduces the mean queue size as well as the overall queue size. It turns out that the mean queue size starts to level out when dispersing the traffic over five paths or more. Simulations were also performed on traffic generated by a chaotic map called the FPDI map, which is an on-off source with heavy-tailed distribution of the sojourn times spent in the on and off states, respectively. The aggregated traffic from a number of such sources has been shown to exhibit long-range dependence. Dispersion is shown to improve the queuing performance for this type of traffic as well. Lastly, simulation results indicate that cyclic dispersion is superior to dispersion of longer sequences of cells, regarding the queuing behavior.

The work in [25] is a study on the equivalent capacity of a connection, and is intended to establish under which circumstances traffic dispersion is useful. The equivalent capacity is calculated for traffic from two-state Markov chain sources, for

buffering as well as nonbuffering links. It turns out that dispersion decreases the equivalent capacity in general, and in particular for sources with high peak-to-mean ratio and high peak-to-link ratio. These are actually the sources that are most difficult to handle in terms of statistical multiplexing. There might, however, be some penalty for using multiple paths for a connection. The values of equivalent capacity are therefore related to three different cost functions. The first cost function is a fixed charge per capacity unit, while the second is composed of a fixed charge per capacity unit and a fixed charge per path used for a transfer. The third cost function is similar to the second but has a progressively increasing charge per number of paths. The results show that on condition that the penalty for using several connections does not dominate the total cost for a transmission, dispersion over a moderate number of paths is practically always profitable. The benefits are most important for sources with a peak-to-mean ratio larger than 10 and a peak-to-link ratio larger than 1/10th. As a general estimate, the number of paths need not exceed five.

Lastly, a practical application of traffic dispersion is presented in a call admission control scheme for ATM networks [27, 28]. The scheme demands the source peak rate as the only traffic descriptor declared at call request. It then uses dispersion to reduce the impact of a single source on each link and to provide for statistical multiplexing gains. Given those conditions, the dispersed traffic from a source on one of the links may be approximated by a Poisson process, and the required capacity can hence easily be calculated. In order to obtain desired performance, measurements are used to improve the capacity estimations. Simulations are performed with traffic generated by two-state Markov chains as well as long-range dependent traffic generated by Pareto on-off sources. They show that the scheme gives low cell loss while keeping high link utilization. Employing dispersion thus provides the basis for an efficient and uncomplicated call admission control scheme.

This concludes the work done so far within the framework of traffic dispersion in communication networks. Work has also been done on dispersion inside switch architectures, and the next section will cover some of those results.

Discussion

There have been several contributions to the area of traffic dispersion, and this article is an attempt to report on the work done so far. Although most of the work is covered in the earlier sections, there are still some assorted topics that are worthwhile to mention.

Traffic dispersion is discussed in the context of switching architectures as well as in communication networks [29–32]. The main idea is to represent a data link by a number of channels, forming a channel group (Fig. 3). A packet that is supposed to follow a certain link can hence be switched to practically any channel within the specified channel group. At connection setup, routing is performed at the channel group level, while routing at the channel level is performed dynamically in real time. It is hereby possible to decrease the blocking probability in the switch and increase the switch performance. The method is particularly useful in networks that use high-speed lightwave technology for transmission, but slower electronic switching devices. Multichannel switching hence makes it possible to increase the switching rate in order to keep up with overall network speed. A brief survey on some of the work done in the field is presented in [32].

Inverse multiplexing provides a means to aggregate several channels to obtain higher capacity [34]. The technique was first used in modem technology for analog voice channels, but now mainly applies to systems with digital switches and optical fibers, where the inverse multiplexers combine digital channels to form a single pipe. The article discusses the Bonding standard, which defines the procedures for using an inverse multiplexer to combine multiple switched 56 and 64 kb/s channels. This technique is claimed to be useful for video conferences, image retrieval, and private line backup, to provide overflow capacity, and for interconnecting local area networks. The technique's future usefulness in matching application needs with available services is also mentioned.

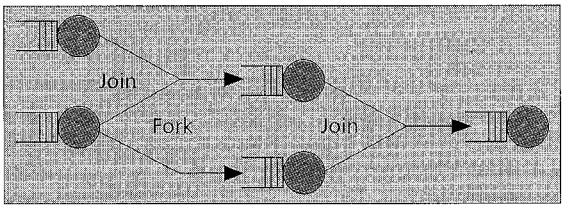
An exhaustive analysis of an acyclic fork-join queuing network (Fig. 10) is given by Baccelli *et al.* [34]. The fork-join mechanism essentially corresponds to nonredundant traffic dispersion, and the queuing behavior in the network nodes is investigated. Fork-join applies to parallel processing applications and flexible manufacturing systems.

Chowdhury analyses the delay introduced by the resequencing function, which is needed when packets are distributed over multiple parallel links [35]. The subject of resequencing is also discussed by Baccelli and Makowski [36], and Jean-Marie and Gün [37]. Gardner *et al.* present a dynamic multipath routing algorithm for use in packet-switched networks, and briefly mention the possibility of increasing throughput by splitting flows among comparable paths [38]. A final example of traffic dispersion is the two-copy routing scheme suggested by Chiou and Li [39]. They claim that delay can be reduced under certain circumstances by sending two copies of each message along two disjoint paths from the source to the destination. This is actually an instance of Maxemchuk's redundant traffic dispersion where the number of routes is restricted to two and a (2, 1) repetition code is used. However, the scheme doubles the network load, and its efficiency might be questioned.

The different results obtained all indicate vast opportunities in using traffic dispersion, in terms of increased network efficiency and improved performance. What characterizes the work done so far is that it is mostly limited to on-off sources in the shape of two-state Markov chains, and rather small and simple network structures. It would be interesting to investigate the strategies for various traffic types and larger networks, and to extensively investigate the full consequences of using paths that are only partially disjoint. In order to implement the schemes, it would be necessary to decide and verify an optimum number of paths and an accurate dispersion strategy for various traffic characteristics. That is, under what conditions should the traffic be dispersed; and, in the case of dispersion, should it be spread cyclically, dynamically, or according to string mode? If the latter strategy is chosen, what is the optimum string length for each traffic and/or network type?

It should be noted that dynamic dispersion does not agree with the requirements of forward error correcting codes. Using redundancy over multiple paths requires a roughly equal amount of a message to be sent on each path. If keeping to those requirements, the main purpose of dynamic dispersion is wasted. Besides, dynamic dispersion does not consider the correlation in the traffic stream, which is often closely related to congestion phenomena in a network and might be significantly reduced with the appropriate dispersion strategy. As mentioned earlier, dynamic dispersion is reactive rather than preventive. If each user were to employ dynamic dispersion, the method might, under heavy load, cause instability in the network. It would be interesting to investigate how fast the dynamic algorithm adjusts to changes in the current network load, and what actually happens when all users independently employ the method.

When it comes to dispersing traffic on a burst-by-burst basis, the question arises of how to define a burst. If the allocated peak rate is not used throughout the duration of a



■ Figure 10. *Example of a fork-join queuing network.*

burst, the effectiveness of the scheme decreases. Moreover, due to the time it takes to allocate capacity, the scheme requires bursts to be rather long, since short and frequent bursts would make the time and effort spent on capacity allocation grow unrestrainedly. Extremely long bursts may, however, cause congestion, and it might hence be necessary to divide them into smaller bursts. The scheme then falls back on the problem of too much overhead caused by the allocation procedures.

Throughout the work done so far, there is no thorough investigation of a routing algorithm capable of finding the right number of paths with the desired properties, nor is there a discussion of what these properties are. Furthermore, implementing the schemes in, for example, an ATM network requires an investigation on how each scheme affects the different protocol functions. This topic also includes an exhaustive study on the resequencing procedure.

Summary

This article reports on the work done on traffic dispersion in communication networks. We have described a number of different strategies, all based on essentially the same concept, and presented some of the results. Traffic dispersion has been shown to alleviate the effects of traffic bursts in the network, to reduce the network queuing delay, and to reduce the probability of packet loss. Redundant traffic dispersion also provides a capability to handle network failures transparently to the application. Despite all the work done, some topics, such as partially disjoint paths, resequencing procedures, and routing algorithms, still remain to be investigated. We hope that this summary may further interest in traffic dispersion and promote its inclusion in broadband networks.

References

[1] W. E. Leland *et al.*, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE/ACM Trans. Networking*, vol. 2, no. 1, Feb. 1994, pp. 1–15.
[2] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, June 1995, pp. 226–44.
[3] W. Willinger *et al.*, "Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level," *ACM Comp. Commun. Rev.*, vol. 25, no. 4, Oct. 1995, p. 100–113.
[4] M. O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance," *J. ACM*, vol. 36, no. 2, Apr. 1989, pp. 335–48.
[5] D. Sidhu, S. Abdullah, and R. Nair, "Congestion Control in High Speed Networks via Alternate Path Routing," *J. High Speed Networks*, vol. 1, no. 2, 1992, pp. 129–44.
[6] S. Bahk and M. El Zarki, "A Dynamic Multi-Path Routing Algorithm for ATM Networks," *J. High Speed Networks*, vol. 1, no. 3, 1992, pp. 215–36.
[7] N. F. Maxemchuk, "Dispersity Routing," *Proc. ICC '75*, San Francisco, CA, June 1975, pp. 41-10–41-13.
[8] N. F. Maxemchuk, "Dispersity Routing in High-Speed Networks," *Comp. Networks and ISDN Sys.*, vol. 25, no. 6, 1993, pp. 645–61.
[9] N. F. Maxemchuk, "Dispersity Routing on ATM Networks," *Proc. IEEE INFOCOM '93*, vol. 1, San Francisco, CA, Mar. 1993, pp. 347–57.
[10] J. H. Déjean, L. Dittmann, and C. N. Lorenzen, "Performance Improvement of an ATM Network by Introducing String Mode," *Proc. IEEE INFOCOM '91*, vol. 3, Bal Harbour, FL, Apr. 1991, pp. 1394–1402.
[11] J.H. Déjean, L. Dittmann, and C.N. Lorenzen, "String Mode — A New Concept for Performance Improvement of ATM Networks," *IEEE JSAC*, vol. 9, no. 9, Dec. 1991, pp. 1452–60.
[12] A. Jagd *et al.*, "Implementation of String Mode: A Multi-Link Broadband Network Protocol," *Proc. 2nd Int'l. Conf. Broadband Services, Sys. and Networks*, Brighton, U.K., Nov. 1993, IEE, pp. 65–69.
[13] T. T. Lee and S. C. Liew, "Parallel Communications for ATM Network Control and Management," *Proc. IEEE GLOBECOM '93*, vol. 1, Houston, TX, Dec. 1993, pp. 442–46.
[14] Q-L. Ding and S. C. Liew, "A Performance Analysis of a Parallel Communications Scheme for ATM Networks," *Proc. IEEE GLOBECOM '95*, vol. 2, Singapore, Nov. 1995, pp. 898–902.
[15] R. Krishnan and J. A. Silvester, "Choice of Allocation Granularity in Multipath Source Routing Schemes," *Proc. IEEE INFOCOM '93*, vol. 1, San Francisco, CA, Mar. 1993, pp. 322–29.
[16] R. Krishnan and J. A. Silvester, "Resource Allocation in Broadband Networks — Cell, Burst or Connection Level?" *Proc. ICC '94*, vol. 1, New Orleans, LA, May 1994, pp. 86–90.
[17] R. Krishnan and J. A. Silvester, "The Effect of Multipath Routing on the Loss Performance of Multiplexed ON-OFF Sources," *Proc. ITC-14*, Antibes Juan-les-Pins, France, June 1994, Amsterdam: Elsevier, pp. 941–50.
[18] H. Suzuki and F. A. Tobagi, "Fast Bandwidth Reservation Scheme with Multi-Link & Multi-Path Routing in ATM Networks," *Proc. IEEE INFOCOM '92*, vol. 3, Florence, Italy, May 1992, pp. 2233–40.
[19] T-H. Cheng, "Bandwidth Allocation in B-ISDN," *Comp. Networks and ISDN Sys.*, vol. 26, no. 9, 1994, pp. 1129–42.
[20] N. T. Plotkin and P. P. Varaiya, "Performance Analysis of Parallel ATM Connections for Gigabit Speed Applications," *Proc. IEEE INFOCOM '93*, vol. 3, San Francisco, CA, Mar. 1993, pp. 1186–93.
[21] A. Banerjea, "Fault Management for Realtime Networks," Ph.D. dissertation, University of California at Berkley, 1994, Ch. 6.
[22] A. Banerjea, "Simulation Study of the Capacity Effects of Dispersity Routing for Fault Tolerant Realtime Channels," *ACM Comp. Commun. Rev.*, vol. 26, no. 4, Oct. 1996, pp. 141–205.
[23] C. B. S. Traw and J. M. Smith, "Striping Within the Network Subsystem," *IEEE Network*, vol. 9, no. 4, July/Aug. 1995, pp. 22–29.
[24] H. Adiseshu, G. Parulkar, and G. Varghese, "A Reliable and Scalable Striping Protocol," *ACM Comp. Commun. Revi.*, vol. 26, No. 4, Oct. 1996, pp. 131–41.
[25] E. Gustafsson and G. Karlsson, "When Is Traffic Dispersion Useful? A Study On Equivalent Capacity," *Performance Modelling and Evaluation of ATM Networks, Volume 2*, D. Kouvatsos, ed., Chapman & Hall, 1996.
[26] E. Gustafsson and G. Karlsson, "Traffic Dispersion in ATM Networks," *Proc. RadioVetenskap och Kommunikation 96, RVK 96*, Luleå and Kiruna, Sweden, June 1996, pp. 593–97.
[27] E. Gustafsson and G. Karlsson, "Call Admission Control with Traffic Dispersion," *Proc. Nordic Teletraffic Seminar, NTS-13*, Trondheim, Norway, Aug. 1996, pp. 370–83.
[28] E. Gustafsson and R. Rönngren, "Fluid Traffic Modelling in Simulation of a Call Admission Control Scheme for ATM Networks," *Proc. 5th Int'l. Symp. Modeling, Analysis and Simulation of Comp. and Telecom. Sys., MASCOTS '97*, Haifa, Israel, Jan. 1997, pp. 110–15.
[29] R. L. Cruz, "The Statistical Data Fork: A Class of Broad-Band Multichannel Switches," *IEEE Trans. Commun.*, vol. 40, no. 10, Oct. 1992, pp. 1625–34.
[30] H. S. Kim, "Design and Performance of Multinet Switch: A Multistage ATM Switch Architecture with Partially Shared Buffers," *IEEE/ACM Trans. Networking*, vol. 2, no. 6, Dec. 1994, pp. 571–80.
[31] A. Pattavina, "Multichannel Bandwidth Allocation in a Broadband Packet Switch," *IEEE JSAC*, vol. 6, no. 9, Dec. 1988, pp. 1489–99.
[32] P. S. Min, H. Saidi and M. V. Hegde, "A Nonblocking Architecture for Broadband Multichannel Switching," *IEEE/ACM Trans. Networking*, vol. 3, no. 2, Apr. 1995, pp. 181–98.
[33] P. H. Fredette, "The Past, Present, and Future of Inverse Multiplexing," *IEEE Commun. Mag.*, vol. 32, no. 4, Apr. 1994, pp. 42–46.
[34] F. Baccelli, W. A. Massey and D. Towsley, "Acyclic Fork-Join Queuing Networks," *J. ACM*, vol. 36, no. 3, July 1989, pp. 615–42.
[35] S. Chowdhury, "An Analysis of Virtual Circuits with Parallel Links," *IEEE Trans. Commun.*, vol. 39, no. 8, Aug. 1991, pp. 1184–88.
[36] F. Baccelli and A.M. Makowski, "Queueing Models for Systems with Synchronization Constraints," *Proc. IEEE*, vol. 77, no. 1, Jan. 1989, pp. 138–61.
[37] A. Jean-Marie and L. Gün, "Parallel Queues with Resequencing," *J. ACM*, vol. 40, no. 5, Nov. 1993, pp. 1188–1208.
[38] M. L. Gardner, I. S. Loobeek, and S. N. Cohn, "Type-of-Service Routing with Loadsharing," *Proc. IEEE GLOBECOM '87*, vol. 2, 1987, pp. 1144–50.
[39] S-N. Chiou and V.O.K. Li, "An Optimal Two-copy Routing Scheme in a Communication Network," *Proc. IEEE INFOCOM '88*, New Orleans, LA, Apr. 1988, pp. 288–97.

Biographies

EVA GUSTAFSSON received her M.Sc. degree in electrical engineering from the Royal Institute of Technology, KTH, in 1992. She is currently a Ph.D. student at the TeleTraffic Systems Laboratory at KTH Department of Teleinformatics, working on traffic dispersion in high-speed networks.

GUNNAR KARLSSON received his master's degree from Chalmers University of Technology in 1983 and his Ph.D. from Colombia University in 1989. He has worked at IBM Zurich Research Laboratory from 1989 to 1992, and at the Swedish Institute of Computer Science since 1992. He is a member of the faculty at the Department of Teleinformatics at KTH, and is currently visiting professor at EPFL, Switzerland.