In Network Management and Control, Volume II (I. Frisch, M. Malek, and S. Panwar, eds.), pp. 423-438. Plenum Publishing Co., NY, NY, 1994.

AN ADAPTIVE INFORMATION DISPERSAL ALGORITHM FOR TIME-CRITICAL RELIABLE COMMUNICATION

Azer Bestavros

Computer Science Department
Boston University
Boston, MA 02215

ABSTRACT

AIDA – a novel elaboration on Michael O. Rabin's IDA [21] – is a communication protocol that uses redundancy to achieve *both* timeliness and reliability. In AIDA redundancy is used to tackle several crucial problems. In particular, redundancy is used to *tolerate* failures, to *increase* the likelihood of meeting tight time-constraints, and to *ration* (based on task priorities) the limited bandwidth in the system. AIDA is a *probabilistic* protocol in the sense that it does not guarantee the fulfillment of hard time constraints. Instead, it guarantees a lower bound on the *probability* of fulfilling such constraints. Such a bound could be lowered so as to satisfy any level of confidence in the timeliness and reliability of the system. In this paper we present AIDA and contrast it with traditional communication scheduling techniques used in conjunction with time-critical applications in general, and distributed multimedia systems in particular. The suitability of AIDA-based bandwidth allocation for a variety of time-critical applications is established and plans for future research experiments are mentioned.

INTRODUCTION

The successful execution of time-critical tasks running in a distributed environment often requires that a set of communication requests be successfully completed before some set deadlines. In many instances, such requests are periodic and require transfer rates *higher* than what a single client-server connection can provide. The large amount of data to be transferred, coupled with the possibility that it may not be available in full when the client-server connection is established (*e.g.* live transmission), limits the usefulness of buffering or data caching. Due to the extended period over which such client-server connections are expected to last, tolerance for failures and network traffic fluctuations becomes of utmost importance. We argue that for such time-critical systems, striping and redundancy must be employed to secure the required bandwidth and fault-tolerance. In this paper we propose a novel technique that allows the secure and fault-tolerant retrieval of information *striped* across a number of sinks in a distributed environment. While the sustained bandwidth that any single sink can confidently provide is low, the aggregate bandwidth achieved through striping and redundancy could be *demonstrably* much higher.

Distributed multimedia represents an important class of time-critical applications with requirements similar to those mentioned above. A distributed multimedia presentation uses audio, video, text, and graphics from local and remote sources, some of which may be live. Multimedia data such as audio and video require special considerations when supported by a computer system. They have well-defined (*natural*) presentation timing constraints that must be satisfied by the system during presentation (or *playout*). Examples of natural timing constraints include the maximum tolerable delay in playing out a voice (or video) packet, beyond which dropping the packet might be necessary to avoid disturbing the continuity (smoothness) of the presentation. Other data types (*e.g.* text and graphics) do not have natural timing constraints, but can be subject to *synthetic* constraints, such as those arising from synchronization requirements (a piece of text or graphics may be required to appear when a particular sequence of video frames is displayed).

In order for any communication protocol to successfully schedule service requests, an accurate knowledge of the delays introduced by the communication network is often required. For communication scheduling purposes (hereinafter referred to as *bandwidth allocation*), such knowledge can be acquired either statically or dynamically. Using static techniques, worst-case delays are determined and accounted for *a priori* when scheduling the communication transactions. Alternatively, using dynamic techniques the average (or maximum) delays experienced through a communication network can be measured and used as an *estimate* for use with future communication transactions. Static communication scheduling (using a priori knowledge about the communication network delays) can be safely and efficiently used in systems with predetermined communication patterns (*e.g.* broadcasting) and systems with predetermined computation requirements (*e.g.* periodic tasks). For systems with unpredictable communication patterns or systems with sporadic computation requirements, dynamic communication scheduling becomes necessary.

Several techniques have been suggested in the literature for dynamic bandwidth allocation. Most of these techniques rely on the use of feedback from the communication network to establish a performance model that can be used in conjunction with a scheduling algorithm to allocate/reserve the communication bandwidth needed for the successful execution of time-critical tasks [15, 17].

While bandwidth allocation is an important consideration in the design of distributed time-critical systems (such as multimedia), it is not the only one. Issues of reliability, availability, and fault-tolerance are equally important. The most common technique used to tackle these issues is *replication*. For example, in distributed database applications [2, 7], several copies of a particular data object may be kept in a number of different sites so that the failure (whether intermittent or permanent [22]) of any proper subset of these sites would not render the data object unavailable.

For distributed applications operating under strict time constraints, replication alone may not be adequate. In particular, failures should not be allowed to increase the retrieval delay for data objects (at least not considerably). In this respect, techniques that rely on watchdog timers and/or retransmit protocols may not be adequate. Instead, techniques that use redundant communication (*e.g.* requesting the same data object from a set of failure-independent sites/paths) may be necessary.

Current fault-tolerant communication protocol do not attempt to use available redundancy (necessary for fault-tolerance) to improve other aspects of the system – timeliness, for example. In other words, the added redundancy serves a unique purpose: to boost the reliability of the system. It is our thesis that this approach is unnecessarily restrictive and, to some extent, inefficient. We argue that redundancy should be used to tackle many problems in an integrated manner.

In this paper we propose AIDA (Adaptive Information Dispersal Algorithm), a novel technique for dynamic bandwidth allocation, which makes use of minimal, controlled redundancy to guarantee timeliness and fault-tolerance up to any degree of confidence. Our technique is an elaboration on the Information Dispersal Algorithm of Michael O. Rabin [21], which we have previously shown to be a sound mechanism that considerably improves the performance of I/O systems and parallel/distributed storage devices [3, 6]. The use of IDA for efficient routing in parallel architectures has also been investigated [20].

REAL-TIME BANDWIDTH ALLOCATION: RELATED WORK

A real-time communication system must manage the communication of time-dependent data to provide timely and predictable data delivery; it provides performance guarantees for the delivery of data from source to destination. These guarantees can be absolute through deterministic scheduling and resource allocation, or probabilistic through the use of statistical approaches. In this section, we review a few of the representative techniques currently being used/investigated for real-time bandwidth allocation in multlimedia applications.

One way to schedule data transmission is to maintain statistics characterizing each of the communication resources (channels) in the system. Whenever the channel characteristics of the network change, the server responsible for delivering the time-critical data can adjust accordingly to maintain predictable service. This can be achieved by decreasing the demand on the network. For example, when a network becomes congested and the percentage of late data elements (missed deadlines) increases, dropping the demand on the network helps clear the congestion [12]. This effectively allows data elements scheduled for transmission to traverse the communication network and reach their destination *on time* rather than be lost due to lateness.

Another mechanism to deal with the adverse effect of network congestion is to distinguish between the various communication requirements. This was proposed in the Asynchronous Timesharing System (ATS) [15], in which data traffic is divided into four classes. A control class C has the highest priority; it delineates a class of communication where data loss or unpredictable communication delays cannot be tolerated. Class I is next on the priority scale; it delineates a class of communication where data loss cannot be tolerated, but a user-specified maximum end-to-end communication delay is allowed. Class II has a set maximum percent of lost packets and a maximum count of consecutive packets lost. Finally, class III has zero loss and no maximum end-to-end delay for communication that is not subject to time constraints.

A similar treatment of the different communication requirements imposed on a distributed system is under investigation at the University of California at Berkeley, where an experimental RAID-II network file server is being implemented. It treats requests with a low end-to-end delay requirement differently from requests with a high bandwidth requirement is being implemented [16].

The network protocol presented in [9, 8] handles performance requirements in a different manner. When a connection is requested, the user provides the network manager with maximum end-to-end delay, maximum packet size, maximum packet loss rate, minimum packet inter-arrival time, and maximum jitter, where jitter is defined as the difference in the delays experienced between two packets on the same connection. Three types of channels can be requested: *deterministic, statistical,* and *best-effort.* For deterministic channels, the communication delay is guaranteed to fall below a given time bound. For statistical channels, the probability that the delay is less than a given time

D is kept greater than or equal to a requested factor q. This can be thought of as establishing a *confidence interval* about the expected delay rather than a *deterministic bound* on that delay. Best-effort channels provide no guarantees for the percentage of messages reaching their destination on-time; they merely attempt to make the best use of the available bandwidth.[1]

Statistical approaches to overcoming delay and bandwidth limitations are attractive because they provide application programs with a flexible framework, in which a continuum of communication priorities can be easily expressed as confidence intervals. In particular, we argue that the distinction between deterministic, statistical, and best-effort channels in the protocol proposed in [9, 8] is artificial. Deterministic and best-effort channels can be thought of as *special* statistical channels, for which the confidence interval (determined by q) describing the communication delay is taken to its limits.[2] Therefore, in this paper (without loss of generality), we consider only statistical channels.

Current techniques for statistical bandwidth allocation [18] rely on choosing an end-to-end time delay T per packet that is larger than the delay expected to be experienced by a percentage P of the retrieved packets. This time T is used as an estimate for the time it will take to retrieve packets from a given source. Figure 1 illustrates a typical relationship between T and P. While such a delay function can accurately represent delay characteristics over a given period of time, network loading does change with time, possibly making the delay distribution (and thus the delay function) outdated.

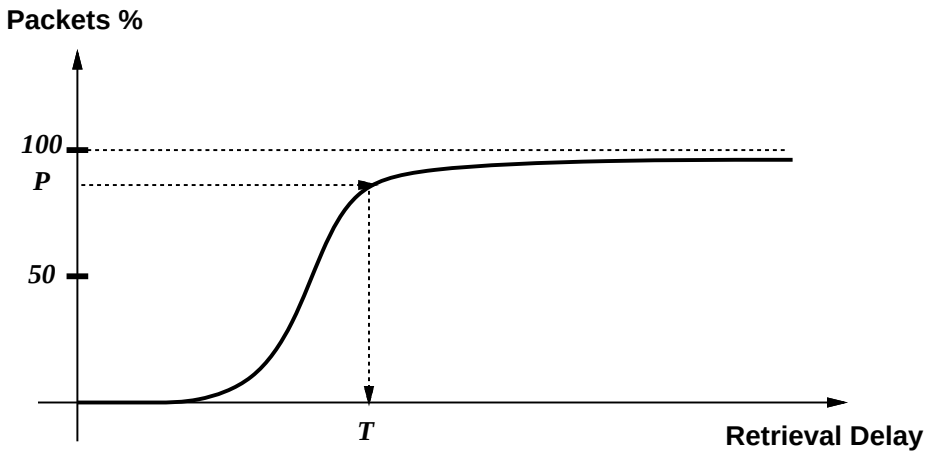

Figure 1: A typical end-to-end delay characteristic function.

One way to accommodate this dynamic behavior is to monitor the delays experienced by retrieved packets and adjust the delay function accordingly. In [10], a mechanism called *Limited A Priori* (LAP) scheduling is proposed, in which adjustments to the delay function are made either periodically or whenever sudden changes in network traffic are detected. Using second and higher order moments, linear and quadratic extrapolation of the network delay characteristics can be more accurately predicted. This research work, however, is yet to be pursued.[3]

All of the bandwidth allocation mechanisms described so far (with the exception of RAID-II) assume a single source of data for a given transaction. In a truly distributed environment, this is not likely to be the case; the storage of a single object may span

[1] Deterministic channels are necessary for computations with *hard* time constraints, whereas statistical channels are appropriate for computations with *soft* time constraints. Best-effort channels are adequate for computations with no time constraints.

[2] For deterministic channels, $q = 1$. For best-effort channels, $q = 0$.

[3] For more information, please contact the author.

a number of nodes to meet fault-tolerance requirements. For example, striping data for a video presentation over N nodes would increase the *availability* of the system by allowing a graceful degradation of the quality of the presentation by $1/N\%$, should any of the N nodes fail. Spreading the storage of a single object over a number of nodes may also be desired for caching purposes, especially for "hot" data objects. Finally, the distribution of a complex data object may be distributed as a result of placement and security constraints. Although it is possible to extend the aforementioned bandwidth allocation mechanisms to deal with data distributed over a number of nodes, the performance of these protocols will deteriorate significantly as a result of such distribution. The mechanism we are proposing in this paper is inherently distributed and, in that respect, is far more superior.

INFORMATION DISPERSAL AND RETRIEVAL

In this section we overview the original Information Dispersal Algorithm (IDA). We refer the reader to the original paper on IDA [21] for a more thorough presentation.

Let F represent the original data object (hereinafter referred to as the *file*) in question. Furthermore, let's assume that the storage of file F is to be distributed over N sites. Using the IDA algorithm, the file F will be processed to obtain N distinct pieces in such a way that recombining *any* m of these pieces, $m \leq N$, is sufficient to retrieve F. The process of processing F and distributing it over N sites is called the *dispersal* of F, whereas the process of retrieving F by collecting m of its pieces is called the *reconstruction* of F. Figure 2 illustrates the dispersal and reconstruction of an object using IDA. The dispersal and reconstruction operations are simple linear transformations using *irreducible polynomial arithmetic*.[4] Both the dispersal and reconstruction of information using IDA can be performed in real-time. This was demonstrated in [4], where we presented an architecture and a CMOS implementation of a VLSI chip[5] that implements IDA.

Let $|F|$ be the size of the file F. The IDA approach *inflates* F by a factor of $\frac{N}{m}$. In particular, the size of each one of the dispersed pieces of F would be $\frac{|F|}{m}$. This added redundancy makes the system capable of tolerating up to $N - m$ faults without any effect on timeliness. More importantly (as we will demonstrate shortly), this added redundancy will boost the performance of the information retrieval process *significantly*.

Several *redundancy-injecting* protocols have been suggested in the literature to deal with fault-tolerance issues. In most of these protocols, redundancy is injected in the form of (potentially distributed) parity blocks, which are only used for error detection and/or correction purposes [11]. The IDA approach is radically different in that redundancy is added *uniformly*; there is simply *no* distinction between data and parity. It is this feature that makes it possible for IDA to be used not only to boost communication fault-tolerance, but also to improve bandwidth allocation and utilization.

Unlike other redundancy-injecting protocols [23, 16], the amount of redundancy that IDA uses with a given object, or in a given session, does *not* have to be constant.

[4] For a concrete implementation and for examples, the reader is referred to our previous work on SETH [4] and IDA-based RAID I/O systems [5].

[5] The chip (called SETH) has been fabricated by MOSIS and tested in the VLSI lab of Harvard University, Cambridge, MA. The performance of the chip was measured to be about 1 megabyte per second. By using proper pipelining and more elaborate designs, this figure can be boosted significantly.

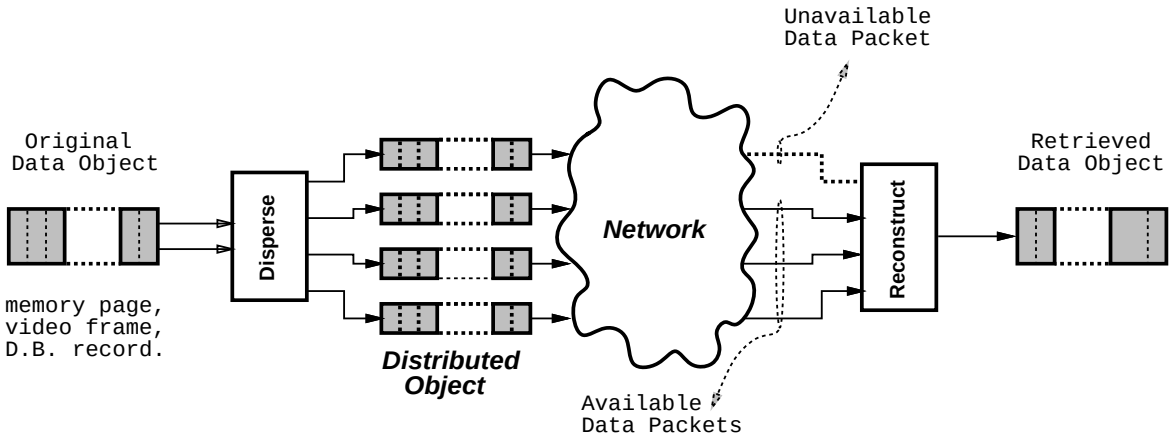

Figure 2: Dispersal and reconstruction of information using IDA.

In particular, as we will describe later, our AIDA-based bandwidth allocation strategy *controls* the amount of redundancy to be used with a particular object in a particular session so as to reflect the *priority* and/or the *urgency* of the transaction at hand. By increasing the redundancy allocated for a given communication session, the expected retrieval delay can be reduced, thus increasing the chances of meeting the possibly tight time constraint imposed on the transaction.

PERFORMANCE CHARACTERISTICS

Let X be a data object dispersed using IDA into N pieces, each residing in a different site. Let m be the minimum number of pieces needed to reconstruct X. Obviously, in order to retrieve X, at least m of the N sites must be consulted. It is possible, however, to consult more than m of these sites. Let n (where $m \leq n \leq N$) denote the total number of sites *consulted* for the retrieval of X. In this section, we derive an expression for the expected communication delay for accessing such an object. Later, we will use this result to establish the merits of our proposed AIDA-based bandwidth allocation protocol.

$$\text{Prob}(t \geq z) = \text{Prob}(\text{Response time of at least } (n-m+1) \text{ of the sites} \geq z) \quad (1)$$

$$= \sum_{r=n-m+1}^{n} \binom{n}{r} P^r (1-P)^{n-r} \quad (2)$$

where P is the probability that the response time of a single site will be z or more. P can be estimated using delay characteristic functions such as the one illustrated in figure 1.

Approximation Using a Uniform Distribution Delay Model

As a first and safe approximation, we will assume that the delays experienced through the communication network are *uniformly distributed* random variables with lower and upper bounds ($D_{\min}$ and $D_{\max}$) as illustrated in figure 3.

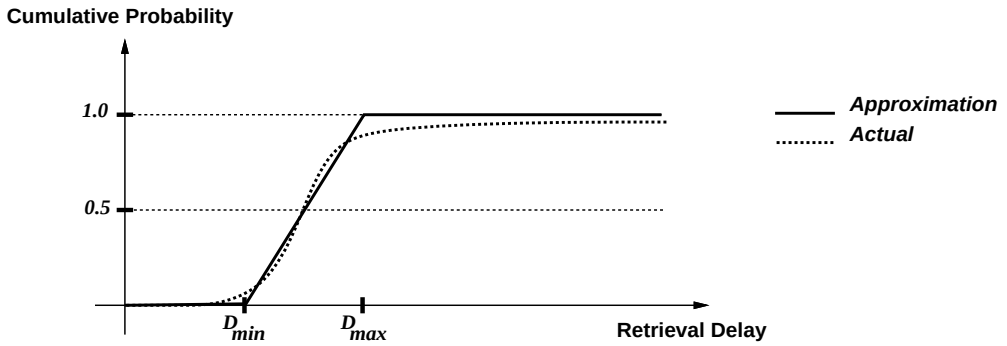

Figure 3: Delay characteristics under a uniform delay model.

We denote by P_u the value of P (in equation 2) under the uniform distribution assumption.

$$P_u = \begin{cases} 1 & \text{if } 0 < z < D_{\min} \\ 1 - \frac{z - D_{\min}}{D_{\max} - D_{\min}} & \text{if } D_{\min} \leq z \leq D_{\max} \\ 0 & \text{if } D_{\max} < z < \infty \end{cases}$$

The random variable t (in equation 2) is simply the $(n - m + 1)^{\text{th}}$ largest of these n uniformly distributed independent random variables. It can be shown that t follows the *beta probability law* and that the mean and standard deviation for t are given by:[6]

$$\tau_u = D_{\min} + \frac{m}{n+1}(D_{\max} - D_{\min}) \tag{3}$$

$$\sigma_u = \sqrt{\frac{m(n-m+1)}{(n+1)^2(n+2)}}\,(D_{\max} - D_{\min}) \tag{4}$$

Approximation Using an Exponential Distribution Delay Model

We denote by P_e the value of P (in equation 2) under the exponential distribution delay model.

$$P_e = \begin{cases} 1 & \text{if } 0 < z < D_{\min} \\ e^{-\lambda(z - D_{\min})} & \text{if } D_{\min} < z < \infty \end{cases}$$

Let τ_e denote the average delay experienced using an IDA-based strategy under an exponential delay model with parameter λ (see figure 4). To compute τ_e, we proceed as follows:

$$\begin{aligned} \tau_e &= D_{\min} + E(t - D_{\min}) \\ &= D_{\min} + \int_{D_{\min}}^{\infty} \text{Prob}(t \geq z) \cdot dz \\ &= D_{\min} + \sum_{r=n-m+1}^{n} \binom{n}{r} \int_{D_{\min}}^{\infty} P^r (1-P)^{n-r} \cdot dz \end{aligned}$$

[6] Derivation is omitted for space limitations. For a reference, refer to [14].

$$
\begin{aligned}
\tau_e &= D_{\min} + \sum_{r=n-m+1}^{n} \left(\begin{array}{c} n \\ r \end{array} \right) \int_0^1 P^r (1-P)^{n-r} \cdot \frac{1}{\lambda P} \cdot dP \\
&= D_{\min} + \frac{1}{\lambda} \sum_{r=n-m+1}^{n} \left(\begin{array}{c} n \\ r \end{array} \right) \frac{\Gamma(r)\Gamma(n-r+1)}{\Gamma(n+1)} \\
&= D_{\min} + \frac{1}{\lambda} \sum_{r=n-m+1}^{n} \frac{\Gamma(n+1)}{\Gamma(r+1)\Gamma(n-r+1)} \frac{\Gamma(r)\Gamma(n-r+1)}{\Gamma(n+1)} \\
\tau_e &= D_{\min} + \frac{1}{\lambda} \sum_{r=n-m+1}^{n} \frac{1}{r} \qquad (5)
\end{aligned}
$$

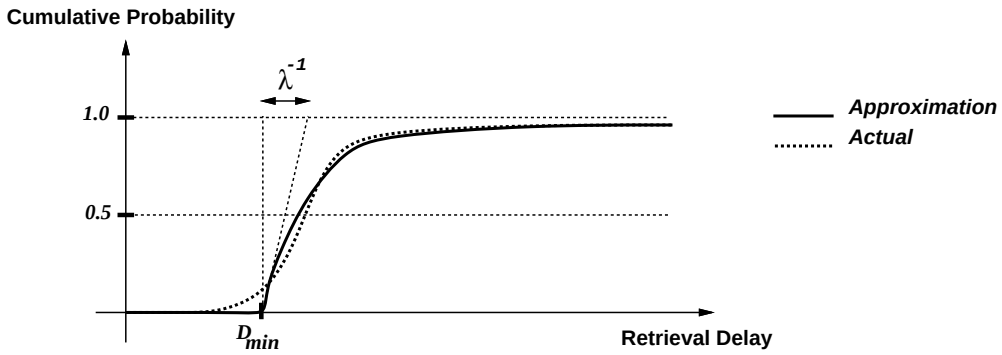

Figure 4: Delay characteristics under an exponential delay model.

The remainder of this paper assumes an exponential delay model.

Effect of Distribution and Redundancy on Delay Characteristics

There are a number of interesting observations to be made from the delay analysis of the previous section. By varying the values of n and m, the negative effect of data distribution and the positive effect of data redundancy on the delay characteristics can be demonstrated. The following cases can be readily examined:

a. $\underline{n = m = 1}$: This is the case when the object X is not distributed. The expected delay reduces to $\frac{1}{2}(D_{\min} + D_{\max})$ under the uniform delay model and reduces to $D_{\min} + \frac{1}{\lambda}$ under the exponential delay model. This corresponds to the average delay for one transmission.

b. $\underline{n > m = 1}$: This is the case when the object X is replicated over n sites. For $n \gg 1$, the expected delay approaches $D_{\min}$, which is the minimum delay for one transmission under both the uniform and exponential delay models.

c. $\underline{n = m > 1}$: This is the case when the object X is distributed without any added redundancy. For $n \gg 1$, the expected delay approaches $D_{\max}$, which is the maximum delay for one transmission under the uniform delay model. Under the exponential delay model, the expected delay approaches $D_{\min} + \frac{1}{\lambda}\ln(n)$, making the communication delay logarithmically proportional to the distribution level.

IDA-based communication attempts at striking a balance between the above three extreme setups. Figure 5 illustrates the improvement (speedup) in communication delay that can be achieved through the use of even remarkably small levels of redundancy.[7]

[7] These results were obtained under an exponential delay model, but can be easily reproduced for any other model.

For example, at a 20% redundancy level ($\frac{1}{5}$ of the communicated data is redundant), IDA cuts the expected delay through a communication network by almost 50% (a 2-fold speedup) for an object distributed over 8 sites. This gain is even larger for objects distributed over a larger number of sites. If the level of redundancy is increased further, the gain is substantial. For example, IDA can deliver a 5-fold speedup in communication with the redundancy level set at 50% for an object that is distributed over 32 sites.

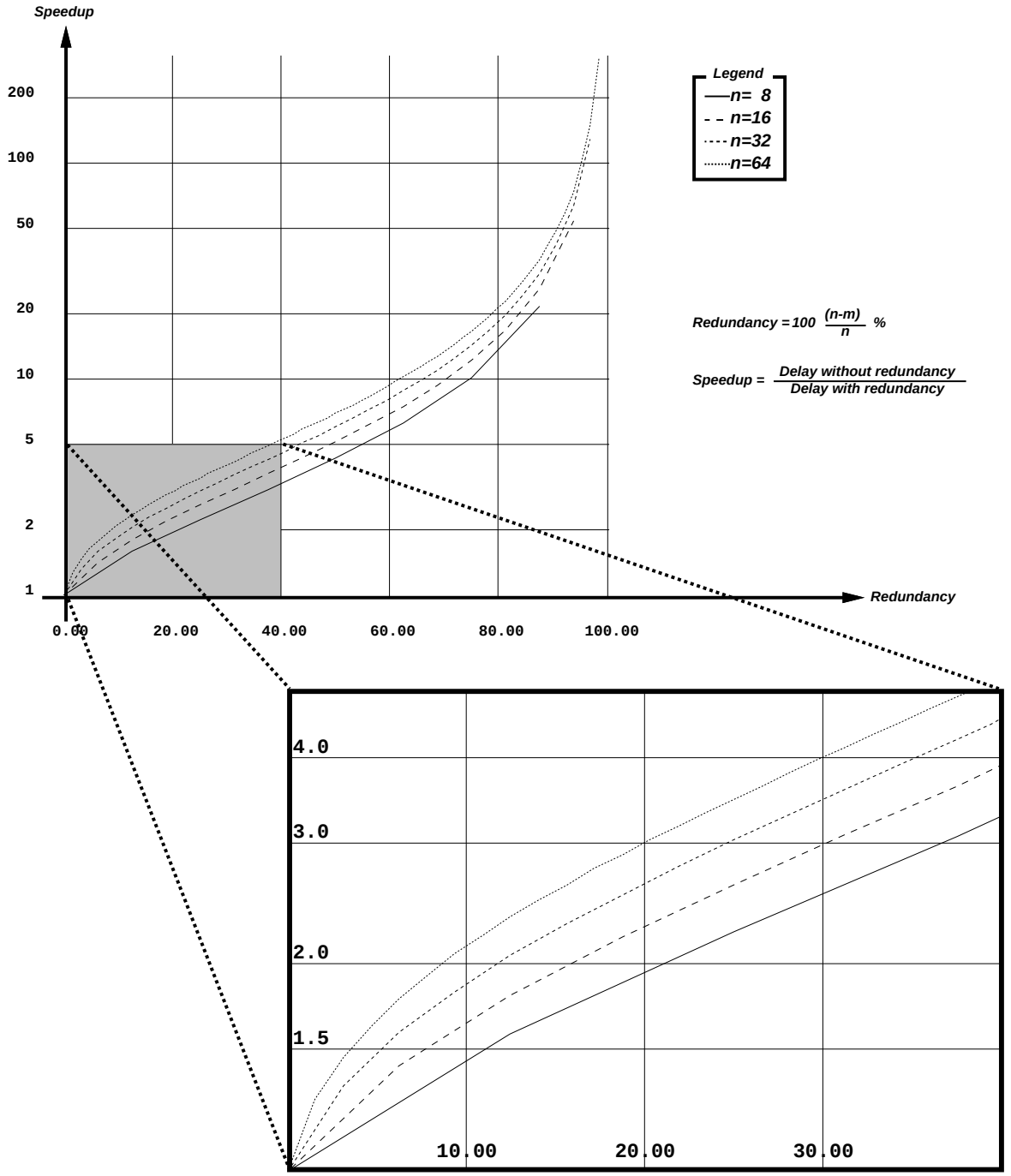

Figure 5: Expected AIDA speedups – Only the random part of the delay is considered.

For the same amount of redundancy, other protocols (such as replication) yield minuscule speedups compared to IDA. For example, if an object is replicated once ($\frac{1}{2}$ of the communicated data is redundant) and each of the two replica is distributed over 16 sites (for a total of 32-site distribution), then it can be shown that under the exponential delay model, the achievable speedup will be less than 1.1-fold. Under the same conditions, IDA delivers over 5-fold speedups.

AIDA-BASED BANDWIDTH ALLOCATION

In this section, we highlight the features of AIDA that enable it to deal effectively with deadline and priority issues in time-critical systems.

Using Redundancy to Control Communication Delays

Let the retrieval of an object X be subject to a soft time-constraint that requires X to be fetched within $T^X_{\max}$ units of time. According to equation 5 the expected delay in retrieving X decreases *predictably* as $n-m$ increases. Incorporating the time constraint in equation 5, we can solve for n as follows.

$$\begin{aligned} T^X_{\max} &\geq \tau_e \\ &\geq D_{\min} + \frac{1}{\lambda} \sum_{r=n-m+1}^{n} \frac{1}{r} \\ \lambda(T^X_{\max} - D_{\min}) &\geq \sum_{r=n-m+1}^{n} \frac{1}{r} \end{aligned}$$

Using the lower bound $[\ln(n) - \ln(n-m)]$ to approximate the value of $\sum_{r=n-m+1}^{n} \left(\frac{1}{r}\right)$, we get:

$$\lambda(T^X_{\max} - D_{\min}) > \ln(n) - \ln(n-m)$$

Solving the above inequality for a lower bound on n we get:

$$n > \frac{m}{1 - e^{-\lambda(T^X_{\max} - D_{\min})}} \tag{6}$$

In order to compute the appropriate value of n using equation 6, it is necessary to evaluate *dynamically* the values of $D_{\min}$ and λ. This can be done using statistical techniques similar to the those described in [10].

Priority-based rationing of redundant bandwidth

Equation 6 establishes a lower bound on n that guarantees an *expected* communication delay, not an *actual* communication delay. In other words, while it is very possible for the actual communication delay to be less than the desired expected delay (thus satisfying the imposed time constraint), it is very possible as well for the actual communication delay to exceed the desired expected delay (thus resulting in a violation of the imposed time constraint). This *randomness* factor can be accounted for and controlled by using second order moments (*e.g.* Standard Deviation) to build a *confidence interval* about the actual communication delay. One way of building such a confidence interval is to set the value of n so as to make $T^X_{\max}$, the available slack for completing the communication session, greater than or equal to $\tau_e + \alpha\sigma_e$ (rather than simply τ_e).

$$n > \frac{m}{1 - e^{-\lambda(T^X_{\max} - D_{\min} - \alpha\sigma_e)}} \tag{7}$$

The value of n in the above equation defines a confidence interval that corresponds to a specific probability of meeting the time constraint imposed on the communication

session. This probability can be made arbitrarily high by increasing the value of α. This, however, is not without cost. In a distributed real-time system, the total communication bandwidth is *finite*, and increasing the amount of *redundant* information flowing in the system may adversely affect the end-to-end delay characteristics that we were aiming to improve in the first place!

One way of solving the aforementioned problem is to set the value of α in such a way that the total available bandwidth in the system is *rationed* among the different communication sessions in a way that reflects the *priority* assigned to these sessions. In other words, the value of α for a particular task is related to its priority *and* the priority of all the other tasks sharing the available bandwidth in the system.

It is important to notice that using AIDA, the priority of the transaction (how critical it is to the mission of the system) and the urgency of the transaction (how tight its time constraint is) are *both* taken into account when the value of n is determined. This stands in sharp contrast with protocols that deal only with either the priority of the transaction *or* its urgency, making it necessary for applications to express (artificially) one of these attributes using the other.

Fault-tolerance and Security Characteristics

The usual technique employed to deal with communication failures is to retransmit on errors (or timeouts). For time-critical applications, this detect-then-recover approach may not be feasible due to the time constraints imposed on the system. Instead, masking techniques are employed. In particular, error-correcting codes are used to tolerate communication failures, whereas replication and/or n-modular redundancy (NMR) techniques are used to protect against site failures [22]. The main drawback of these techniques is their excessive use of redundancy, which may adversely affect performance. For example, to mask one site failure an approach relying on replication will require that a particular object be retrieved from two different sites, thus doubling the network traffic. The blowup is even larger when error-correction for a relatively small number of communication-induced errors is taken into account.

The AIDA-based protocol we are proposing in this paper is a failure-masking protocol that is provably optimal in its use of redundancy. The main reason for AIDA's superiority is that it does not distinguish between communication failures and site failures, thus making the best use of allotted redundancy in the system. To tolerate up to r simultaneous failures, AIDA requires that the total number of sites from which the dispersed object X will be requested exceeds the minimum number of data pieces needed to reconstruct X by r. Thus, a total of $n = m + r$ sites is needed for every m pieces of data, a redundancy of $100(n - m)/m$ percent. For example, if an object X is to be dispersed over $n = 12$ sites and coverage for up to 3 failures is required, then using AIDA, the total redundancy would be 25% (a blowup of 33%). To provide the same coverage using replication, the total redundancy would soar to 75% (a blowup of 300%).

A common technique to insure communication security is to store and communicate information using some form of encryption, where only authorized users are enabled to decrypt the information through the use of appropriate *secret keys* [24]. The proven difficulty of decrypting the information without knowing the secret key guarantees a high level of *security*. The main disadvantage of this technique is that the information (although encrypted) is available in one site – whether stored in or communicated through that site – for long periods of time. This may make it possible for adversaries to break the secret key of the encryption.

The AIDA-based protocol we are proposing in this paper guarantees the security of the communicated information by making it unavailable as a whole in any one par-

ticular site. As a matter of fact, it is hard to get any clue about the original information unless at least m pieces from the dispersed file are collected. This makes the task of the adversaries more difficult, since they have to control m of the sites and not only one. Even if this happens, it is provably very difficult to reconstruct the original file unless the secret key is known.

Simulation Results for AIDA-based Communication

Predicting the effectiveness of AIDA using analytical techniques is limited to simplistic assumptions concerning the network delay characteristics, server response times, and communication capacity. Simulation becomes the only alternative, if realistic assumptions are to be adopted. We have conducted a number of simulations to further evaluate the promise of AIDA and check the accuracy of our analytical expectations. Figure 6 shows the simulation model used.

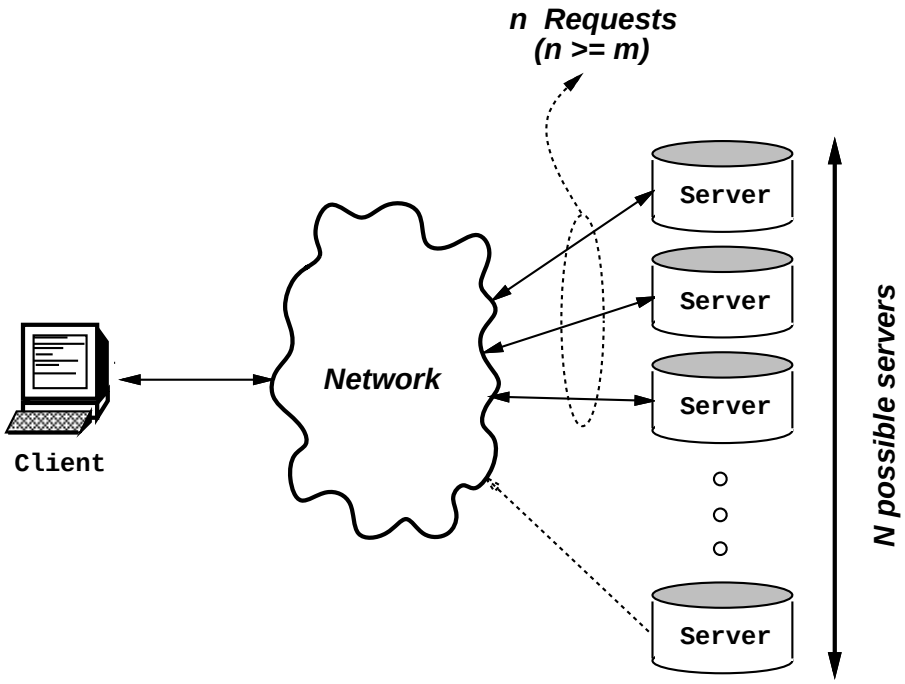

Figure 6: Simulation Model

Figure 7-a shows the speedups obtained by AIDA if the simplistic exponential delay model is replaced with a realistic multi-staged model, where each stage introduces an exponentially-distributed delay. The simulations show that AIDA still reduces the communication delay considerably.

Figure 7-b shows the speedups obtained by AIDA under a limited channel/server assumption. The simulations show that an expected *saturation* point exists. Obviously, increasing communication redundancy beyond that point is counter-productive. This confirms that a feedback redundancy control mechanism must be employed with any implementation of AIDA.

Two parameters determine the *achievable* access time for a given object (e.g. shared memory page, database record, or video frame sequence) in the system: distribution level m and dispersal level N. Figure 8-a shows the effect of varying these two parameters. This confirms the need to manage the distribution/dispersal level for the various objects in the system, based on the desired accessibility and timeliness constraints imposed on these objects. Needless to say, these constraints (and therefore the control mechanism employed to manage their distribution/dispersal levels) need to be dynamic to accommodate the changing priorities and modes of the system operation.

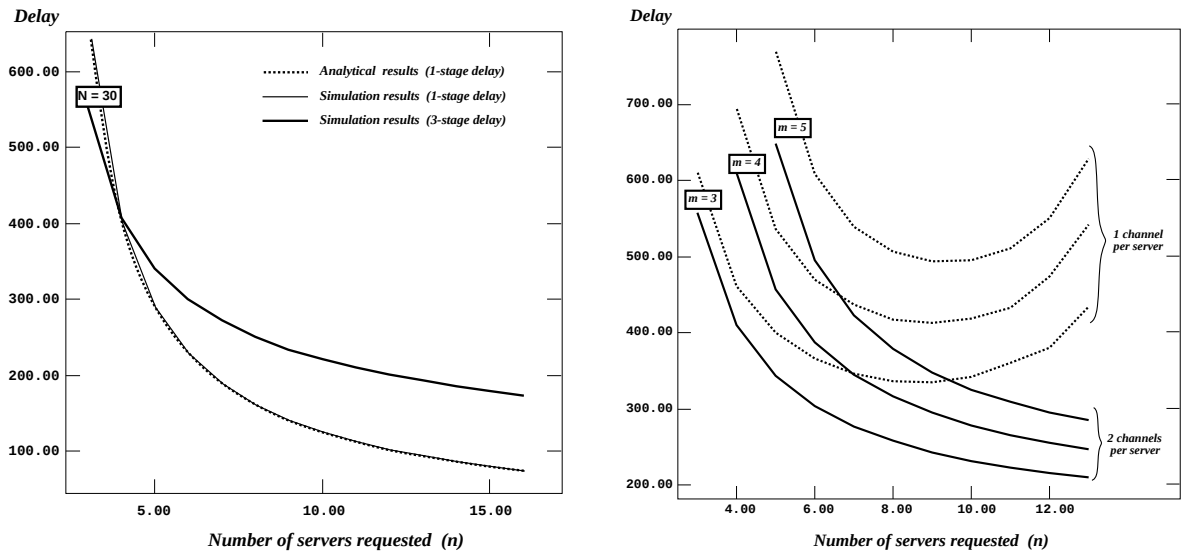

Figure 7: Simulation: (a) Staging effect (b) Channel capacity effect

Our analytical evaluation has assumed that all the servers in the system have identical responsiveness. If this assumption is not true, then the algorithm used to select the n out of N servers to be consulted for the retrieval of an object (where $m \leq n \leq N$) becomes crucial. Figure 8-b shows the performance of two such algorithms: *random selection* and *cyclic selection*. Both of these selection algorithms are blind in the sense that they do not account for the servers load or network traffic.

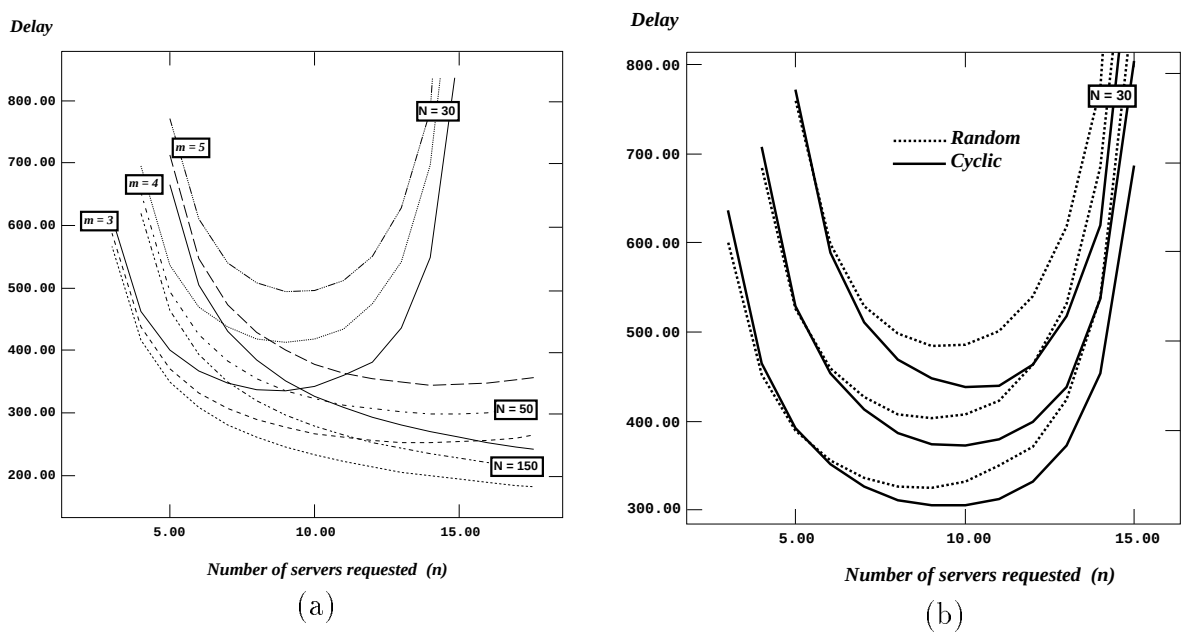

Figure 8: Simulation: (a) Distribution/dispersal effect (b) Server selection effect

We have performed a number of other experiments to study the effect of AIDA on data buffering/caching requirements (necessary for satisfying synchronization constraints in applications like Multimedia). One result was eminent: by reducing/controlling the uncertainty associated with communication delays, the buffering/caching requirements were greatly relaxed. This result suggests that AIDA should be used *in conjunction* with, and as an *integral part* of the I/O and memory management subsystems.

CONCLUSION

AIDA is a novel bandwidth allocation strategy suitable for distributed fault-tolerant time-critical systems. In AIDA redundancy is used to tackle several crucial problems; it is used to *tolerate* failures, to *increase* the likelihood of meeting tight time-constraints, and to *ration* (based on task priorities) the limited bandwidth in the system. Currently, we are in the process of building a prototype for an AIDA-based Network File Server (NFS), whose salient features are discussed below.

A request for data access made to the proposed AIDA-based NFS will specify both a priority level and a time constraint. The priority level will be used to establish the predictability requirement and thus the allotted redundancy (i.e. percentage of system bandwidth) to be granted for that request in order to meet its timing constraint. Using statistical data about the expected delays through the network, a subset of all servers that can grant such a request will be determined. The statistical techniques to be used will be similar to those suggested in [9, 15, 8, 18].

While the correctness and efficacy of AIDA are not dependent on the algorithm used to select the n out of N sites to be consulted for an object retrieval, its performance may benefit greatly. Performance gains may be achieved by classifying communication requests as was done in [15, 16]. This will likely reduce the uncertainty associated with communication delays, thus providing for a superior bandwidth allocation.

For a distributed real-time application, the NFS is only an added layer in the memory hierarchy. The addition of such a layer cannot be done in isolation from other system components (e.g. memory and I/O managers). In particular, the NFS might be able to improve the timeliness of data access considerably, if the access pattern (working set) for a given process can be reliability predicted so as to use local storage (e.g. memory or disk) to cache the NFS objects.

The reliability and accessibility requirements of various data objects in a distributed real-time application depend on the system *mode* of operation. For example, while the timely access of a data object (e.g. "location of nearby aircrafts") could be critical in a given mode of operation (e.g. "combat"), it might be less critical in some modes (e.g. "landing"), or even completely unimportant in others. The proposed AIDA-based NFS will be able to maintain different user-defined profiles of reliability and accessibility requirements for the various objects (files) in the system. This can be done dynamically (when a mode-change takes place) by controlling the levels of distribution and dispersal for the system objects.

In this paper, we have focused on using AIDA for information retrieval. Issues pertaining to information *update* were not tackled. These issues are particularly important in distributed time-critical systems to ensure data consistency and recency. In particular, it is of utmost importance to investigate the interaction between AIDA and other consistency-preserving protocols such as distributed shared memory protocols [25], caching protocols [1], and non-coherent memory protocols [13].

AIDA does not *guarantee* that time-constraints will be satisfied, rather it guarantees a lower bound on the *probability* of meeting these constraints. This probability can be made arbitrarily high if enough redundancy is secured. This, however, may not be feasible if the system is running close to capacity. One possible approach to deal with such a situation is to allow the *quality* of service to degrade gracefully. The integration of AIDA with *best-effort* techniques such as those presented in [8, 12] and *imprecise computation* techniques such as those suggested in [19] is an interesting research problem yet to be pursued.

ABOUT THE AUTHOR

Azer Bestavros (bestavros@cs.bu.edu) is an Assistant Professor in the Computer Science Department at Boston University, Boston, Massachusetts. He received his S.M. and Ph.D. degrees in Computer Science from Harvard University in 1989 and 1992, respectively. He is a member of several societies in the IEEE and the ACM. He is the moderator and maintainer of the electronic newsletter and archives of the IEEE-CS Technical Committee on Real-Time Systems. His research interests include reliable time-constrained communication and computation, embedded responsive systems, real-time databases, and parallel computing.

References

[1] J. Archibald and J-L. Baer. Cache coherence protocols: Evaluation using a multiprocessor simulation model. *ACM Transactions on Computer Systems*, 4(4):273–298, November 1986.

[2] A. Bernstein, A. Philip, V. Hadzilacos, and N. Goodman. *Concurrency Control And Recovery In Database Systems.* Addison-Wesley, 1987.

[3] Azer Bestavros. IDA-based disk arrays. Technical Memorandum 45312-890707-01TM, AT&T, Bell Laboratories, Department 45312, Holmdel, NJ, July 1989.

[4] Azer Bestavros. SETH: A VLSI chip for the real-time information dispersal and retrieval for security and fault-tolerance. In *Proceedings of ICPP'90, The 1990 International Conference on Parallel Processing*, Chicago, Illinois, August 1990.

[5] Azer Bestavros. IDA disk arrays. In *Proceedings of the First International Conference on Parallel and Distributed Information Systems*, Miami Beach, Florida, December 1991.

[6] Azer Bestavros, Danny Chen, and Wing Wong. The reliability and performance of parallel disks. Technical Memorandum 45312-891206-01TM, AT&T, Bell Laboratories, Department 45312, Holmdel, NJ, December 1989.

[7] Ramez Elmasri and Shamkant Navathe. *Fundamentals of Database Systems.* The Benjamin/Cummings Publishing Company Inc., 1989.

[8] D. Ferrari. Design and application of a delay jitter control scheme for packet-switching internetworks. In *Proceedings of the second International Conference on Network and Operating System Support for Digital Audio and Video*, Heidelberg, Germany, November 1991.

[9] D. Ferrari and D.C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, April 1990.

[10] John F. Gibbon, Azer Bestavros, and Tom Little. Limited a priori scheduling for distributed multimedia systems. Work in progress, November 1992.

[11] Garth Gibson, Lisa Hellerstein, Richard Karp, Randy Katz, and David Patterson. Coding techniques for handling failures in large disk arrays. Technical Report UCB/CSD 88/477, Computer Science Division, University of California, July 1988.

[12] M. Gilge and R. Gussella. Motion video coding for packet-switching networks – an integrated approach. In *Proceedings of the SPIE Conference on Visual Communications and Image Processing*, Boston, MA, September 1991.

[13] Abdelsalam Heddaya and Himanshu S. Sinha. An overview of MERMERA: a system and formalism for non-coherent distributed parallel memory. In *Proceedings of the 26th Hawaii International Conference on System Sciences*, January 1993.

[14] Harold Larson. *Probability theory and statistical inference, Third Edition.* John Wiley & Sons, 1982.

[15] Aurel A. Lazar, Adam Temple, and Rafael Gidron. An architecture for integrated networks that guarantees quality of service. *International Journal of Digital and Analog Cabled Systems*, 3(2), 1990.

[16] Edward Lee, Peter Chen, John Hartman, Ann Drapeau, Ethan Miller, Randy Katz, Garth Gibson, and David Patterson. RAID-II: a scalable storage architecture for high-bandwidth network file service. Technical Report CSD 92/672, University of California at Berkeley, Spring 1992.

[17] T.D.C. Little and A. Ghafoor. Scheduling of bandwidth-constrained multimedia traffic. *Computer Communications (Special Issue on Multimedia Communications)*, 15(6):381–387, July/August 1992.

[18] T.D.C. Little and J.F. Gibbon. Management of time-dependent multimedia data. In *Proceedings of the SPIE Symposium OE/FIBERS'92: Enabling technologies for multimedia, multiservice networks*, Boston, MA, September 1992.

[19] Jane Liu and Victor Lopez-Millan. A congestion control scheme for a real-time traffic switching element using the imprecise computations technique. In *Proceedings of the IEEE IPPS 1st Workshop on Parallel and Distributed Real-Time Systems*, pages 89–93, Newport Beach, CA, April 1993.

[20] Yuh-Dauh Lyuu. Fast fault-tolerant parallel communication and on-line maintenance using information dispersal. Technical Report TR-19-1989, Harvard University, Cambridge, Massachusetts, October 1989.

[21] Michael O. Rabin. Efficient dispersal of information for security, load balancing and fault tolerance. *Journal of the Association for Computing Machinery*, 36(2):335–348, April 1989.

[22] B. Randell, P. Lee, and P. Treleaven. Reliability issues in computing system design. *ACM Computing Surveys*, 10:84–98, June 1978.

[23] Martin Schulze, Garth Gibson, Randy Katz, and David Patterson. How reliable is a RAID? In *Proceedings of COMPCON-89, the Thirty-fourth IEEE Computer Society International Conference*, March 1989.

[24] A. Shamir. How to share a secret? *Communication of the ACM*, 22:612–613, November 1979.

[25] Avadis Tevanian and (et al). A Unix interface for shared memory and memory mapped files under Mach. Technical report, Carnegie-Mellon University, Department of Computer Science, February 1987.