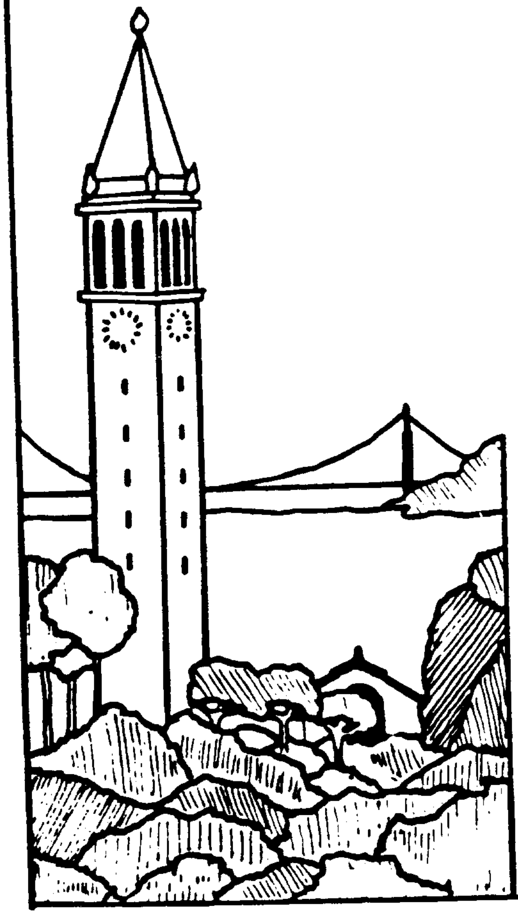
A Protocol for Secure Communication in Large Distributed Systems

D. P. Anderson, D. Ferrari,

P. V. Rangan, and B. Sartirana



Report No. UCB/CSD 87/342

February 1987
PROGRES Report No. 87.1

Computer Science Division (EECS)
University of California
Berkeley, California 94720

A Protocol for Secure Communication in Large Distributed Systems*

D. P. Anderson, D. Ferrari, P. V. Rangan, and B. Sartirana(°)

Computer Systems Research Project
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

Abstract

A mechanism for secure communication in large distributed systems is proposed. The mechanism, called *Authenticated Datagram Protocol* (ADP), provides message authentication and, optionally, privacy of data. ADP is a host-to-host datagram protocol, positioned below the transport layer; it uses public-key encryption to establish secure channels between hosts and to authenticate owners, and single-key encryption for communication over a channel and to ensure privacy of the messages. ADP is shown to satisfy the main security requirements of large distributed systems, to provide end-to-end security in spite of its relatively low level, and to exhibit several advantages over schemes in which security mechanisms are at a higher level. The results of a trace-driven measurement study of ADP performance show that its throughput and latency are acceptable even within the limitations of today's technology, provided single-key encryption/decryption can be done in hardware.

* This research was supported by the Defense Advanced Research Projects Agency (DoD), ARPA Order No. 4871, monitored by the Naval Electronic Systems Command under Contract No. N00039-84-C-0089, by the IBM Corporation, by Olivetti S.p.A., by MICOM-Interlan, Inc., by CSELT S.p.A., and by the University of California under the MICRO Program. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing official policies, either expressed or implied, of any of the sponsoring agencies or corporations.

(°) B. Sartirana's permanent affiliation: Gruppo Sistemi e Reti di Informatica, Divisione Minicomputer e Reti, Ing. C. Olivetti & C., Ivrea (Torino), Italy.

1. Introduction

According to technological forecasts [29], the bandwidths of future wide-area data networks will substantially exceed those offered by current local-area networks. It is natural to expect that those wide-area networks will be used to build *very large distributed systems* (VLDS) that will:

(a) be very large in numerical, geographical, and administrative terms; in other words, they will consist of many interconnected hosts, distributed over a large area, and owned by many organizations and individuals;

(b) allow well-integrated (perhaps transparent) and efficient sharing of resources (processing power, data, communication services) over long distances;

(c) connect individuals and organizations that may wish to share resources, but, since they do not trust each other, demand strict control over their own resources and the ability to function autonomously;

(d) use a variety of communication protocols, including both request/reply and stream-oriented protocols, to access remote resources;

(e) consist of hardware components that in general do not have any physical security (for example, most hosts, including those acting as network gateways, will be loadable with arbitrary kernel software by malicious users).

Most existing and proposed distributed systems make security-related assumptions that are incompatible with one or more of the above properties of VLDSs. Examples of such assumptions are:

- All system-level components trust one another. Once a user is authenticated to the local kernel (via a password mechanism, perhaps), access to remote resources requires no further authentication. Examples are V [10], Eden [3] and Cronus [28].
- The system is controlled by a single administrative agent, so that it is possible to have universally trusted name and/or key servers. Examples are Sun UNIX$^+$ [30], Mach [27], and Grapevine [6,7].
- The system is restricted to a single local area network that, like the Ethernet [21], guarantees that either every host on the network correctly receives a packet, or that no host receives it. Thus a packet cannot be modified by a malicious host. The system may span networks connected by gateways if these gateways are physically secure.
- There is a single administrative agent that can punish security violators, e.g., by firing them. Therefore, violation detection, rather than prevention, is sufficient.
- Resource sharing across administrative domains is possible, but is poorly integrated (e.g., is restricted to mail and file transfer). Typically, there are no global naming or authentication mechanisms, and a user must have independent accounts and passwords on each host. An example is Berkeley UNIX [20].
- The network is physically secure, or the network interfaces are assumed to contain tamper-proof logic, as in Amoeba [22].

A VLDS, as we have defined it, fails to satisfy any of the above assumptions, and thus has security problems that are not present in current distributed systems.

In this paper, we propose a mechanism for secure communication in Very Large Distributed Systems. The mechanism is called *Authenticated Datagram Protocol* (ADP). It is positioned below the transport layer, at the level of datagram communication between hosts. ADP is the basis of a new architecture for fast secure communication; it is at a level sufficiently high for end-to-end security, but sufficiently low for inexpensive and efficient

$^+$ UNIX is a trademark of AT&T Bell Laboratories.

operation. Potentially, all network communication in the VLDS passes through ADP. We are currently building a distributed system called DASH that uses ADP [4].

ADP will be shown to have these properties:

(i) it satisfies the requirements of VLDSs;

(ii) it can provide end-to-end security;

(iii) it has several advantages over designs in which the security mechanisms are at a higher level of the protocol hierarchy;

(iv) its performance can be made acceptably high even on today's machines and LANs, provided single-key encryption is done in hardware.

The rest of the paper is organized as follows. Section 2 describes the security model on which ADP is based, and Section 3 its design and implementation. The experiments that were carried out to evaluate its performance and some of their results are presented in Section 4. Finally, some conclusions are offered in Section 5.

2. The Security Model

ADP's security principals are called *owners*. Each owner has a unique symbolic name and a unique private-key/public-key pair. The mechanisms for delegating naming authority and those for mapping names to public keys are being studied within the DASH Project but are outside the scope of this paper.

The distributed system includes a set of *hosts*, each running an operating system *kernel*. At any point in time, a host has a *kernel owner*. The ownership of a kernel is established at boot time, before network communication takes place; it might be done manually or from a ROM. Kernel ownership may change over time, e.g., as different people boot a public workstation. A crash-free period under a single kernel owner is called a *kernel session*.

The kernel may support multiple *user processes*, each of which has an associated owner, perhaps different from the kernel owner. A kernel possesses the private keys of its kernel owner and of all owners of the user processes it has executed or is executing.

A kernel is *security-correct* if

(1) it limits the information flow out of each process it is executing to that explicitly requested by this process;

(2) it performs name resolution correctly;

(3) it correctly executes the algorithms of ADP, as described below;

(4) it allows only its owner to read the private keys it contains.

All of these conditions require some form of logical correctness on the part of the kernel; methods of ensuring this are outside the scope of this paper (see [9]). Condition 4 requires that either

(a) the host is physically accessible only to its current owner, or

(b) there is no mechanism for reading out the kernel's private key storage that can be used by somebody other than the kernel owner. This is not the case if there is a "reset button" that jumps into a ROM monitor without first destroying the keys, or if a hardware failure generates a memory dump to a file or device readable by anyone other than the kernel owner. However, the condition is satisfied if private keys are held in a write-only storage inside the encryption chip.

An owner X is *kernel-trustworthy* if, whenever X owns a kernel, this kernel is security-correct.

If an owner X issues a command to execute a process remotely on a kernel with owner Y, we infer (and say) that X is placing *kernel trust* in Y.

3. The Design of ADP

3.1. An operational description

As mentioned in Section 1, ADP is a host-to-host datagram protocol that provides secure communications, i.e., authentication and, optionally, privacy of messages.

ADP makes use of an underlying network layer that provides an insecure datagram service. This service could vary according to the remote host involved; for example, the Internet Protocol might be used [24] for a distant host, while a simpler network protocol would suffice for a host on the same LAN. In some cases, ADP must handle fragmentation of long messages; this issue will not be considered here.

ADP does not implement any particular authorization scheme, nor does it provide other security guarantees such as confinement or freedom from denial of service. With ADP as a basis, the kernel may provide higher-level services for user processes. For example, DASH supports a general client/server model in which the authentication provided by ADP underlies authorization mechanisms at both the kernel and the user level.

3.1.1. Secure channel establishment

The operation of ADP is based on a host-to-host *secure channel* that is established whenever a (user or kernel) process executing on a host wants to communicate with another process running on another host and such a channel does not already exist. Public-key encryption (PKE) [15,26] is used to establish a secure channel: when a host A needs to establish a secure channel to a host B, the ADP module running on A sends a channel establishment request to B. This request contains random string *S*, encrypted with the public key of the owner of B, and random string *T*. *S* will be used as the secret single key of the secure channel, and *T* will be used as a digital signature [1,12,13] to authenticate owners from B to A. A marks the secure channel as being *tentative* until it receives an acknowledgement. The secure channel request is included with ADP messages sent while the channel is still tentative.

In the secure channel acknowledgement and in every ADP message until the first signature is received, B sends to A a random string *R* to be used for signatures sent from A to B, i.e., to authenticate owners from A to B. If the two hosts simultaneously try to establish a channel between themselves, the one with the lexicographically greater name determines the channel key *S*.

In many cases, patterns of communication (in terms of local and remote owners and addresses) are predictable; for example, a workstation will always communicate with local file servers. It is then possible for a kernel to establish secure channels, and do authentication on those channels, in advance of user demand (e.g., at boot time).

3.1.2. Owner authentication

When an owner *X* who has not sent any messages to B before wants to communicate from host A with an owner *Y* on host B, and a secure channel between A and B has been established, public-key encryption is used to authenticate *X* to B (more precisely, to the owner of B's kernel). The ADP module running on A encrypts string *R* with the private key of *X* and sends it in a message along with *X*'s name; the ADP module on B decrypts it with the public key of *X* obtained from the name server, and compares the result to string *R*; if the two strings are identical, then B concludes that the message is from *X* (or from a host that possesses *X*'s private key), and caches *X*'s name in the list of the owners authenticated from the other end that it maintains for that channel. A in turn caches the name of *X* in its list of owners authenticated to the other end of the channel (see Figure 1).

When this operation has been done, both hosts are aware, and remember, that *X* has been authenticated to B. This *authentication caching* means that expensive PKE-based authentication need be done only the first time an owner is involved as a sender or receiver on a particular secure channel. In all subsequent messages sent from *X* to *Y*, the presence of *X*'s name in the caches maintained by ADP on A and B is considered sufficient for the authentication of the messages.

3.1.3. Messages

As explained in [5], messages from X to Y, when X has been authenticated to the host where the destination process belonging to Y resides, only require for the purposes of authentication either the encryption (with the secure channel's secret key S) of a message trailer that includes a sequence number or a cryptographic checksum [2,14]. If privacy of the data is also desired, then the entire message may be encrypted with S. Thus, ADP uses a bootstrapping mechanism to combine the advantages of public-key [15,26] and single-key [23] cryptography. As a basis for authentication in large distributed systems, public-key schemes have several advantages over single-key schemes [16]. In VLDSs, replication is essential for performance, availability, and fault-tolerance. Key server replication increases the vulnerability of a single-key scheme to attack on secrecy, whereas it reduces the vulnerability of public-key systems. Current public-key encryption algorithms are too slow to consider using them to encrypt any part of each message sent. On the other hand, single-key operations are fast enough to be employed for each message.

Further reductions in encryption overhead can be achieved if ADP and its users recognize the existence of messages that are not particularly urgent (e.g., most acknowledgements, most writes to remote disk) and that can therefore be delayed and piggybacked onto the next urgent message to be transmitted over the same secure channel. In this case, one *ADP message* on the network may include a number of user messages. Only one sequence number will have to be encrypted, or only one cryptographic checksum will have to be computed, for each ADP message, thereby reducing the cost of encryption per user message.

The urgency of a message will typically be determined by the process originating it: a message marked urgent will be shipped by ADP as soon as possible; one whose maximum delay is specified will have to be sent when this timeout expires, unless it will have already been piggybacked onto an urgent message; finally, one which neither is marked urgent nor has its maximum delay specified will be assigned by ADP a timeout, which will usually be a tunable parameter of ADP and the same for all such messages. Since delayed messages will have to be kept in queues within ADP, another tunable parameter is the maximum size of each queue: the arrival of a non-urgent message which causes a queue to fill or to overflow will be treated as if it were the arrival of an urgent message. This maximum size will depend on the amount of buffer space available, and may be limited by the maximum size of the messages that can be accepted by the lower protocol layers. The one just described is the queue service discipline that has been implemented in the current version of ADP, but is by no means the only possible one, and we do not claim it is the best.

3.2. The properties of ADP

From the security model introduced in Section 2 and the operational description of the protocol in Section 3.1, it can be seen that ADP provides authentication as well as, when desired, privacy. To be more precise, when ADP delivers to owner Y a message tagged with the name of owner X, it guarantees that, if X has placed kernel trust only in kernel-trustworthy owners, then the message was sent at the request of a process owned by X, and was directed to a process owned by Y. ADP also guarantees that, if X has requested privacy in the transmission of that message, and if the kernel where the sending process (belonging to X) was running is security-correct, the text of the message is readable only by kernels with the same owner as the host where the destination process (belonging to Y) resides.

As shown in [5], ADP satisfies the requirements of VLDSs. In particular, it does not require physical security, or kernels to trust each other, or owners to trust all kernels. Nor does it restrict sharing of resources in any way. Other functions like authorization can be efficiently built on top of ADP [4]. In our model of distributed computation, an owner who runs a process on a kernel with a different owner has no privacy from that owner. If the kernel is not security-correct, it can alter or publicize any data accessible to the user process. In this model, no additional security is obtained by doing encryption at higher kernel levels or in user processes. This host-to-host security and the security-correctness of the kernel provide as much end-to-end security as is possible in our model without using external mechanisms.

Since security mechanisms at the level of host-to-host datagrams can exhibit (for example, in ADP) desirable end-to-end characteristics, it is reasonable to discuss their several advantages:

- They allow heterogeneity and modifiability of the implementation without the need to change any of the higher level protocols.
- They simplify transport level protocols. When a host crashes, its secure channels are destroyed. Thus, remote host crashes are detected at the host-to-host level at the time of secure channel establishment. In combination with the elimination of duplicates and the limiting of replay delay [5], this means that three-way handshakes can often be eliminated from transport-level protocols.
- Three-way handshakes can be often eliminated from stream and request/reply protocols. A short transaction then requires just two messages, as opposed to at least six in TCP [17] and four in secure RPC [7].
- Security functions need not be duplicated in multiple transport protocols or user programs.
- In ADP there are two public-key operations (encryption or decryption) per owner per remote host per kernel session. Often these operations can be done at boot time or during idle periods. There are no per-process or per-operation public-key operations, resulting in a substantial performance benefit.
- Since messages from all client processes and higher level protocols pass through ADP, a number of these messages destined to a common remote host can be combined into a single datagram and authenticated once using the channel's secret key. This can appreciably reduce the number of single-key operations.
- Additional specific advantages over transport-layer [7,11,17,30] or higher-layer solutions are discussed in [5].

4. An Experimental Evaluation of ADP

4.1. The design of the experiments

A prototype of ADP has been implemented in C++ (the language we are using to build DASH) and integrated with the necessary portions of the DASH kernel. This implementation runs on Sun 3/50 workstations connected by an Ethernet, and has been used as the basis for a set of experiments. The main goals of these experiments were:

(a) to measure the performance of ADP under a variety of external and internal conditions, especially with an input traffic as high as we expect it to be in VLDSs;

(b) to determine the impact of encryption/decryption overhead on ADP performance;

(c) to evaluate the influence of non-urgent message queueing on ADP performance, hence the value of introducing the notion of urgent message in protocol design;

(d) to study the sensitivity of ADP performance to the variations of such parameters as the maximum delay of non-urgent messages.

Among the interesting investigations that had to be postponed to the near future, as our current experimental setup does not permit them, are the evaluation of the effects of caching, and that of the advantages of running ADP on multiprocessor machines (the protocol has been designed so as to maximize parallelism).

The following primary performance indices were chosen for the study:

- *Throughput T*: the maximum sustained rate at which information can be transmitted by an ADP module and received by another (remote) ADP module.
- *Latency L*: the delay incurred by a message between the instant it is given to an ADP module for transmission and the instant it is delivered by another ADP module to the destination process or to a higher-level protocol on the destination host. Since the latency of non-urgent messages is guaranteed to be acceptable, as it is bounded by their maximum delay, all our measurements of L are for the urgent messages only. In the rest of this paper, unless otherwise specified, we shall use the symbol L to denote the average latency of the messages in a given finite sequence.
- *Secure channel establishment time C*: the time between the instant a channel establishment request is generated by an ADP module and the instant the corresponding acknowledgement is received by that module.

Another index we measured in our experiments is the *utilization of the CPU*, obtained by timing a low-priority idle process that was running whenever no other process was running.

The setup adopted for the experiments is schematically depicted in Figure 2. It consists of two diskless Sun 3/50 workstations, each with a main memory of 4 Mbytes that can be downloaded from the file server. When both memories have been downloaded, the switch may be opened to eliminate the interference that would be caused by the traffic generated by the other workstations. Note that by "DASH kernel" in the figure is meant that portion of the kernel which is needed to support the software modules depicted there, the keyboard, and the display. Also, "IP" represents the Internet Protocol, whose fragmentation and reassembly routines were, however, the only portions exercised in our experiments.

Not shown in the figure is the heart of the instrumentation used to measure the indices introduced above, i.e., the clock; since the resolution of the workstation's clock (10 ms) was insufficient for latency measurements, we used the clock of one of the serial communication controllers to interrupt the CPU with the constant period of 1 ms; this method allowed us to measure times with a 1 ms resolution at the cost of only 2% of the total CPU power [31]. Another important component of the setup not represented in the figure is the Zilog Z8068 DES chip mounted in both workstations and used for single-key hardware encryption and decryption. This chip is seen by the CPU as a polled I/O device with an 8-bit wide data interface. Figure 2 clearly shows that our performance indices do not represent the performance of ADP *per se*, but rather that of the ADP-"IP"-Ethernet driver-Ethernet complex.

The main function of the sender process is to generate input messages for ADP. While protocol performance (especially throughput) is frequently measured in a laboratory setting as a function of message size by sending into the protocol sequences of equally sized messages (see for example [8]), doing this to evaluate ADP was likely to provide misleading results, as ADP performance must be expected to be sensitive to the distributions of individual message sizes, and even to their temporal sequences. Thus, it was decided to run trace-driven experiments.

Since a complete DASH system does not exist yet and a real ADP input trace cannot therefore be measured, the assumption was made that the message traffic on a local-area network interconnecting a variety of machines, including diskless workstations and file servers, would represent a reasonable approximation to the type of traffic that an ADP module will experience in a DASH system. Indeed, we conjecture that remote resource sharing will be substantial in a VLDS, and presumably the sizes of the messages exchanged for that purpose will not drastically differ from page and file transfers to and from file servers in today's LANs. A trace of all packets transmitted on a 10 Mb/s Ethernet among 96 machines of various types, 49 of which were diskless Sun workstations and 6 were Sun file servers [19], was converted into the corresponding message trace, and also decomposed into traces containing only the messages generated by a given transport-level protocol. The three protocols most represented in the trace were TCP (DARPA's Transmission Control Protocol), ND (Sun's Network Disk), and NFS (Sun's Network File System). Five such traces were thus obtained:

- ALL (the one including all message types),
- TCP (representing the type of traffic generated by machines that do paging and access files on local disks only),
- ND (the typical traffic to or from a page server),
- NFS (the traffic directed to or from a file server), and
- ND+NFS (characterizing the traffic to or from a file server that is used also for remote paging).

The ALL trace included TCP messages from and to all machines, but only the ND and NFS messages generated by the busiest file server on the monitored network. The ND trace contained the ND messages generated by that same file server, whereas the NFS trace consisted of the NFS messages transmitted by another file server, which was the one with the highest NFS traffic. The ND+NFS trace was the result of the superposition of the ND and NFS components of ALL.

The urgency of the messages in each trace was chosen considering various attributes of the message (TCP, ND, or NFS; page read or write; file read or write; request, response or acknowledgement; and so on). Other choices, dictated by, for instance, message size or message position in a sequence, were also considered, in order to investigate the impact of message urgency on ADP performance. Non-urgent messages were not assigned an individual maximum delay: the timeout parameter of ADP was used as the common maximum delay for such messages in all cases. The message arrival rate for each trace was also varied around to the one of the original Ethernet trace, while the ratios between any two interarrival times were kept constant, in order to vary the offered load (in latency measurements) or to make it so high that the sending ADP module would never remain idle (in throughput measurements).

The factors whose influence was to be studied in the experiments are displayed in Table 1. Because of the large number of runs that even a partial factorial design would have required, we decided to limit the initial phase of the study (the one reported on in this paper) to a one-factor-at-a-time design [18]; however, the ease with which our setup allows us to change the levels of most factors and the brief duration of most runs caused us to deflect in several occasions from the original design and to extend it in some of the possible directions. The limited space prevents us from presenting and discussing all of these extensions here: only those sample results that may help us provide answers to each of the questions listed at the beginning of this section will be given in the next section.

4.2. The results

Preliminary experiments were performed for both the latency and throughput measurements to determine the minimum length of each trace to be submitted to ADP that would ensure the stability of the results of each run. A number of measurements of latency and throughput with traces of increasing lengths and various levels for the factors were collected. The results that were hardest to stabilize, i.e., that required the longest traces, were those produced by the ALL trace (see Figure 3). On the basis of the results of these preliminary measurements, a trace length of 10,000 messages was chosen for all the experiments.

The main characteristics of the five traces and the values of ADP performance indices obtained with nominal levels for all the factors are shown in Table 2. The mean interarrival times reported in the table are those between the arrivals of the 10,000 messages in each trace at the input of the sending ADP module during most latency experiments (but not, of course, in throughput experiments). The corresponding offered load in kbytes/s is between 5% and 60% of the throughput, depending on the trace, and between 3% and 23% of the Ethernet's bandwidth. The ratio between the mean ADP message size and the mean message size is a measure of the amount of queueing, and depends on the urgency characteristics of each trace, but is also influenced by the value of the ADP timeout (see Table 2). Note the small message size of the TCP trace, which includes a large number of single-character messages, and the correspondingly low throughput, due to the much greater relative weight of non-data bytes transmitted, and of message handling and encryption overhead.

The values of throughput (750 kbytes/s) and urgent-message latency (11 ms) in the nominal case are quite satisfactory. For example, the throughput is about 8 times the maximum measured by Cabrera *et al.* [8] for Berkeley UNIX TCP on an Ethernet between two Sun 2 workstations with constant size messages, and by about the same factor higher than that measured for UDP in the same experimental conditions; it should be noted that these maxima correspond, in the Berkeley UNIX implementations of TCP and UDP, to 1024-byte messages, and that the mean message size in our ALL trace is 1166 bytes, whereas the mean size of ADP messages (after queueing and piggybacking) is 6247 bytes. Note also that the throughput of ADP corresponds to about 60% of the Ethernet's bandwidth. The mean latencies measured by Cabrera *et al.* for TCP and UDP were, in the case of 1024-byte packets, 11.9 ms and 8.6 ms, respectively, that is, much closer to ADP performance than the throughputs. The variability of message size slightly worsens the latency produced by ADP: an experiment with constant message sizes resulted in mean ADP latencies of 3 ms for 32 bytes and 9 ms for 8 kbytes.

In comparing Cabrera's TCP and UDP measurements with our ADP measurements, we are not suggesting that these protocols are interchangeable; they are at different levels in the network architecture, they provide very different services, (TCP's features are especially rich [25]), and Cabrera's data was certainly influenced by the characteristics of the software layer between a user process and TCP or UDP in Berkeley UNIX. We are simply verifying the reasonableness of our results. In any case, hardware encryption/decryption of ADP message headers seems not to be too slow, even though the DES chip's operation cannot be overlapped with other message processing operations. We found that the main limitations to the chip's performance are imposed by the interface rather than by the speed of encryption. In fact, data transfers in and out of the DES chip accounted for 90% of the time spent doing encryption.

Tables 3 and 4 provide some data about the impact of the scope of encryption and of queueing on performance for two of the traces, namely, ALL and ND. The drop in performance due to the encryption of the entire message (for privacy and authentication) is always substantially larger than the one due to the encryption of the trailer (for authentication only). The latter is often very small, especially when non-urgent messages are queued by ADP. When Q = ON, the utilization of the sending CPU with the ND trace only increases from 20.1% to 20.3% because of trailer encryption, and to 47% for full message encryption. When Q = OFF, the corresponding utilizations are 39.9%, 59.8%, and 90.9%. Thus, with our DES chip and its interface, the cost of privacy, especially in terms of latency, is high. The tables also show that queueing always has a very appreciable beneficial effect, especially on latency, even if there is no encryption. This result, which increases with the message arrival rate, is confirmed by the curves in Figure 4. Note that, in this figure, the arrival rate factor on the horizontal axis is a measure of the offered load. Note also that the nominal rates are marked with small squares on the curves. Table 5 shows the devastating impact of software DES encryption and decryption on throughput and even more on latency; the large difference between the latencies of the two traces can be explained with the higher arrival rate of ALL.

The influence of message urgency on performance is illustrated in Table 6, where the various levels of urgency correspond to different policies for selecting urgent messages in the ALL trace. The non-monotonic values of T and L suggests that the sequence and arrival times of urgent messages are probably at least as important as their mean rate. However, infrequent arrivals of urgent messages substantially lower the probability that they will have to wait because of ADP's still being in the process of shipping the previous ADP message; at the other extreme, the case in which all messages are urgent coincides with that of no queueing (see Table 3), which is characterized by a much lower performance.

Figures 5 and 6 show the effect of the ADP timeout parameter (i.e., the maximum delay for non-urgent messages) on throughput and latency, respectively, for some of the traces. The throughput curves are all well-behaved, in the sense that, beyond a certain value of M that is approximately the same for all traces, T becomes quite insensitive to variations of M. This is because, beyond that point, the shipping of an ADP message is determined either by the arrival of an urgent message or by the size of the queue exceeding its upper bound. On the contrary, Figure 6 suggests that different types of traffic will need different timeouts if latencies of urgent messages are to be close to their minimum values (see the ND curve); note that the behavior of the ALL trace in this respect is apparently much more heavily influenced by that of its TCP component than by those of all the other components (NFS and ND+NFS have curves similar to that of ND in Figure 6). The shape of the curves can be explained as follows. Very low timeouts are equivalent to a very large fraction of urgent messages. As the value of M increases, the beneficial effects of queueing make themselves felt. Beyond a certain value of M, the expiration of the timeout is no longer a normal cause for the shipment of ADP messages; the size of the ADP messages increases, and with it the latency of urgent messages. For traces with low arrival rates, like ND, the value of M at which this occurs is much higher than for higher-traffic traces like TCP and ALL.

All values of latency in the tables and figures presented above are averages. What type of distribution does the latency of urgent messages exhibit? Figure 7 answers this question for the ALL trace and its component traces in the case of nominal levels for all factors. The mean of the frequency distribution in the diagram is 11 ms, the median 10 ms, and the standard deviation 6.08 ms. The peaks in the global distribution correspond to the latencies of the most frequent ADP message sizes. Clearly distinguishable are those due to most of the TCP messages (around 7 ms), the shipments of one 8-kbyte page (10 ms), those of two piggybacked 8-kbyte pages (17 ms), and those of three piggybacked 8-kbyte pages (24 ms). The peaks of ND and NFS correspond to approximately the same latency values. Note that most of the ADP message types just mentioned also include some small additional piggybacked messages, and their latencies are therefore somewhat larger than they would be if these messages were not there.

We also measured the mean secure channel establishment time C, and found it to equal about 1.75 seconds. In our experiment, each host knew the public key of the other host, and the time needed to obtain that key from an appropriate server is therefore not included in this result.

5. Conclusion

We have described the security requirements of very large distributed systems and shown that the existing mechanisms for secure communication do not address these needs. A new architecture for fast secure communication in VLDSs has been introduced. The architecture of the Authenticated Datagram Protocol offers efficient end-to-end secure communication in large-scale systems by providing authentication and privacy at the level of host-to-host datagrams. ADP has numerous advantages over designs in which security mechanisms are placed at other protocol levels.

The initial phase of our study of ADP performance has shown that, even without subjecting ADP to careful tuning, the performance penalties due to authentication following the ADP approach are small if hardware encryption/decryption is used. The DES chip used in our experiments has a bandwidth comparable to that of the Ethernet. As network bandwidths increase, encryption speeds will have to be increased, but in order to do this the bottleneck in the current interface to the DES chip will have to be removed first by careful design.

The idea of classifying messages into urgent and non-urgent, and of having the latter wait in a queue until an urgent message comes or given time or space limits are reached, seems to improve the performance of ADP substantially, in particular when arrival rates are high, as we expect them to be in VLDSs. The possibility of introducing this idea into other protocols is worth exploring. The results of some of our latency experiments suggested the use of message-dependent values for the ADP timeout. Finally, the time required for the establishment of a secure channel between two hosts is relatively short, and in any case the operation is expected to be quite infrequent.

A number of investigations, some of which require a more sophisticated experimental setup, remain to be performed. These include further explorations of the ADP parameter and input spaces, and of the interactions among factors; the investigations involving faster encryption hardware (and better interfaces) as well as faster networks will be among the most interesting ones. They also include the study of the impact of such additional factors as the caching of authenticated names and the degree of physical parallelism available for the execution of ADP modules.

Acknowledgements

Numerous individuals made substantial contributions to the research effort described in this paper. Shin-Yuan Tzou designed, implemented, and tested those parts of the DASH kernel that were indispensable to ADP operation, created the higher-resolution clock used in the experiments, and made valuable comments on the manuscript. Riccardo Gusella kindly gave us his Ethernet traces and precious advice about their processing and use. Kevin Fall worked on imported software modules and participated in their integration. Sechang Oh was instrumental in procuring missing hardware, software, and information. Cherie L. Miller of AMD and Gene Banman, Bob Lyon, Marty Rattner, and Brad Taylor of Sun Microsystems gave us the prompt and effective support we needed. Brian Bershad participated in the early phases of the design of ADP, and Peter Danzig and Joan Slobin contributed to the establishment of our experimental facilities. Jean Richter prepared with an amazing combination of accuracy and speed the successive versions of the manuscript. To all of these individuals, without whose help our research would not have been possible or would have taken much longer, our most sincere thanks.

References

[1] S. G. Akl, Digital Signatures: A Tutorial Survey, *IEEE Computer*, vol. 16, n. 2, pp. 15-24, February 1983.

[2] S. G. Akl, On the Security of Compressed Encodings, *Advances in Cryptology: Proc. Crypto '83*, pp. 209-230, 1983.

[3] G. T. Almes, A. P. Black, E. D. Lazowska, and J. D. Noe, The Eden System: A Technicl Review, *IEEE Trans. on Software Engineering*, vol. SE-11, n. 1, pp. 43-59, January 1985.

[4] D. P. Anderson, D. Ferrari, P. V. Rangan, and S.-Y. Tzou, The DASH Project: Issues in the Design of Very Large Distributed Systems, Rept. No. UCB/CSD 87/338, University of California, Berkeley, January 1987.

[5] D. P. Anderson and P. V. Rangan, A Basis for Secure Communication in Large Distributed Systems, *Proc. IEEE Symp. on Security and Privacy*, Oakland, April 1987.

[6] A. D. Birrell, R. Levin, R. Needham, and M. D. Schroeder, Grapevine: An Exercise in Distributed Computing, *Comm. ACM*, vol. 25, n. 4, pp. 260-274, April 1982.

[7] A. D. Birrell, Secure Communication Using Remote Procedure Calls, *ACM Trans. on Computer Systems*, vol. 3, n. 1, pp. 1-14, February 1985.

[8] L. F. Cabrera, E. Hunter, M. Karels, and D. Mosher, A User-Process Oriented Performance Study of Ethernet Networking Under Berkeley UNIX 4.3BSD, Rept. No. UCB/CSD 84/217, University of California, Berkeley, December 1984.

[9] M. H. Cheheyl, M. Gasser, G. A. Huff, and J. K. Miller, Verifying Security, *Computing Surveys*, vol. 13, n. 3, pp. 279-339, September 1981.

[10] D. R. Cheriton, The V Kernel: A Software Base for Distributed Systems, *IEEE Software*, vol. 1, n. 2, pp. 19-42, April 1984.

[11] D. R. Cheriton, VMTP: A Transport Protocol for the Next Generation of Communication Systems, *Proc. Tenth Data Commmunications Symposium*, pp. 406-415, August 1986.

[12] D. E. Denning, Protecting Public Keys and Signature Keys, *IEEE Computer*, vol. 16, n. 2, pp. 27-35, February 1983.

[13] D. E. Denning, Digital Signatures with RSA and Other Public-Key Cryptosystems, *Comm. ACM*, vol. 27, n. 4, pp. 388-392, April 1984.

[14] D. E. Denning, Cryptographic Checksums for Multilevel Database Security, *Proc. 1984 Symp. on Security and Privacy*, pp. 52-61, May 1984.

[15] W. Diffie and M. Hellman, New Directions in Cryptography, *IEEE Trans. on Information Theory*, vol. IT-22, n. 6, pp. 644-654, November 1976.

[16] W. Diffie, Conventional Versus Public Key Cryptosystems, in G. J. Simmons, ed., *Secure Communications and Asymmetric Cryptosystems*, Westview Press, Boulder, Colorado, 1982.

[17] W. Diffie, Security for the DoD TCP, *Proc. Crypto '85*, 1985.

[18] D. Ferrari, *Computer Systems Performance Evaluation*. Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[19] R. Gusella, private communication, November 1986.

[20] W. Joy, E. Cooper, R. Fabry, S. Leffler, K. McKusick, and D. Mosher, 4.2BSD System Manual, Tech. Report, CSRG, Computer Science Division, University of California, UC Berkeley, July 1983.

[21] R. M. Metcalfe and D. R. Boggs, Ethernet: Distributed Packet Switching Local Computer Networks, *Comm. ACM*, vol. 19, n. 7, pp. 395-403, July 1976.

[22] S. J. Mullender and A. S. Tanenbaum, Protection and Resource Control in Distributed Operating Systems, *Computer Networks*, vol. 8, pp. 421-432, 1984.

[23] National Bureau of Standards, Data Encryption Standard, FIPS Publication 46, NBS, U.S. Dept. of Commerce, Washington, D.C., 1977.

[24] J. Postel, Internet Protocol, Information Sciences Institute, University of Southern California, Marina del Rey, California, RFC 791, September 1981.

[25] J. Postel, Transmission Control Protocol, Information Sciences Institute, University of Southern California, Marina del Rey, California, RFC 793, September 1981.

[26] R. L. Rivest, A. Shamir, and L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Comm. ACM*, vol. 21, n. 2, pp. 120-126, February 1978.

[27] R. D. Sansom, D. P. Julin, and R. F. Rashid, Extending a Capability Based System into a Network Environment, Technical Report, Computer Science Department, Carnegie-Mellon University, April 1986.

[28] R. E. Schantz, R. H. Thomas, and G. Bono, The Architecture of the Cronus Distributed Operating System, *Proc. 6th Int. Conf. on Distributed Computing Systems*, Cambridge, Massachusetts, pp. 250-259, May 1986.

[29] E. L. Slate and J. A. Popko, The Next Five Years in Communications, *Telecommunications*, pp. 49-60, January 1986.

[30] B. Taylor and D. Goldberg, Secure Networking in the Sun Environment, *Proc. 1986 Summer Usenix Technical Conf.*, pp. 27-36, June 1986.

[31] S.-Y. Tzou, private communication, December 1986.

Table 1. Factors and levels

FACTOR TYPE	FACTOR	SYMBOL	LEVELS
INPUT	TRACE	W	*ALL* TCP ND NFS ND+NFS
	URGENCY	U	U1: 0.08% U2: 3% *U3: 10%* U4: 19% U5: 27% of the messages are urgent(°)
	ARRIVAL RATE(∞)	R	continuous (nominal value depends on the trace, see Table 2)
ADP	SINGLE-KEY ENCRYPTION SCOPE	E	NE: no encryption *PE: partial encryption (trailer only)* FE: full encryption
	SINGLE-KEY ENCRYPTION TECHNOLOGY	Y	*HW: hardware (DES chip)* SW: software (DES routine)
	MESSAGE QUEUEING	Q	*ON: queueing* OFF: no queueing
	TIMEOUT	M	continuous (nominal value depends on the trace, see Table 2)

Nominal levels are italicized.

(°) Percentages of urgent messages are for the ALL trace.

(∞) Latency experiments only.

Table 2. Characteristics and nominal ADP performances of the five traces

Trace	Mean message size (B)	Mean interarrival time (ms)	Nominal ADP timeout (ms)	Mean ADP message size (B)	T (kB/s)	L (ms)
ALL	1166	4	40	6247	750	11
TCP	219	2	40	2104	180	8
ND	2339	11	100	11073	986	18
NFS	1897	50	100	3382	740	13
ND+NFS	1891	10	60	10332	960	11

Table 3. T and L as functions of E and Q

(ALL trace, other factors at nominal levels)

	T (kbytes/s)		L (ms)	
	Q = OFF	Q = ON	Q = OFF	Q = ON
E = NE	344	760	67	11
E = PE	328	750	101	11
E = FE	208	350	4388	620

Table 4. T and L as functions of E and Q

(ND trace, other factors at nominal levels)

	T (kbytes/s)		L (ms)	
	Q = OFF	Q = ON	Q = OFF	Q = ON
E = NE	464	991	260	18
E = PE	330	986	369	18
E = FE	279	394	694	318

Table 5. T and L versus Y

(ALL and ND traces, all other factors at nominal levels)

	T (kbytes/s)		L (ms)	
	Y = HW	Y = SW	Y = HW	Y = SW
ALL	740	50	11	3300
ND	980	97	18	192

Table 6. T and L versus U

(ALL trace, all other factors at nominal levels)

	T (kbytes/s)	L (ms)
U1	730	8
U2	600	18
U3	750	11
U4	450	27
U5	401	60

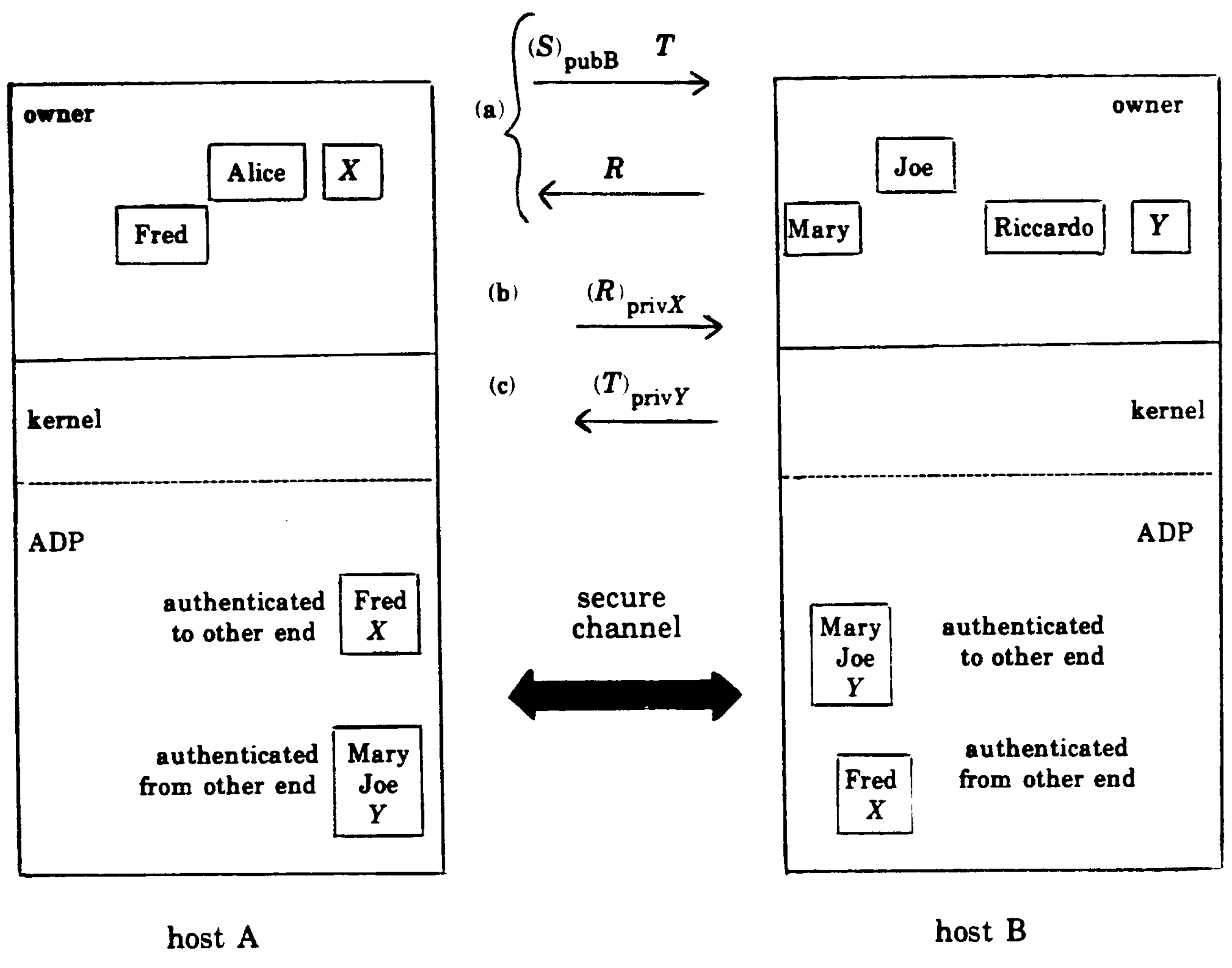


Figure 1.

A schematic diagram of ADP operation ((a) secure channel establishment; (b) authentication of owner X to B; (c) authentication of owner Y to A). Messages exchanged in (b) and (c) as well as during normal communication between X and Y are partially (or, optionally, totally) encrypted with channel key S. $(W)_z$ denotes string W encrypted with key z.

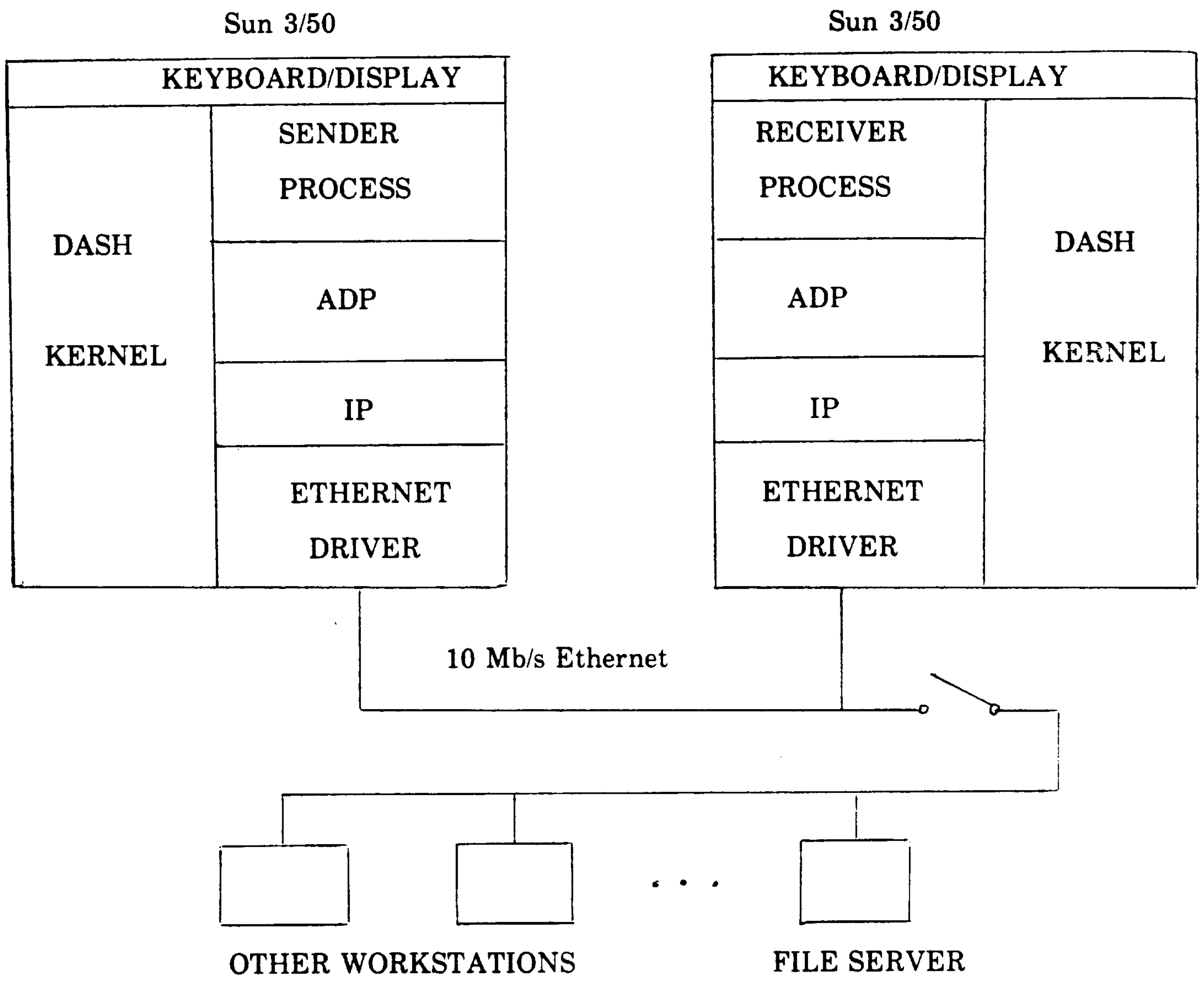


Figure 2. The experimental setup.

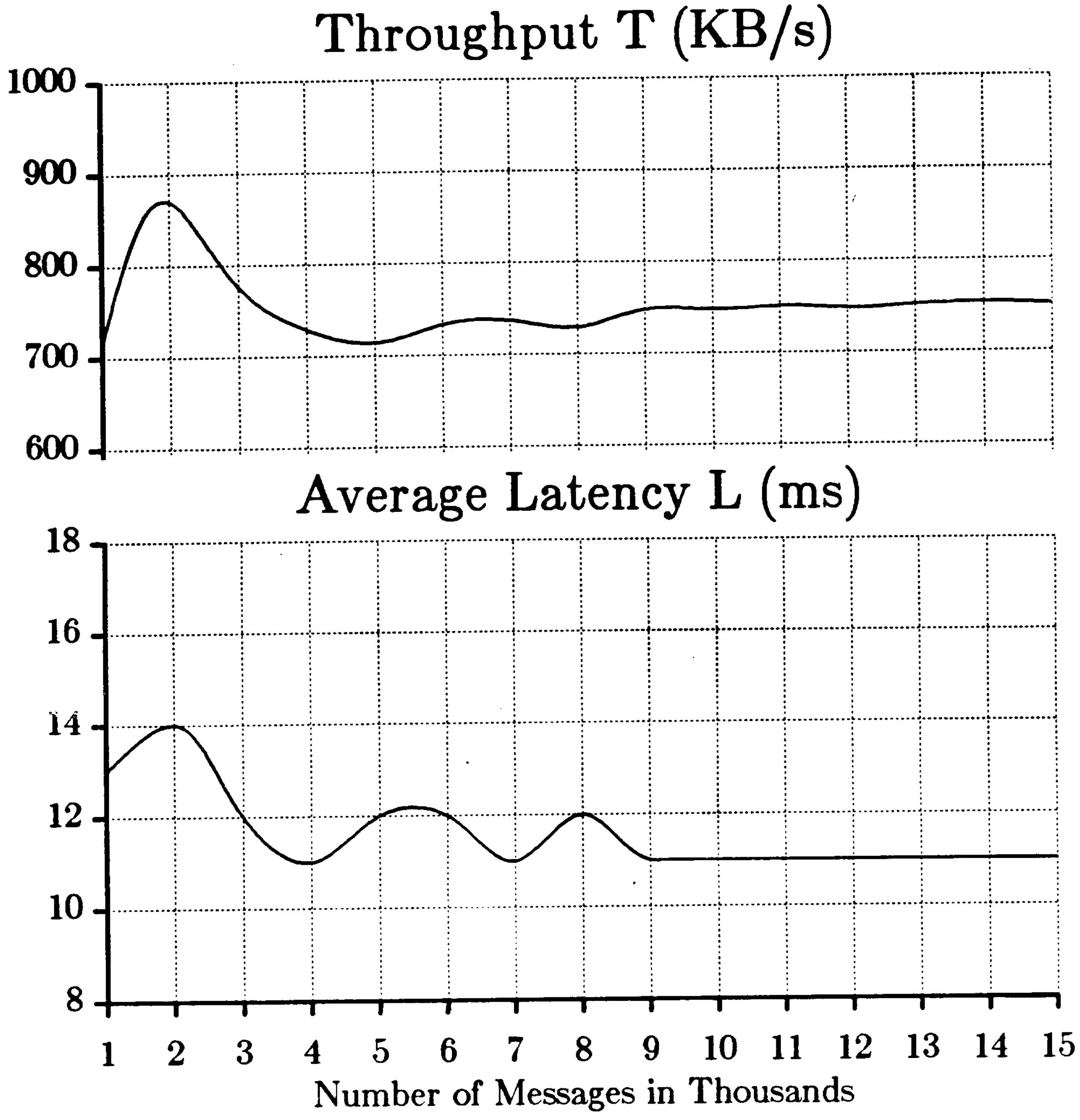


Figure 3.

Throughput and latency versus number of messages (ALL trace, nominal levels for all other factors).

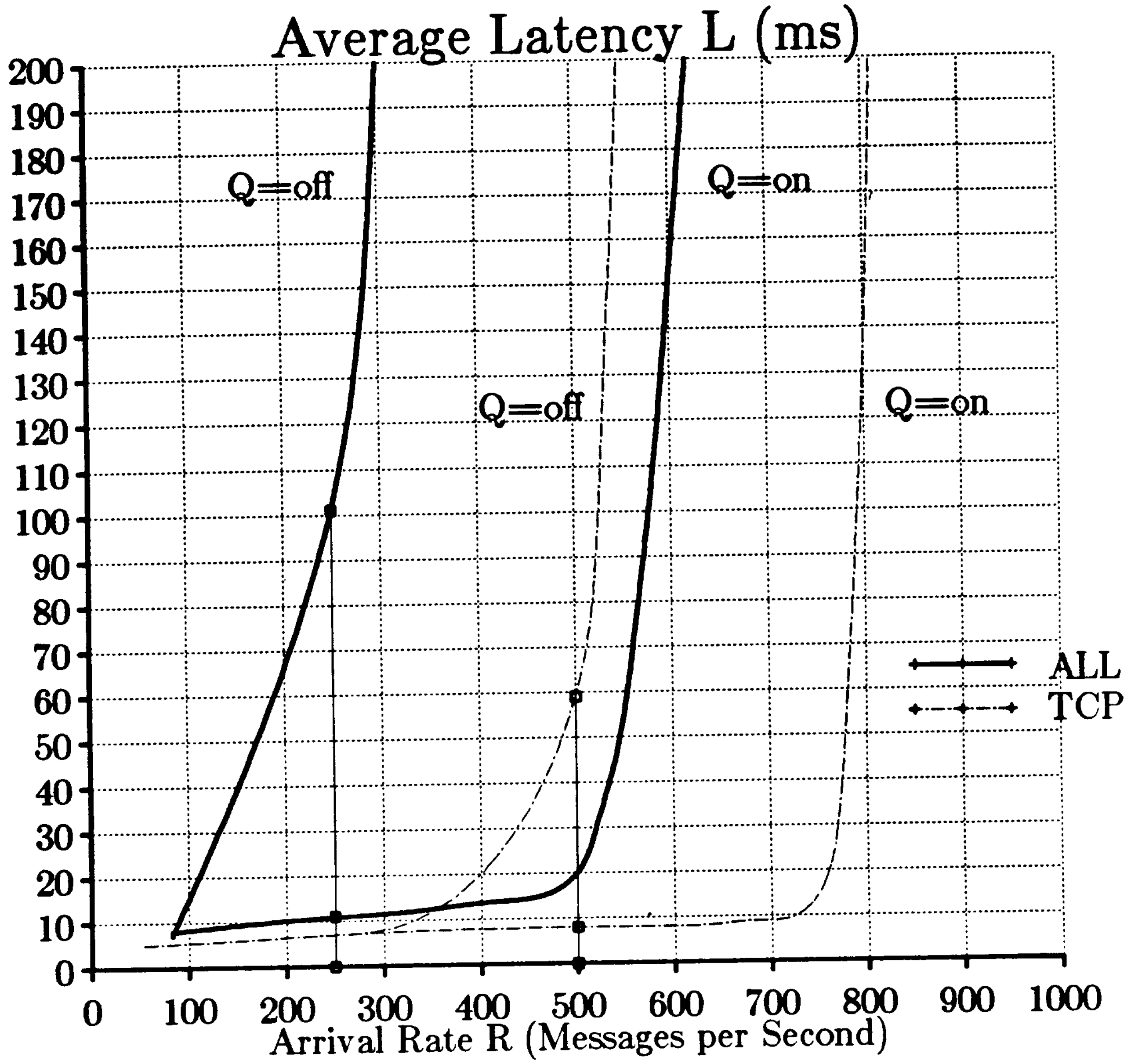


Figure 4.

L versus *R* for the two levels of *Q* and for various traces (M = 40, other factors at nominal levels).

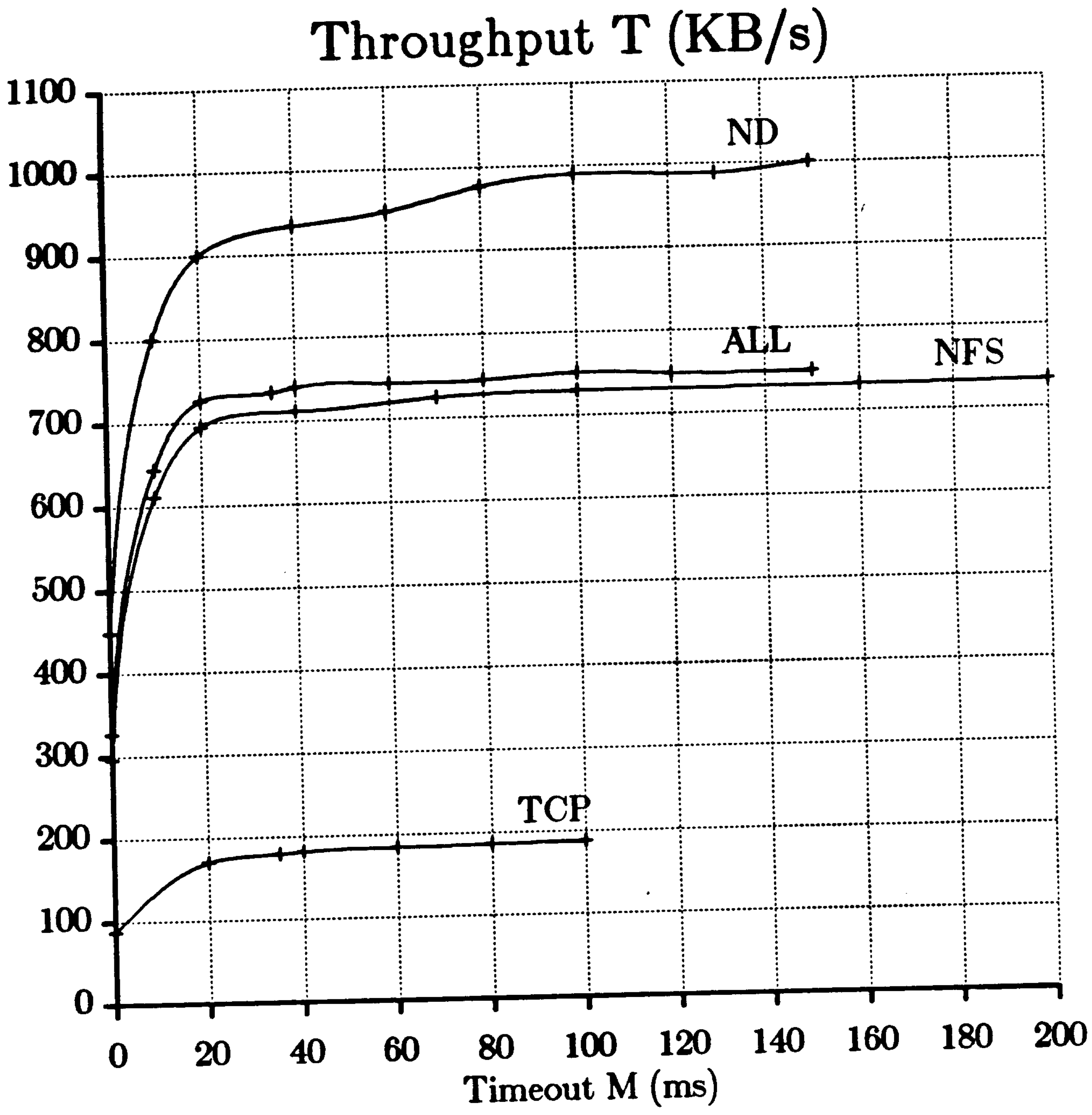


Figure 5.

T versus *M* for various traces (other factors at nominal levels)

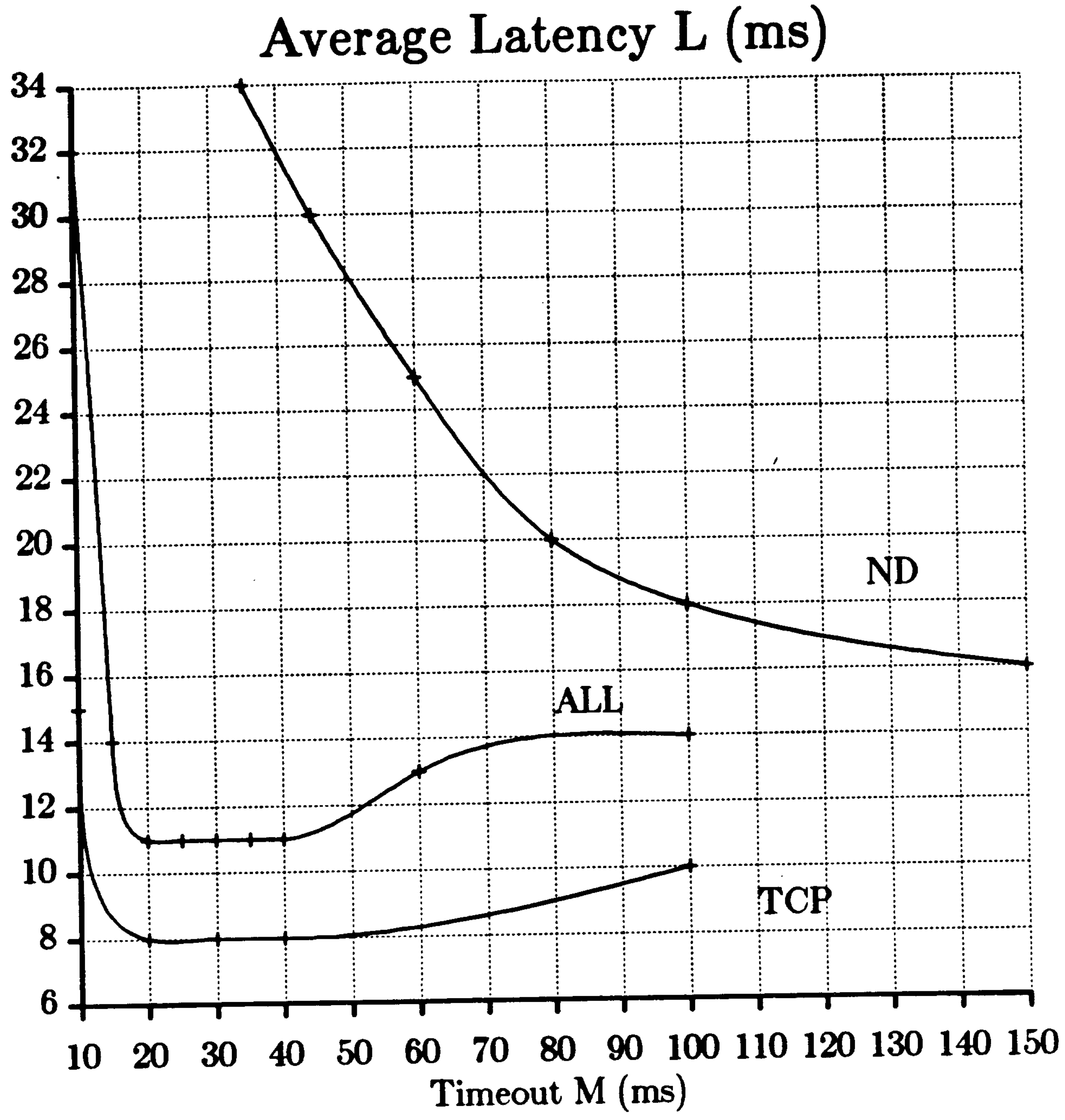


Figure 6.

L versus *M* for various traces (other factors at nominal levels)

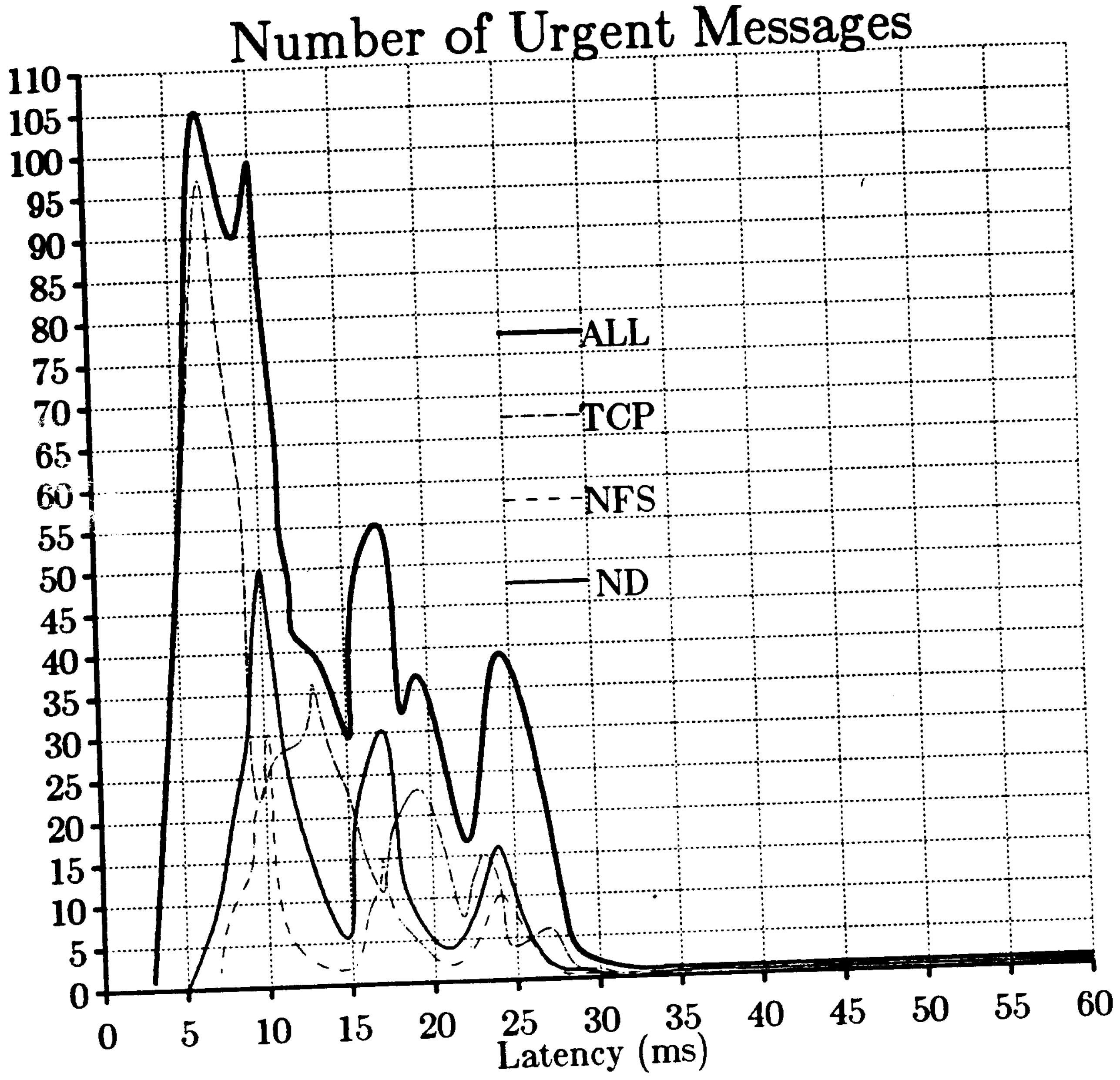


Figure 7.

Latency distributions for the ALL trace and its components