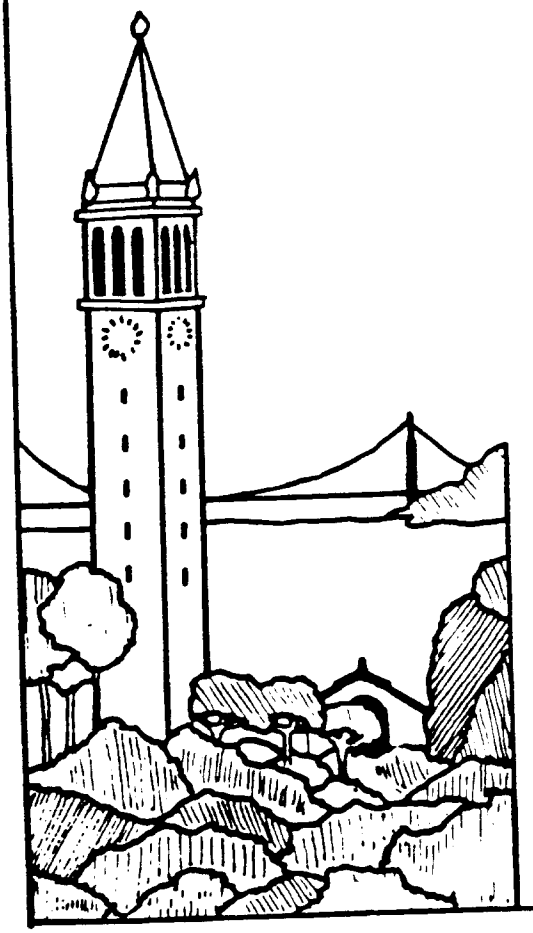
The Empirical Evaluation of a Security-Oriented Datagram Protocol

D. P. Anderson, D. Ferrari,

P. V. Rangan, and B. Sartirana



Report No. UCB/CSD 87/350

April 1987
PROGRES Report No. 87.3

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

87/350

The Empirical Evaluation of a Security-Oriented Datagram Protocol*

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

Computer Systems Research Group

Computer Science Division

Department of Electrical Engineering and Computer Sciences

University of California, Berkeley

Abstract

Performance considerations played an important role in the design of the Authenticated Datagram Protocol (ADP), a subtransport-level host-to-host datagram protocol that contains cryptographic mechanisms for end-to-end authentication and, optionally, privacy of messages. Several performance-motivated features were introduced into ADP. This paper describes the first phase of a measurement-based study of ADP intended to determine the actual effects of each of those features on the protocol's performance. The experiments were trace-driven, and took place between two workstations in a laboratory setting. The results in every case demonstrated the usefulness of the features in question, but not always for the reasons that motivated their introduction into the design.

* 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 a representing official policies, either expressed or implied, of any of the sponsoring agencies or corporations.

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

1. Introduction

Computer security is becoming more and more important in the non-military sector as distributed systems become larger and administratively more heterogeneous. Within the context of a project that is investigating the salient issues to be faced in the design of operating systems for very large distributed systems (the DASH Project [3]), we have designed a communication protocol called ADP (the *Authenticated Datagram Protocol*) that can provide end-to-end authentication and, optionally, message privacy in such systems [4].

Since security mechanisms are known to be expensive in terms of performance, considerable attention has been devoted to performance problems in designing ADP, and a number of provisions intended to reduce the cost of encryption have been incorporated into the design.

To determine whether and to what extent those provisions are effective, we have undertaken a performance evaluation study of ADP. This paper describes the first phase of the study, which has been based on a very simple experimental setup, and discusses its results. Section 2 consists of a description of the protocol. In Section 3, we present the main questions that the study must answer, and the principles that were followed in planning and carrying out its first phase. The factors considered and the instrumentation used in the experiments are described in Section 4. Section 5 presents and discusses the experimental results, and Section 6 offers concluding remarks.

2. The Protocol

ADP is a host-to-host datagram protocol that provides secure communications, i.e., authentication and, optionally, privacy of messages. It 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 [16] for a distant host, while a simpler network protocol would suffice for a host on the same LAN.

2.1. 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) [11,18] 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 a random string S, encrypted with the public key of the owner of B, and a random string T. S will be used as the secret single key of the secure channel, and T will be used in digital signatures [1,8,9] 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.

2.2. Owner authentication

ADP's security principals are called *owners*. Each owner has a unique symbolic name and a unique private-key/public-key pair. When an owner X on host A who has not sent any messages to B before wants to communicate 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 an owner on B, 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.

2.3. Messages

As explained in [4], 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 [2,10]. 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 [11,18] and single-key [15] cryptography.

As a basis for authentication in large distributed systems, public-key schemes have several advantages over single-key schemes [12], which require secure and reliable key servers. In such systems, replication is essential for performance, availability, and fault-tolerance. Key server replication increases the vulnerability of a single-key scheme to attacks on secrecy, whereas it reduces the vulnerability of public-key systems. The need for such key servers does not exist in public-key schemes, where only public keys must be distributed, a task that can be assigned to ordinary name servers. However, current public-key encryption algorithms are too slow to consider using them to encrypt any part of each message sent, whereas 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 contain 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. The Evaluation of ADP Performance

The design and the implementation of ADP were very heavily influenced by performance considerations. These considerations suggested the introduction of security mechanisms at the subtransport (i.e., host-to-host) level rather than at the transport or higher level, where end-to-end mechanisms are usually placed [20]. The combined use of public-key and single-key encryption (the former to establish secure channels over which the latter can be used) was suggested by the desire to maximize performance. The same objective inspired both the caching of authentication information, the piggybacking of non-urgent messages onto urgent ones, and the structure of the code, which was built to exploit the possible physical parallelism existing in a multiprocessor system.

An important question that was repeatedly asked while ADP was being designed was whether and to what extent each of these provisions would be effective in the performance sense. To answer this question, it was felt that the best approach would be to implement ADP and measure its performance by means of purposely designed experiments. To this effect, the question was subdivided into the following subquestions:

(a) Are the levels of performance offered by ADP acceptable for a protocol of its kind?

(b) What is the performance impact of subtransport-level rather than transport-level security mechanisms?

(c) What are the effects of encryption on performance?

(d) What is the performance impact of piggybacking?

(e) What are the performance savings due to authentication caching?

(f) What performance gains can be obtained by executing ADP in a multiprocessor environment?

(g) How do the effects of ADP's performance-oriented features interact with each other?

For these questions to be meaningful, we must define precisely the object whose performance is to be evaluated and the meaning of the term "performance" in this context. As for the object of our study, it definitely is ADP; but, if we look at Figure 2a, the question arises whether we want to evaluate the performance of the protocol in isolation, as it were, or that of ADP and of the layers underlying it in a particular protocol hierarchy. Introducing the concept of *virtual medium* as shown in Figure 2b, the question is roughly equivalent to asking whether evaluating ADP on a variety of virtual media is necessary to answer questions (a) - (g) above, or whether its evaluation on a single virtual medium will be sufficient. To our knowledge, no answer to this question can be found in the published literature. The field of protocol performance evaluation is still in its very early infancy; much work is needed to establish it on firm bases, and we plan to do some of that work soon.

For the current study, we decided to start by evaluating our prototype implementation of ADP, which runs on Sun 3/50 workstations and a virtual medium consisting of an Ethernet, Ethernet driver, and the fragmentation/reassembly routines of the DoD Internet Protocol (IP) [16], as shown in Figure 3. Since this implementation is for a local area network environment only, no other parts of IP (for instance, the routing routines) are needed.

As primary performance indices, we chose the two that are normally used to represent the productivity and the responsiveness of networks:

- *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.

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

Clearly, the values of these indices are not only influenced by ADP design and implementation decisions, but also by the bandwidth and delay characteristics of the virtual medium on which ADP runs. However, questions (a) through (g) above are all, either explicitly or implicitly, comparative in nature, and therefore can be legitimately addressed in the context of the setup in Figure 3, provided care is taken to verify that, in throughput evaluations, the bandwidth is not severely limited by that of the virtual medium, otherwise no design changes would ever make any difference in the value of T. Also, while interpreting the results, we should never forget that the magnitude of a difference in performance is likely to be quite dependent on the particular virtual medium we are considering.

Having specified the goals of the study, defined its object, and selected the indices of performance, we must decide what evaluation techniques should be used. Since ADP is part of a distributed operating systems project (DASH [3]) that requires implementation and experimentation, and since ADP had to be implemented very soon for other reasons, we chose measurement as our first technique. The amount of software involved is small enough that changing it for experimental purposes is not too difficult or error-prone. Subsequent studies, especially those involving a variety of virtual media, are expected to require the use of simulation techniques.

There are in every science two fundamental types of measurement experiments: *in vivo* and *in vitro*. For protocols, *in vivo* experiments are those conducted while the protocol is processing natural message sequences in a working real-world distributed system. Most of the protocol measurements literature describes *in vivo* studies (see, for example, [7]). *In vitro* experiments, in which a protocol is subjected to laboratory tests under artificial inputs (though these inputs may be real message traces), have been so far much less frequently reported. An example of an *in vitro* study of TCP and UDP performance can be found in [6].

Since an ADP implementation running on a real system does not exist yet, we could not perform *in vivo* experiments. However, this does not mean that, whenever such experiments are possible, they should necessarily be preferred: for example, the maximum sustainable rate T of a protocol, its performance under inputs that might occur in the future, or that occur in the present but are hard to capture, and the impact of changes to its design require *in vitro* experimentation in a laboratory setting.

The setup depicted in Figure 3 involves only two diskless workstations exchanging messages processed by ADP and not doing essentially anything else. Each of their 4 Mbyte main memories can be downloaded from the file server before the starting time of an experiment; then, the switch may be opened to eliminate the interference that would be caused by the traffic generated by the other workstations. The "DASH kernel" in the figure represents that portion of the kernel which is needed to support the software modules depicted there, the keyboard, and the display; "IP," as already mentioned above, represents only the fragmentation/reassembly routines of IP. Since only uniprocessor workstations, two user-level processes, and one secure channel were involved in the experiments performed on the setup in Figure 3, we could not obtain any answers to questions (e) and (f) above. These questions will be addressed in a future investigation based on a more complex and flexible setup.

4. The Design of the Experiments

We shall now show how our planning of the experiments was driven by the questions listed at the beginning of Section 3.

(a) *Are the levels of performance offered by ADP acceptable for a protocol of its kind?*

This question calls for the definition of a *nominal* case, in which all the levels of the factors to be defined below are "normal," and for the comparison of the performance of ADP with that produced under similar circumstances by a network protocol (in the OSI model sense) or a transport protocol. Comparing ADP and network protocol performances means adopting the viewpoint of the transport layer and is therefore more relevant, but, since only the order of magnitude of the difference is desired, even a comparison of ADP and transport protocol performances in similar contexts is usually adequate.

(b) *What is the performance impact of subtransport-level rather than transport-level security mechanisms?*

A message sequence at the input of an ADP module (see the downward arrows in Figure 2) will typically result from the mixing of messages pertaining to many distinct transport (process-to-process) connections. In the setup of Figure 3, we can simulate security mechanisms at the transport level by having ADP establish a secure channel whenever a new connection is created, and delete one whenever a connection is destroyed. Since all the other security-related operations have to be performed in both cases, no additional changes are necessary. However, the input sequence must contain information regarding connection establishment and deletion. In particular, if the heavy input traffic needed to measure throughput is obtained by increasing the arrival rate of a given message sequence, the question arises whether and how the rate of connection establishments and deletions should also increase.

Two quite different solutions we have considered are: (i) a constant establishment/deletion rate, and (ii) an establishment/deletion rate linearly increasing with the message arrival rate. The latter corresponds to a constant average frequency of messages generated by each process, the former to a linearly increasing frequency of such messages. Thus, we introduce an experimental factor called *Layer* (to be denoted by the symbol Z), whose levels are SUB (subtransport) and TR (transport). For the TR level, when necessary, we consider the two sublevels CON (constant) and LIN (linear).

(c) *What are the effects of encryption on performance?*

The use of encryption in ADP has three major aspects: first, the choice of the *Scheme* (S), with the two levels PSK (public- and single-key) and PKO (public-key only); second, the *Scope* (E), with its three levels NE (no encryption), PE (partial encryption), and FE (full encryption); and third, the *Technology* (Y), with the two levels HW (hardware single-key encryption) and SW (software single-key encryption).

As far as the S factor is concerned, we found the elapsed time difference between public-key encryption software and single-key encryption hardware (the Zilog Z8068 DES chip) so large that we felt the measurement of performance in the PKO case to be unnecessary, and we just limited ourselves to estimating the results for this case. Level PE of factor E corresponds to the encryption of a sequence number for each ADP message, which is sufficient for authentication in a LAN environment [4]; level FE ensures message privacy as well as authenticity. Factor Y only refers to single-key encryption technology, since public-key encryption hardware is not yet available. Other interesting questions in this area, such as those having to do with the effects of alternative encryption algorithms and chips, or of chip architectures, arise, but so far we have not tried to enable our experimental setup to address them.

(d) *What is the performance impact of piggybacking?*

The *Piggybacking* or *Queueing* factor (Q) has only two possible levels: ON and OFF. However, there are two additional parameters in the current implementation of ADP that

can be regarded as factors related to piggybacking: the timeout and the maximum queue size. To reduce the number of factors, and since these two parameters have similar effects on the performance of piggybacking, we decided to consider only one of them, the *Timeout* (*M*), as a factor, and to fix the value of the other one (which was set equal to 64 Kbytes). *M* has an infinite number of levels, since its value is a positive real number. Another possible factor, the queue service discipline, has been considered fixed in the first phase of our study.

(g) *How do the effects of ADP's performance-oriented features interact with each other?*

The number of binary, ternary, . . . interactions among factors is extremely large, even if we were to restrict ourselves to the 6 primary factors introduced so far. That of the interesting ones is smaller, and we examined almost all of them, but still quite large. Due to shortage of space, only a sample of the results will be given in the next section.

A summary of the primary factors and their levels is displayed in Table 1.

Since we decided that the system configuration should be fixed throughout this study, the only secondary factor is the input sequence of messages. Because of the influences on ADP performance of message sizes and arrival times, which are inherent in the design of the protocol, we could not use a synthetic input such as those of some previous studies, where all messages have the same size and arrive at regular intervals. Thus, it was decided to run trace-driven experiments.

A complete DASH system using ADP 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 such a system. Indeed, we conjecture that remote resource sharing will be substantial in a large distributed system, and that the sizes of the messages exchanged for such sharing 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 [14], 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 (DoD'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 and of the message sequence. The five policies we experimented with are those summarized in Table 2. 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. 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), while the ratios between any two interarrival times were kept constant, the message arrival rate for each trace was varied around the one of the original Ethernet trace.

When the number of messages in a trace is too small, the edge effects [13] may influence the values of the performance indices, which are averages (L is averaged over the messages, T is a time average). Preliminary experiments were performed 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 L and T measurements 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 4). On the basis of the results of these preliminary measurements, a trace length of 10,000 messages was chosen for all the experiments.

Table 2 summarizes the secondary factors we chose to represent the input and their respective levels. Because of the very large number of runs that a factorial experiment would have required, we performed only a partial exploration of the interactions to which question (g) refers. In particular, we did not systematically investigate any of the interactions involving the secondary factors.

The instrumentation required by the experiments was indeed minimal, as only times had to be measured. And yet, the resolution of the clocks available on the Sun 3/50's (10 ms) was not sufficient for latency measurements. Therefore, we used the clock of one of the serial communication controllers to interrupt the CPU with the constant period of 0.2 ms; this method allowed us to measure times with a 0.2 ms resolution at the cost of only 2% of the total CPU power [21].

Figure 2 (b) indicates the times ta, tb, tc, and td corresponding to the times when a message crosses the boundaries a, b, c, and d, respectively. Synchronization of the clocks on the two Sun 3/50's was difficult to achieve without much overhead. Thus, the latency of a message could not be calculated as the difference of times td and ta. Instead, the latency was calculated as follows: if we denote by t21 the time taken by a message to travel from b to c and back from c to b, then t21 = t2 - t1, where t1 is the time the message left b and t2 is the time it came back to b, both times measured on host A. These times can be measured offline before or after the experiment for each message size in the trace. Since the time a message takes from b to c is t21/2, and there are no collisions on the network during our experiments, the latency is equal to (tb - tc) + (t21/2) + (td - tc). Here each time difference is between times measured on the same machine, and thus does not require the synchronization of the clocks of the two machines.

5. The Results and Their Interpretation

We shall now present, among the results of the many experiments to which ADP was subjected in our laboratory, only those directly relevant to questions (a) - (d) and (g) listed in Section 3 above. The results will be interpreted for the purpose of obtaining answers to those questions.

(a) *Are the levels of performance offered by ADP acceptable for a protocol of its kind?*

Table 3 shows some of the basic characteristics of the five traces, and the values of throughput and latency measured for each of them in the nominal case (i.e., when all factors except W are at their nominal levels). 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 3). Note the small message size of the TCP trace, which includes a large number of single-character messages, and the correspondingly low throughput, certainly attributable to the much greater relative weight of non-data bytes transmitted, message handling overhead, and encryption overhead.

The values of throughput (750 kbytes/s) and urgent-message latency (11 ms) for the ALL trace in the nominal case are quite satisfactory. For example, the throughput is about 8 times the maximum measured by Cabrera *et al.* [6] 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 in an otherwise identical environment resulted in mean ADP latencies of 3 ms for a size of 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 [17]), 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. Unfortunately, we do not know of any similar measurement experiments performed on IP, whose comparison with ADP would be more meaningful. An alternative approach that would be very interesting too is the comparison between the performance of TCP or UDP running on top of ADP and Cabrera's results, but ADP has not been interfaced to TCP or UDP yet. The values of latency in Table 3 are averages. What type of distribution does the latency of urgent messages exhibit? Figure 5 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.

In summary, the performance of ADP seems to be quite acceptable, and the penalties of encryption do not seem to be excessive. This is the case in spite of the inefficient design of the DES chip's interface, which prevents its operation from being overlapped with other message processing operations [5]. Indeed because of the design of the chip's interface, the encryption bandwidth goes down from 3.24 Mbytes/s (which is that of the encryption algorithm) to 400 Kbytes/s.

(b) *What is the performance impact of subtransport-level rather than transport-level security mechanisms?*

The results most relevant to this question are displayed in Table 4. The TCP and NFS traces were chosen for these experiments as they represent the two cases of transport-level security mechanisms in a stream-oriented and RPC protocol, respectively. Since

piggybacking would be ineffective in the transport-level case, as the arrival rate for each connection would be too low, these experiments were run with no piggybacking (Q = OFF). The mean inter-arrival times were 20 ms for TCP and 150 ms for NFS trace (this was because without piggybacking, nominal arrival rates result in very high latency values). All other factors were at their nominal levels. From the results in the table, we conclude that the performance gains of subtransport-level security over bothh instances of transport-level security are substantial, even in the case of constant establishment/deletion rate, which is the more favorable for the throughput in the transport-level security case. The inferior performance of transport-level approaches is to be attributed primarily to the higher number of channel establishment/deletion operations that these approaches require with respect to the case of ADP. Establishment/deletion operations are quite time-consuming: on the average, our measurements show that each costs about 1.75 s.

In the transport level experiments, the TCP trace had 76 new connection establishments and the NFS trace had 41 new RPC transactions from new processes, both in a span of 10,000 messages. However, the actual number of secure channel establishments in the latency and the LIN case of throughput experiments were slightly less than 76 and 41 in the TCP and NFS cases, respectively. This was because of corrections we introduced to remove edge effects. The CON case of throughput experiments had 7 new connections in the case of TCP and 4 new transactions from new processes in the case of NFS.

(c) *What are the effects of encryption on performance?*

Of the three aspects of this question mentioned in Section 4, the one relating to the encryption scheme can be addressed by estimating bounds for T and L assuming that all factors except S are at their nominal level and all encryption operations are public-key ones (S = PKO). These bounds are $L \geq 875$ ms, and $T \leq 7.14$ kBytes/s, respectively.

The effects of encryption scope E and technology Y on performance are presented in Tables 5 and 6, respectively. 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 partial encryption (for authentication only). The latter is often very small, but with our DES chip and its interface, the cost of privacy, especially in terms of latency, is high. Indeed, encryption becomes the bottleneck when the entire message is encrypted, as can be seen in Table 5, where the throughputs for E = FE equal the DES chip's speed. Table 6 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 for Y = SW can be explained with the higher arrival rate of ALL, which is responsible for much longer message waiting times within the sending ADP module.

(d) *What is the performance impact of piggybacking?*

The effect on L of factor Q can best be seen in the diagram of Figure 6, where the latency curves for the two levels of Q and for traces ALL and TCP are shown as functions of the arrival rate R. The benefits of piggybacking grow very rapidly with R. When R is below a minimum threshold, piggybacking does not cause any latency improvement. Note that, in Figure 6, nominal rates are marked with small squares on the curves. The effects on T can be summarized by, for instance, the following figures: the value of T for the ALL trace decreases from 750 to 328 kB/s, and that for the ND trace decreases fromm 986 to 330 kB/s (see Table 8).

Figures 7 and 8 depict the impact of the timeout M (i.e., the maximum delay for non-urgent messages) on T and L for various 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 8 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 8). 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.

How sensitive is the performance of ADP to the urgency characteristics of the message sequence when Q = ON? The answer resulting from consideration of the figures in Table 7 is that it may be quite sensitive, and that there seems to be no simle relationship between ADP's performance and the percentage of urgent messages reported for the ALL trace in the column labeled "%". The arrival times of the urgent messages and their positions in the message sequence appear to be more important than their number.

(g) *How do the effects of ADP's performance-oriented features interact with each other?*

Providing even a summary of our study of interactions would require much more space than is available in this paper. We shall therefore discuss the effects of only one of the most interesting binary interactions: that between factors E and Q. As was mentioned in Section 2.3, piggybacking was introduced to decrease the performance penalties of encryption. Above in this section, we saw that piggybacking has a very substantial positive influence on performance. Is this improvement due entirely or predominantly to the reduction in the number of DES encryption/decryption operations (for the ALL trace, from 10,000 to 1,866)? The answer can be found in Table 8, which shows how E and Q interact with each other. Piggybacking improves latency and throughput by factors of 6 and 2.2, respectively, even without encryption. With partial encryption, the improvement factors are about 9.2 and 2.3, respectively; the increases in the improvements must be attributed primarily to savings in the encryption costs. In the full-encryption case, the improvement factors decrease to 7 and 1.7; the improvement for L is close to that measured in the NE case, as it must be, since the entire message sequence is encrypted at both levels of Q, and piggybacking does not provide any encryption savings; and the lower improvement for T can probably be explained by observing that for E = FE and Q = ON the DES chip is likely to be the bottleneck of the whole experimental system. To what phenomenon can we then attribute the greater part of the improvement due to piggybacking? For latency, the most plausible explanation is that, with Q = ON an arriving urgent message is less likely to find the sending ADP module busy than with Q = OFF, as most non-urgent messages are immediately queued rather than processed. For throughput, it is well-known that in the Ethernet it can be increased by shipping longer and less numerous messages [19], which is what piggybacking essentially does.

6. Conclusion

The impact on performance of several design decisions dictated by performance considerations during the development of the Authenticated Datagram Protocol has been evaluated by laboratory measurement experiments. All decisions have turned out to have been useful in enhancing the throughput and reducing the latency of the protocol, though not always for the reasons they had originally been made (e.g., in the case of piggybacking).

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, when arrival rates are sufficiently high, as we expect them to be in most large distributed systems in the future, at least in the case of multiuser machines. The possibility of introducing this idea into other protocols even not containing any security mechanisms is worth exploring. The results of some of our latency experiments suggested the use of message-dependent values for the ADP timeout.

A number of investigations, which require a more sophisticated experimental setup, remain to be performed. These include the study of the effects of faster encryption hardware (and better interfaces) as well as faster networks. 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] 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.

[4] 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.

[5] D. P. Anderson and P. V. Rangan, High-performance Interface Architecture for Cryptographic Hardware, Proceedings of Eurocrypt '87, Amsterdam, April 13 - 17, 1987.

[6] 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.

[7] D. R. Cheriton and C. L. Williamson, Network Measurements of the VMTP Request-Response Protocol, *Proc. 1987 ACM SIGMETRICS Conf. on Meaurement and Modeling of Computer Systems*, May 1987.

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

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

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

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

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

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

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

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

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

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

[18] 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.

[19] J. R. Shoch and J. A. Hupp, Measured Performance of an Ethernet Local Network, *Comm. ACM, 23*, no. 12, pp. 711-721, December 1980.

[20] V. L. Voydock and S. T. Kent, Security Mechanisms in High-Level Network Protocols, *Computing Surveys, 15*, no. 2, pp. 135-171, June 1983.

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

Table 1. Primary factors

Factor	Symbol	Levels(°)
Layer	*Z*	*SUB*: subtransport TR: transport connection est./del. rate: CON: constant LIN: linear with arrival rate
Encryption Scheme	*S*	*PSK*: public- and single-key PKO: public-key only
Encryption Scope	*E*	NE: no encryption *PE*: partial encryption (sequence no.) FE: full encryption
Encryption Technology	*Y*	*HW*: hardware (DES chip) SW: software (DES routine)
Piggybacking	*Q*	*ON*: queueing of non-urgent messages OFF: no queueing
Timeout	*M*	real positive values (the nominal value depends on the trace, see Table 3)

(°) Nominal levels are in italics.

Table 2. Secondary factors

Factor	Symbol	Levels (°)
Trace	*W*	*ALL* TCP ND NFS ND+NFS
Arrival Rate (°°)	*R*	real positive values (the nominal value depends on the trace, see Table 3)
Urgency	*U*	U1: Messages with size > K urgent, others non-urgent (K given) U2: If time to next timeout expiration > M/2 and message not low-priority, then message urgent *U3*: If time to arrival of next message > A, and message not low-priority, then message urgent (A given) U4: If message size < 1 Kbytes, and previous message size > 1 Kbytes, then message urgent U5: Every n-th message urgent (n given)

(°) Nominal levels are in italics.

(°°) Latency experiments only.

Table 3. 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 4

Performance comparison of subtransport-level and transport-level security mechanisms (TCP and NFS traces, Q = OFF)

W	L (ms)		T (kB/s)		
	Z		Z		
	SUB	TR	SUB	TR/CON	TR/LIN
TCP	6	30	90	76	18
NFS	8	42	325	305	152

Table 5

Performance effects of encryption scope E

(nominal case)

W	L (ms)			T (kB/s)		
	E			E		
	NE	PE	FE	NE	PE	FE
ALL	11	11	620	760	750	350
ND	18	18	318	991	986	394

Table 6

Performance effects of encryption technology Y

(nominal case)

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

Table 7

Performance effects of message urgency U

(ALL trace, nominal case)

U	%	L (ms)	T (kB/s)
U1	0.08	8	730
U2	3	18	600
U3	10	11	750
U4	19	27	450
U5	27	60	401
*	100	101	328

* case corresponding to Q = OFF

Table 8

Performance effects of the interaction between E and Q

(ALL trace, nominal case)

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

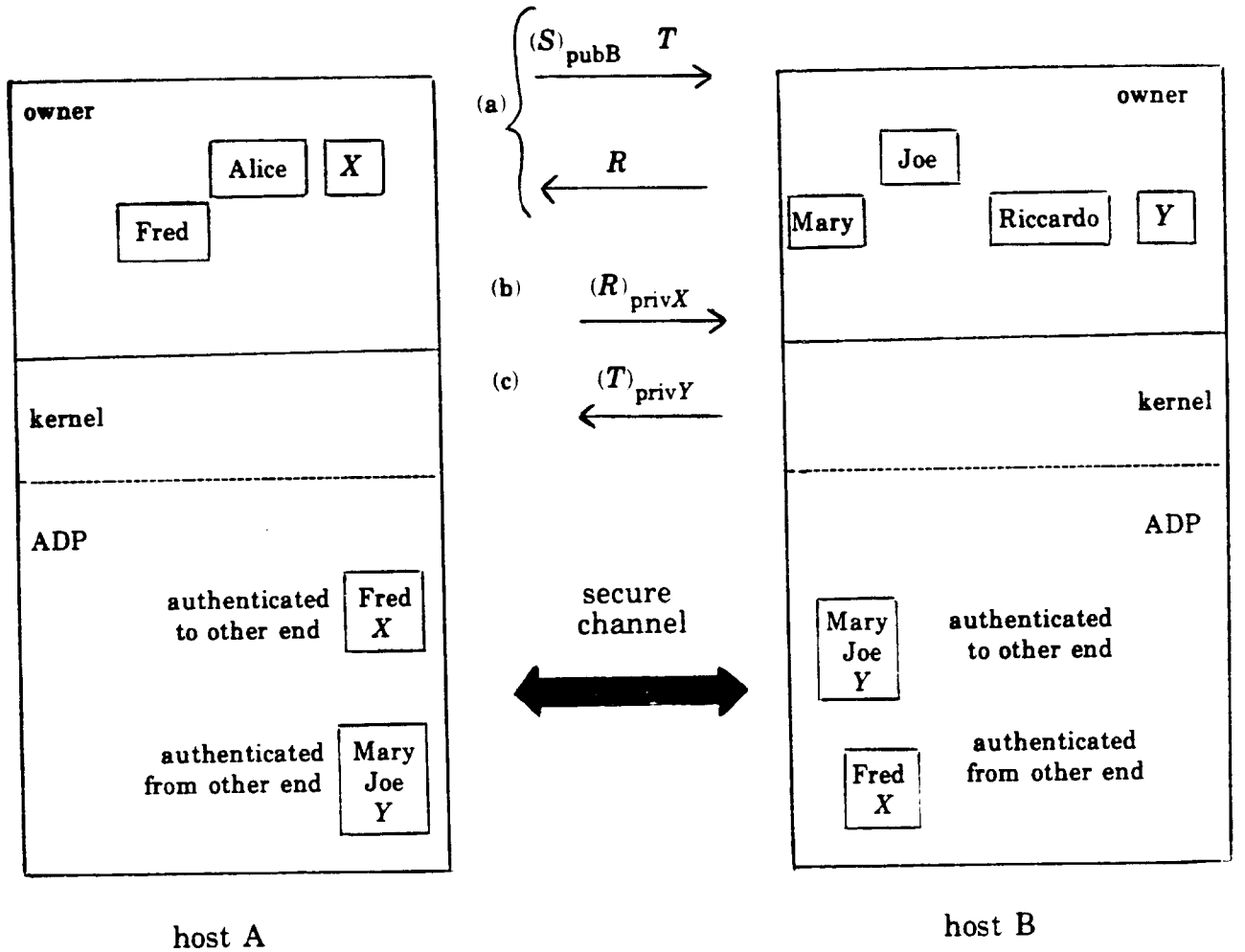
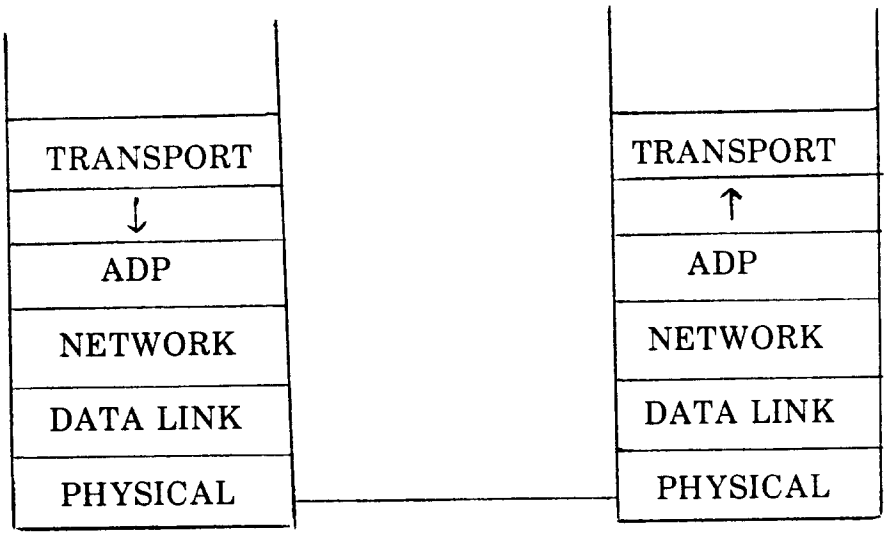
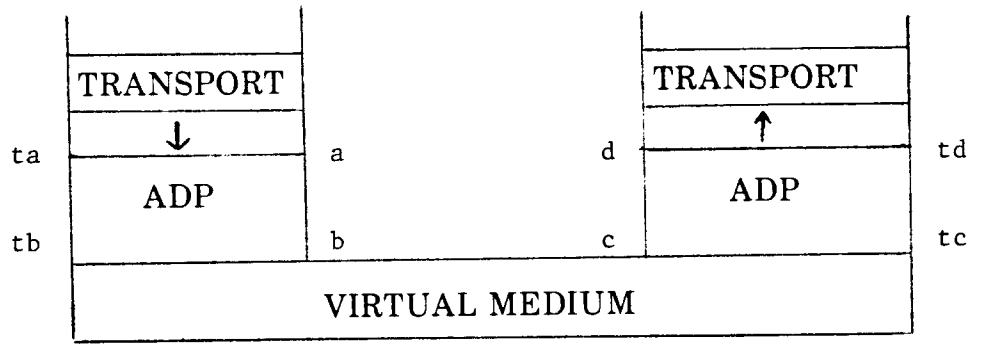


Figure 1.

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.



(a)



(b)

Figure 2. (a) The position of ADP in the OSI model of network architecture; (b) The *virtual medium* on which ADP runs.

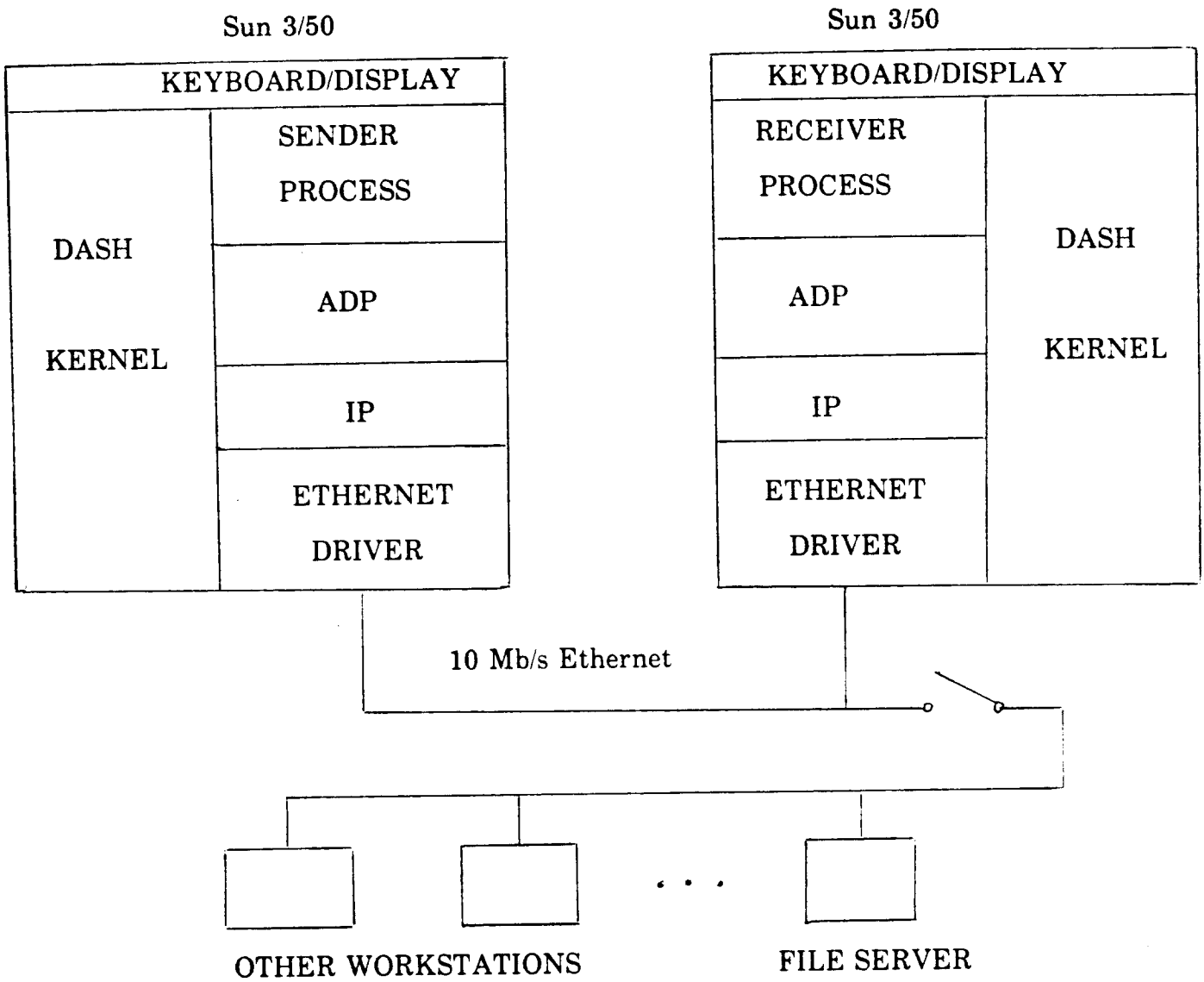


Figure 3. The experimental setup.

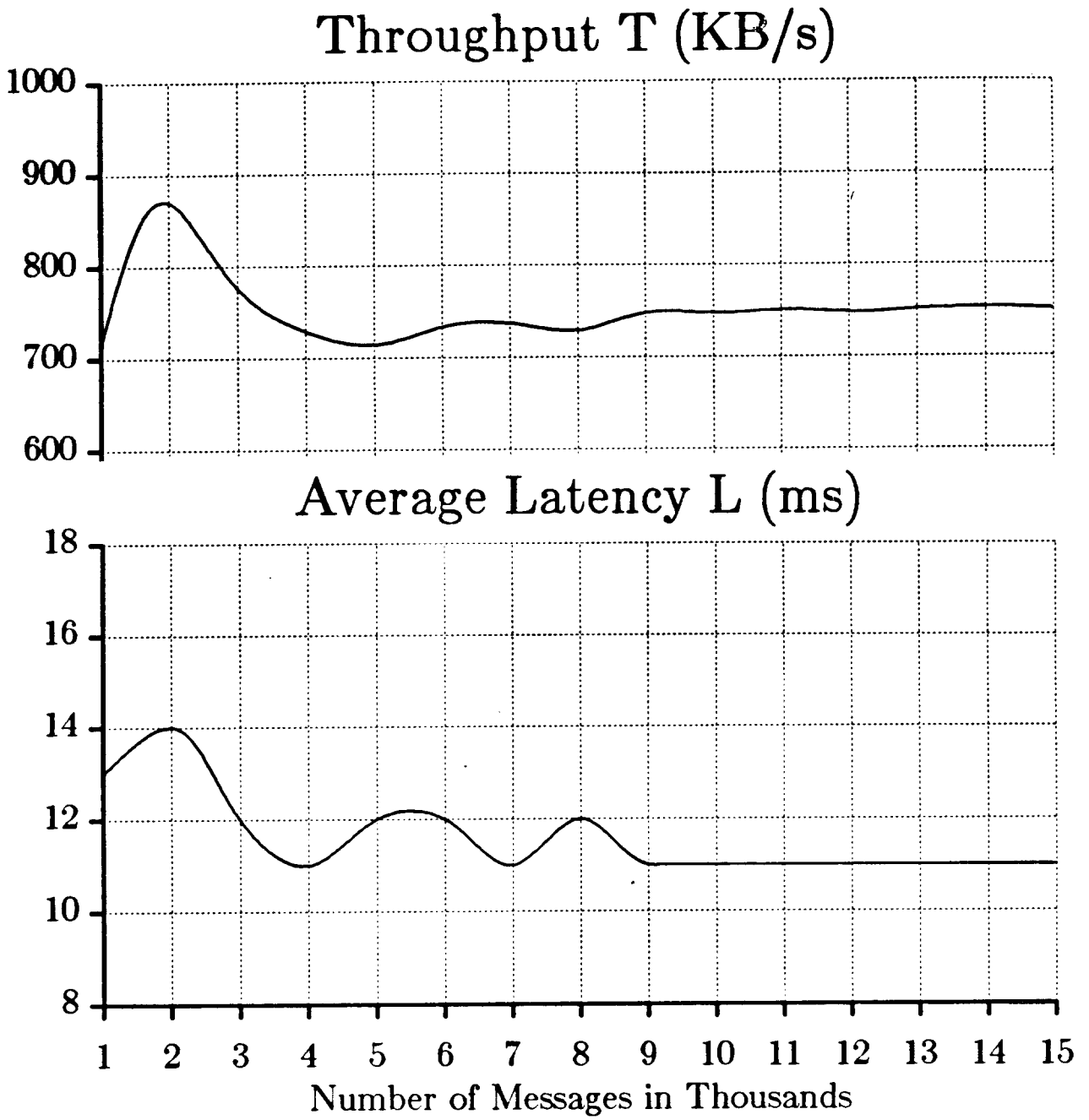


Figure 4.

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

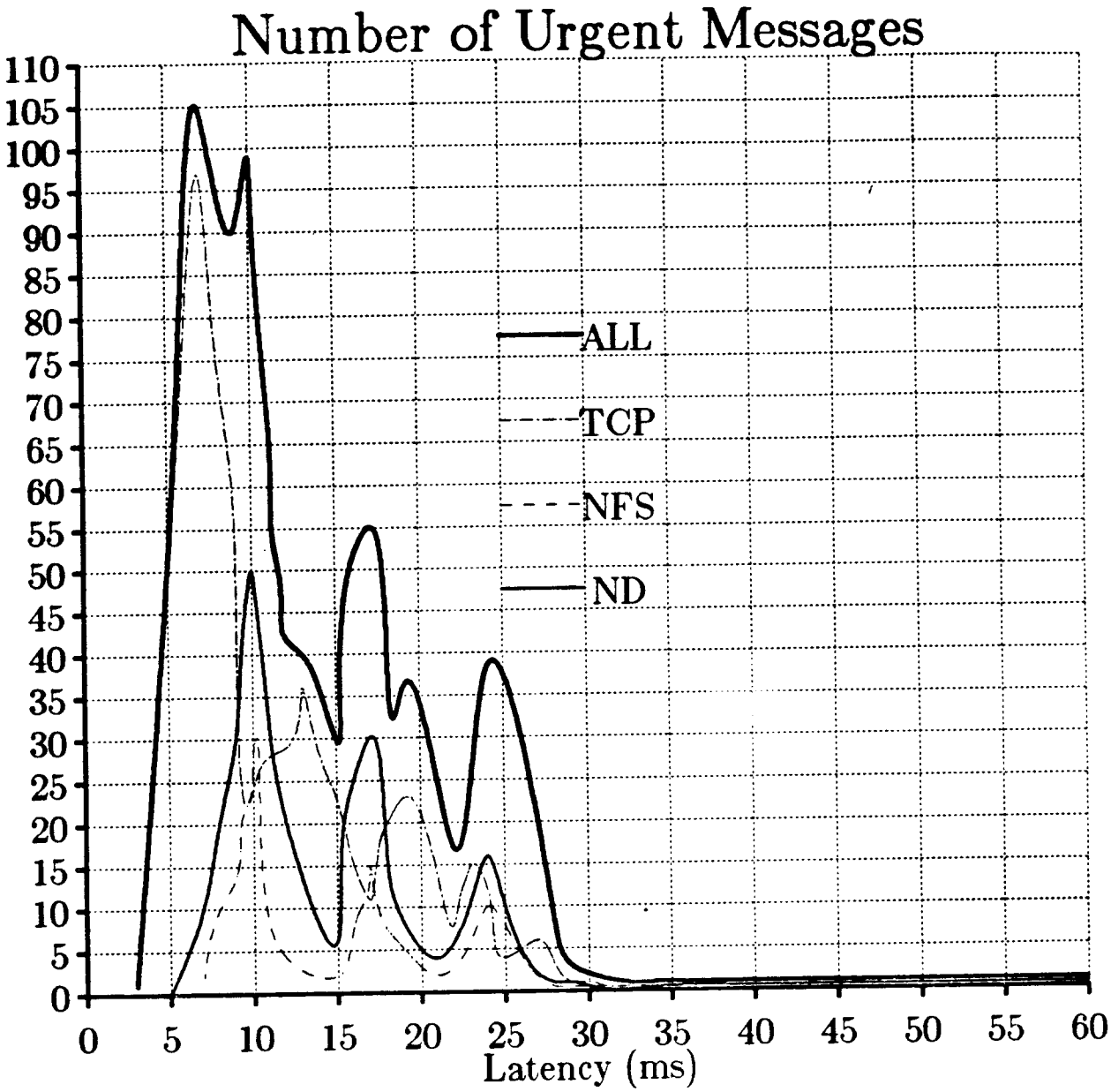


Figure 5.

Latency distributions for the ALL trace and its components

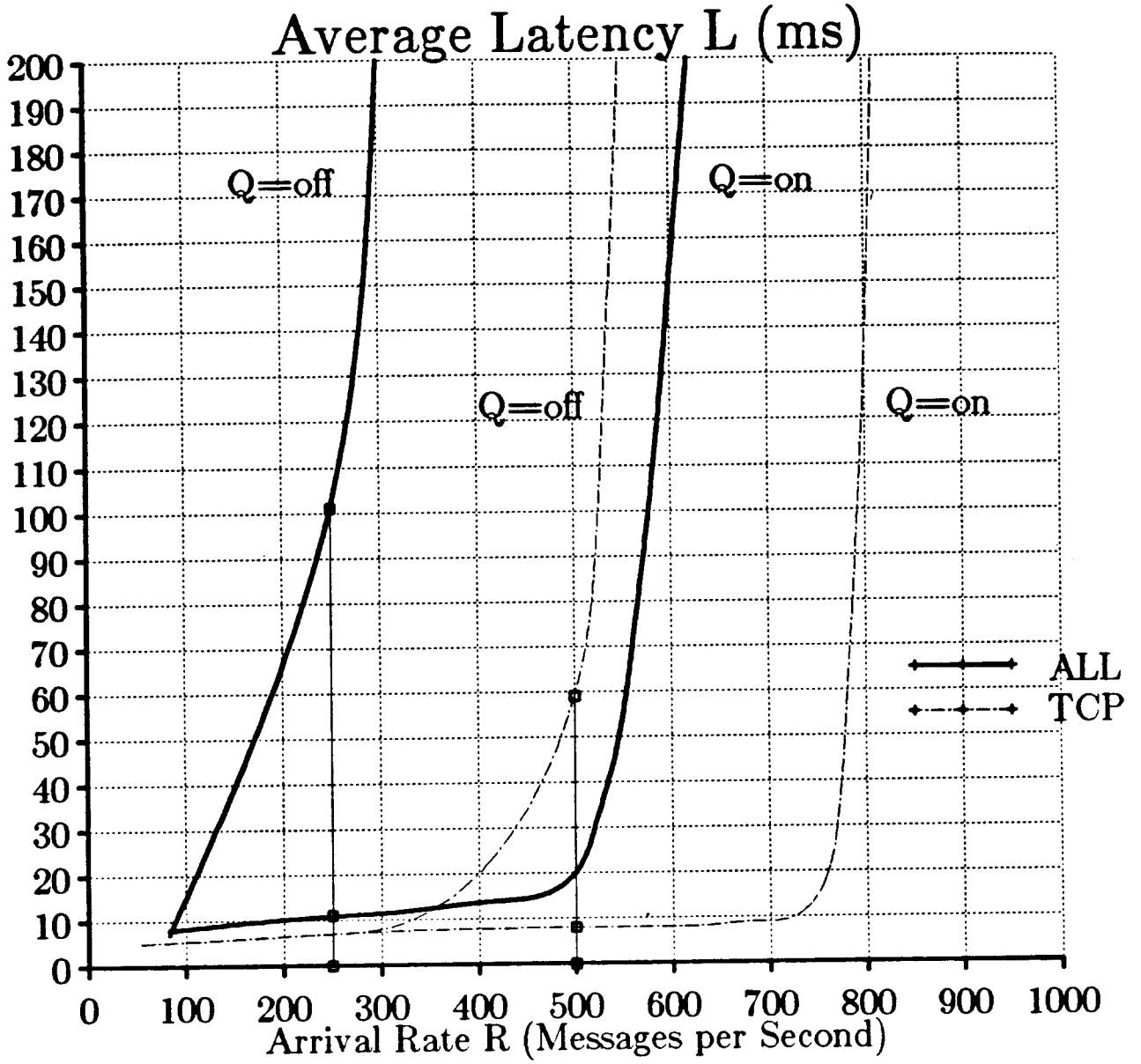


Figure 6.

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

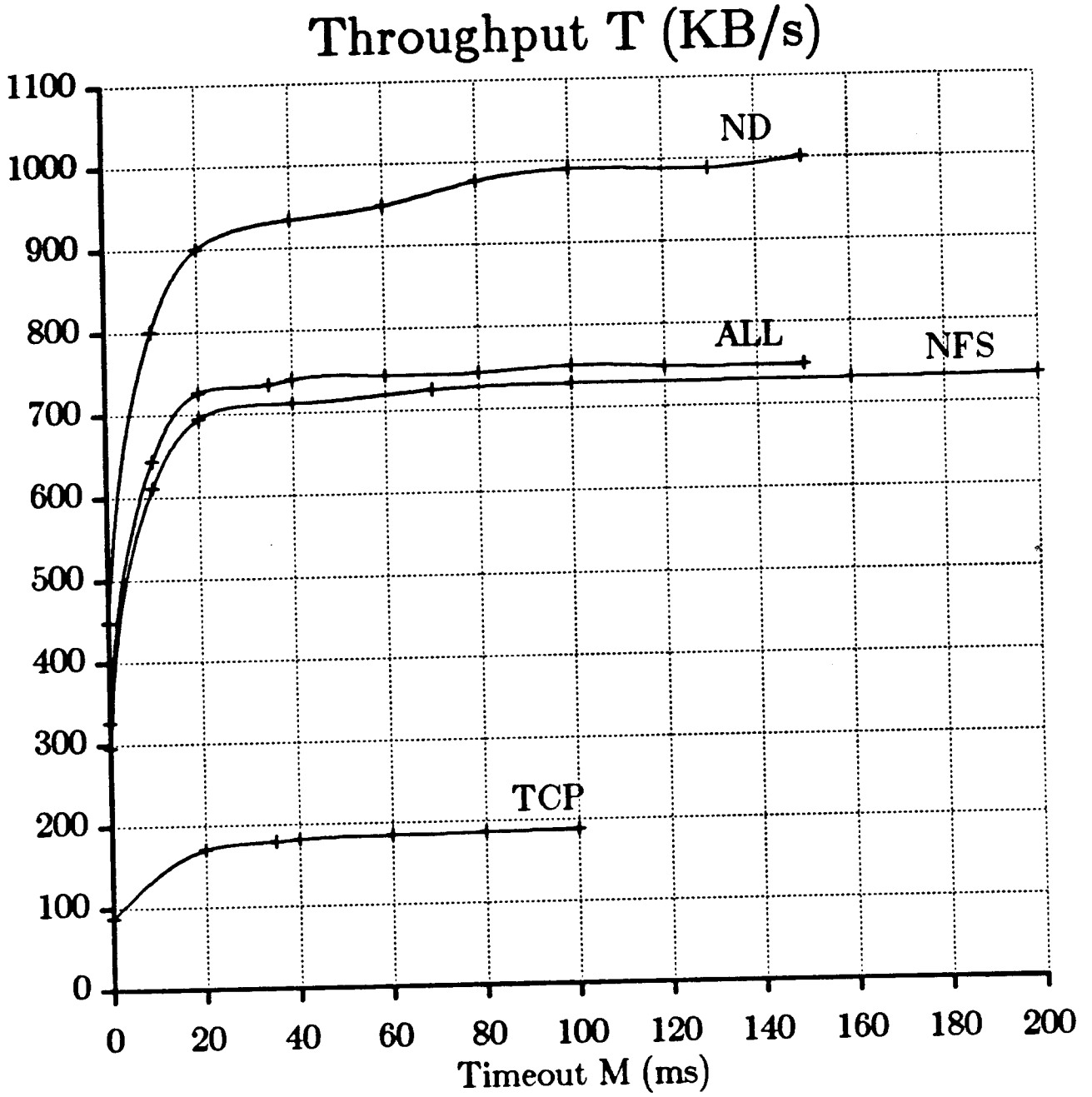


Figure 7.

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

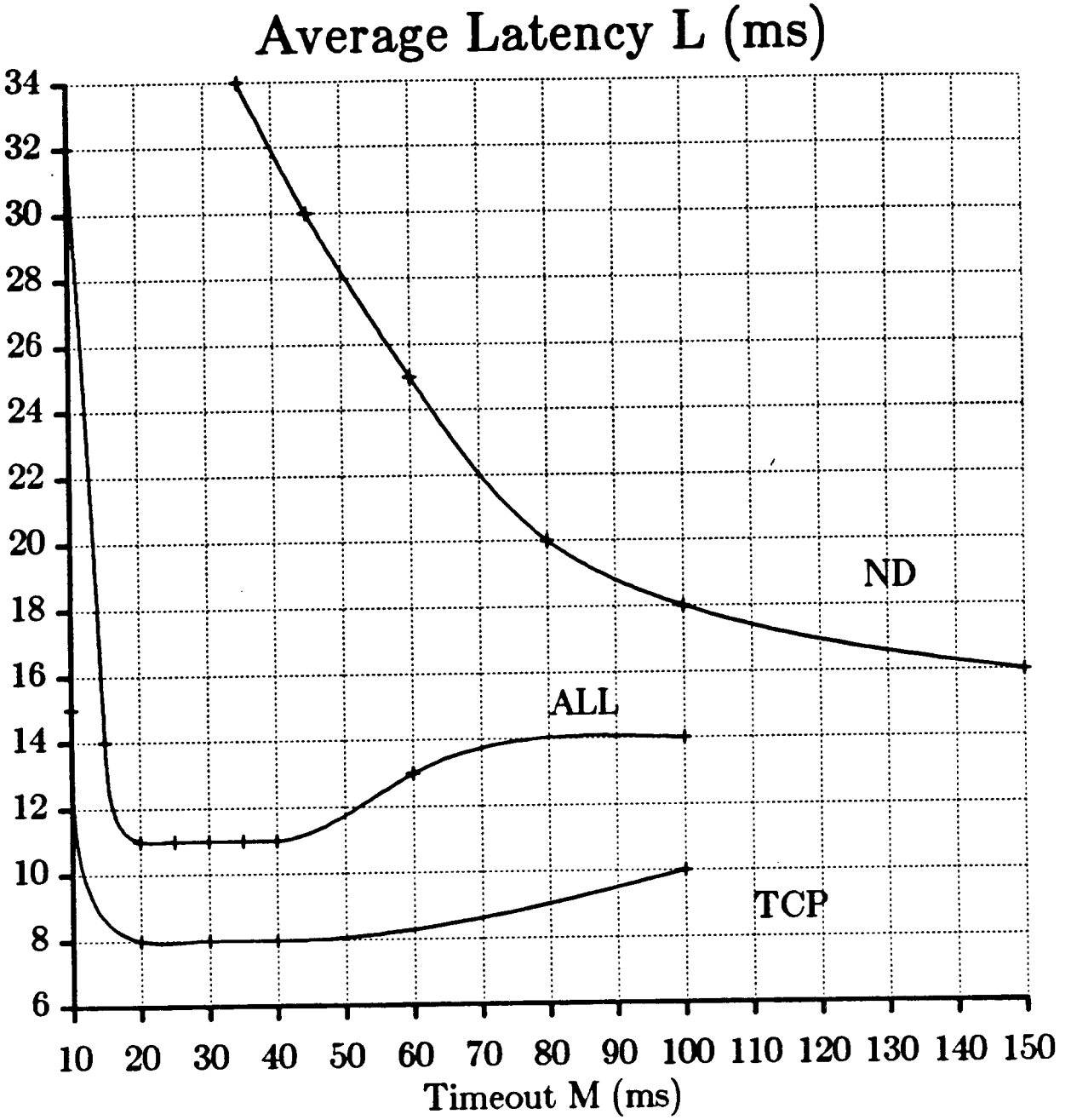


Figure 8.

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