Efficiency of Asynchronous Transfer Mode Networks in Transporting Wide-Area Data Traffic

Ramón Cáceres

Computer Science Division
University of California
Berkeley CA 94720
ramon@tenet.berkeley.edu

ABSTRACT

For performance and economic reasons, ATM networks must efficiently support the Internet family of protocols. We calculate the transmission efficiency achieved by a range of ATM-related protocols when transporting TCP and UDP wide-area traffic. We also compare the efficiency effects of several non-standard compression techniques. To assure an accurate workload characterization, we drive these calculations with millions of wide-area packet lengths measured on the current Internet.

We find that networks using standard ATM procedures are dismally inefficient in carrying traditional data traffic – depending on the protocols used, efficiency as seen by an application program ranges between 40 and 53%. Moreover, due to interaction between TCP-IP datagram lengths and ATM cell padding, efficiency responds abruptly to changes in certain protocol parameters – for example, a 4-byte increase in ATM cell payload size can yield a 10% increase in efficiency. Using one compression technique in isolation can improve efficiency by 12%, and simultaneously using three techniques can improve it by 34%. These issues should be considered when designing future ATM networks.

1. Introduction

This paper presents the transmission efficiency achieved by Asynchronous Transfer Mode (ATM) networks when transporting traditional wide-area data traffic, in particular traffic sent by the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) of the Internet family of protocols. ATM is the recommended multiplexing technique for future broadband integrated services networks [13] [22] [25] and the Internet family of protocols [4] will continue to dominate wide-area data transport, an important application of these networks. Network protocol designers avoid transmission overhead to give users better access to available bandwidth. Transmission efficiency, or the ratio of useful bytes to total bytes carried by a network, measures how well protocols achieve this goal. Inefficient protocols waste network bandwidth. Protocols and techniques that improve efficiency should be considered when designing a network.

Transmission efficiency will remain an important issue as networks evolve. Granted, the performance effects of protocol overhead decrease as communication lines become faster. However, there remains a significant economic cost to wasted bandwidth. Historically, high-speed communication outside the local area has carried a high price. This situation may continue due to

---

This research was supported by NSF and DARPA under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, by AT&T Bell Laboratories, Hitachi, Ltd., the University of California under a MICRO grant, and the International Computer Science Institute.

the non-decreasing cost of upgrading land lines. Even if long-haul prices fall due to sharing of lines among many users, not everyone will benefit from the economies of scale. Consider an integrated services network that reaches to the home. Installing a high-speed access line between a home and the nearest network point of presence will continue to be costly. This cost will not be shared among many users. In some cases the old, slower line will not be upgraded because of economic considerations. Whether an access line is upgraded to high speeds or is kept at slower speeds, both the user and provider of the line will remain conscious of the quality of service provided for a certain price. It will always be necessary for networks to operate efficiently.

There are three major sources of transmission inefficiency. First, as data flows through the network, protocols add header and trailer overhead to the data generated by an application program. Second, some protocols add overhead to satisfy byte-alignment requirements for their headers and trailers. Finally, in ATM networks, another source of inefficiency is the fragmentation of variable-size packets into fixed-size cells. When a packet is presented to an ATM network, it is separated into a sequence of cells. In general, the last cell in the sequence will be only partially filled with packet data. Any unused space in the last cell is padded with dummy bytes that add to the overall inefficiency. For a given stack of protocols, the extent of these three problems is highly dependent on the workload presented to the network, particularly on the size distribution of application data units.

The success of the Internet family of protocols suggests they will continue to be a significant portion of the traditional data traffic carried by wide-area networks. The NSFnet backbone links over 2,300 university, industry, and government networks [21], and is growing rapidly. More than 720 new subnetworks joined the NSFnet backbone in the eighteen months between January, 1990, and June, 1991, and 38 subnetworks joined in the second week of June, 1991, alone [20]. This network carries predominantly Internet protocol traffic – we observed that more than 94% of wide-area Internet traffic is due to TCP and UDP. Future networks will carry audio and video traffic with different characteristics from current traffic. However, traditional forms of traffic will prevail for several years and will continue to be used for much longer. As a result, future B-ISDN networks based on ATM must efficiently support TCP and UDP.†

Figure 1 illustrates some of the efficiency concerns addressed in this paper. It shows transmission efficiency as a function of payload size, that is, the portion of an ATM cell used to carry higher-level data. The input to the calculation was a histogram of wide-area TCP and UDP application data lengths measured on the current Internet. Application data bytes plus datagram header bytes were considered useful, all others were not. The protocol overhead used was: 40 bytes per TCP-IP datagram and 28 bytes per UDP-IP datagram at the transport and internetwork levels, 8 bytes per datagram at the framing level, 4 bytes per cell at the adaptation level, and 5 bytes per cell at the data link level. The solid line represents the efficiency obtained by following standard ATM procedures, including padding partially-filled cells. The cross-marked line represents the efficiency obtained when these cells are not padded. The difference between the two lines is the efficiency lost to cell padding.

We observe the following:

- Efficiency responds abruptly to changes in payload size. The sharp drop in efficiency near 48 bytes of payload is due to the abundance of wide-area TCP-IP datagram lengths between 40 and 50 bytes. After ATM-related overhead, these short datagrams generate one full ATM cell and a second cell that is mostly padding. Moving from the 48-byte CCITT standard payload to a 54-byte payload increases efficiency by almost 10%. The previously proposed payload sizes of 32 bytes and 64 bytes both perform better than the eventual standard.
- Discarding cell padding not only reduces overhead but also smooths the efficiency curve. At the standard payload size, not padding partially-filled cells increases efficiency by 12%. As evident by the smoothness of the upper curve, the jaggedness in the lower curve is almost entirely due to cell padding.

---

† Appendix 2 contains a glossary of networks, protocols, and applications.

- With padding, transmission efficiency as seen by a host is only 62.9% at the standard operating point and never rises above 72%. Under the same conditions, a similar calculation shows that efficiency as seen by an application program is considerably worse − it never rises above 49%.

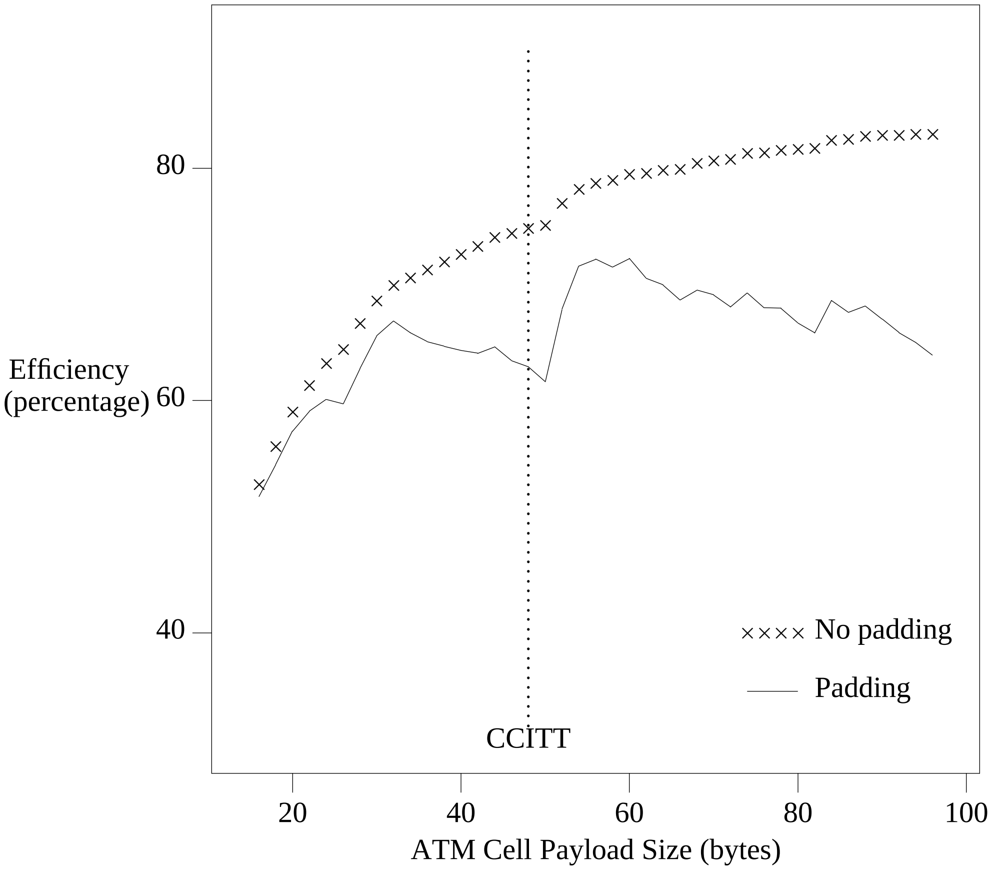

Figure 1 - Efficiency of ATM Networks in Transporting TCP and UDP Traffic

These results suggest that ATM networks are inefficient in transporting traditional wide-area data traffic. We investigate whether the phenomena evident in Figure 1 extend to other ATM-related protocol choices. We also study the efficiency effects of non-standard compression techniques such as discarding cell padding. To isolate the effects of overhead contributed by different levels in the protocol hierarchy, we calculate efficiency from the point of view of a network application, a host transmitting IP datagrams, and certain per-datagram framing protocols. The following are selected results:

- The non-linear effects due to interaction between short TCP-IP datagrams and cell padding are common to all protocol combinations − efficiency is sensitive to small changes in certain design dimensions and insensitive to large changes in others.
- Discarding cell padding can improve efficiency by 6.9 to 11.9% as seen by a host.
- Grouping all the cell payloads corresponding to a datagram and adding to the group only one cell's worth of overhead can improve efficiency by 4.7 to 8.8% as seen by a host.
- Predictive encoding of TCP-IP headers can improve efficiency by 5.0 to 12.7% as seen by an application program.
- Simultaneously applying these three compression techniques improves efficiency by 24.6 to 34.0% as seen by an application program.

The rest of the paper explains these and other results in more detail. Section 2 surveys related work. Section 3 presents the transmission overhead introduced by Internet-related and

ATM-related protocols. Section 4 describes our measurements of wide-area Internet traffic taken at two universities and two industrial research laboratories. Section 5 presents a characterization of TCP and UDP datagram sizes derived from the traffic measurements. Finally, Section 6 discusses our efficiency calculations.

2. Related Work

Kleinrock et. al. [18] studied line overhead in the ARPANET during the mid-1970's. They noted the critical dependence of network efficiency on traffic mix – they found that average transmission efficiency could be as low as 1% for single-character traffic and as high as 79% for file transfers. They suggested that designers of future network protocols take more account of the effect of line overhead on network performance. We note that ATM networks also exhibit severe efficiency problems for an important class of traffic, and make a similar suggestion to designers of protocols for ATM networks.

DeSimone [7] calculated ATM transmission efficiency based on a statistical model of TCP traffic. His model is based partly on network measurements; we drive our calculations directly with measured TCP and UDP traffic data. From him we borrow the definitions of application, datagram, and frame efficiency. He presents results for one of the four protocol combinations and two of the three principal compression techniques we discuss here. Because of the difference in assumed workload, he finds the effects of discarding ATM cell padding less significant than we do – he saw an improvement in efficiency of only 2-3%; we see 4-12%. Similarly, after compressing TCP-IP headers he saw improvements of 0-7%; we see 5-12%. As a result, we draw different conclusions regarding the merits of these two compression techniques.

Recently, there has been considerable activity in network traffic measurement and analysis [5] [12] [14] [19] [24]. Traffic measurements are valuable in the study of network performance because they provide a snapshot of real network workloads. Analysis of the same traffic traces used in this study appears in two related publications: one presents packet length characteristics broken down by the type of application responsible for the traffic [2], and the other presents additional characteristics further broken down by individual conversations of each application type [3]. Here, we apply aggregate TCP and UDP packet length statistics to the issue of transmission efficiency.

3. Protocol Overhead

In an ATM network transporting TCP and UDP traffic, protocols introduce transmission overhead at all levels in the hierarchy. This overhead takes three forms: header and trailer bytes, alignment bytes, and padding bytes. Header and trailer overhead are fixed per data unit, while alignment and padding overhead are variable. In this paper, we ignore overhead added to user data by network applications such as TELNET and FTP, and consider only overhead added below the application level.

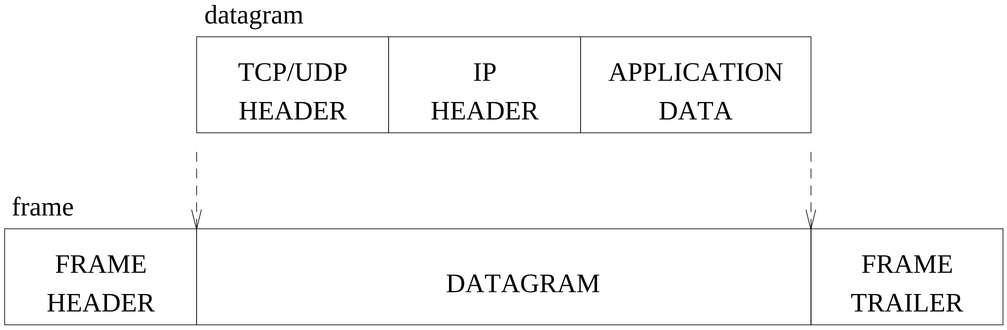

Figure 2 - Frame Format

After an application data unit is prepared for transmission, Internet protocols add their share of overhead while forming a *datagram*, as shown in Figure 2. In the absence of rarely-used protocol options, TCP adds 20 bytes of header at the transport level and UDP adds 8. IP adds another 20 bytes at the internetwork level.

A *frame* is then formed by wrapping each datagram with an optional frame header, trailer, or both, as shown in Figure 2. Examples of proposed services and protocols at the frame level are AT&T's Message Stream Protocol (MSP) [10], AT&T's Frame Relay Service for XUNET 2 (XFRS) [16] [11], and Bellcore's Switched Multi-Megabit Data Service (SMDS) [26]. All three specify the length of the enclosed datagram so it can be reconstructed from a stream of fixed-size cells. In addition, MSP provides sequencing to detect the loss of full frames. XFRS also provides frame sequencing and detects bit errors in both the datagram and the frame trailer. This combination of techniques allows XFRS to handle both the corruption of single bits and the loss of full ATM cells. SMDS detects similar types of errors and supports a larger source and destination address space than IP does. MSP adds 8 bytes of header, XFRS adds 12 bytes of trailer, and SMDS adds 24 bytes of header and 4 bytes of trailer. An XFRS trailer must be also 8-byte aligned and reside entirely in one ATM cell, which may introduce additional overhead in the form of alignment bytes.

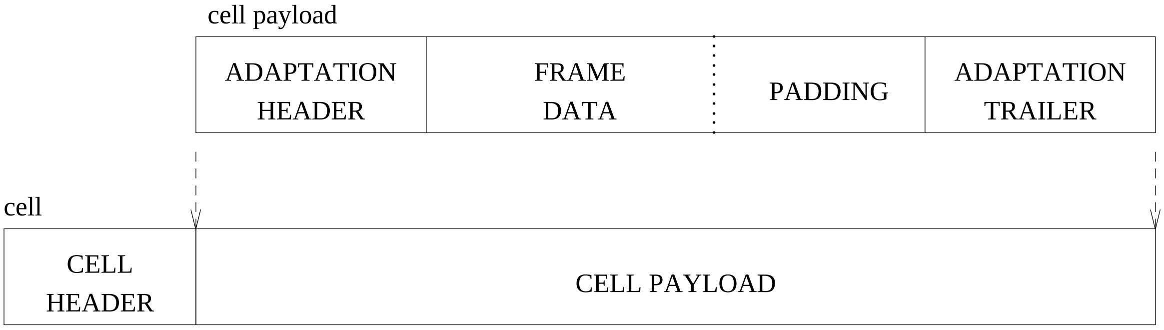

Figure 3 - Cell Format

An ATM network will then break up a frame into fixed-length *cells* whose format is shown in Figure 3. The CCITT ATM standard calls for 53-byte cells with 5 bytes of header and 48 bytes of payload. The ATM header provides virtual circuit identifiers and single-bit error detection on the contents of the cell header alone. An optional adaptation layer inserts in each payload a header, a trailer, or both, leaving the remaining space to carry frame data. Examples of proposed adaptation protocols are Bolt, Beranek and Newman's Segmentation and Reassembly (SAR) protocol [9] and IEEE's 802.6 segmentation and reassembly layer [8]. SAR provides sequencing to detect the loss of full cells, and also detects bit errors in the payload; 802.6 provides similar error-detection. In order to account for partially-filled cells, 802.6 also specifies the number of higher-level bytes actually carried in the payload; SAR leaves this responsibility to higher levels. SAR adds 3 bytes of trailer and 802.6 adds 2 bytes of header and 2 bytes of trailer. Table 1 summarizes the fixed overhead contributed by the services and protocols just discussed.

Level	Data Unit	Protocol	Bytes of Overhead
Transport	Datagram	TCP	20
		UDP	8
Internetwork	Datagram	IP	20
Framing	Frame	SMDS	28
		XFRS	12
		MSP	8
Adaptation	Cell	802.6	4
		SAR	3
Data Link	Cell	ATM	5

Table 1 - Fixed Protocol Overhead

In addition to header, trailer, and alignment overhead, ATM networks introduce fragmentation overhead in the form of padding bytes. When the length of a frame is not an integral number of usable payload lengths, the last cell for that frame is padded to the required fixed length. The amount of overhead contributed by fragmentation is equal to the number of bytes left unused in the last cell for the frame.

Finally, the physical layer also introduces overhead. For example, the DS3 transmission standard offers a raw 44.7 Megabits/second. The IEEE 802.6 standard calls for 12 ATM cells per 125-microsecond DS3 frame, which yields 40.7 Megabits/second, or 91% transmission efficiency [7]. The effects of the physical layer are ignored in our efficiency calculations.

4. Traffic Measurements

4.1. Tracing Sites

To accurately gauge the workload of current wide-area networks, we obtained Internet traffic traces from two universities and two industrial research laboratories: the University of California in Berkeley, the University of Southern California (USC) in Los Angeles, AT&T Bell Laboratories in Murray Hill, New Jersey, and Bellcore in Morristown, New Jersey. We gathered packet headers from the gateway Ethernet carrying all traffic between the local networks at each site and the corresponding regional access network: the Bay Area Regional Research Network (BARRnet) for Berkeley, Los Nettos for USC, and the John Von Neumann Center Network (JVNCnet) for Bell Labs and Bellcore. We filtered out local and transit traffic but kept all IP datagrams flowing between each site and the rest of the world.

We analyzed 24 hours of traces from each site. Trace collection started at 10:30 on Tuesday, October 31, 1989, in Berkeley; 14:24 on Tuesday, January 22, 1991, in USC; 7:59 on Thursday, July 13, 1989, at Bell Labs; and 14:37 on Tuesday, October 10, 1989, at Bellcore. Tables 2 and 3 show the amount of traffic attributed to TCP and UDP at each site.

Protocol	Site							
	Berkeley		USC		Bell Labs		Bellcore	
	#	%	#	%	#	%	#	%
TCP	5,891,621	83.3	5,094,504	97.5	393,305	78.7	1,466,297	88.5
UDP	1,046,027	14.8	52,068	1.0	104,799	20.9	103,355	6.3
other	135,406	1.9	77,696	1.5	1,896	0.4	86,587	5.2

Table 2 - Packet Counts from each Tracing Site

Protocol	Site			
	Berkeley	USC	Bell Labs	Bellcore
TCP	504,969,623	662,737,646	5,007,0497	154,093,992
UDP	63,033,070	4,351,976	3,362,217	5,532,723

Table 3 - Byte Counts from each Tracing Site †

4.2. Trace Applicability

Given that wide-area TCP and UDP traffic will be an important application of ATM networks, we believe our traces are representative of such traffic. A look at aggregate statistics for the NSFnet backbone [20] lends weight to this claim. The breakdown of traffic in our traces shows strong similarities to the breakdown of NSFnet backbone traffic during October, 1989, when our Berkeley and Bellcore traces were gathered, and January, 1991, when our USC traces were gathered.‡ Most importantly, the seven applications we identified as transmitting the most packets (TELNET, RLOGIN, FTP, SMTP, NNTP, DOMAIN, and VMNET) [3] also appear as such on the NSFnet backbone.

---

† Byte counts for protocols other than TCP and UDP are not available.

‡ Bell Labs network activity is not listed in the NSFnet statistics database.

NSFnet statistics for the total packet activity generated by a stub network also suggest that our traces are representative. Of 1,174 stub networks for which backbone activity was detected during October, 1989, Berkeley was the 3rd busiest and Bellcore was the 130th busiest. Of 2,345 stub networks measured during January 1991, USC was the 31st busiest. Thus, our traces capture activity generated by a range of stub networks, from an very active one like Berkeley, to a moderately active one like USC, to a less active but non-trivial one like Bellcore.

The uniformity of our trace data further justifies its use as representative of wide-area Internet traffic. Although our data comes from four different organizations and spans a period of one year and a half, traffic from all four sites shares many characteristics [3]. Of particular relevance to this work are the length distributions of application data units. These distributions are very similar among the four sites and, consequently, transmission efficiency results obtained using all four traces are also similar.† All efficiency results in the body of this paper are derived from the Berkeley data.

4.3. Trace format

The traces from Berkeley, USC, and Bellcore were saved on 8-millimeter helical scan magnetic tapes. Every network packet generated a trace record consisting of a time stamp and the first 56 bytes of raw Ethernet data. The time stamp records the arrival time of the packet at the tracing apparatus. The 56 bytes of data are enough to hold the Ethernet header, the IP header, and either a TCP header or a UDP header. ‡ Due to hardware limitations, the Bell Labs traces were not saved on tape. Rather, packet counts were broken down by application type and data unit length, and written to disk; the raw trace data was discarded.

4.4. Tracing Instruments

The Berkeley traces were gathered using a Sun 3/50 workstation equipped with a microsecond timer [6]. The Ethernet driver in the SunOS kernel was modified to write trace records to a circular kernel buffer big enough to hold 128 full-size Ethernet packets. The resulting time stamp resolution was 10 microseconds. A dedicated user-level program inspected the buffer and wrote new records to tape. No packet losses due to buffer overflows were detected during the Berkeley measurements.

The USC data was collected using the NNStat program suite [1] on a Sun SparcServer 4/490. The NNStat package uses the standard SunOS timer, which on that system has a 20-millisecond resolution. During tracing similar to that reported here, the loss rate was estimated by injecting a Poisson stream of short packets; only 0.6% of these packets were missing from the tape.

Bell Labs traffic was inspected with a DEC microVAX II. A new streams module was inserted in the Ninth Edition UNIX kernel between the standard Ethernet driver and user space. It prepended a 10-millisecond-resolution time stamp to each incoming Ethernet packet and forwarded the packet to user space. A user-level program read the time stamps and Ethernet packet data, extracted the statistics of interest, and wrote them to a set of disk files. No packet losses due to buffer overflows were detected during the Bell Labs measurements.

The Bellcore traces were collected with a Sun 3 workstation augmented with: a single-board computer with a Motorola MC68030 processor and an on-board LANCE Ethernet controller, an interval timer board, several local hard disk drives, and two 8-millimeter tape drives. A system of hierarchical buffers stretching between the single-board computer and the tape drives carried the trace records to tape. No packet loss was recorded anywhere in the monitoring system

---

† Section 5 discusses the application length distributions for Berkeley, and Appendix 1 presents a comparison of efficiency results using data from the four sites.

‡ We did not encounter any packets carrying IP or TCP options, and ignored packets containing IP fragments. IP fragments accounted for only 0.02% of all IP packets at Berkeley, 0.05% at USC, and 0.02% at Bellcore.

during the Bellcore measurements [19].

5. Datagram Size Characterization

Since transmission efficiency is strictly a function of the number of bytes carried by a network, the efficiency of ATM networks in transporting wide-area Internet traffic is critically dependent on the length distributions of IP datagrams. Figure 4 is a histogram of TCP application data lengths extracted from the Berkeley data, and Figure 5 is a similar histogram for UDP.

TCP is the dominant protocol on the Internet – Table 2 shows it was responsible for more than 80% of all packets observed at Berkeley. Consequently, the distribution shown in Figure 4 will dominate our efficiency calculations. It is sharply bimodal, not random, with very large peaks near 0 bytes and 512 bytes. The 0-byte peak represents almost 40% of all TCP datagrams and is due to acknowledgments and other control information traveling without accompanying data.† The next largest peak represents 30% of all TCP datagrams; it is due to data units between 0 and 10 bytes exchanged by interactive network applications like TELNET and RLOGIN. The 512-byte and 536-byte peaks are due to the Maximum Segment Size (MSS) for wide-area networks configured in most Internet hosts.‡ Figure 5 shows similarly well-defined peaks in UDP data lengths.

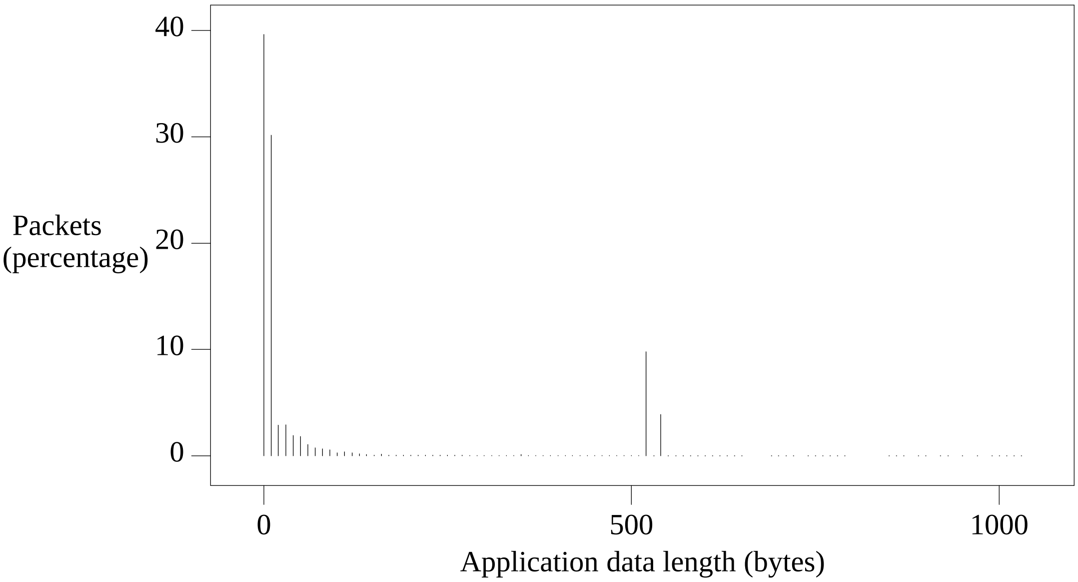

Figure 4 - Histogram of TCP application data lengths

These length characteristics have a strong negative effect on efficiency. As we observed, TCP datagrams are often the smallest possible, and both TCP and UDP datagrams are predominantly small. Almost 70% of TCP datagrams are smaller than 10 bytes, and more than 84% are smaller than 256 bytes. Almost 80% of UDP data lengths are smaller than 48 bytes, and more than 97% are smaller than 150 bytes. In general, the efficiency of a network carrying small data units is more adversely affected by protocol overhead than the efficiency of a network carrying large ones.

---

† A majority of 40-byte TCP datagrams contain only acknowledgements. The fact that TCP is seldom able to piggy-back acknowledgements on reverse-flowing data suggests that TCP connections between wide-area applications are often one-way. Piggy-backed acknowledgments would improve efficiency.

‡ To avoid IP fragmentation [17], the TCP MSS is set to roughly 40 bytes less than 576, the commonly-used IP Maximum Transmission Unit (MTU) for wide-area networks. The choice of 576 is historical and should be reconsidered since the actual MTU of the NSFnet backbone is at least 1500 bytes. Methods for setting a more accurate IP MTU are discussed in [23]. A higher MTU would result in higher efficiency.

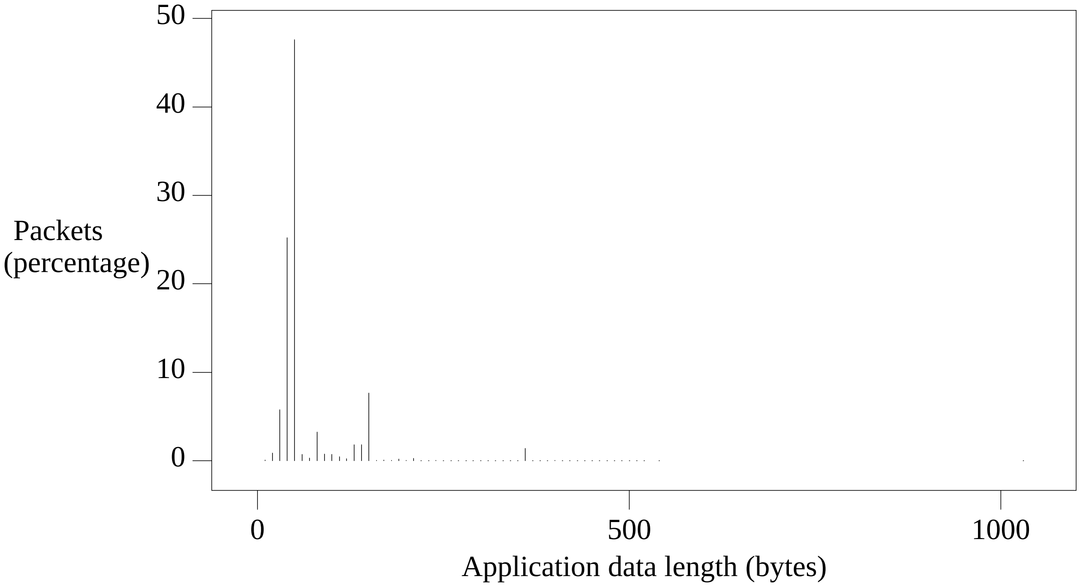

Figure 5 - Histogram of UDP application data lengths

The abundance of TCP application lengths between 0 and 10 bytes has a particularly strong effect on ATM efficiency. In ATM networks, the loss of efficiency due to cell padding depends on how application data lengths map into cell payload lengths. TCP datagrams with 0-10 bytes of application data are 40-50 bytes long. In comparison, the CCITT standard ATM cell payload is 48 bytes long. After framing and adaptation overhead are added, these short TCP datagrams often fill one complete 48-byte payload and very little of another. Cell padding overhead becomes significant when for every two cells transmitted one consists almost entirely of padding. This phenomenon was responsible for the sharp efficiency loss near 48 bytes of payload evident in Figure 1.

6. Transmission Efficiency

6.1. Definitions

We define transmission efficiency as the ratio of useful bytes to total bytes carried by a network's communication lines, expressed as a percentage. Useful bytes can refer to bytes originally sent by an application program, or also include header and trailer bytes added by lower level protocols. We calculate three measures of transmission efficiency to show the effects of overhead contributed by three different levels in the protocol