Speeding up TCP/IP: Faster Processors are not Enough

Evangelos P. Markatos
Institute of Computer Science (ICS)
Foundation for Research & Technology – Hellas (FORTH)
P.O.Box 1385, Heraklio, Crete, GR-711-10 GREECE
http://archvlsi.ics.forth.gr markatos@csi.forth.gr
appears in the
21st IEEE International Performance, Computing, and Communication Conference (IPCCC'02)

Technical Report 297
Inst. of Computer Science - FORTH

Abstract

Over the last decade we have been witnessing a tremendous increase in the capacities of our computation and communication systems. On the one hand, processor speeds have been increasing exponentially, doubling every 18 months or so, while network bandwidth, has followed a similar (if not higher) rate of improvement, doubling every 9-12 months, or so. Unfortunately, applications that communicate frequently using standard protocols like TCP/IP do not seem to improve at similar rates.

In our attempt to understand the magnitude and reasons for this gap between processor performance and interprocess communication performance, we study the execution of TCP/IP on several processors and operating systems that span a time interval of more than eight years. To be able to compare the performance of such different platforms, we define *mileage:* a new performance metric that shows how effective is each platform in using processing power to transfer data. We also propose, calibrate, and experimentally validate a simple model that can accurately characterize TCP/IP performance of a computer based on its processor speed and memory bandwidth.

The main conclusion of this paper is that TCP/IP performance does not scale comparably to processor speeds. To make matters worse, this poor scalability is magnified and propagated to higher-level protocols like HTTP.

1 Introduction

Over the last decade we have been witnessing a tremendous increase in the capabilities of our computation and communication systems. Processor speeds have been increasing exponentially, doubling every 18 months or so [11]. This rate of increase, which is also known as Moore's Law, has been sustained since the seventies and is expected to continue to hold (at least) in the near future. Similarly, network bandwidth, has followed a similar (if not higher) rate of improvement, doubling every 9-12 months, or so, as indicated by Gilder's Law [5].

Based on Moore's Law and Gilder's Law, one could conclude that the performance of distributed/networked applications would improve at similar rates, i.e. doubling every 9-18 months. Unfortunately, this is not the case, partly because, interprocess communication does not improve at the rates predicted by Moore's and Gilder's laws. In this paper we study the performance improvements in TCP/IP-based interprocess communication over the last decade, aiming to answer the following questions:

- *Will future networked applications be able to capitalize on Moore's and Gilder's laws and improve at comparable speeds?* i.e. double their performance every 9-18 months?
- *Does interprocess communication performance improve at rates comparable to those suggested by Moore's and Gilder's laws?* If interprocess communication performance does not scale similarly to

processor and network speeds, then it will soon become a significant bottleneck in the deployment of networked applications.

- *What are the most significant bottlenecks in the performance of TCP/IP-based interprocess communication?* Identifying and resolving these bottlenecks is essential for the efficient execution of networked applications.

- *Can we characterize computers using simple metrics which can be used to predict the performance of communication protocols?* For example, most researchers characterize a processor's speed in "SPECmarks" [10]. If we know the SPECmarks of a processor, we can compare its speed to the speed of other processors. Can we define such a simple and easy-to-measure performance metric that will rate processors according to their performance on running communication protocols?

- *Can we define a simple model of TCP/IP performance, based on easy-to-measure parameters, like processor speed and memory bandwidth?* Using such models we can easily extrapolate and predict the performance of TCP/IP in the years to come.

To answer the above questions we experimentally evaluate the performance of TCP/IP using a variety of communication benchmarks. The contributions of our study are:

- We study the performance of TCP/IP execution on three different processor families (SPARC, Alpha, and Pentium), on 30 computers that span a time interval 8-years long, located in three different countries, connected through several networks ranging from a 100 Mbps Ethernet LAN to a 6 Mbps WAN.

- We show that the execution cost of TCP/IP does not improve at rates similar to processor speeds; actually, although processor speeds have improved by a factor of 35 over the last decade, TCP/IP performance has improved only by a factor of 15.

- We show that the relative performance of operating system primitives, which lie in the critical path of communication operations, has decreased by more than a factor of 3.5 over the last decade; actually, although processor speeds have improved by more a factor of 50, operating system primitives have improved by only a factor of of 15.

- To capture the interprocess communication performance of computers in a single metric, we define *mileage*: a new performance metric that characterizes the power of a computer with respect to its communication abilities. Mileage expresses the amount of data that the computer can transfer by investing one unit of computing power.

- Based on the data we have collected, we define a simple model of TCP/IP performance, which fits very closely (within 5%) the experimental data. The model can be used not only to infer TCP/IP performance for recent computers, but also to extrapolate and predict the performance of TCP/IP for future ones.

The rest of the paper is organized as follows: Section 2 places our work in the context of previous work and recent technology trends. Section 3 presents our experimental measurements, and section 4 defines *mileage* and presents a simple model of TCP/IP performance. Then, section 5 discusses the major factors that limit TCP/IP performance, and finally section 6 concludes the paper.

2 Related Work

TCP/IP performance had received significant interest in the past [3, 19, 2, 18, 9]. Most of this work has focused on studying and improving TCP/IP throughput and latency over limited bandwidth, possibly wireless, and usually congested networks. This line of research, which focused on improving TCP/IP bandwidth and latency, was natural, since, traditionally, communication links had limited (or very expensive) bandwidth, and therefore the improvement of TCP/IP throughput and the reduction of TCP/IP latency were of paramount importance. However, technology trends suggest that, currently, network bandwidth is neither the most limited, nor the most precious commodity in a distributed system. Recent results suggest that raw network bandwidth has been improving much faster than processor speeds

[5], and we have reached a point when processor execution of TCP/IP has become a significant overhead in any TCP/IP-based communication. Actually, it was a couple of years ago, that network bandwidth started to exceed processor speed: that is, a single state-of-the-art processor does not any more have the computing power to execute the network protocols needed to keep a state-of-the-art communication line saturated with data [15]. This trend between processor speeds and network bandwidth is expected to continue in the near future, and therefore the gap between TCP/IP performance and processor speeds will continue to widen. In fact, some researchers, have given up hope on general-purpose processors and have started working on *network processors*, a new breed of special-purpose processors that are specifically tuned for network processing [7, 6]. Our hope is that by studying TCP/IP performance on traditional general-purpose processors we will be able to identify and improve on the main factors that contribute to its limited scalability.

3 Experiments

3.1 Hardware

To quantify the performance improvement of interprocess communication over the last several years, we have used a variety of computers based on SPARC, Alpha, and Pentium Processors. The oldest and slowest of the computers used is a 40 MHz SPARCstation 10 rated at 0.96 SPEC Int95. The most recent and fastest computer used is a 1200 MHz Pentium rated at 51 SPEC Int95. These computers represent a time spectrum more than eight years apart.

Table 1 lists the platforms used, their processors, their operating systems, their name (for identification purposes), their processor clock speed, and their performance. The performance metric we list is the SPEC CPUint95 [1].

3.2 Benchmarks

In this paper we want to accurately characterize the interprocess communication performance of the studied computing systems and identify performance and scalability trends that may exist. To do so we use a variety of popular benchmarks, including `lmbench` [14], `ttcp` [21], and `netperf` [8].

We use `lmbench` to measure (among others things), clock speed [20] and local TCP/IP bandwidth/latency. We use `ttcp` to measure bandwidth between systems communicating over a network. Finally, we use `netperf` to measure the communication latency (using TCP/IP) between different computers.

The traditional performance metrics that have been used in the literature to characterize the performance of communicating systems, are *bandwidth*, *latency*, and, in some cases, *wall clock time*. However, in this paper we are not only interested in the actual performance of the studied communicating systems, but also in the improvement of this performance over the past few years. Therefore, instead of reporting the actual bandwidth and/or latency measured, we report the *improvement* of the bandwidth (or latency) compared to the bandwidth (or latency) of our base case. [2]

3.3 Kernel Entry-Exit Overhead

Most interprocess communication operations usually require the assistance of the operating system kernel. [3] For example, all the operations to send and receive data, to manipulate sockets, and in general to communicate with other processes require operating system calls. Therefore, it is important to understand how the performance of system calls that are in the critical path of interprocess communication operations has improved over the last decade.

Thus, in our first benchmark, we measure the cost of an empty operating system call: that is, the latency to enter and exit the operating system kernel. Figure 1 plots the cost (latency) of an empty operating system call (normalized to the cost of an empty operating system call of our baseline system).

[1] Although SPEC95 has been recently retired and replaced by the SPEC2000, we use SPEC95, because, for most of the computers we studied, only SPEC95 performance results were available.

[2] Our base case is a Sparcstation 10 clocked at 40 MHz running the Solaris 2.6 operating system, rated at 0.96 CPUint95 SPECmarks.

[3] For some noticeable exceptions in the area of high-speed communication for workstation clusters see [16] and [13].

Architecture	Operating System	name	MHz	SPEC Int95 (relative to base line)
Alpha	OSF1 V4.0	isnogood	122	1.4
Alpha	OSF1 V4.0	sarayu	597	30.9
Pentium	Linux 2.2	fiat	199	5.0
Pentium	Linux 2.2	bonny	267	10.8
Pentium	Linux 2.2	kameleon	449	17.0
Pentium	Linux 2.2	ralou	731	35.2
Pentium	Linux 2.2	ga	933	39.4
Pentium	Linux 2.2	dual	1200	51
MIPS	IRIX64 6.	graphite	192	9.7
MIPS	IRIX 6.5	oxygen	283	11.8
MIPS	IRIX64 6.	silicon	250	14.9
SPARCstation-10	SunOS 4.1	atalante	36	1.0
SPARCstation-10	Solaris 2.7	okyalos	36	1.0
SPARCstation-5	Solaris 2.6	osiris	69	1.4
Ultra-1	Solaris 2.7	garbis	143	4.8
Ultra-1	Solaris 2.6	hector	167	5.3
Ultra-1	Solaris 2.6	pandora	167	5.3
Ultra-2	Solaris 2.7	zefyros	168	5.5
Ultra-Enterprise	Solaris 2.7	brain	168	5.7
Ultra-2	Solaris 2.6	apal	200	6.9
Ultra-4	Solaris 2.7	crete	248	7.7
Ultra-5_10	Solaris 2.7	graegos	360	10.2
Ultra-Enterprise	Solaris 2.7	arion	400	14.0
Ultra-4	Solaris 2.7	calliope	400	14.0
Ultra-5_10	Solaris 2.7	horsetail	440	15.0
Ultra-5_10	Solaris 2.7	snoopy	440	15.0
Ultra-60	Solaris 2.7	iris	450	16.2
Ultra-30	Solaris 2.8	bark	296	12.6
Ultra-60	Solaris 2.8	athena	450	16.2
Ultra-60	Solaris 2.8	ourania	600	25.1

Table 1: Hardware Platforms on which the communication benchmarks were run.

Architecture	Processor speed improvement	Kernel entry-exit speed improvement	Relative improvement
SPARC	25	3.4	0.13
PENTIUM	52	15	0.28
Alpha	32	7.7	0.24
MIPS	15	1.9	0.12

Table 2: Relative performance improvement of kernel entry-exit calls.

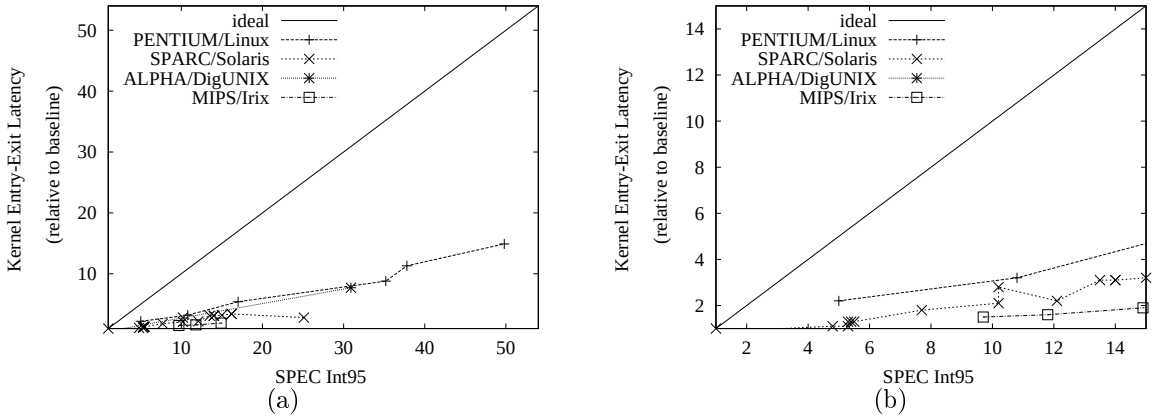

Figure 1: **Kernel entry-exit latency as a function of processor speed.** Figure (b) is a zoomin of figure (a).

We plot the kernel entry-exit latency as a function of the processor speed (measured in SPECint95 SPECmarks). We immediately see that the performance of the empty operating system call does not scale well with processor speed. For example, we see that although PENTIUM processors have gotten more than 50 times faster, the overhead of entering the operating system kernel has improved at best only by a factor of 15. Similarly, although SPARC processors have gotten faster by more than a factor of 25, the overhead of entering the (Solaris) kernel has improved only by a factor of less than 3. To put it in perspective, the kernel entry-exit overhead in a 40 MHz SPARC 10 was 150 cycles, while in a recent 600 MHz Sun-Blade-1000 it is 780 cycles. Table 2 shows the relative performance improvement for each architecture studied. We see that in all cases the kernel entry-exit latency did not scale proportional to the processor speed. In one case though (MIPS), although the processor speed improved by a factor of 15, the kernel entry-exit latency improved only by a factor of 2. Overall, we see that the relative improvement of the time to invoke an empty operating system call was 12%-28% of what one would expect based on processor performance.

Although this disparity between processor speed and operating system performance has been reported before [17], our results indicate that the disparity is getting worse. For example, Ousterhout, in his seminal paper in the early 90's [17], reported that kernel entry/exit performance had relatively improved only 50%-80% compared to the processor's speed, while our results (in the late 90's and early 2000's) as reported in table 2 suggest that kernel entry/exit performance has relatively improved only 12%-28% compared to the processor's speed. Therefore, operating system kernel performance not only continues to get relatively worse compared to processor speed, it does so at a higher rate.

Although the overhead of calling the operating system is rather small, [4] it is especially important for networked applications, since (i) operating system invocation lies in the critical path of almost all data transfer operations, and (ii) Amdahl's law suggests that any component of a system, no matter how small, that does not scale at similar rates with the rest of the system, will eventually become the bottleneck and limit the scalability of the whole system [1]. The bottleneck of the operating system calls has already started to manifest itself in busy web servers. Indeed, the repeated crossing between user

[4] The operating system entry-exit cost in all our SPARC-based computers was 1-3 microseconds.

space and kernel space has reported to contribute a significant overhead to busy web servers [12].

3.4 TCP/IP in Localhost

Although kernel entry-exit performance lies in the critical path of interprocess communication, and may be significant for short data transfers, long data transfers are dominated by the overhead of communication protocol execution. The dominant communication protocol on the Internet today is TCP/IP, and therefore, it is important to understand how the performance of TCP/IP scales with processor speed.

In our next experiment we study the performance of TCP/IP between communicating processes that run on the same computer. [5] To reduce any start-up overheads, data are given for transfer to the socket layer in chunks 64 Kbytes large. Figure 2 plots the TCP/IP throughput achieved in each experiment (relative to the TCP/IP throughput for the base case). We separate the computers into four categories: SPARC uniprocessors, SPARC multiprocessors, DEC Alpha, and PENTIUM, so as to show the trends that exist within each family of processors. We see that in Alpha-based and in Pentium-based computers the TCP/IP throughput scales very poorly. For example, although processor speed has improved by more than a factor of 35 in both cases, TCP/IP throughput has improved only by a factor of 5 for Alphas, and by a factor of 12 for Pentiums. Contrary to these trends, SPARC-based uniprocessors have somewhat better performance. In most of them (esp. the slow ones), TCP/IP throughput scales similarly with processor speed, a trend, however, that seems to diminish in computers rated faster than 10 SPECmarks. Finally, TCP/IP bandwidth on SPARC-based multiprocessors seems to follow the trends of the other computers and does not scale as fast as processors do. [6]

3.5 TCP/IP in a Local Area Network

Although TCP/IP performance between communicating processes located on the same computer revealed significant insight about the scalability of TCP/IP execution, it is important to understand, what is the scalability of TCP/IP execution between communicating processes that reside in different computers, connected through a network. Therefore, in our next experiment we investigate the performance scaling of TCP/IP-based data transfers between processes communicating over a 100Mbps and a 10 Mbps Local Area Network, using the `ttcp` benchmark. We use various computers as sources of traffic (only one computer transmits at-a-time). The traffic destination for the 10 Mbps network is an ULTRASPARC-1 clocked at 167 MHz connected to a 10 Mbps Ethernet adapter, and the traffic destination for the 100 Mbps network is an ULTRA-4 clocked at 400 MHz.

The performance metric we use in this and all subsequent measurements is not the number of Mbytes transfered per second, but the number of Mbytes transfered per sender's CPU-second, (expressed in Mbytes per CPUsec), that is ratio of the amount of data transfered over the (sender's) CPU seconds (including both kernel and user time) that were required for the transfer. Thus, we essentially measure how much data were transfered for each second of CPU time invested. For example, if we transfer 100 Mbytes of data, over a period of 10 seconds, during which the sender has invested 2 seconds of computing power, then the performance we report is 100/2=50 Mbytes/CPUsec. [7]

Figure 3 shows the achieved throughput over the achieved throughput of our base case, for the 100 Mbps LAN. We see that TCP/IP performance generally scales well with processor speed. Although it

[5] By locating both processes on the same computer we are able to focus on the execution cost of TCP/IP, without the interference from a intermediate communication network that may introduce unpredictable performance factors. In our subsequent experiments we will gradually introduce such factors by benchmarking processes running on the same LAN, MAN, and eventually WAN.

[6] The careful reader will notice that the SPARC-based multiprocessors rate as high as 35 SPEC marks in figure 2, while they were reported to rate up to only 17 SPECmarks in figure 1. This is because, in multiprocessors, the TCP/IP bandwidth benchmark is executed on two processors (one processors executes the sender process and one processor executes the receiver process), while the kernel entry-exit benchmark was executed on a single processor. Therefore, in the TCP bandwidth benchmark, each multiprocessor contributed two processors, twice as much processing power, and therefore we doubled the SPECmarks reported.

[7] We did not use the traditional definition of throughput, that is the Mbytes transfered over the wall clock time elapsed, because, the computers we use are rather fast and can easily saturate the 10/100 Mbps Ethernet network that was connecting them. Therefore, the interconnection network, by being a bottleneck, it would not let us explore how TCP/IP execution scales on the different processors studied. In order to understand the scaling of TCP/IP execution we needed to shift the bottleneck from the Ethernet network to the processor executing the TCP/IP. Thus, measuring "Mbytes per CPUsec", instead of "Mbytes per sec", puts the processor, instead of the network, in the spotlight.

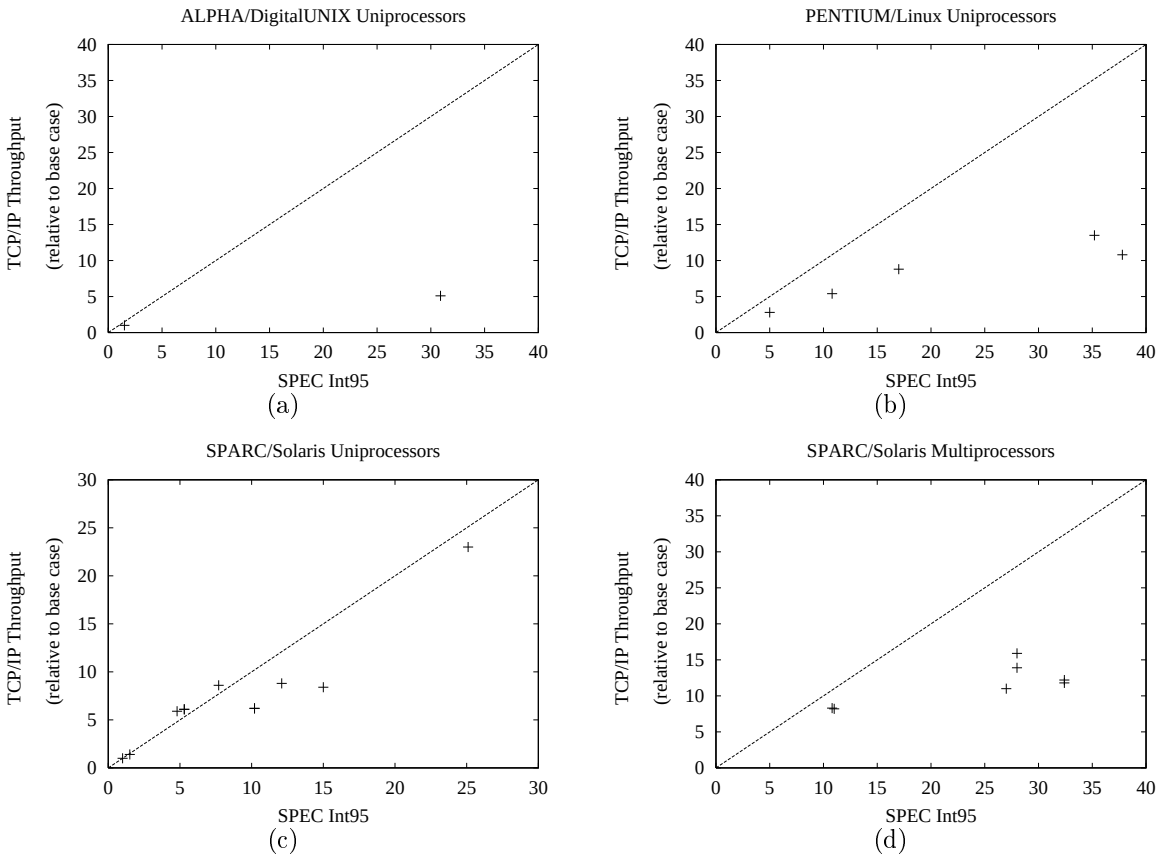

Figure 2: **TCP/IP bandwidth between processors located in the same host as a function of processor speed.** Message size = 64 Kbytes.

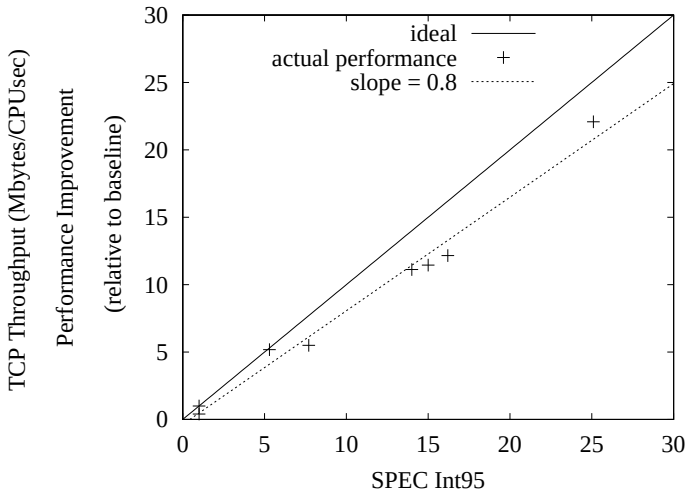

Figure 3: **TCP/IP Bandwidth to a host in the same 100 Mbps LAN .**

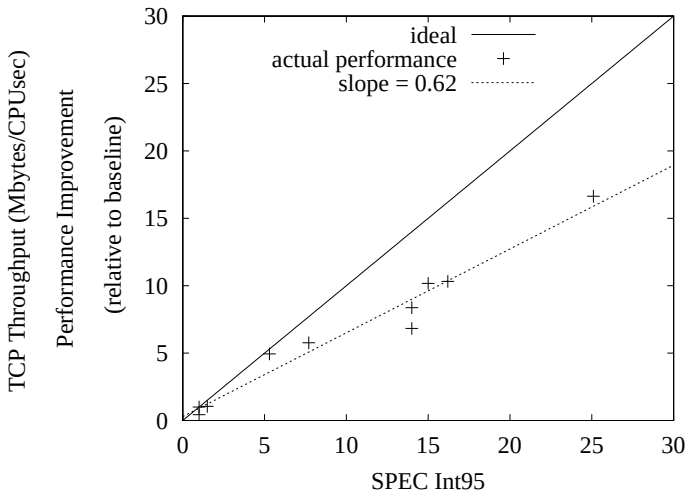

Figure 4: **TCP/IP Bandwidth to a host in the same 10 Mbps LAN .**

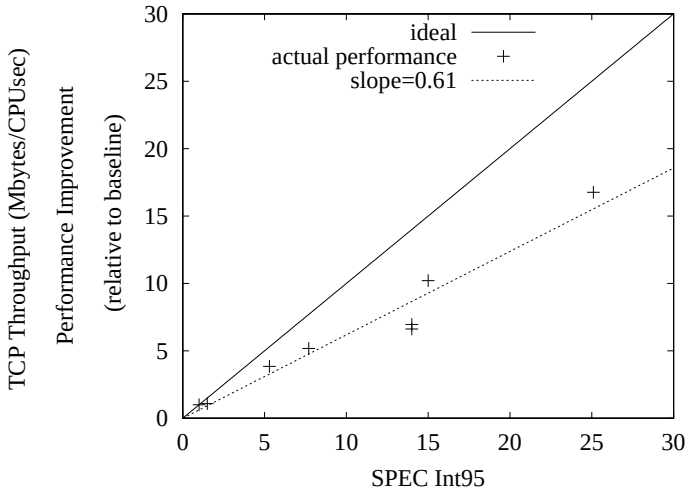

Figure 5: **TCP/IP Bandwidth to a host in the same MAN.**

does not scale exactly the same as processor speed, the line that fits the data has a slope of 0.8, which implies that TCP/IP execution follows processor performance within 80%.

Figure 4 shows the achieved throughput (in Mbytes per CPUsec) over the achieved throughput of our base case for the 10 Mbps LAN. We can see easily that TCP/IP performance, in this case scales rather poorly with processor speed. Actually, although processor speeds have improved by a factor of 25, the achieved TCP/IP bandwidth has improved only by a factor of 17. As can be seen in figure 4 we fitted the measured data with a straight line, whose slope turned out to be 0.62. This implies that TCP/IP execution scaled (about) 60% as fast as processor speeds.

3.6 TCP/IP in a Metropolitan Area Network

In our next experiment we investigate the performance scaling of TCP/IP-based communication in a 10 Mbps Metropolitan Area Network (MAN), using the `ttcp` benchmark. We use various computers as sources of traffic (only one computer transmits at-a-time). The traffic destination is an Sun Ultra 5 clocked at 360 MHz. All the sources are more than 10 kilometers away from the destination computer. Figure 5 shows the achieved throughput (in Mbytes per CPUsec) over the throughput of the base case. It is easy to see the same trend: TCP/IP performance in a Metropolitan Area Networks scales much

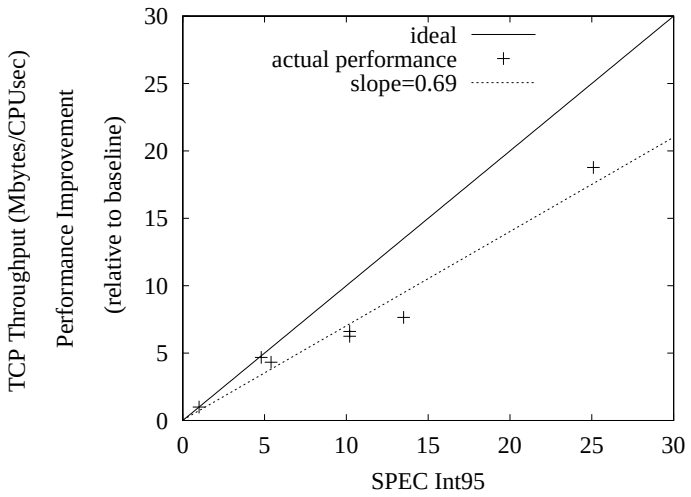

Figure 6: **TCP/IP Bandwidth to a host across a WAN.**

worse than processor speed. [8] We fitted the data with a line that had a slope of 0.61, which implies that TCP/IP performance scaled only 60% with respect to processor speed.

3.7 TCP/IP in a Wide Area Network

In our last bandwidth experiment we investigate the performance scaling of TCP/IP-based communication in a Wide Area Network, using the `ttcp` benchmark. All source computers were located in Greece, while the destination computer (a SUN Ultra 2 clocked at 200 MHz) was located in Norway. Figure 6 plots the measured throughput (in Mbytes per CPUsec) over the throughput of the base case. These results confirm the poor scalability of TCP/IP execution compared to processor performance. We see that in all network we have studied, ranging from Local Area Networks, to pan-european Wide Area Networks, TCP/IP performance scales much worse than processor performance.

3.8 TCP/IP Latency

Our experiments so far indicate that TCP/IP bandwidth clearly does not scale well with processor speed. In our next experiment we will investigate whether TCP/IP latency is influenced by processor speed. We measure the latency between two processes using the `netperf` benchmark. Figure 7 plots the latency between processes running on different computers that are connected in the same 10 Mbps Ethernet LAN. We see that TCP/IP latency scales very little with processor speed, a fact which should be expected: TCP/IP latency between two different computers depends more on network latency and less on processor speed. Our experiments with TCP/IP latency between computers in the same MAN and WAN (not shown here) indeed confirm that there is little (if any at all) relation between processor speed and TCP/IP latency.

3.9 HTTP Performance

Our experiments so far have demonstrated that TCP/IP execution does not scale as well as processors do. It is interesting to know, however, whether this poor scalability propagates to upper-level protocols as well, or is it a minor detail confined within the TCP/IP software. To understand the effect of this poor scalability to upper-level protocols, we use `lat_http`, one of the programs of `lmbench` to measure the performance of the `http` protocol. In our setting, `lat_http` initiates a very simple web server and a client on the same computer. The client requests an html page (about 10 Kbytes large), and its 12 embedded images that are about 40 Kbytes large in total. The benchmarks measures the latency to receive all

[8] Note that this limited TCP/IP performance scalability is not due to the limited bandwidth of the network, because we do not report TCP/IP network bandwidth: we report the Mbytes transfered for each CPU-second.

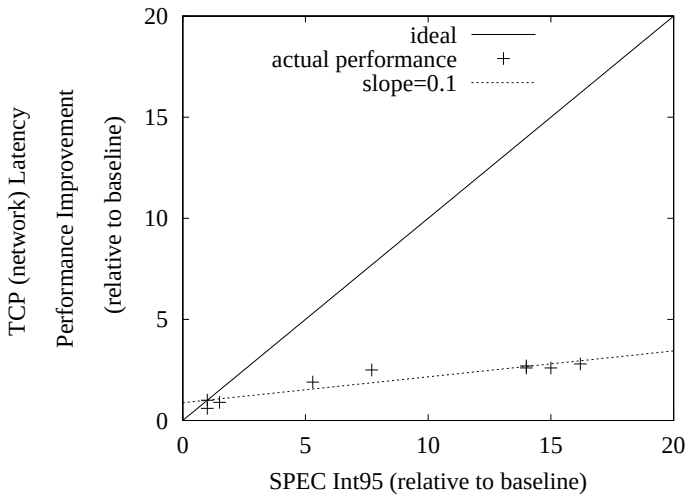

Figure 7: **TCP/IP latency to a host in the same 10 Mbps LAN .**

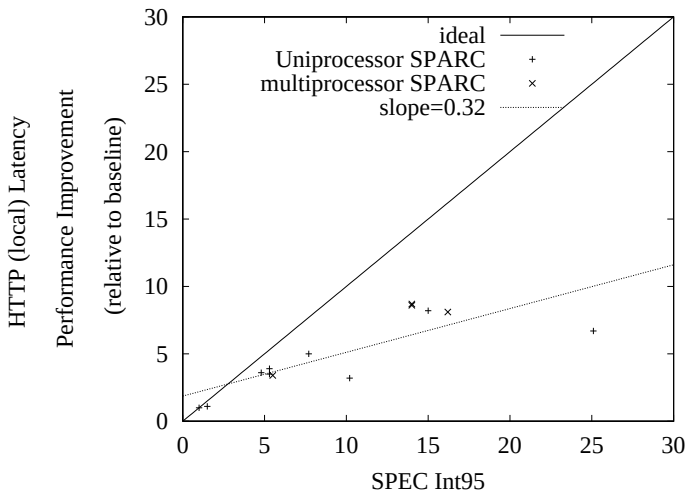

Figure 8: **HTTP latency in localhost as a function of processor speed.**

these files. Figure 8 plots the performance measured by `lat_http` (normalized to the base case). We see that the http performance scales worse than processor speed. It scales even worse than the TCP/IP performance we observed previously. Figure 8 suggests that HTTP performance scales about 30% as fast as processor speeds. The reason behind this obviously bad performance is that `lat_http` transfers html pages and images a few Kbytes large, while our previous TCP/IP benchmarks transfered chunks of data 64 Kbytes large. Recent studies report that the average file size on the web is 8 Kbytes, while the median file is even smaller: only 3 Kbytes large [4]. When transferring such small files, operating systems are not able to optimize the transfers and necessarily suffer large overheads.

3.10 Summary

Our experimental results so far suggest that interprocess communication performance does not scale similarly to processor speeds. Basic operating system functions (like kernel entry-exit overhead) scale as fast as 12%-28% as fast as processor do. TCP/IP execution in all kinds of networks scales between 60% and 80% as fast as processors do. Similarly, TCP/IP latency does not seem to improve noticeably with processor speeds. Unfortunately, this poor scalability is magnified and propagated to higher level communication mechanisms like web data transfers, where HTTP communication performance scales

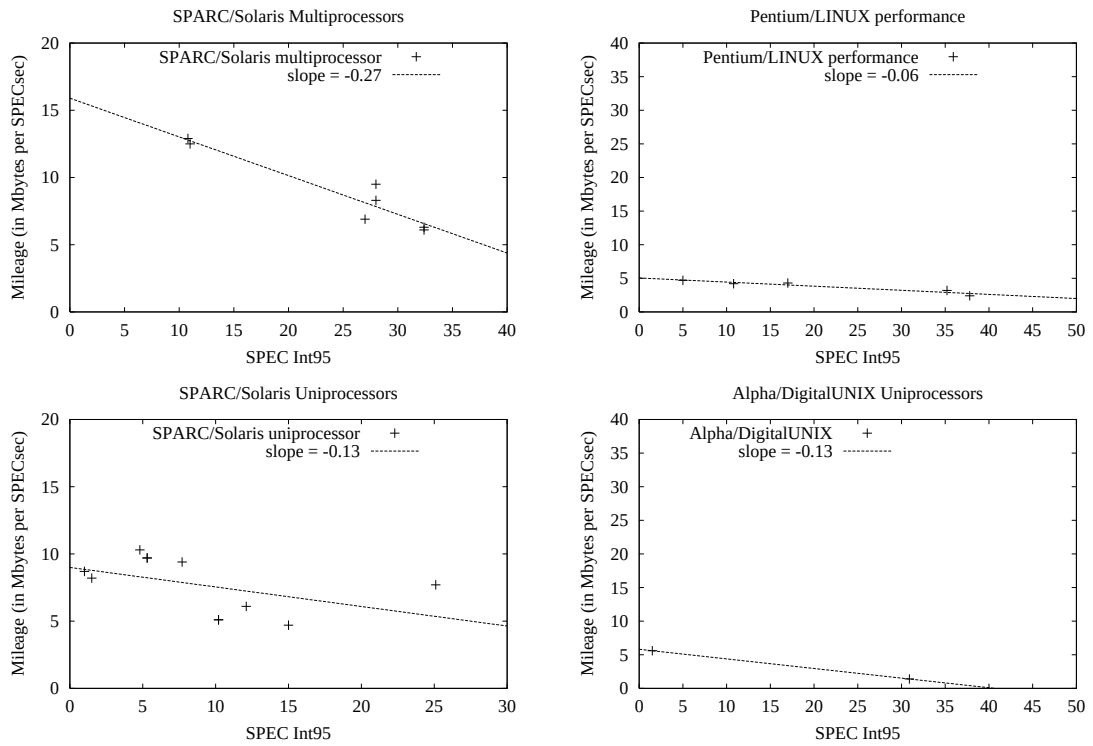

Figure 9: **TCP/IP mileage for data transfers in localhosts.** Message size = 64 Kbytes.

only 30% as fast as processors do.

4 Modeling TCP/IP performance

4.1 Communication Efficiency: Mileage

Our experiments so far indicate that most recent computers are not able to effectively capitalize on the increasing processor speeds and directly translate these speeds into communication protocol performance. To accurately characterize the changes in the disparity between communication protocol performance and processor performance, we define a new performance metric we call *mileage*. We define the *mileage* of a data transfer to be the size of data transfered over the computing power (SPECmarks per sec) invested for the transfer. Mileage is measured in Mbytes per SPECmark per second (Mbytes/SPECmark/sec), and shows how much data (Mbytes) can be transfered by investing one unit of processing power (SPECmark) in one unit of time (sec). For example, if a computer achieves a mileage of 10 Mbytes/sec/SPECmark shows that the computer by investing one SPECmark of computing power for one second, it can transfer 10 Mbytes. Mileage can be computed by dividing the "Mbytes per CPU second" (as have been reported in figures 2-6) we measured with the SPECmarks of the sender computer. Mileage captures the communication capabilities of a particular processor: it measures how fast the processor is, when transferring data. Mileage allows as to compare the communication capabilities of processors that have very different speeds, and thus allows us to understand how the communication capabilities of different processors have scaled with time.

Figure 9 shows the mileage of the computers in our study (separated in groups) for the TCP/IP data transfers to localhosts as reported in figure 2. We can easily see that the mileage of computers decreases with processor speed. That is, faster (and more recent) processors have decreased mileage compared to slower (and older) processors. For example, although early SPARC multiprocessors had a mileage of 13 (Mbytes/sec/SPECmark), recent SPARC multiprocessors have a mileage of 7. Similarly, although early PENTIUM processors had a mileage of 5, recent PENTIUM processors have a mileage of 2.5. Similar trends exist in SPARC uniprocessors, and in Alpha-based workstations. Figure 9 indicates that mileage

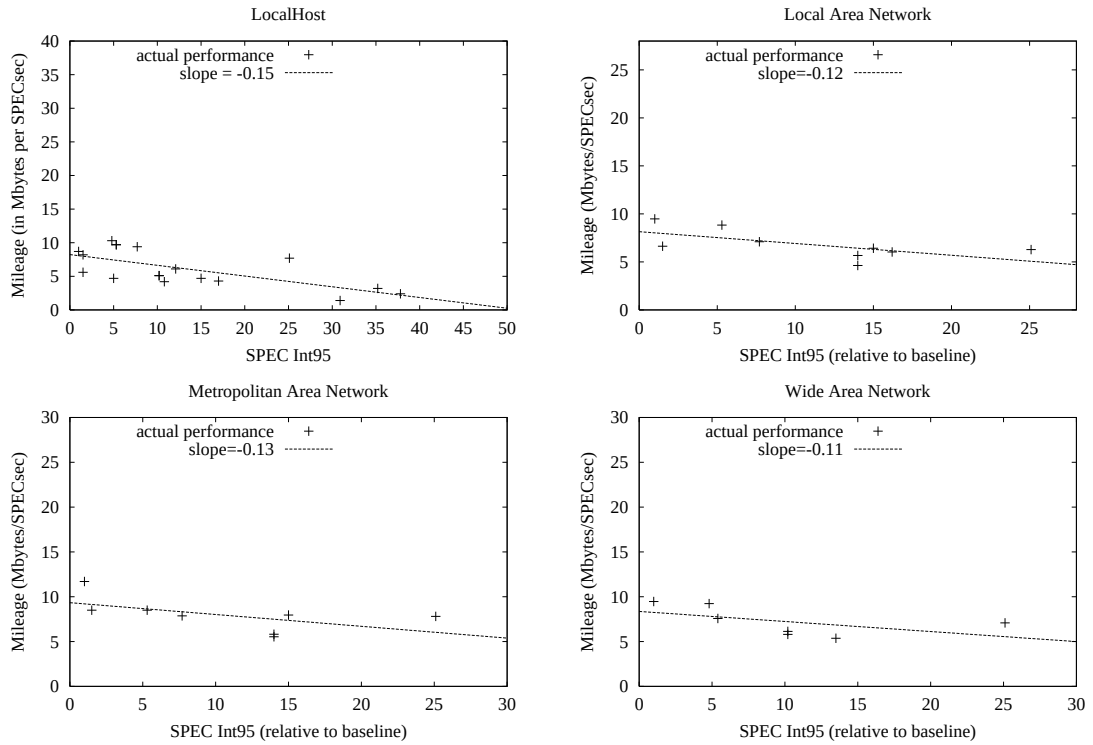

Figure 10: **TCP/IP mileage for transfers in the same host, LAN, MAN, and WAN.** Message size = 64 Kbytes.

has been reduced by factor of 2 over the last 8 years, which implies that, current processors spend twice as much as processing power in order to transfer the same amount of data.

Similar decrease in mileage, can be shown in data transfers between processes in the same network, whether LAN, MAN, or WAN. Figure 10 plots the mileage for the TCP/IP transfer experiments we have conducted in the localhost, the LAN, the MAN, and the WAN. In all cases we see that mileage decreases with processor speed. Essentially, network communications will require a continually increasing amount of processing power in order to transfer the same amount of data.

4.2 An analytic model for TCP/IP throughput

Based on the experimental data that we have collected, we will now develop an analytic model to characterize TCP/IP throughput. The model will not only help us understand the factors that influence TCP/IP performance, but most importantly will enable us to extrapolate (and predict) TCP/IP performance on future computers. Since all our experiments were based on communication using 64-Kbyte chunks, we will restrict our study to TCP/IP communication using 64-Kbyte buffers.

The time to transfer a 64-Kbyte buffer ($T_{TCP/IP}$), consists of the time spent reading from and writing the data to memory (T_{mem}), and the time spent processing these data (T_{comp}). Such processing may include fragmentation, CRC calculation, etc. Thus: $T_{TCP/IP} = T_{mem} + T_{comp}$

The time to read the data from and write the data to memory T_{mem} can be very accurately approximated by the size of the message msg_size over the memory bandwidth mem_band:

$T_{mem} = msg_size/mem_band$

The time to process each message for transfer is proposrtional to the message size and inversely proportional to the speed of the computer: $T_{comp} = Const \cdot \frac{msg_size}{SPEC_rating}$

Therefore, the TCP/IP throughput of the computer is

$$Throughput = \frac{msg_size}{T_{TCP/IP}} = \frac{msg_size}{\frac{msg_size}{mem_band} + \frac{msg_size \cdot Const}{SPEC_rating}} = \frac{1}{\frac{1}{mem_band} + \frac{Const}{SPEC_rating}} \quad (1)$$

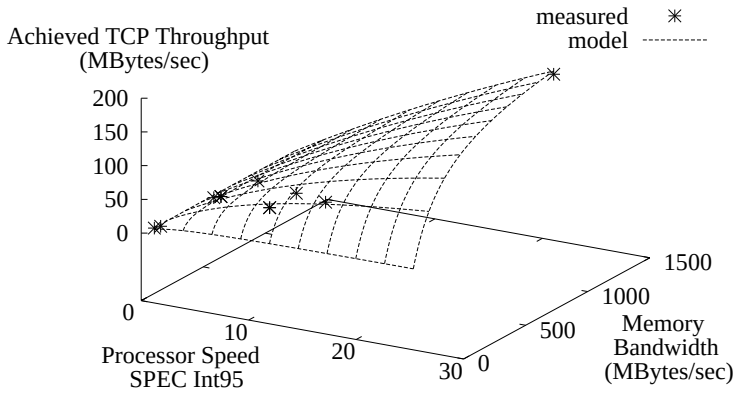

Figure 11: **TCP/IP Throughput in uniprocessor SPARC systems.**

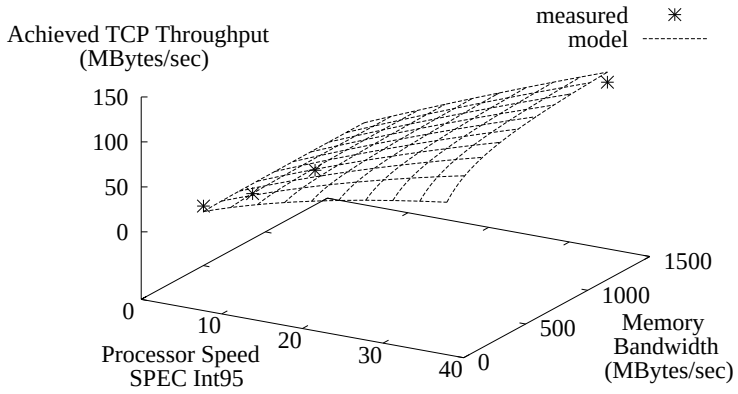

Figure 12: **TCP/IP Throughput in multiprocessor LINUX/PENTIUM systems.**

In equation (1) the mem_band and $SPEC_rating$ can be easily measured using standard benchmarks, like the ones we have already used. The only unknown factor is $Const$ which we found by interpolating the data we measured.

Based on our interpolation on SPARC processors equation (1) becomes:

$$TCP/IP\ Throughput\ for\ SPARC\ uniprocessors: \frac{1}{\frac{1}{mem_band} + \frac{0.105}{SPEC_rating}} \tag{2}$$

Similarly, we interpolated the data for Linux-based computers, and the predicted TCP/IP throughput is:

$$TCP/IP\ Throughput\ for\ LINUX\ uniprocessors: \frac{1}{\frac{1}{mem_band} + \frac{0.26}{SPEC_rating}} \tag{3}$$

Figures 11 and 12 show the TCP/IP throughput for SPARC and LINUX uniprocessors respectively. In the same figures we plot both the measured data points (as stars) and the throughput model (as a grid). In both cases we see that the model matches very closely the experimental data.

5 Why isn't TCP/IP getting faster as fast as hardware?

All our experiments so far indicate that TCP/IP processing does not get faster as fast as hardware does. There are several reasons that contribute to this performance disparity.

- **Architectural Innovations do not necessarily apply to protocol processing.** Recent processors incorporate several architecture innovations, including larger caches, out-of-order execution, deep pipelines, and superscalar execution, all of which can not necessarily be exploited by networking code. For example, although large caches improve the performance of programs that repeatedly access large amount of data, TCP/IP code typically does not exhibit large amount of temporal locality. That is, TCP/IP, and similar communication protocols, do not repeatedly access their data several times, and therefore large caches may not improve their performance significantly. To make matters worse, recent highly optimized protocols (i.e. like zero-copy protocols) reduce the number of accesses they make to their data, and therefore, they take even less advantage of the large caches that may exist. Therefore, processors do get faster, but not for protocol-processing-type of applications.

- **Operating system performance lags behind processor speed.** This is partly because, recent processors include lots of registers, deep pipelines, and in general a large amount of state, all of which needs to be saved and restored during context switches. Therefore, operating system activities on recent processors, get relatively slower compared to similar activities on older processors.

- **Memory bandwidth can be a limiting factor.** Although protocol processing for small messages is dominated by operating-system related overheads, large message transfers can be limited by memory throughput. It is possible that significant improvements in processors speed do not translate in improvements in interprocess communication performance if not accompanied by similar improvements in memory systems. For example, in Fig. 2 (c) we see that one computer rated at 15 SPECmarks and another rated at 7 SPECmarks achieve the same TCP/IP throughput. The reason is that first computer, a SPARC Ultra 5-10 at 440 MHz, had half the memory bandwidth of the latter, a SPARC Ultra 4 clocked at 83 MHz with a 2-way interleaved memory. Therefore, memory bandwidth is a very significant issue which influences the performance of communication protocols.

6 Conclusions

In this paper we experimentally studied the performance cost of TCP/IP in several different processors, ranging from an old 36 MHz SPARC, to a recent 1200 MHZ Pentium. To capture the performance of the processors with respect to their capacity in executing network protocols, we proposed *mileage*: a performance metric that allows us to compare the communication capabilities of different processors. We have also proposed, calibrated, and experimentally validated a model for TCP/IP performance. The model which is based on simple to measure metrics enables us to predict the TCP/IP performance of a computer based on its processor speed, and memory bandwidth.

Based on our experimental results we can safely conclude the following:

- *Future networked applications will probably not be able to take full advantage of computation and communication improvements as expressed by Moore's and Gilder's laws.* The performance improvements of networking code lag significantly behind the performance improvements of computation and and communication systems.

- *TCP/IP protocol execution improves about 40%-50% each year, effectively utilizing only 60%-80% of the processor's improvements.* For example, Fig. 2 (b) suggests that Pentium processor speeds have improved by a factor of 35, while TCP/IP protocol execution on these processors has improved by only a factor of 15.

- *Basic operating system performance improves about 20%-40% each year, unitilizing only 12%-28% of processor speed.* Indeed, although SPARC processor speed have improved by a factor of 25, kernel entry-exit overhead has improved by only a factor of 3.4.

- *The poor scalability of TCP/IP is magnified and propagated to higher level protocols like HTTP.* Our experiments indicate that http-based transfers using realistic file sizes scale poorly (about 30%) with processor speeds; that is, over the last 7 years, SPARC-based processor speeds have improved by a factor of 25, while http transfers have improved by only a factor of 8.
- *Current processors need to invest more processing power to transfer the same amount of data.* Our results show that mileage decreases with time, and has been reduced almost by a factor of two over the last decade.

Our results suggest that the disparity between processor speed and TCP/IP performance will probably continue to widen. Therefore, it becomes increasingly important to understand and optimize the execution of TCP/IP in particular, and protocol software in general, on recent (and future) processors.

7 Acknowledgments

We would like to thank Dionisions Pnevmatikatos and Catherine Chronaki for their constructive comments in earlier versions of this paper. Katerina Gialama, Antonis Danalis, and Xeni Asimakopoulou provided the lmbench results for the LINUX operating system.

References

[1] G.M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the AFIPS Conference*, pages 483–485, 1967.

[2] H. Balakrishnan, V. Padmanabhan, S. Sehan, and R. Katz. A comparison of mechanism fro improving TCP performance over wireless links. *IEEE/ACM Trans. Networking*, 5(6):756–769, 1997.

[3] Hari Balakrishnan, Venkata N. Padmanabhan, and Randy H. Katz. The Effects of Asymmetry on TCP Performance. In *Mobile Computing and Networking*, pages 77–89, 1997.

[4] Paul Barford, Azer Bestavros, Adam Bradley, and Mark Crovella. Web Client Access Patterns: Characteristics and Caching Implications. *World Wide Web Journal*, 2:15–28, 1999.

[5] Jay S. Bayne. Unleashing the POWER of Networks. http://www.johnsoncontrols.com/Metasys/articles/article7.htm.

[6] Tzi-Cker Chiueh and Prashant Pradhan. Cache Memory Design for Network Processors. In *Proceedings of the Sixth International Symposium on High-Performance Computer Architecture*, pages 409–418, 2000.

[7] Intel Corporation. Intel IXP1200 Network Processor (white paper), 2000. http://developer.intel.com/design/network/products/npfamily/ixp1200.html.

[8] Hewlett-Packard Company Information Networks Division. Netperf: A Network Performance Benchmark, 1995. http://www.netperf.org/netperf/training/Netperf.html.

[9] C. Fang, H. Chen, and J. Hutchins. A simulation study of TCP performance in ATM networks. In *Proc. of IEEE Globecom*, 1994.

[10] J.D. Gee, M.D. Hill, D.N. Pnevmatikatos, and A.J. Smith. Cache Performance of the SPEC-92 Benchmark Suite. *IEEE Micro*, August 1993.

[11] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach (second edition)*. Morgan Kaufmann Publishers, Inc., 1996.

[12] Philippe Joubert, Robert King, Richard Neves, Mark Russinovich, and John Tracey. High-Performance Memory-Based Web Servers: Kernel and User-Space Performance. In *Proc. of the 2001 Usenix Technical Conference*, 2001.

[13] E.P. Markatos and M. Katevenis. Telegraphos: High-Performance Networking for Parallel Processing on Workstation Clusters. In *Proceedings of the Second International Symposium on High-Performance Computer Architecture*, pages 144–153, February 1996.

[14] L. McVoy and C. Staelin. lmbench: Portable Tools for Performance Analysis. In *Proc. of the 1996 Usenix Technical Conference*, pages 279–294, January 1996.

[15] SUN Microsystems. THE NET EFFECT: How increased bandwidth is driving innovation in business and technology, 2000. http://www.sun.com/neteffect/.

[16] S. Mukherjee and Mark D. Hill. A Survey of User-Level of Network Interfaces for System Area Networks. Technical report, Computer Science Department - University of Wisconsin-Madison, 1997.

[17] J.K. Ousterhout. Why aren't Operating Systems Getting Faster As Fast As Hardware? In *Proceedings of the Summer 1990 Usenix Technical Conference*, pages 247–256, June 1990.

[18] C. Partridge and T. Shepard. TCP Performance over Satellite Links. *IEEE Network*, 11(5):44–99, 1997.

[19] Michael Perloff and Kurt Reiss. Improvements to TCP Performance in High-Speed ATM Networks. *Communications of the ACM*, 38(2):90–100, 1995.

[20] C. Staelin and L. McVoy. mhz: Anatomy of a micro-benchmark. In *Proc. of the 1998 Usenix Technical Conference*, 1998.

[21] USNA. TTCP: a Test of TCP and UDP Performance, 1984.