Analysis of TCP's Computational Energy Cost for Mobile Computing *

[Extended Abstract] †

Bokyung Wang
Department of Computer Science
Portland State University
Portland, OR 97207
wangbok@cs.pdx.edu

Suresh Singh
Department of Computer Science
Portland State University
Portland, OR 97207
singh@cs.pdx.edu

ABSTRACT

In this paper we present results from a measurement study of TCP (Transmission Control Protocol) running over a wireless link. Our primary goal was on obtaining a breakdown of the computational energy cost of TCP at the sender and receiver (excluding radio energy costs) as a first step in developing techniques to reduce this cost in actual systems. We analyzed the energy consumption of TCP in FreeBSD 5 running on a wireless laptop. Our initial results showed that 60 - 70% of the energy cost (for transmission or reception) is accounted for by the Kernel – NIC (Network Interface Card) copy operation. Of the remainder, ∼15% is accounted for in the copy operation from user space to kernel space with the remaining 15% being accounted for by TCP processing costs. We then further analyzed the TCP processing cost and determined the cost of computing checksums accounts for 20 – 30% of TCP processing cost.

Categories & Subject Descriptors C.2 [Communication Networks]; C.2.2 [Network Protocols]

General Terms Measurement, Performance

Keywords Energy, mobile, wireless, TCP

1. INTRODUCTION

Most internet applications run by users (mobile or not) use TCP as the underlying protocol. For mobile users, since energy is a constraint, it is important to understand and reduce the *energy consumed* by TCP. Prior work in this domain has has looked at the energy consumption of various wireless interface cards [1], the impact of ARQ (Automatic Repeat reQuest) protocols on overall energy consumption [2], the effect of channel conditions on energy consumption [3], and comparative energy and throughput-based studies of different TCP flavors. While all of these studies have contributed a great deal to understanding the impact of the wireless link on the overall end-to-end energy consumption, none of these studies has looked at the *computational energy* cost of running the TCP protocol itself. The computational cost of TCP includes the cost of various copy operations, the cost of computing checksums, the cost of responding to timeouts and triple duplicate ACKs, and other bookeeping costs. Our goal in this study is to estimate the energy (in Joules) taken by each of these operations and then develop techniques to reduce this cost.

1.1 Energy consumption in TCP

The computational energy cost of a TCP session established by a wireless device can be viewed as the total energy cost of the session minus the cost of the radio and the idle energy cost of the connection (i.e., when the node is idle awaiting ACKs or data segments). We can view the TCP computational energy cost as being composed of the following primary components:

- The cost of moving data from the user space into kernel space that we denote *user-to-kernel copy*. Note that this cost can be eliminated by using zero-copy, if available.
- The cost of copying packets to the network interface card that we denote as *kernel-to-NIC copy*.
- The cost of processing in the TCP/IP protocol stack (*TCP Processing Cost*) which includes:
  - The cost of computing the checksum (at sender and receiver),
  - The cost of ACKs (at sender and receiver),
  - The cost of responding to timeout events and the cost of responding to triple duplicate ACKs,
  - Other processing costs such as window maintenance (at sender on receiving ACKs), estimate round trip time (at sender), obtaining the TCB (Transmission Control Block), interrupt handling, and timer maintenance[1,2]

*This research was funded by NSF under grant ANIR-0196043.

†A full version of this paper is available on request

SIGMETRICS'03, June 10–14, 2003, San Diego, California, USA.
ACM 1-58113-664-1/03/0006.

[1]In this paper we chose not to analyze TCP options such as SACK (Selective ACK) but rather concentrate on the core TCP implementation alone.

[2]We have not isolated the cost of interrupt processing but rather folded it into the various processing costs.

2. METHODOLOGY

We performed our measurements on a Toshiba Satellite 2805-S201 laptop with a 650MHz P3 Celeron processor, 256MB RAM running FreeBSD 5.0. The network card used was a 2.4GHz Lucent 802.11b WaveLAN "Silver" card. Finally, power management was turned off in the laptop as well as the 802.11 card for our experiments (this was done so that the power spikes measured by the multimeter would correspond to TCP processing and not some other side effect of power management). The display was also off and the x-server was not running. We subtract out the idle energy consumption of the laptop in order to obtain TCP's energy cost.

Energy consumption was determined by measuring the input voltage and current draw using two Agilent 34401A digital multimeters that have a resolution of one millisecond. We did not use batteries in the devices because avoiding the use of batteries allows for a more steady voltage to be supplied to the device. Since our goal was to measure the computational cost of TCP alone, we needed to separately but simultaneously measure the current draw of the laptop and the current draw of the 802.11b PCMCIA card. To do this, we followed the methodology described in [1]. Our final experimental set up was as follows: the laptop's (or iPAQ's) power supply is measured by one Agilent 34401A multimeter while the 802.11b's power supply is measured by attaching a second Agilent 34401A multimeter to the appropriate terminals on the Sycard CardBus extender. Both these multimeters are triggered simultaneously by a second laptop that also collects data from these multimeters. The average current and voltage measurements were: 1.22A (Idle), 1.905A (Transmit), 1.709A (Receive) and the voltage was 15.08V/DC.

We used ttcp (the buffer size for ttcp was set at 8192 bytes) to measure the energy consumption for transmitting 1 MByte of data. We simultaneously ran tcptrace and tcpdump. The information collected from these tools was used to identify the radio events and appropriately subtract the corresponding radio energy cost from the overall system cost. We used a TCP window size of 16K and RTS/CTS was turned off.

3. MAIN RESULTS

Data sent through a TCP socket is first queued in the socket send buffer and it is then copied into kernel space for further processing. In order to determine the cost of performing this copy (that we call user-to-kernel space copy), we implemented a kernel program using the *copyin()* and *copyout()* functions. We then ran this program and copied large files composed of random data *in memory*. We measured the current draw while this program was running and used this measurement to obtain the user-to-kernel copy cost. To determine the copy cost from the kernel to the NIC card, we used UDP for data transmission. Unlike TCP, data sent through a UDP socket goes all the way down to the network interface and the socket send buffer is not used. We note the following (see Figure 1):

- By far the most expensive operation is the Kernel – NIC copy cost. This accounts for between 60 – 75% of the total cost incurred at the sender. To better understand these costs, note that the PCMCIA card is not very efficient and it takes a significant amount of time to copy the data from memory to the 802.11b card. The device driver calls a copy operation for each packet resulting in an interrupt and context switch since the PCMCIA 2.1 specification does not allow DMA (Direct Memory Access) or bus mastering.

MTU	1500	1000	552	296	168
TCP processing (μJ)	1,546	1,514	1,520	1,489	1,416
Cksum Cost (μJ)	512	502	462	443	422
Cksum %	32.13%	33.14%	30.42%	29.72%	29.81%

Table 1: Sender-side TCP processing cost.

- As a percentage of the total cost, we note that the TCP processing cost decreases with an increase in MTU size. This is explained by the fact that each packet results in the same code being executed on the output (Table 1 shows the TCP processing cost per packet at the sender as a function of MTU size, as we can see, the processing cost is fairly independent of the MTU size). Thus, smaller MTUs result in more packets sent with the corresponding increase in overall TCP processing cost and hence a larger share of the overall cost.

The above studies were done with *zero copy* disabled. When we use zero copy, we see that there is a 10 – 14% reduction in the total sender and receiver cost. This happens in conjunction with an increase in throughput of approximately 18%.

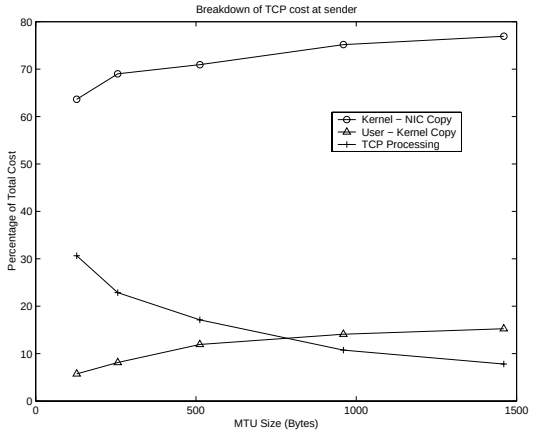

Figure 1: Relative costs for sending 1MByte data.

4. REFERENCES

[1] Laura Feeny and Martin Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proceedings INFOCOM 2001, Anchorage, Alaska*, 2001.

[2] M. Srivastava P. Lettieri, C. Schurgers. Adaptive link layer strategies for energy efficient wireless networking. In *Wireless Networks*, volume 5, pages 339 – 355, 1999.

[3] M. Zorzi, M. Rossi, and G. Mazzini. Throughput and energy performance of tcp on a wideband cdma air interface. In *Journal of Wireless Communications and Mobile Computing, Wiley 2002*, 2002.