The design and performance of Mobile TCP for wireless networks

Article *in* Journal of High Speed Networks January 2001

Source: DBLP

2 authors, including:

Haas J. Zygmunt
Cornell University
325 PUBLICATIONS **23,230** CITATIONS

SEE PROFILE

Journal of High Speed Networks 10 (2001) 187–207
IOS Press

The design and performance of Mobile TCP for wireless networks

Zygmunt J. Haas and Abhijit Warkhedi
School of Electrical Engineering, Cornell University, Ithaca, NY 14853-2801, USA
E-mail: haas@ece.cornell.edu, warkhedi@cisco.com

Abstract. In this paper, we propose a novel approach to reduce the wireless communication overhead associated with the transport layer. Portability is a key element of mobile computing. As mobile devices shrink in terms of size and weight, their computing capabilities are also reduced, constrained by battery capacity. To sustain a good performance level of mobile applications, we devise a solution that reduces the processing complexity on computing-limited mobile devices. Additionally, we aim to minimize the use of wireless resources to further improve performance. Specifically, we develop a streamlined protocol architecture, Mobile TCP (MTCP), that achieves the elimination of IP processing on the wireless segment of the TCP connection. MTCP operates over a single hop wireless link, and, consequently, it eliminates the unnecessary overhead placed on mobile devices, such as the TCP congestion control mechanisms. In addition, we investigate the impact of streamlining the socket layer on offloading the processing overhead. Our experimental results indicate a substantial improvement in the efficiency of protocol processing. For instance, results show that the MTCP processing time per packet is approximately only one fourth that of TCP and the use of CPU resources is reduced by up to 50%. Furthermore, the protocol incorporates various robust, yet simple, loss recovery techniques to considerably improve the throughput in lossy wireless conditions.

1. Introduction

The vision behind mobile computing [1,2] is to support ubiquitous access to various forms of information, such as data, voice and video. Integration of mobile devices into the existing internetwork consisting of stationary hosts has played a critical role in bringing the state-of-the-art a step closer to realizing this vision. At the same time, continued use of legacy technology in conjunction with wireless networks has given rise to certain problems due to the requirements of the portable mobile devices and the special characteristics of the wireless link.

Portability is an important element of mobile computing. As mobile devices shrink in physical parameters (size and weight), their computing and power capabilities are also reduced, constrained by battery capacity. Unfortunately, battery technology has been improving only at a modest pace in terms of increased capacity per unit weight and volume [3]. To maintain a constant level of performance of mobile applications, there is a need to minimize processing load on portable mobile devices [4].

The primary focus of our work is the design of transport protocols in computing-limited wireless environments. We devise a lightweight protocol architecture, Mobile TCP (MTCP), that allows the emulation of TCP functionality between a mobile host and a stationary host, in a way that reduces the processing load on the mobile host and minimizes the use of the wireless medium. Most of our protocol enhancements stem from the fact that the communication between the mobile host and the basestation is over a single link. Thus, the design of MTCP resembles that of a link-layer protocol. Consequently, as we see in this paper, many functions can be either simplified or totally eliminated in the wireless segment of the TCP connection, leading to a lean protocol code on the mobile machine. Furthermore, our implementation streamlines the protocol stack to additionally offload processing from the mobile devices.

Our protocol implementation is based on the split-connection model, where the connection between the mobile host and the basestation is split into two connections: the connection between the fixed host and the basestation, and the connection between the basestation and the mobile host. The first connection is called the wired segment, while the second portion is the wireless segment. The division, by itself, is not a new idea and was already employed in

previous studies [5,6]. We chose to employ the split-connection approach because it provides a convenient proxy-style model to enable the development of a new lightweight protocol on the wireless segment. It should be noted, however, that MTCP could be implemented using other approaches as well. We address this point in Section 6.

In addition to reducing the communication overhead, MTCP incorporates various robust yet simple techniques to combat heavy losses over the wireless link. Losses are prevalent on networks with wireless links due to various factors, such as noise, interference, channel fading, and user mobility. It has been identified that TCP performs poorly in wireless networks, since it assumes all losses occur as a result of congestion problems in the network. In response to a loss, TCP [7,8] takes various steps to throttle its transmission rate to alleviate congestion. However, this leads to a degraded performance in networks that exhibit non-congestive losses [9]. Several studies have been proposed in an effort to alleviate the effects of non-congestion-related losses on TCP performance in wireless or similar high loss links [6,10–12]. The split-connection model, which is the basis of our implementation, addresses this issue by separating loss recovery on the wireless segment from the wired segment. Thus, in this paper, our aim is *not* to solve the congestion issue, but rather to improve loss recovery on the wireless segment.

In accordance with the goals of our research, we evaluate the performance of MTCP in terms of its ability to offload processing from mobile devices, as well as its ability to improve throughput in lossy conditions. Our measurements, which are based on metrics such as throughput, protocol latency, CPU-usage, CPU energy consumption, and inter-buffer delay, indicate that our protocol is indeed effective in addressing both of these goals.

The rest of this paper is organized as follows. Section 2 briefly summarizes some previously conducted work on energy conserving protocols for portable devices. This section also outlines some solutions that have been proposed to address the performance issues of TCP over networks with wireless links. In Section 3, we present the design of MTCP followed by some implementation details. Section 4 describes our experimental testbed, including the error model and performance metrics for evaluating the protocols. Section 5 presents the performance results and analysis of several experiments. In Section 6, we propose some alternative approaches that can be used to implement MTCP. We discuss our future plans in Section 7 and conclude with a summary in Section 8.

2. Related work

Considerable research work in the past has focused on improving the performance of TCP over wireless and other lossy links, as it is the most dominant protocol in the Internet today. Some work has also been undertaken in developing energy efficient protocols and techniques for portable devices. In this section, we present a brief overview of each area of research.

2.1. Energy efficient protocols and techniques

Many portable devices today are equipped with power-saving components. Two such components are the processor and the wireless communication device [3]. Processors typically offer two types of power saving features. One power-saving feature is the ability to slow down the clock rate. Energy consumption (E) is essentially proportional to the switching capacitance (C), and the square of the voltage (V) ($E \propto CV^2$). Consequently, energy consumption can be reduced with a decrease in clock rate or by a reduction in voltage. Various software techniques can be employed to dynamically change the speed of the clock [13,14]. The general strategy is to dynamically adjust the CPU clock depending on the processor utilization during a particular time interval. Another power-saving feature of the processor is the ability to shutdown the processor when the system is idle and turn it back on when the next interrupt occurs. A well-designed operating system can deduce whether the system is idle using the current state of all processes. Whenever all the processes are blocked, the processor is turned off and subsequently restarted upon the occurrence of an interrupt [15–17]. Examples of operating systems that use this strategy are Windows [18,19] and UNIX. Wireless communication devices also provide similar power-saving features that enable the devices to enter low-power consumption mode during periods of inactivity.

From protocol design standpoint, general strategies to promote the transition of processors and communications devices into energy conserving modes can be stated as the following: minimize the processing complexity of

protocols, and decrease the overall use of the network device [3,20] without sacrificing any functionality. Various power-saving techniques have been proposed at different layers of the protocol stack.

A link-layer error control scheme, proposed by [21], conserves energy by reducing the data transmission rate when the channel is impaired, so that the number of unnecessary transmissions is minimized. While the channel is presumed to be impaired, short probe packets are continuously sent until some feedback is received for these packets. The protocol proceeds with the transmission of data packets upon the receipt of the first feedback packet indicating an improvement in the link quality. This scheme has been shown to be more energy efficient, however, with a slight loss in throughput.

Another link-layer scheme, proposed by Lettieri et al. [22], follows an adaptive error control strategy that uses Forward Error Correction (FEC) and Automatic Repeat reQuest (ARQ) to minimize energy consumption. ARQ mechanisms, such as Go-Back-N (GBN) and Selective Repeat (SRP), achieve reliability by retransmitting lost packets. Although ARQ mechanisms are relatively simple, they can cause a significant number of retransmissions when losses are excessive. FEC, on the other hand, reduces retransmissions by correcting erroneously received packets, however, at the cost of greater computational load, latency, and overhead in the form of additional bits. The adaptive error control scheme investigates the tradeoff between ARQ and FEC, and concludes that optimum energy savings can be achieved through an adaptive adjustment of the two mechanisms.

AIRMAIL [23] is a reliable link-layer protocol, which uses asymmetric protocol design to reduce processing load on the mobile device, while placing much of the complexity on the basestation. This protocol also minimizes the amount of data transmitted over the wireless channel by combining several acknowledgments into a single one, thus conserving power at the cost of increased latency. In addition, AIRMAIL employs FEC to minimize the energy expended on retransmissions in highly lossy conditions.

Another approach to minimizing the utilization of the communication device is to use header compression. In [24], the authors propose a low-loss header compression technique for TCP/IP, which significantly improves throughput performance of TCP. Their study also indicates that Van Jacobson header compression algorithm [25] can cause throughput degradation in lossy conditions due to inconsistencies in the compression-state.

Another energy conserving strategy is to use a medium access protocol that dictates in advance when each wireless device may receive data. This allows each device to sleep when no data is to be received. Variations of this technique are employed in 802.11 LAN standard [26], LPMAC [27], POCSAG protocol, and Motorola's FLEX™ protocol [28].

Considerable work has also been done in the area of lightweight transport protocols. This work is motivated by the need to provide a low overhead protocol, without performance bottleneck for high-speed networks. To minimize processing bottlenecks, lightweight protocols employ various techniques, such as smaller header sizes, simpler flow and congestion control strategies, and partial or complete implementation of the protocol in hardware. Similar ideas can be applied for reducing the energy consumption and the processing overhead on mobile devices. Examples of experimental lightweight protocols include VMTP [29], LNTP [30], high-speed transport protocol [31], Delta-t [32], and XTP [33].

2.2. Wireless TCP optimizations

Figure 1 shows a typical network model for wireless networks. There are fundamentally three different approaches to improving TCP performance in the presence of non-congestive losses, such as those on wireless links [34]. The first approach is to use a reliable link-layer protocol on the wireless link in order to hide wireless losses from the TCP sender. Link-layer solutions recover from losses either by retransmitting lost packets (ARQ) or by correcting erroneously received packets (FEC). Although link-layer protocols fit well into the layered structure of network protocols, they can cause adverse interactions with the error recovery mechanisms of higher layer protocols, such as TCP. Specifically, competing retransmissions between the link-layer and the transport-layer caused by incompatible timer settings can result in redundant retransmissions [35]. Another concern is that error recovery mechanisms (such as ARQ, FEC, checksums) are duplicated at both the layers to provide reliability. This duplication of effort can impose a greater processing demand on mobile devices. TCP-aware link-layer

solutions, such as Snoop [11], solve some these problems. Snoop introduces an agent at the basestation to cache TCP segments sent across the link. The agent detects losses by leveraging upon acknowledgments sent by TCP. This way it avoids additional control exchanges to perform error recovery. Upon detecting a packet loss, the Snoop agent retransmits the lost packet from the local cache and suppresses any duplicate acknowledgements heading toward the TCP sender in order to shield it from non-congestive losses.

The second approach is to split the end-to-end TCP connection into two connections: one instance of TCP operates over the wired segment and another instance of TCP or a different protocol operates over the wireless segment. A transport layer proxy (referred to here as the *Redirector*) at the basestation bridges the two segments together. An important consequence of the split-connection model is the separation of loss recovery on the wireless segment from the wired segment. As a result, losses over the wireless link can be recovered locally, while shielding the TCP sender on the fixed host from such losses. Indirect TCP (I-TCP) [5,10] is a split-connection solution that uses standard TCP over the wireless segment. The design of I-TCP is particularly intended for shielding TCP from packet losses during handoffs. I-TCP exploits the fact that lost packets are buffered at the basestation in the split-connection model. As part of the handing off process, I-TCP transparently moves the state of all connections (including send/receive buffers) associated the mobile from the old basestation to the new one. This is followed by an immediate retransmission and recovery of unacknowledged packets in the connection buffers, without the knowledge of the TCP sender on the fixed host. Studies, however, show that I-TCP is ineffective at recovering from wireless losses due to its choice of standard TCP for the connection over the wireless links [34]. Alternatively, a transport protocol with faster recovery mechanisms can be substituted, as in [6]. One problem with the split-connection model is that end-to-end semantics of TCP acknowledgments (ACKs) are not preserved. Specifically, ACKs for segments can now reach the source even before packets are delivered to the mobile host. However, proponents of this approach argue that most applications using TCP also have some kind of support for application-layer acknowledgments and error recovery [10]. In [12], the authors develop a protocol that does not violate the end-to-end semantics of split-connection approach by taking advantage of the persist state in TCP. Specifically, the source is completely throttled by shrinking the advertised window when the mobile host is disconnected, and activity is resumed as soon as connectivity is restored. Another technique called TCP splicing [36] preserves semantics by designing the transport proxy in such a way that ACKs (destined to the source) are not generated until the receiver acknowledges the forwarded data.

The third class of solutions, the end-to-end schemes, improve TCP performance by distinguishing between congestive and non-congestive losses [34]. Specifically, an agent residing on the basestation marks all duplicate ACKs corresponding to lost packets in order to explicitly notify the sender of non-congestion losses. The sender uses this information to decide whether to invoke congestion control mechanisms or not. One problem with the end-to-end schemes is that they are often slow to detect and recover from losses, since recovery is not performed locally. This results in considerable throughput degradation in WAN environment, where the round-trip delays are much greater than in a LAN [10,34]. Another major problem with the end-to-end approach is the need to modify the existing TCP implementations on the fixed host, making the solution impractical for wide scale deployment.

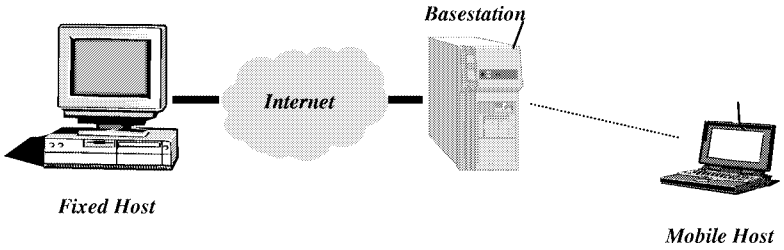

Fig. 1. Wireless network model.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.