ENHANCING ENERGY EFFICIENT TCP BY PARTIAL RELIABILITY

L. Donckers, P.J.M. Havinga, G.J.M. Smit, L.T. Smit

University of Twente, department of Computer Science,
PO Box 217, 7500 AE Enschede, the Netherlands
havinga@cs.utwente.nl

Abstract – In this paper we present our study on the effects on energy efficiency of a mobile system by enhancing our energy-efficient TCP variant (E^2TCP) with partial reliability. Partial reliability is beneficial for multimedia applications and is especially attractive in wireless communication environments. It allows applications to trade a controlled amount of data loss for a higher throughput, lower delays, and lower energy consumption. In the design of E^2TCP we provide solutions to four problem areas in TCP that prevent it from reaching high levels of energy-efficiency. We have implemented a simulation model of the protocol, and the results show that E^2TCP has a significant higher energy efficiency than TCP.

I. INTRODUCTION

The Transport Control Protocol (TCP) is a reliable, end-to-end, transport protocol that is widely used to support various applications. Despite its success as a transport protocol, there are certain communication environments and types of applications for which the design of TCP is less suitable. Prominent examples are wireless communication environments and multimedia applications.

Although TCP is very efficient on wired networks, it has been shown to perform poorly on wireless networks in both performance and energy efficiency, which are the two most prominent issues in current and future mobile systems. In the presence of a high packet error rate and periods of intermittent connectivity of wireless links, TCP may overreact to packet losses, mistaking them for congestion. TCP responds to all losses by invoking congestion control and avoidance algorithms. Another area in which TCP is less suitable, is for those types of applications that can tolerate a limited amount of data loss. Examples are multimedia applications that encompass transfer of audio, video and image data. They place very different requirements on the network compared to what the network was originally designed for. Most multimedia applications require a service with high throughput, low network delays, and low variations in these delays. They can however, tolerate a limited amount of data loss. TCP offers full reliability through the use of retransmissions, but at the expense of increased delays and lower throughput.

To address part of the problems with TCP and UDP the partial reliability transport protocol (PRTP) [1] was developed. PRTP provides a partially reliable service, i.e. a service that does not insist on recovering all, but just some of the data loss. As compared to TCP, PRTP enhances the service by allowing applications to trade a controlled amount of data loss for higher throughput, lower delays, and higher energy efficiency. The transport service provided by PRTP could be especially beneficial when the channel is lossy and round trip times are non-negligible, as can be the case in a wireless environment.

Studies on the *energy efficiency* of TCP have been very limited so far. E.g. [9] analyses the energy consumption performance of various versions of TCP. They argue that energy efficiency can be improved by avoiding periods of bad channel conditions. In fact, this is exactly what the window adaptation algorithm of TCP does. However, since their model is based on many simplifications, it is unclear how accurate their results are. Moreover, in their analysis they only considered the energy spent in transmitting data, and did not consider energy consumption while the interface is idle. Tsaoussidis et al. [8] compared the energy performance of several TCP variants. They make fewer simplifications, and their results are based on simulations. Their results indicate that the energy performance of the various TCP variants is fairly similar.

In this paper we present our study on the effects of enhancing our energy efficient TCP variant (E^2TCP) [5] with PRTP. We will here merely introduce the essential background of E^2TCP, and present simulation results and compare other TCP variants with E^2TCP, taking into account both energy spend in communication as well as spend in idling. Then we will show the effects on energy efficiency when partial reliability is used.

A. E^2TCP objectives and assumptions

E^2TCP is suitable for the wireless link between base station and mobile host in a split-connection scheme. In a split-connection scheme errors caused by the wireless link can be addressed near the site of their occurrence. The protocol makes no assumptions about the link- and media access control (MAC) layers, or requires any interaction between them. Therefore, the protocol can be used as a drop-in replacement for TCP/IP on the mobile host and should be usable on all wireless links. All modifications to TCP to extend it with PRTP are localized at the receiver.

B. Energy consumption

The wireless network interface of a mobile computer consumes a significant fraction of the total power [7].

Basically there are two characteristics that influence the energy efficiency and overhead of a protocol. The first characteristic is the *data overhead of a protocol*. When a protocol uses more bytes to transmit the same amount of data, more bytes are wasted, and thus the protocol becomes less energy efficient. The second characteristic that influences the energy efficiency of a protocol is *time overhead*. In general, the longer the protocol needs to transmit the same amount of data, the longer the radio has to be active, and consequently the more energy is wasted.

C. *Metrics*

For a given data transmission medium there is a minimum amount of energy M that is required to send data from source to destination. The actual spent energy is called S. The difference between those two values is called W: the amount of wasted energy. *Energy efficiency* ε then is:

$$\varepsilon = M / S \qquad (1)$$

However, energy efficiency is not always a good metric to compare various protocols since it is related to the amount of data. Therefore we introduce another metric that is closely related to energy efficiency: *energy overhead* o. Energy overhead is the amount of wasted energy compared to the minimum amount of energy, or:

$$o = W / M \qquad (2)$$

II. E^2TCP

Because E^2TCP is derived of TCP, its architecture and mechanisms are roughly the same. On four points, however, adjustments were made to increase the energy efficiency of the protocol. These points are the reliability requirements, the headers, the acknowledgements, and the window management [4].

A. *Partial reliability*

When transmitting streaming media, energy can be saved when unwanted retransmits can be avoided. Partial reliability provides a way to do this, by enabling the application to set the minimum desired reliability of the channel.

The implementation of partial reliability in E^2TCP is rather straightforward. PRTP is completely receiver-based, which means that all necessary modifications to TCP are localized to the receiver. The receiver keeps track of how much data was successfully received and how much was lost. An application is able to set a certain amount of reliability for each connection with a Quality of Service-like parameter. This parameter: the *reliability level*, can be set from 0% to 100% in one percent steps. When the overall reliability does not drop below the limit, the receiver will not ask for a retransmission and sends an ack. When the receiver encounters lost packets, it checks its current reliability level. If it is still above the specified limit, the receiver will falsely acknowledge as much lost packets as possible without violating the reliability demand, so the sender will not retransmit them.

B. *Header format*

E^2TCP headers are general much smaller than standard TCP headers, and are still robust for errors. Reducing the size of the header not only implies that less data has to be transmitted; it also makes the packet less susceptive to errors. Both effects increase energy efficiency.

The E^2TCP header may contain some of the standard TCP fields like source- and destination IP and port numbers, sequence and acknowledgement numbers, window, urgent pointer, and checksum. Because the source- and destination IP addresses and ports are large, and will not change during a connection, they will only be sent during connection startup. Until the connection is terminated, a one-byte connection identifier will be used instead. This is comparable to other header compression mechanisms (e.g. [2]). Furthermore E^2TCP has fields to indicate SACK blocks to support selective acknowledgement (SACK). Another optimization is realized by having a special field that indicates whether a certain option is included in the header or not. Because almost all fields in the E^2TCP headers are optional and only need to be transmitted when they are required, E^2TCP headers are usually quite small, just 8 bytes. Normal acknowledgements will have a size between 8 and 16 bytes depending on how many SACK blocks are used. This is considerably less than TCP acknowledgements that have a size of 40, 50 or 60 bytes (with none, one and two SACK blocks respectively) up to a maximum of 80 bytes if more options are used.

C. *Selective acknowledgements*

Standard TCP can only generate positive cumulative acknowledgements. This means that when the end station receives an out-of-order packet (due to packet reordering or packet loss) it is unable to send this information to the sender. Selective acknowledgements (SACK) can convey information to the sender about multiple non-contiguous blocks of successfully transmitted segments [6]. E^2TCP not only supports SACK but also relies on them to effectively increase its energy efficiency. Because E^2TCP will work on a *single-hop* link and performs *local* retransmissions, it will know when a packet is received out of order, that the intermediate packets were lost. Upon noticing out of order packets, the receiver will indicate to the sender (with SACK), that it has not received the intermediate packets. The sender can immediately retransmit the lost packets and does not have to wait on timeouts or duplicate acknowledgements. This will reduce the time overhead of E^2TCP without increasing the data overhead.

D. *Window management*

E^2TCP features a window management scheme that is optimized for energy efficiency on wireless single-hop links. The window management mechanism of E^2TCP differs on four points from TCP.

- First of all, E^2TCP features *immediate retransmits*. When the receiver indicates it has received an out-of-order packet, the sender can immediately retransmit the missing packets, because E^2TCP will be used on a single-hop link and no packet reordering can take place on such a link. Under the same conditions standard TCP would wait on a timeout before it would retransmit the lost packet, causing substantial delays.
- The second change is that E^2TCP takes into account the *error characteristics of the wireless channel*. If few errors occur, E^2TCP considers this to be the result of normal random errors on the wireless link. When lots of errors occur, E^2TCP considers this to be caused by a burst error and *drastically* reduces its transmission speed since the next packets are likely to be lost anyway. This way, E^2TCP reacts to (burst) errors in a very energy efficient way.
- E^2TCP also features a *minimum window size*, which is the third point on which the window management of TCP and E^2TCP differ. This minimum window size causes E^2TCP to quickly recuperate after a burst error, which will decrease time overhead.
- The final change to the window management of TCP is the use of *a retransmission timer*. The timers used in E^2TCP are similar to the transmission timer in TCP, only one is used for transmissions and one is used for retransmissions. The extra timer increases the responsiveness of the protocol to changes on the channel and thus decrease time overhead.

III. SIMULATION

A. *Simulation setup*

To measure the performance and energy efficiency of E^2TCP and compare the protocol with other versions of TCP (Tahoe, Reno and NewReno), an implementation of E^2TCP was made in ns2. The simulation setup consists of two hosts connected by a wireless LAN. Each host is running TCP and together they create one TCP connection that connects both hosts. The *default simulation setup* has a bandwidth of 1 Mbps. The default delay will be 50 ms, which is an estimation of the delays introduced by a typical IEEE 802.11 physical layer, link layer and MAC layer combined, based on measurements by [3]. The default-simulated traffic is a mass data transfer of 20 MB in total.

B. *Error model and setup*

All packets on the wireless LAN are transparently routed through the error model, which randomly corrupts the packets with a chance that corresponds to the error rate of the state it is currently in. We use a two-state error model with a *good state* that resembles a high quality channel with some modest random noise and a *bad state* that resembles a burst error with a very high error rate. The state will switch between the states after a certain time.

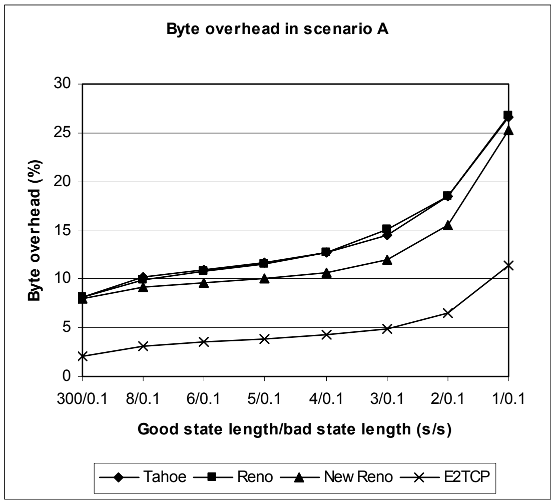

Figure 1: Data overhead of various protocols in scenario A.

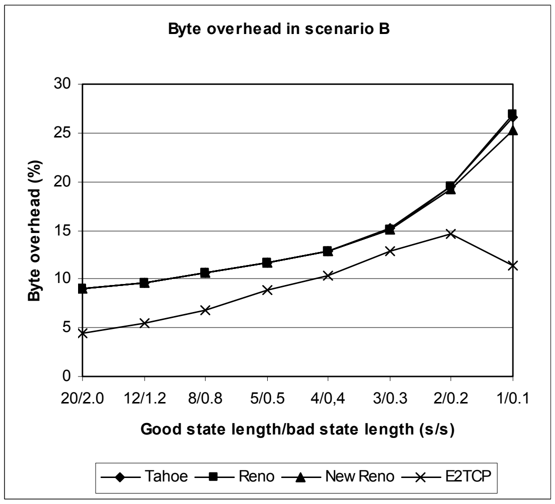

Figure 2: Data overhead of various protocols in scenario B.

We have performed all our simulations using two Scenarios, which represent two quite different error characteristics. The first Scenario (*Scenario A*) has a fixed bad state length of 0.1 second and the good state length varies from 300 seconds to 1 second. This corresponds to a nearly perfect channel (the simulations finish within 300 seconds of simulated time) to a very bad channel. In this Scenario the proportions between the good state and bad state length are gradually worsened. In the other Scenario (*Scenario B*) the good state lengths vary from 20 to 1 second, with the bad state length always being one tenth of the good state length. This allows the protocols energy efficiency to be examined with varying bad state lengths while the proportions between the good state and bad state length remain the same.

C. Data and time overhead of E^2TCP without Partial Reliability

As discussed before, energy efficiency is influenced by both time overhead and data overhead. In this simulation setup, we evaluate these issues separately, and with full reliability.

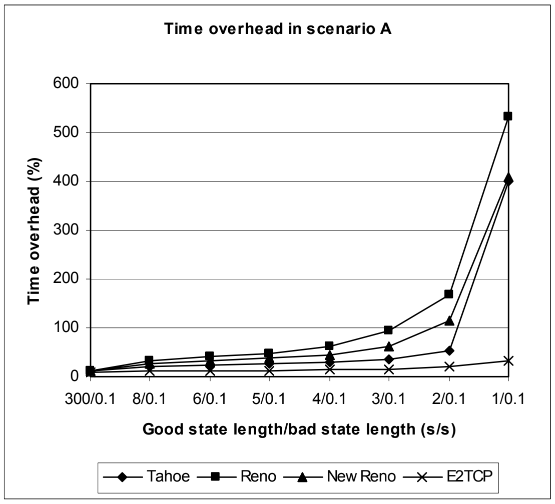

Figure 3: Time overhead of various protocols in scenario A.

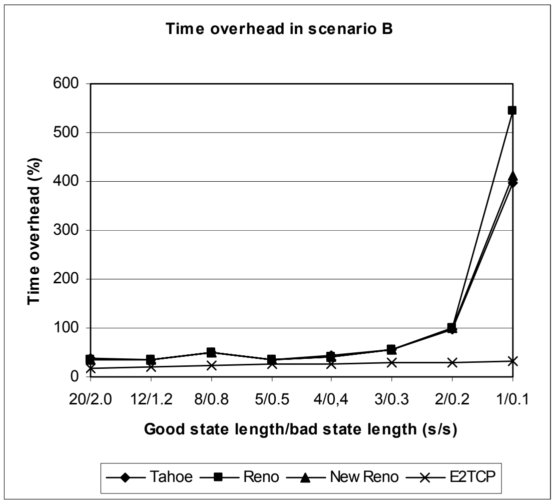

Figure 4: Time overhead of various protocols in scenario B.

As can be seen in Figure 1 and 2, E^2TCP has less *data overhead* than the other TCP versions, in both scenarios, at all points. This can be attributed to the small headers and its optimized window management in combination with selective acknowledgements. Especially in scenario A it is clear that when the quality of the channel deteriorates, the data overhead increases. It is interesting to note the decrease in data overhead in scenario B for E^2TCP at the right side of the graph. Unlike standard TCP, E^2TCP does not decrease its transmission speed for small burst errors, resulting in a very low data overhead.

The *time overhead* of E^2TCP (Figure 3 and 4) is far less than other protocols in all cases. Especially when the quality of the channel deteriorates (the right side of the graphs), the difference in time overhead between E^2TCP and the other protocols increases. This means that (considering time overhead) E^2TCP scales much better than the other protocols when the quality of the channel gets worse.

It should also be clear that the other versions of TCP have much more time overhead than data overhead. Note that all three standard TCP versions behave the same, which is in line with the results presented in [8]. Since Tahoe tends to perform slightly better, in the following simulations we will compare E^2TCP with Tahoe only.

D. Energy overhead of E^2TCP with partial reliability

In this simulation the partial reliability of E^2TCP will be examined. The default setup will be used with the following protocols: Tahoe, PRTP and E^2TCP. Tahoe is of course fully reliable. E^2TCP will be set to 95% and 90% reliability while PRTP will be used at 90% reliability.

It is only useful to use partial reliability on certain types of traffic. Streaming media applications and sometimes mass data transfer (images, audio and video) applications are suited to adapt to partial reliability. Therefore only the constant bit rate and mass data transfer models are used in this simulation. The energy overhead for both scenarios is shown in Figures 5 and 6.

From the graph of scenario A, a few things can be deduced. First of all, PRTP (with a reliability of 90%) clearly has less energy overhead than Tahoe. Another interesting thing to note is that both PRTP and E^2TCP at 90% reliability score (almost) the same independent of the quality of the channel. Because of the loose reliability constraints both protocols can deal very efficiently with errors. E^2TCP with 95% reliability clearly has more trouble when the quality of the channel worsens because the reliability constraints are tighter. Still it manages to surpass PRTP in all but the worst channel conditions.

In scenario B, the last point is also valid. That is: PRTP is more efficient than Tahoe, while E^2TCP with a reliability of 95% surpasses the performance of PRTP in all but the worst channel conditions. Just like in scenario A, E^2TCP with a reliability of 90% is the most energy efficient protocol.

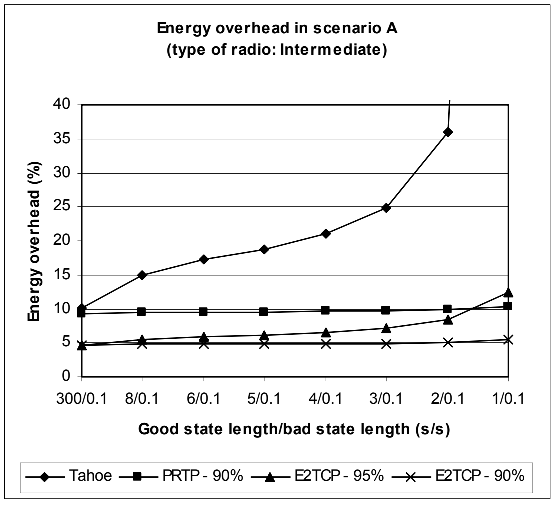

Figure 5: Energy overhead of various protocols in scenario A.

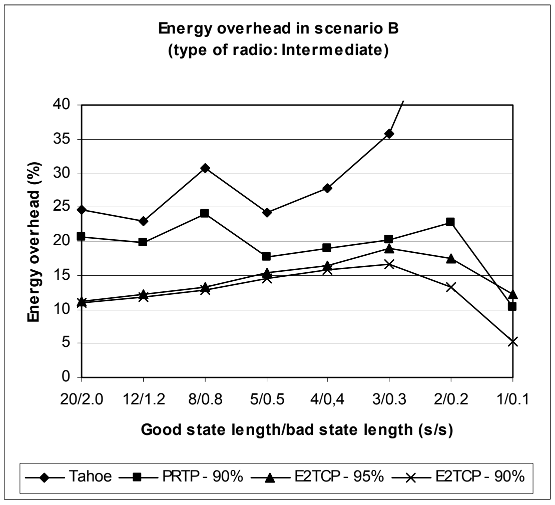

Figure 6: Energy overhead of various protocols in scenario B.

IV. CONCLUSIONS AND RECOMMENDATIONS

E^2TCP is optimized for energy efficiency of wireless communications on four points. The first point is the acknowledgement scheme of TCP, which is unable to provide the sending host with enough information about the state of the destination host. E^2TCP uses an *efficient* selective acknowledgement mechanism to overcome this problem. These selective acknowledgements are also required for the second optimization: the window management. This optimization is the result of efforts to make TCP aware of burst errors. Because burst errors are a major cause of packet loss on wireless links and TCP considers all packet loss to be the result of congestion, TCP was unable to react to burst errors in an energy efficient way. These two optimizations cause the greatest decrease in energy overhead: about 75% of the total decrease in energy overhead. The third optimization is the use of partial reliability to limit unwanted retransmits during the transmission of streaming media. This optimization is the cause of about 13% of the total decrease in energy overhead. The final optimization is the use of custom headers, which rely on techniques from header compression standards to minimize wasted energy. This optimization is the cause of the last 12% of the total decrease in energy overhead.

E^2TCP has been compared to standard versions of TCP, like Tahoe, Reno and NewReno. From the results can be concluded that E^2TCP has less energy overhead than TCP for each bandwidth/quality of channel combination. In most cases TCP even has an energy overhead that is at least twice as large as that of E^2TCP. Also, E^2TCP scales better than TCP when channel conditions deteriorate. Other simulation, not presented here due to lack of space, show that also on traditional metrics like throughput and latency E^2TCP outperforms the others easily.

REFERENCES

[1] Brunstrom A., Asplund K., Garcia J., “Enhancing TCP performance by allowing controlled loss”, *Proceedings of SSGRR 2000 computer & e-business conference*, L’Aquila, Italy, August 2000.

[2] Casner S., Jacobson V., “Compressing IP/UDP/RTP headers for low-speed serial links”, *RFC 2508*, February 1999

[3] Chen K., “Medium access control of wireless LANs for mobile computing”, *IEEE Network magazine*, V. 8 N. 5, September 1994.

[4] Donckers L.: “Energy Efficient TCP”, MSc thesis University of Twente, department of Computer Science, 2001.

[5] Donckers L., Havinga P.J.M., Smit G.J.M., Smit L.T.: “Energy Efficient TCP”, *Proceedings 2nd Asian International Mobile Computing conference (AMOC2002)*, Malaysia, ACM Sigmobile, ISBN 983-40633-1-8, pp 18-28, May 2002.

[6] Mathis M., Mahdavi J., Floyd S., Romanov S., “TCP selective acknowledgement options”, *RFC 2018*, October 1996.

[7] Stemm, M, et al.: “Reducing power consumption of network interfaces in hand-held devices”, *Proceedings mobile multimedia computing MoMuc-3*, Princeton, Sept 1996.

[8] Tsaoussidis V., Badr H., "Energy / Throughput Tradeoffs of TCP Error Control Strategy", *IEEE Symposium on Computers and Communications, IEEE ISCC 2000*, France, 2000.

[9] Zorzi M., Rao R.R.: “Is TCP energy efficient? ”, *Proceedings IEEE MoMuC*, November 1999.