Comparison of ECN-ELFN and SACK on TCP's Performance for Ad Hoc Networks*

H. Singh, S. Saxena, and S. Singh
Department of Computer Science
Portland State University
Portland, OR 97207

ABSTRACT

In this paper we study the *energy cost* (protocol processing and communication cost) and *goodput* of two different flavors of TCP in ad hoc networks. We implemented a testbed and measured the actual energy cost as well as goodput of running TCP-SACK in ad hoc network scenarios. We also implemented Explicit Link Failure Notification (ELFN) [1] and Explicit Congestion Notification (ECN) [2] in Newreno and measured its performance. We see that the use of ECN & ELFN does yield higher goodput in most cases with a corresponding lower total energy cost. We see an energy savings of between 20% and 500% depending on the network conditions.

Categories & Subject Descriptors

C.4 [Performance of Systems]: Measurement Techniques; D.4.4 [Operating Systems]: Communications Management.

General Terms

Performance, Experimentation.

Keywords

Mobile, Protocol, TCP, Energy.

1. INTRODUCTION

In ad hoc networks, communication plays a significant role in the deployed applications and thus accounts for a large proportion of the overall energy usage. Since energy is the key constraint that determines the useful life of ad hoc networks, it is important to reduce the communication energy cost. Various techniques have been proposed for reducing the communication energy cost including transmission power control, using directional antennas, adapting data rates, MAC protocols that power the radio off, and routing protocols that use energy-based routing metrics. While all of these approaches clearly reduce the cost of communicating we believe that additional savings are possible at the TCP layer as well. In this paper we explore the energy cost of TCP connections by comparing two variations of TCP for reducing this cost.

The two protocols we compare are TCP-SACK and TCP-ECN-ELFN. TCP-SACK was shown to be better [3] than Reno and Newreno in reducing the energy cost of TCP connections in multi-hop wireless networks under most networking scenarios. However, [3] did not consider the impact of routing failure on TCP-SACK. As we show in this paper, routing failure can result in dramatically reduced performance of TCP-SACK. The second protocol, TCP-ECN-ELFN, is based on previous proposals of ECN [2] and ELFN [1] developed by other researchers. In ELFN, the TCP timers are frozen until the network layer informs TCP that a new route has been found to the destination. ECN has been proposed as a mechanism to enable TCP senders to quickly respond to incipient congestion in the network. We studied the performance of these two variations of TCP in a testbed. We measured the goodput of the protocols as well as the total energy and idealized energy consumed for ttcp data transfers. The idealized energy corresponds to the energy consumed by the sender when transmitting or receiving or processing but does not include the idle energy of the node. We observe that under most ad hoc networking scenarios, TCP-ECN-ELFN results in significantly lower energy consumption as compared with TCP-SACK and it also has a higher goodput.

The remainder of the paper is organized as follows. In the next section we discuss related work; section 3 develops the energy model for characterizing protocol cost; section 4 presents the details of ECN and ELFN that we incorporated into TCP's implementation; we used a hybrid approach for measuring TCP's energy and this is described in section 5; results are presented in section 6 where we compare the TCP-ECN-ELFN protocol against TCP-SACK.

2. RELATED WORK

There have only been a few papers dealing with the problem of TCP's energy consumption over wireless links. Some of these papers propose link layer solutions while others compare various versions of TCP with respect to the energy cost. The link layer approaches include [4] who consider the effect of ARQ, FEC and a combination of the two on energy consumed in ad hoc networks. Unlike our work here,

*This work was funded by DARPA under contract number F33615-C-00-1633.

MSWiM'02, September 28, 2002, Atlanta, Georgia, USA.

however, this paper was primarily concerned with link layer schemes to improve TCP's energy behavior. [5] also considers the effect of ARQ strategies on energy consumption. The key idea is to suspend packet transmission when channel conditions worsen and probing the channel state prior to packet transmission. When channel conditions improve, packet transmission is resumed.

[6] compares the energy and throughput-efficiency of TCP error control strategies for three implementations of TCP Tahoe, Reno, and New Reno. They implemented the three versions of TCP using the x-kernel protocol framework and their focus was to study heterogenous wired/wireless environments. [7] analyzes the energy consumption performance of various versions of TCP for bulk data transfer in an environment where channel errors are correlated. Interestingly, the energy cost is modeled as the ratio of the number of successful transmissions to the total number of transmissions. This paper does not consider processing or other costs associated with running a higher layer protocol. Furthermore, the paper only considers a one-hop wireless link with zero propagation delay.

Our study is different in many ways, first while all the previous studies were based on simulation, we measured energy using a wireless test-bed (3-hop ad hoc network) and a real TCP/IP stack (FreeBSD); second, in all the previous studies total energy is considered to be the primary metric. Unfortunately, total energy includes the energy consumed when the connection is idle and this can be the dominating factor in computing this metric. We therefore also measured the actual protocol processing energy, which excludes the connection idle periods.

3. ENERGY COST OF TCP

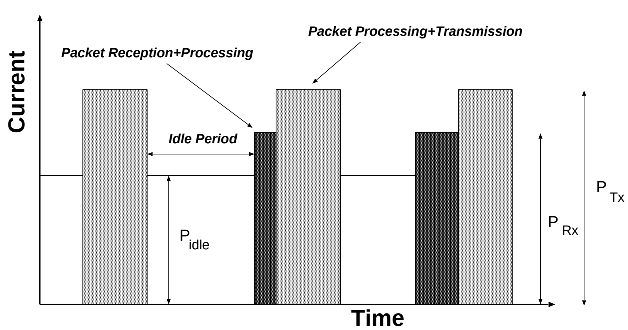

Figure 1: Simplified energy consumption profile.

Figure 1 shows a simplified view of the evolution, in time, of the energy consumption of a TCP sender. In the figure, we plot the instantaneous current draw as a function of the time. As we show in the figure, the node consumes P_{Idle} amount of Power (watts) when idle (essentially waiting for packets or ACKs), P_{Tx} Power in *processing and transmitting* TCP segments, and P_{Rx} Power in *receiving and processing* TCP ACKs. Let t_{total}, t_{Tx}, and t_{Rx} denote the total time of the connection, the time spent by the node in processing and transmitting packets, and the time spent in receiving and processing ACKs. Then, the total energy consumed by the TCP connection is[1],

$$\begin{aligned} E_A &= P_{\text{Idle}}(t_{\text{total}} - t_{Tx} - t_{Rx}) + P_{Tx}t_{Tx} + P_{Rx}t_{Rx} \\ &= P_{\text{Idle}}t_{\text{Idle}} + E_{Tx} + E_{Rx} \\ &= P_{\text{Idle}}t_{\text{Idle}} + E_I \end{aligned} \quad (1)$$

Where E_I denotes the energy consumed only for transmitting/receiving packets and the associated processing cost. In a sense, E_I denotes the *ideal* energy consumed if the node and the radio can be *powered off for exactly the duration of the idle periods*.

Next, assume that B bytes of data are sent during the lifetime of the connection at an average throughput of τ bytes/sec. If the transmission speed of the radio is r bytes/sec, we can write,

$$t_{\text{Idle}} \approx B/\tau - B/r - (64B)/(2Dr) \propto 1/\tau(sec)$$

where D is the packet size used, B/r is the time to transmit the packets, and $(64B)/(2Dr)$ is the time to receive the 64-byte ACKs (assuming one ACK is sent for every two packets). If we substitute $1/\tau$ for t_{Idle} in equation 1, we get,

$$E_A \propto P_{\text{Idle}}/\tau + E_I \propto E_{\text{Idle}} + E_I \quad (2)$$

As equation 2 shows, the total energy consumed by a TCP connection is inversely proportional to the connection's throughput and is proportional to the idealized energy E_I. In equation 2, we treat E_{Idle} as a constant but it is easy to see that its value has a significant impact on the measured total energy E_A. The value of E_{Idle} depends on the behavior of the node during periods when the node is idle. Typically, laptops and PDAs enter a sleep state when they have been idle for some period of time and wake up when an event occurs. The energy consumed when the node sleeps versus when it is idle but awake can be quite dramatic and can make a significant impact on any energy comparison.

4. TCP-ECN-ELFN

Table 1 summarizes the changes made to the operation of TCP to include ECN and ELFN. The table also describes the intuition behind these changes.

Routing Failure: Using ELFN

[1] describes the interplay between routing failure (due to link outage or propagation of stale routes) and TCP throughput, in detail.

Briefly, successive route failures (due to link failure) lead to timeouts hence resulting in a small congestion window.. Hence, the throughput of the connection is small. The fix proposed in [1] and used by us is as follows. A *route failure* message is propagated back to the TCP sender from the intermediate node that detects the route failure. This message has the effect of freezing TCP's state and initiating the transmission of probe packets. When there is a response to the probe packet (i.e., the route is up), TCP's state is unfrozen and transmission resumes. This solution ensures that there are no timeouts (and hence no unnecessary retransmissions), and that the TCP sender begins sending packets soon after the route is up.

Out-of-order Packets, Timeouts, & Triple Duplicate ACKs

[1]The assumption here and in the remainder of the paper, is that there are no other applications running over which the energy cost can be amortized.

Event	*TCP's Behavior*	*TCP-ECN-ELFN*
Routing Failure	Timeout, CWND ← 1 Retransmissions Exponential backoff timer	Freeze state Probe network Unfreeze when route restored
Triple Duplicate (TD) ACKs	Retransmit packet CWND ← CWND/2 + 3	Retransmit packet
Timeout	CWND ← 1 Retransmit Exponential backoff timer	Retransmit packet
Explicit Congestion Notification	No action	CWND ← CWND/2

Table 1: Summary of changes made to TCP.

Mobility of nodes can cause packets belonging to the same connection to be routed along different routes. This can result in the receiver getting out-of-order packets which causes duplicate ACKs to arrive at the sender. Likewise, packet loss due to link-layer errors can result in triple duplicate ACKs or timeouts. On receiving three duplicate ACKs, the sender reduces its congestion window by a half and retransmits the out-of-sequence packet while in the case of timeouts, the window is reduced to one or two segments. This congestion avoidance behavior has the net effect of reducing the throughput of the connection (due to the smaller congestion window) and thus increasing overall energy consumption. We believe that the appropriate fix for this problem is for the TCP sender to retransmit the offending packet but *not adjust its congestion window*. We made this modification to TCP-ECN-ELFN in our implementation.

Network Congestion: Using ECN

A problem with our approach above is that if the triple duplicates (or timeout) were generated as a result of packet drops due to congestion, then the solution of simply retransmitting the packet without reducing the congestion window will have negative consequences (this is the reason why TCP reduces its congestion window). In our design, we rely on *explicit congestion notification* [2] to signal imminent congestion along a route[2]. Here, a node whose buffer occupancy crosses some threshold, sets a bit (the CE bit) in all data packets it sees. Receivers reflect this flag back in the ACKs they generate by setting the ECN-ECHO bit. Upon receiving an ACK with the ECN-ECHO bit set, TCP senders enter a recovery phase in which they reduce the congestion window by a half. The sender sets a CWR (Congestion Window Reduced) bit in new data packets. If the receiver sees another CE bit set in a future packet and sees that the sender had sent a CWR bit, this indicates that there is still congestion in the network. The receiver again sets the ECN-ECHO bit in new ACKs thus forcing the sender to enter another recovery phase. This can go on until the sender's window has shrunk to one or two segments.

5. MEASURING ENERGY

Most of the research in ad hoc networking uses the ns2 simulator and to a lesser extent other simulators like glomosim [8]) to run experiments. The benefit of this approach is that researchers can build upon the work of others and use a standard platform to check competing ideas. While ns2 is a good tool for measuring traditional networking metrics such as throughput, loss, and delay, it is ill-suited to measure energy consumption of a protocol like TCP. This is because the energy consumed includes not only the radio costs (which are modeled to some extent in ns2) but the node-level protocol processing and data copy costs. An alternative idea would be to use a node-level energy simulator/emulator that gives fairly accurate energy readings for processing code. The problem, however, is that these tools do not simulate the ad hoc network environment. Thus, an idealized simulator would be one which combined a detailed node-level emulator and ns2. However, we are not aware of any such simulator that we could have used.

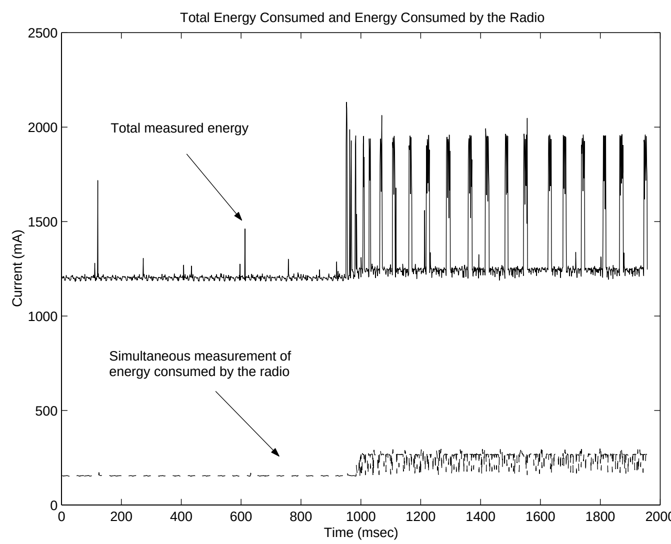

Figure 2: Sample energy trace: note simultaneous measurement of radio and system-level energy cost.

Given the above constraints, we decided to use a hybrid approach to measure the node-level TCP energy. Specifically, we used a 4-node network (see Figure 3) in which we measured the energy of the sender node directly using two Agilent 34401A multimetes (resolution of 1msec) – one measured the total system energy while the second measured the radio-level energy alone (Figure 2 shows a sample data trace). We also ensured that there is no other traffic was present on the channel. Each node in the network is a Toshiba laptop that has a Lucent 802.11 Silver (11 Mbps) WaveLAN DSSS PC card. Further, the two intermediate nodes are set up to act as routers. To simulate multi-hop ad hoc network behavior, we ran Dummynet [9] at node C. Dummynet[3] is a freely available kernel-level patch that al-

[2]ECN has been proposed as RFC 2481 to the IETF and has been put forward as a Proposed Standard for use over the Internet.

[3]Dummynet has been used by researchers and by industry

lows us to control a wide-variety of network behaviors such as delay, loss, and bandwidth. For instance, Dummynet can add delays to packets to simulate variable RTT, drop packets to simulate lossy networks, vary the bandwidth, and implement different queuing mechanisms[4]. We ran Dummynet with a setting of HZ = 1000, a kernel option that gives us a time granularity of 1msec. We also implemented RED (Random Early Detection) for AQM (Active Queue Management) to detect and report incipient congestion in order to implement ECN. All of the implementation was done in FreeBSD. *Ad hoc mode of these WaveLAN PC cards do not allow Transmit Power control or explicit data rate control, and hence we did not not investigate the impact of these on TCP throughput.*

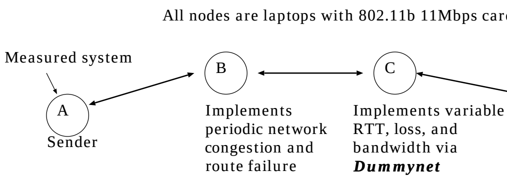

Figure 3: Measurement setup.

6. EXPERIMENTAL RESULTS

Since our ad hoc network is emulated as in Figure 3, we needed to feed appropriate values for various parameters such as bandwidth, loss probability, and RTT range into Dummynet. In addition, we needed to generate out-of-order packets and congestion scenarios to model similar scenarios in ns2-based simulations. To this end, we conducted simulations in ns2 and used the results of others to determine appropriate values for these various parameters. The final selection of parameter values was somewhat optimistic but seemed to cover a wide range of ad hoc network behavior. These values are summarized in Tables 2 and 3.

All experiments were conducted at least ten times and we computed 95% confidence intervals (which are also shown in the figures). Furthermore, in order to get statistically significant results, we transmitted 5M of data (TTCP flow) for each run (transmitting smaller amounts of data resulted in high measurement error due to the 1msec granularity of the multimeter). We measured E_A, E_I, and goodput for the connections. One note about the figures. We normalize the total energy measured (E_I or E_A) by the data transferred and plot the energy in units of micro-Joules per bit. The x-axis in all plots is the average RTT value.

The remainder of this section is broken in three: in section 6.1 we look at the impact of mobility-induced factors on protocol performance; in section 6.2 we consider the relative protocol performance when nodes do not move; finally, we summarize the main results in section 7.

6.1 Mobile ad hoc network case: *impact of mobility*

6.1.1 Routing Failure Case

to simulate a variety of network conditions in order to study protocol behavior in a controlled setting. .

[4]The overhead of running Dummynet on network parameters like RTT etc., is negligible because Dummynet does not perform data copies, it works with pointers only.

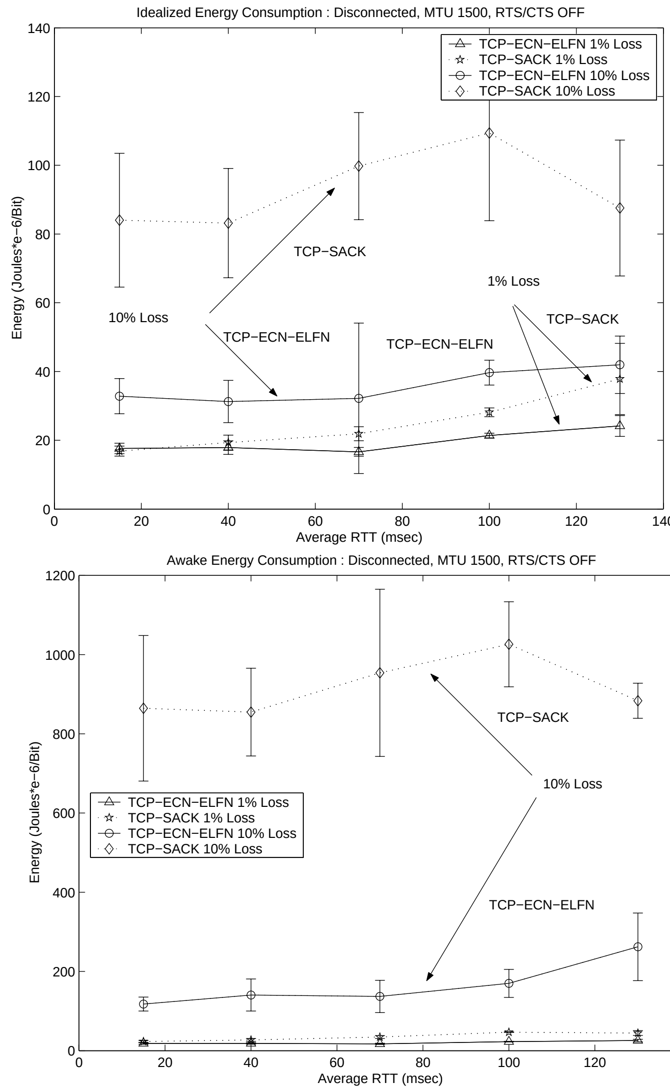

Figure 4: Idealized and awake energy cost for the route failure case.

We simulated routing failure by breaking the route for 5 seconds after every 15 seconds during the run. This was done for the three packet loss probabilities of 1%, 5%, and 10% and for all the different RTT ranges. Figure 4 plots the energy E_I and E_A for different RTTs. We have left out the 5% loss case from the energy graphs for clarity. We also left out the goodput plots because we observed that goodput is inversely proportional to E_A and the plots add nothing new. The TCP-ECN-ELFN protocol outperforms TCP-SACK on all measured metrics because of two reasons:

- Due to route failure, TCP-SACK has a very poor throughput (and goodput) due to many timeouts. In the TCP-ECN-ELFN protocol, on the other hand, an ICMP route failure packet has the effect of freezing TCP state and resuming it when the route is up. Thus, the goodput and energy E_A of the TCP-ECN-ELFN are better than TCP-SACK.

- TCP-SACK having no mechanism of detecting route failure ends up retransmitting many packets and it has

Parameter	*Values*	*Comments*
RTT	10-20ms, 30-50ms, 60-80ms, 90-110ms, 120-140ms	For a given experiment, we use one of the RTT ranges; each packet had a RTT randomly uniformly selected from this range
MTU size	512 and 1500 bytes	These extreme values explore the dependence of energy on MTU size
RTS/CTS	ON or OFF	We performed experiments with both cases but show graphs for the OFF case only (the ON case was similar)
Protocols studied	SACK and ECN-ELFN	SACK implemented in FreeBSD4.3 based on RFC2018; the TCP-ECN-ELFN also implemented in FreeBSD4.3

Table 2: Experimental parameters for all experiments.

Experimental Factors	*Values*	*Comments*
	Mobile ad hoc networks	
Route failure	Route down for 5 sec every 15 sec	These values are very optimistic and were selected because even at these values TCP's energy cost is significantly higher than the cost of the TCP-ECN-ELFN protocol
Packet reordering	1% and 5% packets reordered	Packets as well as ACKs reordered randomly; these values are dependent on the routing protocol and could be higher
	Static ad hoc networks	
End-to-end Packet Loss	1%, 5%, 10%	This range is somewhat optimistic and ignores some high loss cases
Bursty loss	85% loss for 1 sec every 12 sec	Models the case when the route fails at a node that has a buffer full of packets
Congestion	Router B congested for 5 sec every 30 sec	Two RTTs used:15 ms and 130 ms; this case shows that the TCP-ECN-ELFN protocol reacts correctly to congestion

Table 3: Summary of ad hoc network conditions studied.

	1% reordered	5% reordered
MTU 512	$E_I^{SACK} < E_I^{ECN-ELFN}$ $E_A^{ECN-ELFN} < E_A^{SACK}$ $\tau^{ECN-ELFN} > \tau^{SACK}$	$E_I^{ECN-ELFN} < E_I^{SACK}$ $E_A^{ECN-ELFN} < E_A^{SACK}$ $\tau^{ECN-ELFN} > \tau^{SACK}$
MTU 1500	$E_I^{SACK} < E_I^{ECN-ELFN}$ $E_A^{ECN-ELFN} < E_A^{SACK}$ $\tau^{ECN-ELFN} > \tau^{SACK}$	$E_I^{ECN-ELFN} < E_I^{SACK}$ $E_A^{ECN-ELFN} < E_A^{SACK}$ $\tau^{ECN-ELFN} > \tau^{SACK}$
ECN-ELFN has approx 10% less E_I and 20% less E_A		

Table 4: Summary of relative protocol performance for the reorder case.

  to do extra processing in maintaining SACK blocks thus has a higher *idealized enrgy* cost E_I as well. Further, this cost of maintaing SACK blocks becomes significant at higher losses. TCP-ECN-ELFN on the other hand will send a zero window probes in freeze state.

- At a 10% loss, the idealized energy for TCP-SACK is 2.5x greater than that for TCP-ECN-ELFN and the awake energy is 5x greater! However, at a 1% loss, the difference is quite small.

6.1.2 Packet Reorder Case

Dummynet was configured to reorder 1% or 5% packets randomly in its buffer. In Figure 5 we plot the idealized and awake energy for the 5% packet reorder case as a function of RTT. Let us first consider the idealized energy for these two protocols. In the 1% case, we note that TCP-SACK has a lower idealized energy cost than the TCP-ECN-ELFN protocol (see Table 4. This is due to the fact that the TCP-ECN-ELFN protocol retransmits[5] more packets than TCP-SACK Interestingly, at a 5% reorder rate, we see that TCP-SACK has a *higher* idealized energy than the TCP-ECN-ELFN protocol even though it retransmits fewer packets! The explanation is that the additional reception and processing cost of SACKs becomes appreciable at the 5% reorder level and thus increases the E_I value ([10] also contemplates the additional cost of using SACK). SACK is a TCP option in which the receiver can specify up to three blocks of out-of-order data it has received. Each block is specified by the starting and ending 32-bit sequence number. Thus, specifying each block consumes eight bytes. In the case of a 5% packet reordering, the number of duplicate ACKs is larger and each duplicate ACK will contain a SACK that can contribute between ten and twenty four bytes of additional information. The sender also needs to maintain additional data structures to process SACKs. The cost of receiving more data coupled with the cost of processing and storing SACK data structures increases the overall idealized energy cost for TCP-SACK. The TCP-ECN-ELFN protocol has none of this overhead and thus has a lower idealized energy cost. In the 1% reorder case, the number of duplicate ACKs is much smaller and hence the SACK overhead is minimal.

6.2 Static ad hoc network case: *impact of loss*

6.2.1 Random Packet Loss Case

[5]We used tcpdump and netstat before and after each run to gather these statistics.

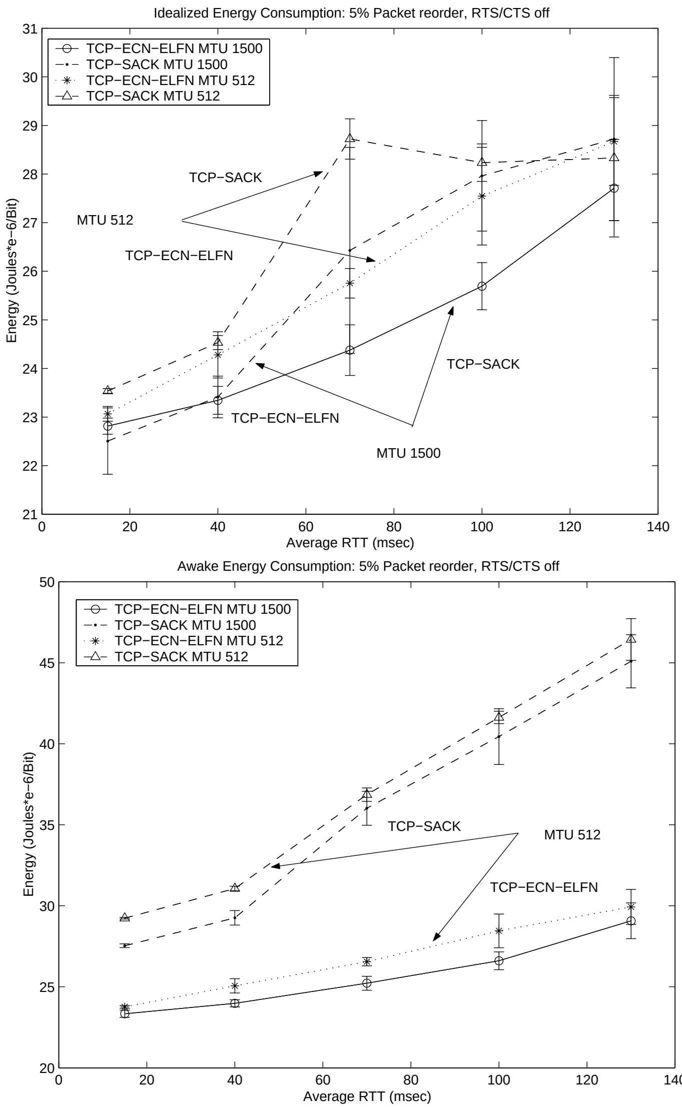

Figure 5: Idealized and awake energy for 5% packet reordering.

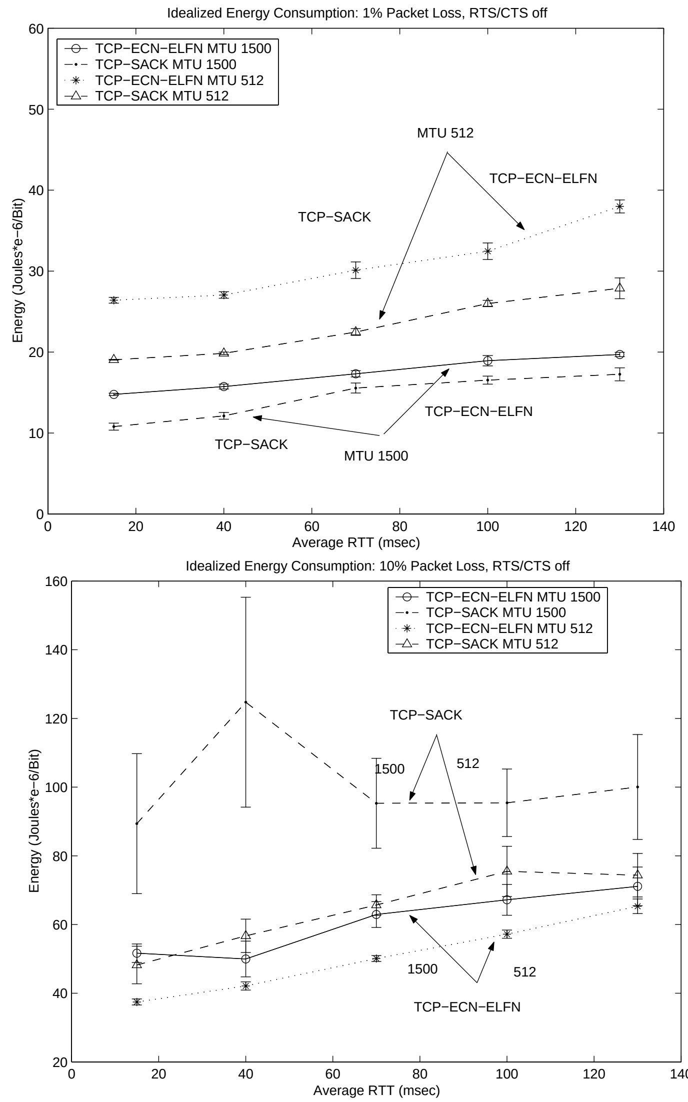

Figure 6: Idealized energy cost for loss case.

One of the primary contributors to lowered protocol performance in static ad hoc networks is packet loss resulting from link-layer errors and congestion. In order to study the impact of packet loss on protocol behavior, we used three values for packet loss (1%, 5%, and 10%), for the five values for RTT and two different MTU values (see Table 2). Figure 6 plots the idealized energy cost E_I as a function of RTT for both MTU values (1% and 10% loss). We left out the goodput and E_A plots as well as plots for the 5% loss case for reasons of space.

We can now make the following observations based on Table 6 and Figure 6 (we have left out the awake energy plots for space reasons).

- In general, smaller MTUs are better (i.e., consume less energy E_I and E_A) at higher loss rates whereas larger MTUs are better at low loss rates. This a well-known result for wired networks as well and is well understood.

Goodput and E_A:

- For the 1% loss case with a MTU of 512 bytes, TCP-SACK has a higher goodput than the TCP-ECN-ELFN protocol for cases when the RTT < 100 ms. The reason the goodput of TCP-SACK is higher for these cases is two-fold: first, the TCP-ECN-ELFN protocol has a higher number of retransmissions (for the reason explained eralier) and second, TCP-SACK can recover from multiple losses occurring in a window within one RTT thus maintaining its throughut.

  However, at a high RTT (e.g., 100ms), we see that the goodput of the TCP-ECN-ELFN protocol overtakes that of TCP-SACK. This is because the TCP-ECN-ELFN protocol does not shrink its congestion window on receipt of triple duplicate ACKs (as TCP-SACK does) thus it maintains a high throughput. At a low RTT, TCP-SACK is able to quickly build up its congestion window to maintain high throughput but at high RTT this process takes much longer and we there-

Total transmissions (Timeouts)	MTU 512		MTU 1500	
	1%	10%	1%	10%
TCP-SACK	11004 (0.51)	11538 (104)	3642 (0.64)	4111 (340)
TCP-ECN-ELFN	11661 (0.52)	14150 (6.48)	4067 (0.16)	5551 (40)

Table 5: Number of transmissions (and timeouts) for the loss case.

fore see the TCP-ECN-ELFN protocol pulling ahead! For an MTU of 1500 bytes the trend is the same for identical reasons. However, the crossover point occurs at lower RTTs due to the larger MTU.

- For the 10% loss case, the TCP-ECN-ELFN protocol has a higher goodput (and thus a lower E_A) as compared with TCP-SACK because, unlike TCP-SACK, the TCP-ECN-ELFN protocol does not reduce its congestion window on receipt of triple duplicate ACKs or timeouts. At a loss probability of 10%, we see multiple losses within a window due to which TCP-SACK's congestion window does not grow much thus keeping the goodput small.

Idealized Energy E_I:

- At a loss of 1%, the idealized energy of the TCP-ECN-ELFN protocol is *higher* than that of TCP-SACK. The reason for this is that the TCP-ECN-ELFN protocol actually retransmits many more packets than TCP-SACK. Table 5 summarizes the total number of transmissions for the two protocols (averaged over all RTTs). As we can see, for a MTU of 1500, the TCP-ECN-ELFN protocol transmits 425 more packets than TCP-SACK.

- At a loss of 10%, we note that the TCP-ECN-ELFN protocol still transmits more packets than TCP-SACK but it has a lower idealized energy cost. The reason for this dichotomy is the dependence of energy cost on the both the transmission/reception cost as well as on the *processing cost*. At a 10% loss rate, TCP-SACK will see a large number of duplicate ACKs containing SACKs for blocks received out-of-order. As we discussed in section 6.1.2, the added cost of processing, storing, and receiving these SACKs is non-trivial and accounts for the higher idealized energy E_I value for TCP-SACK. At a 1% loss rate, however, the number of SACKs is significantly smaller and the energy associated with SACK processing is not statistically significant and thus does not affect the relative values of E_I for TCP-SACK and the TCP-ECN-ELFN protocol.

6.2.2 *Bursty Loss Case*

When several packets are lost in the network, TCP-SACK and the TCP-ECN-ELFN protocol both retransmit the missing packets when the sender receives triple duplicate ACKs. However, TCP-SACK shrinks its congestion window whereas the TCP-ECN-ELFN protocol does not. Thus, the goodput of the TCP-ECN-ELFN protocol is higher and its awake energy E_A & idealized energy E_I are lower than that of TCP-SACK. Interestingly, as shown in Figure 7, the idealized energy E_I for the TCP-ECN-ELFN protocol is quite

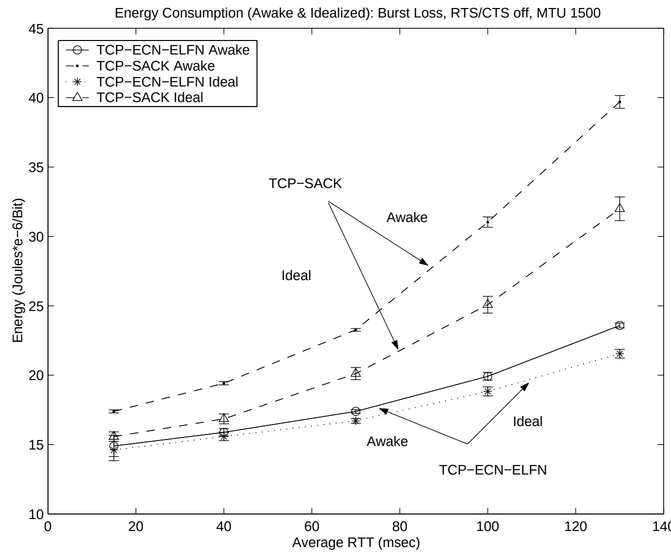

Figure 7: Energy for bursty losses.

close to that of the awake energy. This is due to the fact that in these experiments the only loss suffered was due to bursty loss (i.e., no random packet loss was simulated) and there was no network congestion or route failure. Thus, the TCP-ECN-ELFN protocol operated at the maximum possible rate resulting in minimal idle time. This causes the values of E_A and E_I to be quite close. In the case of TCP-SACK, on the other hand, the congestion window shrinks every time there is a triple duplicate ACK or timeout that causes the goodput and E_A to fall. Thus, there is a larger difference between the idealized and awake energy for TCP-SACK. Overall, we see a 2x improvement in E_A when using ECN-ELFN and a 10 – 25% improvement in E_I.

6.2.3 *Congestion Case*

In this set of experiments we wanted to investigate the effects of congestion on TCP-ECN-ELFN and TCP-SACK. Using packet traces we observed that TCP-ECN-ELFN did indeed respond to congestion appropriately, the sender reduced its congestion window in response to one or more ECNs and build up its congestion window when it no longer received the ECNs. In this section we examine the energy consumed by the two protocols under study for the case when congestion occurs every 30 seconds and lasts for 5 seconds. We ran the experiments for two cases when the average RTT was 15 msec and 130 msec. For each of these RTT values we had background packet loss of 1%, 5%, and 10%.

Figure 8 plots the idealized energy consumed by the two protocols (the awake energy plots are similar but with a much greater difference between ECN-ELFN and SACK). It is interesting to observe that the idealized energy consumed is the same for the two RTT values for TCP-SACK and for the TCP-ECN-ELFN protocol with an RTT of 130 msec. However, the idealized energy for the TCP-ECN-ELFN protocol at an RTT of 15 msec is the smallest by far. In the case of TCP-SACK, the periodic congestion coupled with packet loss ensures that its congestion window is always small (for both RTTs) and thus the idealized energy consumed is almost the same (i.e., on the average the same number of

	1% Loss	5% Loss	10% Loss
MTU 512	$E_I^{\text{SACK}} < E_I^{\text{ECN-ELFN}}$ $E_A\text{SACK} < E_A^{\text{ECN-ELFN}}$ $\tau^{\text{SACK}} > \tau^{\text{ECN-ELFN}}$	$E_I^{\text{SACK}} < E_I^{\text{ECN-ELFN}}$ $E_A^{\text{ECN-ELFN}} < E_A^{\text{SACK}}$ $\tau^{\text{ECN-ELFN}} > \tau^{\text{SACK}}$	$E_I^{\text{ECN-ELFN}} < E_I^{\text{SACK}}$ $E_A^{\text{ECN-ELFN}} < E_A^{\text{SACK}}$ $\tau^{\text{ECN-ELFN}} > \tau^{\text{SACK}}$
MTU 1500	$E_I^{\text{SACK}} < E_I^{\text{ECN-ELFN}}$ $E_A\text{ECN-ELFN} < E_A^{\text{SACK}}$ $\tau^{\text{ECN-ELFN}} > \tau^{\text{SACK}}$	$E_I^{\text{ECN-ELFN}} < E_I^{\text{SACK}}$ $E_A^{\text{ECN-ELFN}} < E_A^{\text{SACK}}$ $\tau^{\text{ECN-ELFN}} > \tau^{\text{SACK}}$	$E_I^{\text{ECN-ELFN}} < E_I^{\text{SACK}}$ $E_A^{\text{ECN-ELFN}} < E_A^{\text{SACK}}$ $\tau^{\text{ECN-ELFN}} > \tau^{\text{SACK}}$
ECN-ELFN shows a 10% improvement in E_A at 1% loss to a 2x – 8x improvement at 10% loss The improvement in E_I is at best 33%			

Table 6: Summary of energy and goodput for the loss case.

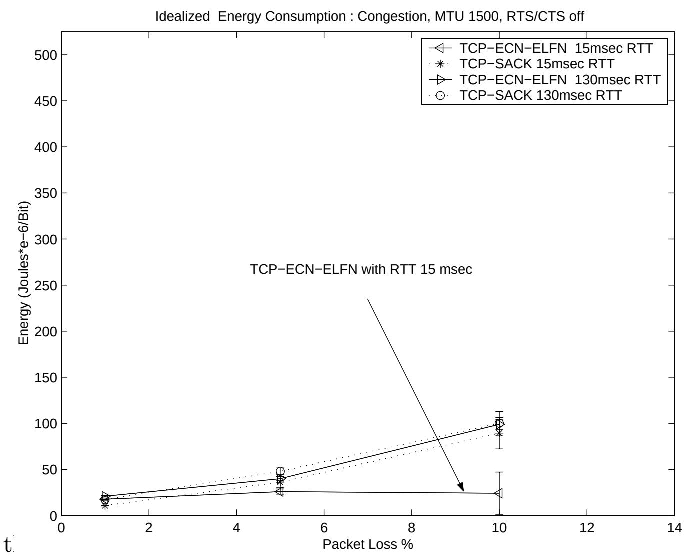

Figure 8: Idealized energy for the congestion case.

SACKs are processed and the same number of packets are retransmitted). In the case of the TCP-ECN-ELFN protocol, however, the idealized energy is higher for the larger RTT case because the protocol mistakenly retransmits more packets at a higher RTT. Recall from our discussion in section *6.2.1* that the TCP-ECN-ELFN protocol may retransmit the same packet more than once because it may receive multiple cases of triple duplicate ACKs – the probability of this happening is higher at a larger RTT.

7. CONCLUSIONS

In this paper we have characterized the energy cost of TCP-SACK and a modified version of TCP that appears to be better suited for operation in ad hoc networks (See Table 7. The TCP-ECN-ELFN protocol relies on explicit routing failure notifications to freeze TCP state allowing faster recovery when the route is back up. In addition, it uses ECN to respond to network congestion. We showed that the TCP-ECN-ELFN protocol uses less energy and delivers a higher goodput as compared with TCP-SACK, under most ad hoc network conditions. One of the areas of concern in using the TCP-ECN-ELFN protocol, however, is the issue of fairness. That is, will this protocol share bandwidth fairly between multiple connections? This question is fairly complex and is presently being studied in a ns2 simulation.

Ntwk condition	*Lower E_I*	*Higher Goodput*
Mobile ad hoc networks		
Route failure	ECN-ELFN	ECN-ELFN
Pkt reordering	1%: SACK 5%: ECN-ELFN	ECN-ELFN ECN-ELFN
Static ad hoc networks		
Packet Loss	1%: SACK 10%: ECN-ELFN	MTU 512: SACK 1500: ECN-ELFN (Both) ECN-ELFN
Bursty loss	ECN-ELFN	ECN-ELFN
Congestion	ECN-ELFN	ECN-ELFN

Table 7: Summary of results.

8. REFERENCES

[1] Gavin Holland and Nitin H. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," in *ACM Mobile Computing and Networking (MOBICOM'99)*, 1999, pp. 219–230.

[2] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, no. 5, pp. 10–23, 1994.

[3] H. Singh and S. Singh, "Energy consumption of tcp reno, newreno, and sack in multi-hop wireless networks," in *ACM SIGMETRICS 2002*, June 15 - 19 2002.

[4] M. Srivastava P. Lettieri, C. Schurgers, "Adaptive link layer strategies for energy efficient wireless networking," in *Wireless Networks*, 1999, vol. 5, pp. 339 – 355.

[5] M. Zorzi and R.R. Rao, "Error control and energy consumption in communications for nomadic computing," in *IEEE Transactions on Computers*, March 1997.

[6] V. Tsaoussidis, H. Badr, X. Ge, and K. Pentikousis, "Energy/throughput tradeoffs of tcp error control strategies," in *In Proceedings of the 5th IEEE Symposium on Computers and Communications, France*, July 2000.

[7] M. Zorzi and R.R. Rao, "Is tcp energy efficient?," in *Proceedings IEEE MoMuC*, November 1999.

[8] Xiang Zeng, Rajive Bagrodia, and Mario Gerla, "Glomosim: a library for parallel simulation of large-scale wireless networks," in *Proceedings of the 12th Workshop on Parallel and Distributed Simulations – PADS '98*, May 26-29 1998.

[9] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM Computer Communication Review*, vol. 27, no. 1, January 1997.

[10] L. Rizzo, "Issues in the implementation of selective acknowledgements for tcp," 1996.