Energy Efficient Congestion Control Operation in WSNs

Adel Gaafar A. Elrahim

Electrical Engineering Dept. Red Sea University, Port Sudan, Sudan

Abstract: The development of wireless technologies makes it necessary to find a way of improving Transmission Control Protocol (TCP) efficiency with regards to optimal utilization of wireless channel capacity. This paper describes a modification to the TCP congestion control for use in wireless sensor networks. It shows that by slightly modifying the algorithm of the TCP, it can be made to respond better to wireless links, while maintaining its advantages on the wired networks at the same time. This is certainly a very desirable feature as the conventional TCP in most cases contradicts the demands of the wireless links of the network. Our simulation results indicate that the modified TCP gives better performance in terms of higher throughput, delay, retransmission, and energy consumption than conventional TCP protocol.

Keywords: Wireless Sensor Networks; Transmission Control Protocol; Congestion control; Wireless links.

I. INTRODUCTION

Many applications of wireless sensor networks require connectivity to external networks to let monitoring and controlling entities communicate with the sensors. By using the TCP/IP protocols inside the sensor network, external connectivity can be achieved anywhere in the sensor network. In such IP-based sensor networks, TCP can be used for remote management and reprogramming of sensor nodes. However, the high bit error rates in multi-hop sensor networks lead to energy-inefficiencies that reduce the lifetime of the sensor network. When a packet loss occurs due to corruption TCP incorrectly activates anti-congestion procedures, causing overall throughput over the channel to drop. Constant losses caused by high bit error rates lead to sub-optimal average data throughput due to TCP’s inherent inefficiencies. TCP optimizations to deal with increased random packet losses should preferably maintain TCP’s end to-end semantics with minimal dependence on intermediate nodes.

In a sensor network, error rate is much higher and bandwidth is smaller than those of fixed networks. As a consequence, running conventional TCP protocol on a sensor network will suffer from severe performance degradation [1]. To handle a packet loss, conventional TCP retransmits the lost packet from its source. However, when error rate is high, it may have to take several retransmissions to deliver a packet to its destination successfully. Furthermore, packets losses may also activate TCP’s congestion control causing too many what is so called (slow start) that will further impair the performance of packet delivery. As a result, the effective throughput is much lower and the average packet delivery time will be much longer. In the worst cases, a TCP may even completely stale when error rate is too high.

In fact, most applications on a sensor network prefer faster and reliable packet delivery to higher throughput. However, most versions of TCP are all designed to achieve higher throughput, not faster packet delivery. It may even completely stall in a highly error prone environment. Therefore, it is beneficial to redesign a TCP by trading throughput for faster and reliable packet delivery. The standard fare in TCP implementations today can be found in RFC 2581 [2]. This reference document specifies four standard congestion control algorithms that are now in common use. The four algorithms are Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery.

Congestion occurs when over a prolonged period of time more packets are generated than the network (as a whole or locally) can actually carry. Usually, nodes have some buffer space available which can handle transient overloads. Any packet in excess of the available buffer space is dropped, wasting all the energy spent on this packet so far. Clearly, the larger this buffer space is, the more overload can be carried and packet dropping occurs later. On the other hand, longer queues impose longer end-to-end delays and the protocols need longer time to react on congestion states. Congestion control is very important in wireless sensor networks, because overloading a wireless network by too many transmissions can increase the collision probability. TCP congestion control limits the maximum window size according to the slow start congestion control algorithm. However, it even might make sense to further limit the window dependent on the number of intermediate hops in a wireless multi-hop network, because the optimal window size in terms of throughput might be below the window size of standard TCP [3].

Improving TCP performance in wireless networks can be classified into two groups. The first group of techniques requires new protocol mechanisms or changes to the existing TCP protocols, while the second groups of methods are
those that do not need new mechanisms or changes. Improving TCP may not be easy to implement on a Wide Area Network (WAN) because upgrading a large number of routers in a WAN is almost a business impossible. However, a sensor network has no such concern so that it is easy for a sensor network to embrace any new approach. The following sections study the modification of TCP protocol over sensor networks. The rest of this paper is organized as follows: Section II presents related work. Section III describes the proposed algorithm. Section IV describes the details of the simulation model. Simulation results and discussions are presented in section V. Section VI concludes this paper.

II. RELATED WORK

Here we discuss some recently proposed protocols for reliable transport layer in wireless sensor networks. Distributed TCP Caching (DTC) and TCP Support for Sensor networks (TSS) are inspired by the Snoop [4] protocol that has been developed for supporting TCP over wireless access networks. The Snoop agent is deployed at an intermediate system between the wireless and wired part of the network. The agent buffers TCP segments that have not yet been acknowledged by the receiver and detects TCP segment loss by analyzing TCP acknowledgments. In that case, the agent can perform local retransmissions as well as suppress TCP acknowledgments in order to avoid duplicate acknowledgments at the sender. Duplicate acknowledgments might cause end to end retransmissions for packets that could also be recovered locally by the agent. DTC can be considered as a generalization of Snoop for wireless multi-hop networks, while TSS further reduces the number of transmissions and adds mechanisms to throttle the transmission rate in case of packet losses.

Event-to-Sink Reliable Transport (ESRT) [5] aims to support reliable sensor data transport in wireless networks. ESRT is a centralized protocol that regulates the reporting rate of sensors in response to congestion detected by a sink. Each sensor node monitors its local buffer level and sets a congestion notification bit in the packets forwarded to the sink if the buffers overflow. When the sink receives a packet with the congestion notification bit set, it infers congestion and broadcasts a control signal notifying all source nodes to reduce their reporting frequency. It includes congestion control and mechanisms to achieve reliability. The reliability is controlled by adapting a rate at which the sink sends state reports back to the source. The frequency of the reports depends on the observed and desired reliability as well as the needs from congestion control.

Congestion Detection and Avoidance (CODA) [6] is based on congestion detection by monitoring channel utilization and buffer occupancy at the receiver. Detected congestion situations are signaled towards the source using backpressure signals (open-loop). Nodes receiving backpressure signals throttle down their transmission. In addition, a closed-loop mechanism operates on a longer time-scale. Based on acknowledgments received from the sink, sources regulate themselves. Lost acknowledgments result in reducing the rate at the source. Again, in contrast to TSS, new signaling messages need to be introduced into CODA.

In [7] it proposed to hold copies of forwarded packets in a cache. When a downstream node encounters an error with packet forwarding, a route error message might be sent to the upstream node. The cached packet can then be retransmitted possibly on multiple alternative routes in order to repair the route break. TCP with Buffering capability and Sequence information (TCP-BUS) [8] proposes to buffer packets during route disconnection and re-establishment. After a route becomes available again, buffered packets are retransmitted by intermediate nodes. Special control messages are used to indicate route breaks and re-establishments.

III. PROPOSED METHODS OF MODIFICATION TCP FOR USE TCP IN WSN

A. Increased Timeout Threshold

In this section, we begin by discussing the selection of Retransmission Timeout (RTO) value in standard TCP operations [9]. Then, we present our enhancement method by increasing RTO value. A TCP sender constantly tracks the Round Trip Time (RTT) for its packets and uses a timeout mechanism to trigger retransmissions if an ACK is not received before the timer expires. As a de facto standard, TCP sender uses the tracked average (RTT) plus m times the mean deviation of RTTs as the RTO value [9] for the next packet where the typical value of the factor m is 4. More precisely, let RTT (k) denote the k-th measurement value of RTT, where its value is the time interval between the beginning of the packet transmission until an ACK for the packet is received by the sender. Let SRTT (k) be the smoothed average RTT given as

\[ SRTT(k) = (1 - \alpha_o)SRTT(k - 1) + \alpha_oRTT(k) \]  \hspace{1cm} (1)

where \( \alpha_o \) is the exponential smoothing parameter with typical value of 1/8 [9]. Similarly, we use \( S(k) \) to denote the smoothed mean deviation of the RTTs, which is calculated by:

\[ S(k) = (1 - \beta)S(k - 1) + \beta \mid RTT(k) - SRTT(k) \mid \]  \hspace{1cm} (2)

Page | 73
where $\beta$ is the smoothing parameter, which has a typical value of $\frac{1}{4}$ [9]. Finally, the retransmission timeout (RTO) value is obtained by

$$RTO(k) = SRTT(k) + mS(k) \tag{3}$$

where $m$ is typically set to 4 as a de facto standard.

According to the standard TCP protocol, the TCP sender records the time when it just starts to forward some packets to the receiver. When the sender eventually receives an ACK associated with the packet, the RTT for the packet is thus computed. Using (1) to (3), the smoothed average and mean deviation of the RTTs, and the new RTO value are obtained, respectively. The RTO value is used for setting the timeout period for the next packets sent by the sender, until the next RTT measurement is obtained and the RTO is updated according to the equations. We note that unless the time-stamping option is activated, a TCP sender does not keep track of the RTT for every packet. Instead, only one packet among the outstanding packets is tracked for RTT at any time. As a result, equations (1) to (3) are invoked to update the RTO value when the ACK for a tracked packet is received by the sender.

We note that setting $m$ to 4 in (3) in determining the RTO value can avoid virtually all spurious timeouts with a probability close to 1, if the RTTs do not demonstrate a high degree of variability, as found in wired network. However, due to reasons such as channel fading, retransmissions, packets scheduling and so forth, RTTs in wireless networks tend to have much higher variability than their counterparts in wired networks. For this reason, we propose to use a value larger than 4 for $m$ in computing the RTO threshold value. Clearly, increasing $m$ in (3) increases the RTO value for the given smoothed average and mean deviation of RTTs. Consequently, a larger RTO value can help absorb the high variability of RTTs in wireless networks, thus avoiding unnecessary spurious timeouts and maintaining throughput performance. On the other hand, the RTO value should not be chosen arbitrarily large to enable speedy recovery of actual packet losses. There is a tradeoff between the throughput gain by avoiding spurious timeouts and the throughput degradation due to the delay in recovering from actual packet losses.

We have a few notes in place. Although it is normal to increase the RTO value to avoid spurious timeouts in wireless networks, to the best of our knowledge, such a method has not yet been proposed in the literature. This may have been due to the fact that the standard way of calculating the RTO value by (1) to (3) has been widely implemented. It will not be easy, if possible at all, to modify the value of $m$ on existing systems and devices. Nevertheless, our proposed method could be considered in future TCP implementation for wireless sensor applications.

**B. Modified Congestion Control Algorithm for WSN**

To estimate the number of packets that can be in transit without causing congestion, TCP maintains a congestion window (cwnd). New packets are only sent if allowed by both the window and the receiver’s advertised window. The TCP sender can detect a packet loss in two ways: Timeout and Duplicate ACKs.

A timeout occurs when the sender does not receive any acknowledgment from the receiver within a RTO. When a timeout occurs, TCP interprets this as severe congestion in the network and sets the slow start threshold (ssthresh) to 1/2 of the minimum of the current cwnd and the receiver’s advertised window. Then it decreases the cwnd to 1 segment and performs a slow start. During the slow start, the TCP sender increases cwnd by 1 segment every time an ACK is received. When the cwnd reaches the ssthresh, it leaves the slow start phase and switches to the congestion avoidance, where new ACK increments the cwnd by 1/cwnd. On the other hand, when duplicate ACKs are received (duplicate ACKs signify that the receiver received out-of-order packets, TCP interprets this as less severe congestion. When the third duplicate ACK is received, TCP goes into the fast retransmit. The ssthresh is set to 1/2 of the minimum of cwnd and the receiver’s advertised window and the TCP sender retransmits the lost packet immediately without waiting for the retransmission time-out. During the fast retransmit stage, each duplicate ACK is considered as an acknowledgment of an out-of-order segment and the cwnd is increased by one correspondingly and continues to send packets. Summary of Slow Start and Congestion Avoidance is shown in Figure.1.

Concept: The principal problem is the congestion control algorithm. Nearly all TCP implementations nowadays assume that timeouts are caused by congestion, not by lost packets. Consequently, when a timer goes off, TCP slows down and sends less vigorously (e.g., Jacobson’s slow start algorithm). The idea behind this approach is to reduce the network load and thus alleviate the congestion. Unfortunately, wireless transmission links are highly unreliable. They lose packets all the time. The proper approach to dealing with lost packets is to send them again, and as quickly as possible. If, say, 20 percent of all packets are lost, then when the sender transmits 100 packets/sec, the throughput is 80 packets/sec. If the sender slows down to 50 packets/sec, the throughput drops to 40 packets/sec. In effect, when a packet is lost on a wired network, the sender should slow down. When one is lost on a wireless network, the sender should try harder. When the sender does not know what the network is, it is difficult to make the correct decision. It should be borne in mind that TCP is an end to end protocol and a connection will almost always be mixed (part wired and part wireless). So, a balance...
has to be kept while making any modifications to suit the wireless links. This balance is between segments dropped due to conventional congestion as opposed to wireless related reasons like high bit error rate and mobility [9].

![Diagram of Slow Start and Congestion Avoidance]

**Figure 1. Summary of Slow Start and Congestion Avoidance**

Keeping the above considerations in mind, we have proposed some modifications to the TCP model. The proposed modifications are:

a) Reducing the number of duplicate acknowledgements required from 3 to 2 before a Fast Retransmit takes place. This supports the theory of resending a lost segment without delay. TCP does not wait for an entire window’s worth of ACKs to add one packet’s worth to the congestion window, but instead increments Congestion Window by a little for each ACK that arrives. Specifically, the congestion window is incremented as follows each time an ACK arrives:

\[ I = \frac{MSS}{cwnd} \times (MSS / cwnd) \]  

\[ cwnd_{new} = cwnd + I \]

where I is Increment value.

That is, rather than incrementing Congestion Window by an entire Maximum Segment Size (MSS) bytes each RTT, we increment it by a fraction of MSS every time an ACK is received. The important concept to understand about Additive Increase Multiplicative Decrease (AIMD) is that the source is willing to reduce its congestion window at a much faster rate than it is willing to increase its congestion window.

b) Raising the Slow Start initialization point from 1 MSS to 4 MSS. This will prevent the cwnd from closing all the way and will aid a fast recovery of the cwnd.

c) Raising the Slow Start threshold from half the previous cwnd size to \( \frac{3}{4} \) of the previous cwnd. This is done so that the process of slow start continues for a longer period and the cwnd gets back to its original size quickly.

\[ ssthresh = \frac{3}{4} \times cwnd \]

**IV. SIMULATION MODEL**

A. Simulation Tool (OPNET)

The well known OPNET simulation tool is used. Different simulations results are presented with different number of nodes in order to check performance of the proposed algorithm. The goal of the study was to investigate the behavior of new TCP congestion control, conventional TCP congestion control for throughput, delay, retransmission, and energy consumption.
B. Simulation Setup

Extensive simulations are carried out to analyze the effectiveness of the proposed congestion control mechanism. 40, 50, 60, 70, 80, 90, 100 sensor nodes are randomly placed in a space of 300 x 300 m. TCP packet size is assumed to be fixed at 128 bytes and each source node generating data at a maximum rate of 8 packets per second, the Slow Start initial count set to 4 Maximum Segment Size (MSS). We assume the presence of one sink located in the centre of WSN. We also assume the presence of three routing protocols which realize multiple paths from the source to sink; these are EAGRP, DSR, and AODV [10]-[13]. In physical layer 1Mbps data rate is used. Simulation time for each scenario was set to 500 seconds and repetitive simulations for each scenario were performed to verify the reliability of our results. Table I summarizes the simulation parameters used in our experimental setup. Evaluation metrics used performance test are: average throughput, average packet delivery time (delay), average number of retransmissions, energy consumption, and the stability of congestion window size (cwnd).

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>SIMULATION PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing algorithms</td>
<td>EAGRP, DSR, AODV</td>
</tr>
<tr>
<td>Simulation time</td>
<td>500 sec</td>
</tr>
<tr>
<td>Simulation area</td>
<td>300 m x 300m</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>40, 50, 60, 70, 80, 90, 100</td>
</tr>
<tr>
<td>Packet size</td>
<td>128 bytes</td>
</tr>
<tr>
<td>Packet rate</td>
<td>8 packets/sec</td>
</tr>
<tr>
<td>Data Rate</td>
<td>1 Mbps</td>
</tr>
<tr>
<td>Initial node energy</td>
<td>1 Joule</td>
</tr>
<tr>
<td>Slow start initial count</td>
<td>4 MSS</td>
</tr>
<tr>
<td>Duplicate ACK threshold</td>
<td>3</td>
</tr>
<tr>
<td>Retransmission Timeout</td>
<td>10 sec</td>
</tr>
</tbody>
</table>

C. Selected Performance Metrics for Evaluation

1) Throughput: The throughput reflects the effective network capacity. It is defined as the total number of bits successfully delivered at the destination in a given period of time. Throughput shows protocol’s successful deliveries for a time; this means that the higher throughput the better will be the protocol performance

2) End-to-End Delay of Data Packet: This metric measure the average time it takes to route a data packet from the source node to the destination node. The lower the end-to-end delay the better the application performance. If the value of End-to-end delay is high then it means the protocol performance is not good due to the network congestion.

3) Retransmission : Total number of TCP retransmissions in the network. Calculated when data is retransmitted from the TCP unacknowledged buffer.

4) Energy Consumption: The energy metric is taken as the average energy consumption per node calculated through simulation time.

V. RESULTS & DISCUSSIONS

With the above TCP modifications applied, simulations were run and the results recorded and compared. As shown in Figures 2, 3, and 4, when an error causing a loss of multiple segments occurs, a problem arises with traditional TCP’s congestion window. Fast Retransmit closes the congestion window every time a non-duplicated acknowledgement is received. A loss of multiple segments makes the window close to half of its size for every dropped segment, as shown in Figure 2. Besides, the usable window is decreased during the fast recovery phase, as more information is sent; every time a duplicated acknowledge is received. When the congestion window is divided by two for the second time, the usable window closes to zero, thus blocking the communication and forcing the retransmission timeout. When the retransmission timer expires the congestion window is closed to 0 and starts growing with slow start and congestion avoidance algorithms.

Traditional TCP problems, when multiple segments are lost, come from applying the Fast Recovery algorithm. Fast Recovery allows sending data while recovering from the first error. This makes the usable window close while this data is not acknowledged. When TCP realizes that another error has happened, it divides the congestion window again by two. Usable window, (calculated as the congestion window minus the sent but not acknowledged data), comes down to a value that does not allow any more data to be sent. At this point the transmission is blocked because duplicated acknowledgements are not received and there are no other mechanisms that inflate the congestion window. So the only event that can trigger the retransmission is the retransmission timer expiration.
Modified TCP does not apply Fast Recovery in order to avoid the problems described previously. When this algorithm is used, Fast Retransmit is the only algorithm applied. This means that Slow Start and Congestion Avoidance will be working when recovering from an error.

As shown in Figures 2 to 4, modified TCP closes the congestion window when the first error is detected. This stops the communication suddenly, but allows the congestion window, and consequently the usable window, to open exponentially. Then the usable window allows the sender to retransmit the lost segments. Figure 2 also shows that the cwnd not only drops at a higher value than in the traditional TCP case but also continues the Slow Start process for a longer time. This has the dual effect of keeping a larger window open and also continuing to open it at a faster rate for a longer time.

Modified TCP provides us better performance than traditional TCP, because the former closes the usable window when the first error of a burst is detected, and uses Slow Start from the beginning. From Figures 2 to 4, Modified TCP yields significantly larger values for the time of first Congestion occurred, compared to Traditional TCP. The improvement in the time of first congestion occurred for Modified TCP is 310 sec compared to Traditional TCP is 140 sec when we used EAGRP routing protocol, the improvement in the time of first congestion occurred for Modified TCP is 305 sec compared to Traditional TCP is 185 sec when we used DSR routing protocol, and the improvement in the time of first congestion occurred for Modified TCP is 370 sec compared to Traditional TCP is 205 sec when we used AODV routing protocol.
As we can see in Figures 2 to 4 the cwnd of modified TCP has larger congestion window size, for most of the time while traditional TCP is fairly unstable. The stability of cwnd at its upper bound implies high throughput and short delay time (packet delivery time) as we can see from Figure 5 and Figure 6. When error rate is high, modified TCP not only have a higher throughput, its delay time is significantly lower than its counterparts when the number of hops is high.

It was shown that in Figure 7, our modification protocol has less number of retransmissions and hence less incorrect triggering of the congestion control window reduction. This in turn resulted in more stable TCP throughput. Figure 8 presents the energy consumption for the modified TCP and traditional TCP protocols. Modified TCP exhibits the lowest energy overheads as shown for these three routing protocols used. Hence, modified TCP approach conserves more energy and is more efficient than traditional algorithm. This helps to considerably extend the lifetime of the nodes in heavy traffic scenarios.
Figure 6. The delay versus number of nodes

Figure 7. The retransmission versus number of nodes

Figure 8. The energy consumption versus number of nodes
VI. CONCLUSION

The limited energy, memory, and computational resources of sensor nodes require an energy-efficient transport layer. TCP support in wireless sensor networks is desirable to allow direct communication of sensor nodes with other systems for various purposes such as configuration, re-programming or management. In this paper we show that even in scenarios with high losses, TCP can be used and implemented in an energy-efficient way. Concept presented in this paper drastically reduced the number of TCP segment transmissions that are needed to transfer a certain amount of data across a wireless sensor network with relatively high bit/packet error rates. Simulation results indicate that in wireless sensor networks, modified TCP outperforms traditional TCP algorithm in terms of sending data and information due to its better average throughput, average end-to-end delay, average retransmission, congestion window size, and energy consumption in both high and low traffic.

REFERENCES