STCP: A Generic Transport Layer Protocol for Wireless Sensor Networks

Yogesh G. Iyer
Telecommunications Engineering Program
University of Texas at Dallas
Email: yogesh@student.utdallas.edu

Shashidhar Gandham
Dept. of Computer Science
University of Texas at Dallas
Email: gshashi@utdallas.edu

S. Venkatesan
Dept. of Computer Science
University of Texas at Dallas
Email: venky@utdallas.edu

Abstract— **We consider the issue of designing a generic transport layer protocol for energy-constrained sensor networks. We present the requirements for such a transport protocol and propose Sensor Transmission Control Protocol (STCP). STCP is a generic, scalable and reliable transport layer protocol where a majority of the functionalities are implemented at the base station. STCP offers controlled variable reliability, congestion detection and avoidance, and supports multiple applications in the same network. We present the design and implementation of STCP and evaluate the protocol with different scenarios and network characteristics.**

I. Requirements of a Generic Transport Layer

Sensor networks are deployed for a wide range of applications in the military, health, environment, agriculture and office domain. Every application has different characteristics and requirements such as sensed data type, rate of data transmission and reliability. Existing transport layer protocols for sensor networks are either tailored for certain applications or assume that the nodes employ a particular network layer or MAC layer protocol. As a result, their approaches may not be applicable across many sensor network deployments.

Heterogenity: Sensor nodes may have multiple sensors (light, temperature, seismic, etc.) with different transmission characteristics. Packets from a sensor for an application constitute its data flow. Each flow can be either continuous or event-driven. In continuous flow applications, nodes transmit packets periodically to a base station. In event-driven applications, nodes transmit data only when a pertinent event occurs. Both types of flows may exist in the same network. The transport layer protocol should support multiple heterogeneous applications in the same network.

Reliability: Every application may require different reliability. For example, in military surveillance, data transmitted by the sensor nodes must always reach the base station. While in temperature monitoring, a few packets may be lost. The transport protocol should exploit this variable reliability model and save network resources.

Congestion Control: Packets from all the nodes in the network converge at nodes located near the base station. These nodes forward more packets and hence, there is a possibility of congestion close to the base station. High data rates, sudden burst of data and collisions are other reasons of congestion in sensor networks [13]. Nodes might drop packets due to memory limitations and energy spent in forwarding these dropped packets is effectively wasted. Congestion also increases latency. Hence, the transport layer should support congestion detection and avoidance.

It is desirable to design a transport layer protocol that can support multiple applications in the same network, provide controlled variable reliability, address congestion issues, reduce latency and maximize throughput. We propose Sensor Transmission Control Protocol (STCP) and show, through simulations, that it meets these requirements.

We summarize the requirements of a transport layer protocol for sensor networks as follows:

1) *Generic:* The transport layer protocol should be independent of the application, Network and MAC layer protocols to be applicable for several deployment scenarios.
2) *Heterogeneous data flow support:* Continuous and event-driven flows should be supported in the same network.
3) *Controlled variable reliability:* Some applications require complete reliability while others might tolerate the loss of a few packets. The transport layer protocol should leverage this fact and conserve energy at the nodes.
4) *Congestion detection and avoidance:* The congestion detection and avoidance mechanism helps in reducing packet retransmissions, thereby conserving energy.
5) *Base station controlled network:* Since sensor nodes are energy constrained and limited in computational capabilities, majority of the functionalities and computation intensive tasks should be performed by the base station.
6) *Scalability:* Sensor networks may comprise of large number of nodes, hence the protocol should be scalable.
7) *Future enhancements and optimizations:* The protocol should be adaptable for future optimizations to improve network performance and support new applications.

II. System Model

We consider sensor networks where each node is equipped with one or more sensing devices, a low computation processor, limited battery-supplied energy and a short-range wireless transceiver. The nodes are preconfigured with a unique identifier. We assume that the base station has adequate energy, memory and processing power to implement all the functionalities of STCP. Sensor nodes and the base station communicate via bidirectional multihop wireless links. If the base station can reach all the nodes in a single hop, it would

enhance the performance of our solution, but this is not a requirement. The maximum transmission range of each node is fixed for its lifetime as 10m. The transceiver exhibits first order radio model characteristics [15]. According to this model, energy dissipated to run the transmitter or receiver circuitry is constant per bit transmitted or received and energy spent in transmitting a bit over a distance *d* is proportional to d^2.

We assume that clocks of all the nodes in the network are synchronized with that of the base station for continuous flow applications. Clock synchronization for sensor networks is addressed in [7], [9]. Nodes employ a network layer algorithm for routing packets. The sensor nodes and base station communicate on the same frequency and employ either a TDMA or CSMA/CA Medium Access Control (MAC) protocol.

III. Related work

We first examine TCP [11] and UDP [5] for sensor networks. The TCP protocol stack is complex to be implemented in a resource constrained sensor node. The overhead from headers can be quite large, particularly for small messages. Consider a simplex connection. TCP is designed to make the receiver side as simple as possible. It acknowledges the sender for reliability and for flow control mechanism. However, for sensor networks, the receiver (base station) has unlimited energy and hence, should control the communication. Also, TCP provides complete reliability, which is not required in many sensor deployments. UDP is a best-effort service and does not guarantee reliable delivery of information.

Reliable Multi-Segment Transport (RMST) [4] is designed to run in conjunction with directed diffusion network layer algorithm [3]. There is a dependency on the network layer. RMST is a selective NACK-based protocol and works in two modes of operation: caching and non-caching. In the caching mode, intermediate nodes cache data fragments which may cause buffer overflow due to memory limitations. Also, the timer associated for detecting fragment loss at every node is fixed and do not adapt to adverse network conditions. RMST does not guarantee reliability when a node fails before successfully transmitting the fragments or when a node fails after receiving all the fragments. Moreover, RMST does not address the issue of congestion in sensor networks.

Event-to-Sink Reliable Transport (ESRT) [2] is designed for networks with data-centric applications. The underlying assumption is that the base station is interested in reliable detection of events from the collective information provided by numerous sensor nodes. ESRT does not support end-to-end reliable data delivery. In ESRT, the base station controls the network congestion by requesting nodes to increase or decrease the rate of transmission depending on current network characteristics. In real world applications, nodes can transmit data only when they detect an event; varying the transmission rate of nodes may not be practical across all applications.

Pump Slowly, Fetch Quickly (PSFQ) [1] is proposed for reliable reprogramming of nodes in a sensor network. It provides a mechanism for reliable broadcast of data from the base station to sensor nodes. This is appropriate for control and management in the reverse direction. Note that bulk of the communication in sensor networks is from sensor nodes to the base station. PSFQ can be used to compliment our solution if necessary.

IV. STCP: Sensor Transmission Control Protocol

STCP provides a generic, scalable and reliable transport layer paradigm for sensor networks. Majority of STCP functionalities are implemented at the base station. Each node might be the source of multiple data flows with different characteristics such as flow type, transmission rate and required reliability. STCP supports networks with multiple applications and provides additional functionalities such as controlled variable reliability and congestion detection and avoidance.

A. Data transmission sequence in STCP

Before transmitting packets, sensor nodes establish an association with the base station via a *Session Initiation Packet*. The session initiation packet informs the base station of the number of flows originating from the node, the type of data flow, transmission rate and required reliability. When the base station receives the session initiation packet, it stores all the information, sets the timers and other parameters for each flow, and acknowledges this packet. It is important for the sensor node to wait for the ACK to ensure that the association is established. The nodes can now start transmitting data packets to the base station. In the reverse path, the base station transmits an ACK or NACK depending on the type of flow.

B. STCP Packet formats

The format of session initiation packet is shown in Figure 1. The session initiation packet for STCP is motivated by the concept of multiple-streams in SCTP [10]. If there are multiple sensing devices in a sensor node and some or all of them will be transmitting their sensed data, the node can transmit a single session initiation packet. Note that the source node will transmit packets associated with each flow independently, since the transmission characteristics may be different.

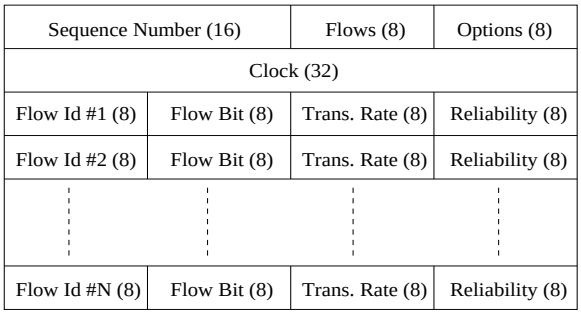

Fig. 1. Session Initiation Packet Format

The first field in the packet is the *sequence number* (16 bits long) which is zero for the session initiation packet. *Flows* indicate the number of flows originating at the node. The local clock value at the time of transmission is included in the *Clock* field. *Flow Id* is used to differentiate packets

from different flows. The *Flow Bit* field specifies whether the flow is continuous or event-driven. For continuous flows, the *Transmission rate* field indicates the rate at which a packet will be transmitted by the source node. The *Reliability* field gives the expected reliability required by the flow.

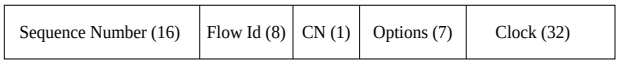

Fig. 2. STCP Data Packet Header

STCP data packet header is shown in Figure 2. The *Sequence number* for a data packet is a non-zero positive integer which distinguishes it from a session initation packet. The *Flow Id* indicates the flow type which helps the base station identify the characteristics of the packet for that node. The packet header includes a *Congestion Notification* (CN) bit field for supporting congestion detection and avoidance. The *Clock* field gives the local time at which the packet was transmitted. The base station uses the clock value to calculate the Estimated Trip Time (ETT) for that node and flow Id.

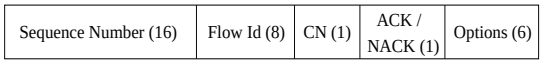

Fig. 3. STCP Acknowledgement Packet

STCP acknowledgement packet is shown in Figure 3. All fields are as explained before. The *ACK / NACK* bit represents a positive or negative acknowledgement.

STCP uses the 32 bit *clock* field in conjunction with the *sequence number* field to avoid issues related to wraparound. All the three packets have an *Options* field for future purposes.

C. Continuous Flows

Since the base station knows the rate of transmission from the source, the expected arrival time for the next packet can be found. The base station maintains a timer and sends a negative acknowledgement (NACK) if it does not receive a packet within the expected time.

When the base station receives a packet from a sensor node, it calculates the Estimated trip time[1] (ETT) for the packet to reach the base station. The base station calculates an expected time for successive packets by one of the following methods:

1) Timeout is calculated by the expression (T + $\alpha \times$ETT), where *T* is the time between successive transmissions and alpha (α) is a positive integer that varies with *ETT*.
2) The second approach is Jacobson/Karels algorithm [14] which considers the variance of the trip time. We use ETT instead of Round trip time.

In the first approach, the base station constantly checks to see if it has received a packet within (T + $\alpha \times$ETT) time units for each sensor node. If a packet has been received within time, it decreases alpha (α) by 0.5. If a packet is lost (timeout) or if the base station receives a packet after transmitting a NACK for it, it increases alpha (α) by 0.5.

[1]ETT = Current clock value at base station - Clock value in packet.

In the second approach, we modify Jacobson/Karels algorithm by considering ETT. The base station dynamically varies the values of delta (δ), mu (μ) and phi (ϕ) in the following expressions:

SampleETT = base station clock - packet clock value
Difference = SampleETT - EstimatedETT
EstimatedETT = EstimatedETT + ($\delta \times$Difference)
Deviation = Deviation + δ(|Difference| - Deviation)
TimeOut = $\mu \times$ETT + $\phi \times$Deviation

Sensor nodes retransmit packets only on receiving a NACK. No state information is maintained. The transmitted packets are buffered for possible retransmission. To prevent buffer overflow, a buffer timer is maintained, which periodically fires when the buffer size reaches a threshold and the buffer is cleared. The timer depends on the rate of transmission of packets and existing network conditions. For example, if packets are lost frequently, then the packets are cached in the buffer for a longer time for retransmission.

If the source node does not receive a NACK, the packet must have reached the base station, unless the NACK is lost. So the base station maintains a record of all packets for which it has sent a NACK. If a NACKed packet arrives, the base station clears the corresponding entry from the record. The base station periodically checks this record and, if it finds an entry, retransmits a NACK.

D. Event-driven Flows

In event-driven flows, the base station cannot estimate arrival times of data packets. Thus, clock synchronization is not needed. Because of reliability requirement, positive acknowledgements (ACK) are used by source to know if a packet has reached the base station.

The source node buffers each transmitted packet till an ACK is received. When an ACK is received, the corresponding packet is deleted from the buffer. The nodes maintain a buffer timer that fires periodically. When the timer fires, packets in the buffer are assumed to be lost and are retransmitted.

E. Controlled Variable Reliability

Sensor nodes specify the required reliability for each flow in the session initiation packet. For continuous flows, the base station calculates a running average of the reliability. Reliability is measured as the fraction of packets successfully received. Even if the base station does not receive a packet within the expected time interval, it will not send a NACK if the current reliability satisfies the required reliability. The base station transmits NACKs only when the reliability goes below the required level.

For event-driven flows, the base station calculates reliability as a ratio of packets received to the highest sequence numbered packet received. The sensor node, before transmitting a packet, calculate the effective reliability assuming that the packet will not reach the base station. If the result is still more than the required reliability, the node does not buffer the packet, thus saving memory space. If the node receives an ACK, the reliability increases.

F. Congestion Detection and Avoidance

Congestion detection and avoidance is an important aspect in sensor networks. The random early detection (RED) mechanism designed by Floyd and Jacobson [12] proposes that an intermediate node drop a packet when it experiences congestion. The source is, therefore, effectively notified by a subsequent timeout or a NACK. Since dropping of packets is detrimental to sensor networks, we consider other solutions. In Ramakrishnan and Jain's DECbit [6], intermediate nodes monitor the load experienced and explicitly notify the end nodes by setting a binary congestion bit in the packets.

STCP adopts this method of explicit congestion notification with some modification. Each STCP data packet has a *congestion notification* bit in its header. Every sensor node maintains two thresholds in its buffer: t_{lower} and t_{higher}. When the buffer reaches t_{lower}, the congestion bit is set with a certain probability. The value of this probability can be determined by an approach similar to that employed in RED. When the buffer reaches t_{higher}, the node will set the congestion notification bit in every packet it forwards.

On receiving this packet, the base station informs the source of the congested path by setting the congestion bit in the acknowledgement packet. On receiving the congestion notification, the source will either route successive packets along a different path or slow down the transmission rate. Note that the nodes rely on the network layer algorithm to find alternate routes.

G. Data-centric Applications

In data-centric applications, collective network-wide information is of interest. A few examples are monitoring of seismic activity, finding maximum temperature in the network, etc. In such applications, the intermediate nodes may aggregate the correlated data as part of the data aggregation process.

Since the number of source nodes may be very high, acknowledging all the source nodes by an ACK or NACK will deplete network resources and energy. Hence, for data-centric applications, STCP does not provide any acknowledgement scheme, similar to UDP. It assumes that data from different sensors are correlated and loss tolerant to the extent that events are collectively and reliably sent to the base station. This view is supported by the authors in ESRT [2].

V. Simulation Results

To understand the end-to-end behavior of wireless sensor networks, we performed extensive simulations of STCP using TinyOS Simulator (TOSSIM) [8].

A. Simulation setup

We simulated networks with 50, 75 and 100 sensor nodes randomly distributed in a 100m.$\times$100m. square sensor field. The transmission range of the nodes was set to 10 meters. Each sensor node was provided with an initial energy of 0.5 J. We assumed the radio dissipates 50 nJ/bit to run the transmitter or receiver circuitry and 0.1 nJ/bit - m^2 for the transmitter amplifier to achieve an acceptable SNR [15]. The data packet length was fixed as 200 bits. To route packets from the sensor nodes to the base station, we implemented distributed Dijkstra's shortest path algorithm. The simulator had a default CSMA MAC layer protocol for channel access. Simulations were run for 1000 simulator seconds unless stated otherwise. On an average, sensor nodes in the network transmitted a packet every 50 simulator seconds for continuous flows and at random intervals for event-driven flows.

To evaluate the performance of STCP, we identified the following metrics. We describe them in detail with results to support our protocol design:

- Number of NACKs for varying alpha (α).
- Average packet latency.
- Energy spent for different reliability.

B. Number of NACKs for varying alpha (α)

For continuous flows, the base station maintains a timer for successive packets from a sensor node. We evaluated two types of timers at the base station:

1) Timeout calculated by the expression (T + $\alpha\times$ETT).
2) Jacobson/Karels timeout mechanism which considers the variation of the trip time.

Consider the case where timers are implemented by the expression (T + $\alpha\times$ETT). ETT value of the previous packet is considered because it would give the base station a fairly accurate indication of the current network condition. Another approach would be to use a weighted average of ETT over several packets. We choose ETT instead of hop-count because we observed that the packet trip time for nodes having the same hop-count were very different. This is due to different network characteristics in individual nodes' neighborhood. The base station waits for more than (T + ETT) time units to compensate for unpredictable network delays due to link instability, node failures and channel contention. To study the effect of improper timeouts, we conducted several simulations by varying alpha (α) for different ETT ranges.

Before simulating STCP, our intuition was that nodes near the base station will have a small ETT value as packets traverse a few hops, and hence a small value of alpha (α) should be enough to wait for the next packet. Packets from nodes that are far away from the base station will traverse long paths, have a high value of ETT, and hence alpha (α) should be high to compensate for network delays.

After simulating several scenarios, we found that the nodes that were closer to the base station had a smaller ETT value. However, as they were in the forwarding path for several nodes, their packets experienced a high variance in ETT. Hence, a higher value of alpha (α) was required to compensate for variation in ETT. Nodes that were further away from the base station had a comparatively high ETT. Here a small alpha (α) value was enough to wait for the next packet because the variation in ETT was negligible. The base station thus dynamically changes the waiting time for the next packet corresponding to ETT and network conditions.

We assumed ideal network conditions for this simulation setup to observe the number of unnecessary NACKs trans-

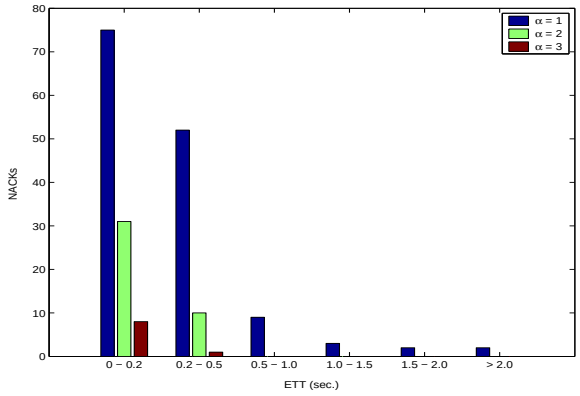

Fig. 4. NACKs for a 50 node network

mitted by the base station. We observed that the number of NACKs was very high for low values of alpha (α), especially for lower ETT. As alpha (α) increases, the number of NACKs decreased. From Fig. 4, for ETT (0 - 0.2 sec), α = 4 would give the correct waiting period without transmitting unnecessary NACKs. For nodes with higher ETT, a small value of alpha (α) was sufficient to compensate for delays.

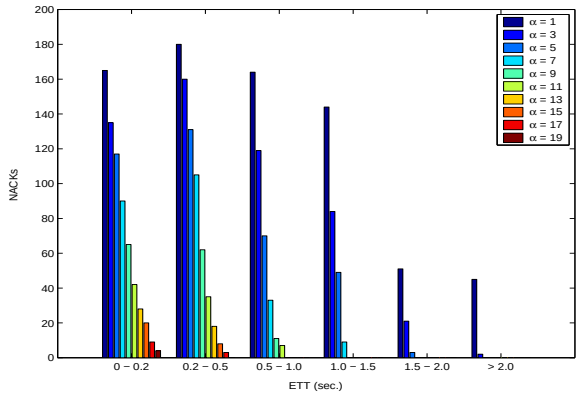

Fig. 5. NACKs for a 100 node network

From Fig. 5, we observe that the number of NACKs transmitted is more for ETT (0.2 - 0.5 sec). This is because when network size increases, latency increases due to congestion and channel contention. Hence, the ETT values for nearby nodes also increases and fall in the second range. As before, lower values of alpha (α) in this range causes more number of unnecessary NACKs to be transmitted.

From the above results, we were interested in finding the optimum[2] value of alpha (α) for all ETT ranges considering the network as a whole. As network conditions change, the base station dynamically varies alpha (α). Fig. 6 shows the optimum values of alpha(α) after 1000 simulator seconds for different network sizes of 50, 75 and 100 nodes. An interesting observation is that alpha (α) is very high for low ETT values in a 100 node network compared to a 50 node network; and alpha (α) is almost the same for higher ETT. This supports the fact that nodes near the base station face more congestion, which increases with network size.

[2]Optimum value is the smallest value at which unnecessary NACKs are not transmitted.

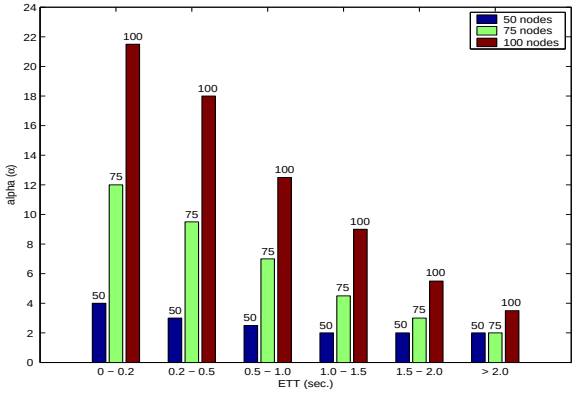

Fig. 6. Optimum alpha(α) for different network sizes

We implemented Jacobson/Karels timeout mechanism and experimented with different combinations of delta (δ), mu (μ) and phi (ϕ). After several simulations, we observed an ideal value of delta (δ) to be between 0.3 and 0.5. For a 100 node network, varying mu (μ) and phi (ϕ), we obtained the following values where the base station does not transmit unnecessary NACKs:

- For ETT (0 - 0.2), μ = 1 and ϕ = 8
- For ETT (0.2 - 0.5), μ = 3 and ϕ = 6
- For ETT (0.5 - 1.0), μ = 6 and ϕ = 6
- For ETT (1.0 - 1.5), μ = 7 and ϕ = 5
- For ETT (1.5 - 2.0), μ = 9 and ϕ = 5
- For ETT ($>$ 2.0), μ = 11 and ϕ = 4

We observe that the Jacobson/Karels timeout mechanism was analogous to our intuition. This is because of the variance considered in the equations to calculate the timeout period. As seen in the previous experiment, nodes near the base station observed a high variance in ETT, hence a high value of phi (ϕ) was observed for lower ETT. We obtained an interesting result for ETT greater than 2.0 seconds. We observed that 1% of the packets were NACKed too early. We traced ETT and its variance for all the nodes falling in this ETT range. We observed that in the same time interval, ETT decreased for some nodes while it increased for others. Since the base station dynamically varies the value of mu (μ) and phi (ϕ) to decrease waiting time when ETT reduced for some nodes, it had a negative effect on other nodes in the same range.

C. Average packet latency

An important goal of a transport layer protocol is to reduce packet latency. For finding the average packet latency, we implemented both continuous flows and event-driven flows in the same network. One-fourth of the nodes generated a continuous flow, one-fourth generated an event-driven flow and the remaining nodes generated both type of flows in the network. We introduced packet losses in the network by dropping packets at intermediate nodes and observed the

average latency. Latency was measured as the time taken by a packet to reach the base station from the time it was generated. From Fig. 7, we observe that the latency is proportional to ETT because packets from far away nodes have to traverse many hops to reach the base station. We also observe that the maximum average latency observed was less than 2.5 seconds for a 100 node network.

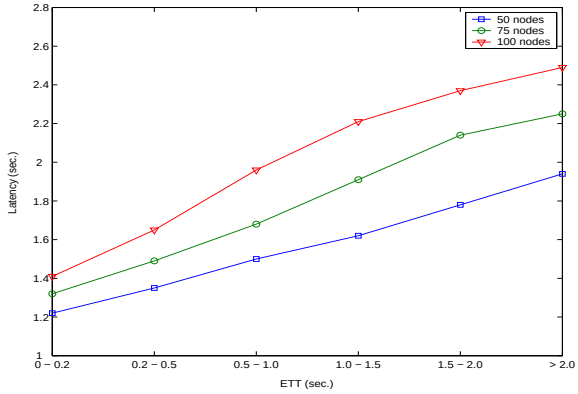

Fig. 7. Average packet latency

D. Energy spent for different reliability

We simulated a 100 node network for 5000 simulator seconds to study the energy spent in the network for different levels of reliability. Nodes generated both continuous and event-driven flows in the same network. To simulate an error-prone network and generate packet loss, every node was forced to randomly drop 30% of the packets flowing through them.

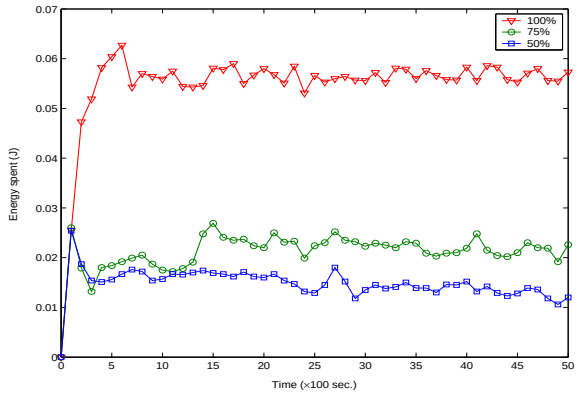

Fig. 8. Energy consumption for a 100 node network

Fig. 8 shows the energy spent in the network every 100 seconds. We observe that the nodes spend significant amount of energy in providing complete reliability to the applications. After 5000 seconds, the total energy spent in the network for providing 100% reliability was 2.78 J, 1.06 J for 75% and 0.77 J for 50% reliability. The energy spent in providing 100% reliability was 2.61 times the energy spent for 75% reliability and 3.59 times the energy spent for 50% reliability. Through the controlled variable reliability mechanism, STCP saves considerable energy and increases network lifetime.

VI. Conclusions and Future work

In this paper, we have presented Sensor Transmission Control Protocol (STCP): a generic, scalable and reliable transport layer protocol for sensor networks. Data flows generated by sensors were classified as continuous and event-driven. Based on flow characteristics, rate of transmission and required reliability, STCP adapts itself to maximize throughput in an energy-efficient manner. Most of the functionalities of STCP are implemented at the base station, thereby saving considerable energy at the sensor nodes. We implemented our protocol in TinyOS and conducted exhaustive simulation experiments using TOSSIM. We studied the impact of incorrect timers and verified that the latency induced was within tolerable limits. We also showed that STCP increases network lifetime through controlled variable reliability.

In the future, we plan to implement STCP on a real sensor test-bed and compare the results with those obtained in our simulations. We also plan to study the effect of different MAC layer and network layer algorithms on STCP.

Acknowledgement

We would like to thank NIST for partly supporting this work through a STTR grant.

References

[1] C. Wan, A. Campbell, L. Krishnamurthy. PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. *In Proc. of WSNA02, Atlanta, Georgia, USA*, Sept. 2002.

[2] Yogesh S., O. B. Akan, Ian F. Akyildiz. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. *In Proc. of MobiHoc03, Annapolis, Maryland, USA*, June 2003.

[3] C. Intanagonwiwat, R. Govindan, D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. *In Proc. of MobiCOM '00, Boston, Massachussetts*, August 2000.

[4] F. Stann, J. Heidemann. RMST: Reliable Data Transport in Sensor Networks. *In Proc. of SNPA, Anchorage, Alaska*, April 2003.

[5] J. Postel. RFC 768: User Datagram Protocol. August 1980.

[6] K. Ramakrishnan and R. Jain. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks. *ACM Transactions on Computer Systems*, May 1990.

[7] P. Blum, L. Meier, L. Thiele. Improved Interval-based Clock Synchronization in Sensor Networks. *In Proc. of IPSN04, Berkeley, California, USA*, April 2004.

[8] Philip Levis, Nelson Lee, Matt Welsh and David Culler. TOSSIM: Acurate and Scalable Simulation of Entire TinyOS Applications. *Proceedings of 1st International Conference on Embedded Networked Sensor Systems*, 2003.

[9] Qun Li, D. Rus. Global Clock Synchronization in Sensor Networks. *IEEE Infocom04, Hong Kong, China*, March 2004.

[10] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson. RFC 2960: Stream Control Transmission Protocol. *Network Working Group*, Oct. 2000.

[11] RFC 793. Transmission Control Protocol. Sept. 1981.

[12] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, August 1993.

[13] S. Tilak, N. B Abu-Ghazaleh and W. Heinzelman. Infrastructure Tradeoffs for Sensor Networks. *In Proc. of WSNA02, Atlanta, Georgia, USA*, Sept. 2002.

[14] V. Jacobson. Congestion Avoidance and Control. *Proceedings of the ACM SIGCOMM Symposium*, August 1988.

[15] W.R. Heinzelman, A. Chandrakasan and H. Balakrishnan. Energy-effi cient communication protocol for wireless micro sensor networks. *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, 2000.