Reliable Wireless Data Transfer for Wireless Sensor Networks

Helen Chung, Reena Gupta Knight, Alex Liu, Evong Nham, Joshua Waldman

Turned in for INFS612 on July 17, 2006 and Presented on July 19, 2006
George Mason University
Fairfax, Virginia, USA

Abstract

Wireless Sensor Networks (WSN) are an emerging technology being used for a broad range of applications, including environmental monitoring, acoustic detection, military surveillance, and structure monitoring. The criticality of many of these missions require that data be reliably collected and transported across the network. Existing network protocols, such as TCP, are inefficient in WSNs because they are extremely resource intensive and their behavior is not conducive to supporting wireless technology. WSNs require communications protocols that minimize energy consumption because of sensor node resource limitations, and also require limited hardware because of sensor node size. A number of protocols and concepts have been proposed to address these issues and increase reliability of data transmission in WSNs. PSFQ (Pump Slowly Fetch Quickly) is one such proposed protocol that has been applied to wireless networks that minimizes error propagation by handling errors in-network (i.e. Hop-to-Hop) as opposed to end-to-end. RMST (Reliable Multi-Segment Transport) is another proposed concept that involves a new data transport layer that guarantees delivery and loss detection and repair. PSFQ and RMST introduce advantageous ways for increasing WSN transmission reliability; however, through further analysis it has been proposed that a combination of these protocols in conjunction with other networking concepts results in a more reliable implementation.

1. Introduction

Computer networks are an essential part of today's operating infrastructure. Many aspects of the implementation of these networks are taken for granted – in particular, data reliability and integrity. The basic concept of data reliability has been critical to the advancing development of

these networks. Without a means of reliable data transfer (hereafter known as RDT), it was impossible to ensure that the received data was identical to the data that originated from the sender. With the advent of the TCP/IP stack in the 1970's, reliable data transfer had become a de-facto requirement in the development of future computer networks.

Wireless sensor networks (hereafter known as WSNs), like other communication networks, also require a means of reliable data transfer. The natures of most WSN applications – remote surveillance networks, detection networks, and monitoring networks – essentially dictate that reliable data transfer must exist in one form or another. However, since WSNs are composed of diminutive sized nodes that are shackled by many design constraints, existing RDT protocols cannot be easily imported to WSNs.

Because WSN nodes are miniature and are to be deployed typically in remote locations, they must be self sufficient (in terms of resources) and completely autonomous. As a result, energy management and hardware complexity are the leading critical design constraints which affect nearly all other aspects of the WSN. The Internet's TCP/IP protocol provides a means of RDT through the use of re-transmissions and ACKs. While this works well in networks where energy management and hardware size is of no concern, it is not a suitable solution for WSNs. In WSNs, the goal is to create a minimalist design that is highly efficient. The proposed solution, Reliable Wireless Data Transfer for Wireless Sensor Networks (RWDT for WSN) addresses these issues.

[illegible]e earch Pro e

The miniature size of the WSN nodes critically affects the design of both the hardware and the software that goes into it. The need for the node to be energy efficient in addition to its small size complicates the design tremendously. Because the hardware is size-constrained, it cannot be overly complex, which results in software algorithms that also cannot be overly complex; in fact, the software must be designed such that hardware is not bottlenecked during software execution. Therefore, a cursory glance at the TCP/IP implementation of RDT indicates that while it is an excellent example of data reliability, integrity, and robustness – it is clearly over-engineered for WSN applications. The overhead of sending too many ACK packets and too

many re-transmissions due to timeouts or losses are counterintuitive to an ideal RDT implementation specific to WSNs.

***.*e a ed or**

Various protocols and concepts have been introduced to provide reliable data transport for wireless sensor networks. According to an article, "RMST: Reliable Data Transport in Sensor Networks" by Fred Stann and John Heidemann of USC/Information Sciences Institute, current sensor networks experience a high loss rate in comparison to wired networks as a result of hardware energy constraints and radio interference. As a result, to improve reliability of data transport within wireless sensor networks, it is imperative to institute protocols that are not resource intensive and can also handle loss detection and repair [6]. Fred Stann and John Heidemann introduce the Reliable Multi-Segment Transport (RMST), a new transport layer for Directed Diffusion that provides guaranteed delivery and fragmentation reassembly in support of improving WSN reliable data transport.

RMST involves increasing reliability at not only the transport layer, but at the Medium Access Control (MAC) layer, as well [6]. However, because our research is focused to transport layer protocols, we will only address the two transport layer paradigms introduced by Stann and Heidemann for RSMT: *End-to-End Selective Request NACK* and *Hop-by-Hop Selective Request NACK and Repair from Cache.* The *End-to-End Selective Request NACK* is a scheme in which only sinks (receiver end-points) make repair requests to the source for missing fragments. These repair requests travel from the sink to the source on a reverse reinforced path and the missing data is retransmitted by the source node[6]. The *Hop-by-Hop Selective Request NACK and Repair from Cache* scheme, on the other hand, allows nodes on the reinforced path from source to sink to make repair requests when they sense a missing data fragment [6]. Each of these nodes caches the fragments that make up the larger data entity, and when missing fragments are sensed, they send a repair request to the next hop on the reverse reinforced path [6]. Because each node caches the fragments, if the requested fragment is in the local cache of the next node on the reverse reinforced path, a response is sent. If not, the NACK is forwarded to the next hop toward the source [6].

The Pump Slowly Fetch Quickly (PSFQ) is another proposed reliable transport protocol that introduces "hop-by-hop error recovery in which intermediate nodes also take responsibility for loss detection and recovery so reliable data exchange is done on a hop-by-hop manner rather than an end-to-end one [7]."

PSFQ eliminates error buildup as well increases scalability by dividing transmission into a series of single hops [7]. PSFQ is comprised of three functions: message relaying (pump operation), relay-initiated error recovery (fetch operation) and selective status reporting (report operation) [7]. A source node "injects" the message and the intermediate nodes buffer and relay messages[7].

The pump operation is associated with the "inject message," whose header consists of the following four fields: file ID, file length, sequence number, and time-to-live (TTL) field[7]. The source continues to "inject" messages to its neighbors every T_{min}. Neighbors that receive this packet will compare it to their local data cache and will discard any duplicates[7]. If the message is not found in the cache, PSFQ will buffer the message and decrease the value in the TTL by 1. If this value in the TTL field is not zero and there is no gap in the sequence number, then PSFQ forwards the message. If there is a gap in the sequence number, the node goes into "fetch mode" [7]. A fetch operation is an act of requesting a transmission from neighboring nodes when loss is detected at a receiving node. PSFQ aggressively sends out NACK messages to its immediate neighbors to request missing segments [7]. The neighboring node replies when the missing segment is found in its data cache. The NACK message is never propagated unless the missing data is not found in the cache [7]. In addition, PSFQ supports a report operation designed to feedback data delivery status information to users in a simple and scalable manner [7].

IV. Solution/Analysis

One of the reasons that research needs to be done into reliable data transfer in wireless sensor networks is because TCP is the reliable transport protocol for the internet, but TCP is not the best solution for wireless sensor networks. TCP was developed for network use before wireless networks really came into play. Therefore, when TCP was optimized and refined, only wired networks and their uses/needs were taken into consideration. Wireless sensor network

devices have the additional limitations in that the devices are usually small and energy consumption is a huge factor in how long the network can survive.

One of the problems with TCP and wireless networks in general is in relation to how TCP handles packet loss. When packet loss is observed, it assumes that it is probably a result of congestion. But, in wireless networks, this should not be assumed. When loss is detected, transmission rate lowers and the data is resent. Since congestion is often not the cause of loss in wireless networks, a lost packet now results in slower transmission when it is not necessary to slow down [1],[6].

Hardware limitations exist because of the size and power consumption. The amount of code and RAM necessary to implement TCP is too big for a small, embedded system [3]. Also, making the hardware too complex can make the sensor consume more power, which is one of the top concerns with wireless sensor networks.

Yet another problem with TCP/IP is that the header size is not appropriate for the data size and frequency that is typical with wireless sensor networks. The data collected and sent is in the realm of tens of bytes, but TCP/IP header can cost 40 bytes or more [3]. In general, it is not very efficient for the header to double the size of the packet being sent because that is a lot of overhead.

Because of these factors, traditional TCP – the protocol used for reliable data transfer over the internet, is not the best solution for wireless sensor networks. WSNs need to have a protocol formulated with their capabilities and limitations in mind. Some protocols have already been designed to address some of these issues, but they are not perfect.

Benefits of the PSFQ protocol design is that it is simple and has minimum requirements [7]. PSFQ divides transmission into a series of single hops in order to eliminate error buildup and increase scalability, thus reducing communication cost by minimizing signaling [7].

One of the major drawbacks of PSFQ, however, is that it does not provide any reliability for single package messages because it uses a NACK based scheme [4]. In order to achieve the pump slowly operation, PSFQ forwards messages in sequence which results in using more bandwidth than necessary [4]. Additionally, PSFQ uses hop by hop error recovery where intermediate nodes take the responsibility for loss detection as well as recovery to ensure a reliable date exchange [7]. When a source “injects” messages into the network, the intermediate

nodes “buffer and relay messages with the proper schedule to achieve loose delay bounds [7].” A relay node maintains a data cache which is used to identify data loss [7]. The involvement of these intermediate nodes results in higher costs as well a requirement for more allocated cache space [7].

Similarly, advantages and disadvantages exist for each of the RMST paradigms with respect to WSN data transport [6]. First, the *End-to-End Selective Request NACK* paradigm. Advantages of the *End-to-End Selective Request NACK* paradigm include the handling of loss detection and repair[6]. Selective Request NACK messages help to ensure that the network is not overloaded with NACK messages being sent by multiple nodes, in addition to the sink, thus reducing the potential for network congestion. A disadvantage posed by this end-to-end message exchange, however, is the lapse in time that it takes to identify missing data and retrieve it from the source[6].

This drawback is addressed, however, in the RMST *Hop-by-Hop Selective Request NACK and Repair from Cache* paradigm [6]. The Hop-by-Hop and caching reduces retransmission time because nodes along the path can retransmit data locally stored in their cache, rather than requiring that the missing data come from the source[6]. This also helps to limit power loss that often results from end-to-end transmissions. (Stann) Although caching poses a number of benefits, it does pose disadvantages, as was aforementioned above in the discussion about PSFQ caching [6]. In the RMST *Hop-by-Hop Selective Request NACK and Repair from Cache* paradigm, all nodes along the reinforced path have the capability to cache, which increases the hardware requirements and energy consumption of nodes as they maintain buffers to identify data loss[6].

This analysis of existing WSN transport protocn92(r)9a6loss[. dchintaidiiry-11(uln-2(h)-10()-2(r)3(ba)

acting as a router and directing data to the appropriate destination via other seed node interfaces. Communications between seed nodes is done via PSFQ. By concentrating PSFQ communications between seed nodes, energy consumption, hardware requirements, and therefore hardware costs are greatly reduced because the number of nodes in the network required to cache and maintain a history of data is reduced. A vast majority of nodes within the network are slaves, requiring minimum hardware requirements, unlike most networks implementing PSFQ.

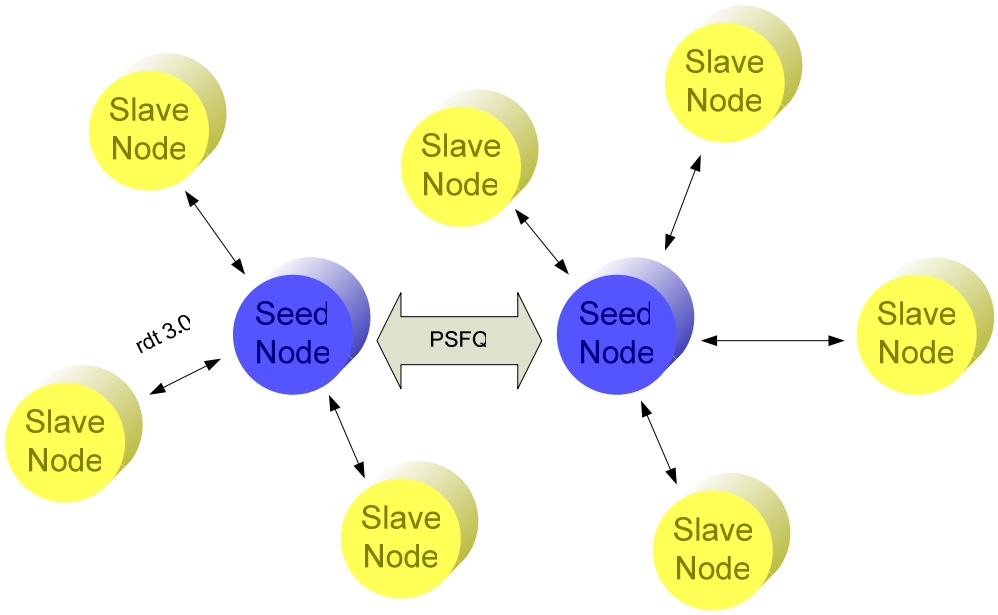

Figure 1

Methodology

Context

The RWDT protocol consists of two types of nodes, a seed node and a slave node. The slave node is a low-power sensor with limited communicative range in an ad hoc distribution. On the other hand, a seed node is a higher-powered machine, with greater computational capacity and communicative range. The distribution of the seed nodes can vary depending on application, from static or ad hoc. Each node in the network is uniquely identifiable by an ID.

Connection Establishment

In order to establish a reliable connection, RWDT inherits TCP's concept of a three-way handshake. Seed nodes initiate the connection establishment handshake by broadcasting a connection request message to nodes local to it as illustrated in Figure 3. The connection request message consists of the seed node's ID and a 1-bit field that indicating that the seed node is accepting new connections. Unconnected slave nodes within the seed node's communicative range will receive this message and respond according by returning a connection request ACK addressed to the seed node that contains the ID of that slave node. Upon receipt of this ACK, the seed node sends a final connection confirmation ACK, echoing the seed and slave node IDs. At this point, a connection is established between the nodes, and the slave nodes can begin reporting and receiving information. A single seed node can connect with multiple slave nodes local to it.

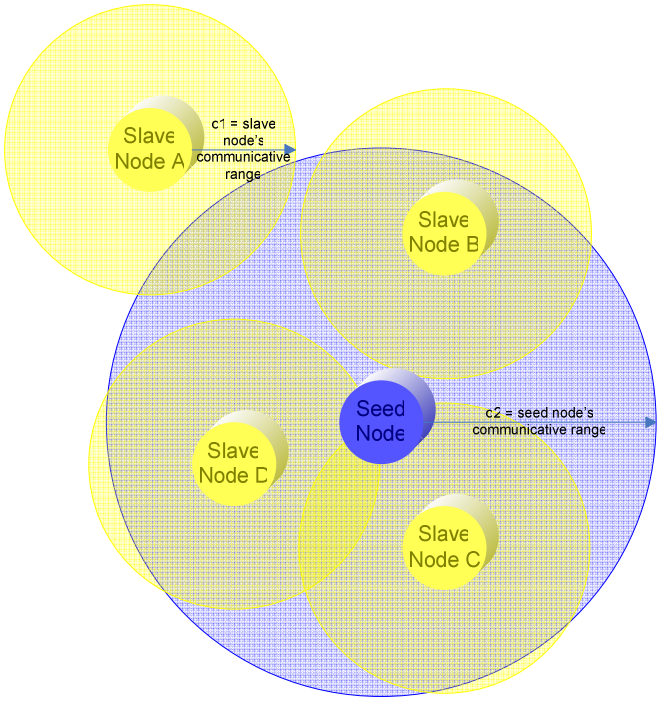

Figure 2

Figure 2. This figure demonstrates how a connection establishment message is broadcasted. The seed node would be able to connect with Slave Node B, C, and D but not A.

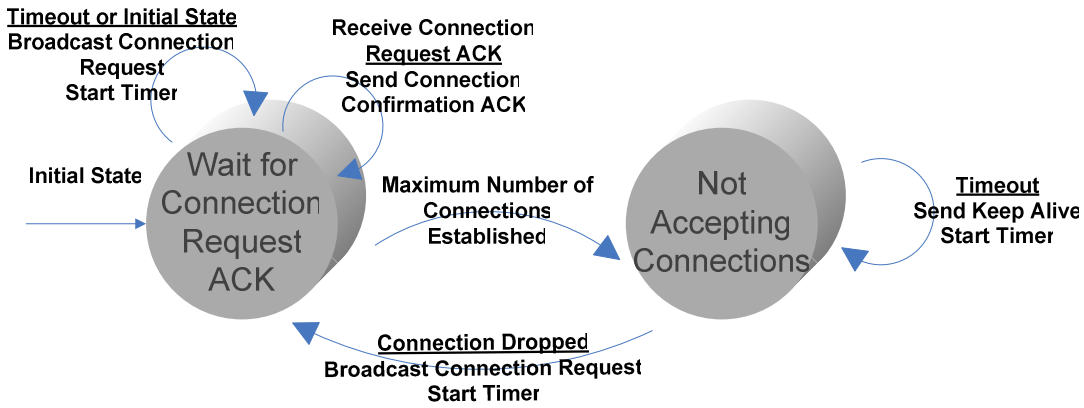

Figure 4

The slave node state diagram is shown in Figure 5. The slave node begins in the *wait for connection request* state, where it waits for the seed node's initial broadcast. If a broadcast message is received, it will send an ACK to confirm that it has received the broadcast, and then wait for the seed node's confirmation ACK. If the window for the seed node expires (due to either timeout or packet loss), it will resend the ACK, then stay in the *wait for ACK* state until it receives the confirmation ACK from the seed node. Once the confirmation ACK has been received, the nodes are synched and a connection is established.

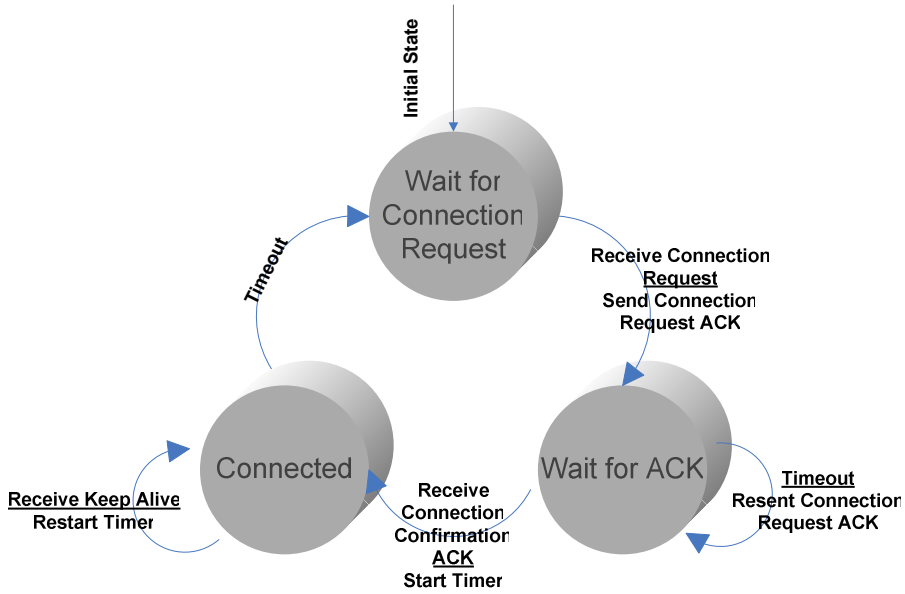

Figure 5

In the event that a seed node suddenly becomes inaccessible, its children slave nodes will soon realize this when it does not receive the *keep alive* message within the proper time limit. Generally, the timeout period for a dropped connection should be relatively longer because re-establishing a connection implies overhead of communication and this should be avoided. Especially in monitoring or surveillance applications, where the information is not as critically time sensitive, longer timeouts result in overall less retransmissions and a longer operating life for the sensors. It should also be noted that for wireless connections, lost packets are more often a result of changing environments and surroundings, rather than as a result of packet congestion in the network.

Communication to Parent Seed

Communication between the slave nodes and their respective seed nodes is simple and straightforward. Because it is a one hop transmission, a complex protocol such as TCP (or even RWDT) would be far over-engineered for this application. Instead, a very basic protocol such as one based on the reliable data transfer model 3.0 is employed. The slave nodes should not have to deal with the overhead of pipelining or threading, since they are not responsible for reporting to any nodes other than the seeds nodes.

While using a model based on rdt 3.0 implies a stop and wait protocol, it is the most ideal implementation in the context of this research, which is assumed to be basic surveillance and monitoring. Clearly, in particular applications of WSN this can prove to be a bottleneck – especially when sensor updates become increasingly frequent, or when the slave node data payload exceeds the size of a single packet. In such cases, it is recommended that a protocol implementation designed for multiple packet communication such as the Go-Back-N or Selective Repeat models be used.

Since the seed nodes operate on a PSFQ protocol, they require that an initial injection request packet sent from the slave node. This injection request packet notifies the seed node that the forthcoming information is to be broadcasted to the other seed nodes within the WSN.

Communication Between Parent Seeds

The PSFQ (Pump Slowly, Fetch Quickly) protocol, described in previous section, will be implemented for inter-seed communication.

According to RWDT protocol, a typical transaction would be as follows:

1. The seed node receives an injection request message from one of its slave nodes. The seed node broadcasts the packets of this message every T_{min} minutes until all data fragments have been sent.
2. Upon receiving a packet, the neighboring node will check its buffer for the message. If it has not been received before, i.e. it is new, it will save it to the buffer and decrease the time-to-live (TTL) value.
3. If the TTL value is not zero and sequence number matches that it is expected, the message will be relayed to its neighboring nodes.

V. Summary

We have discussed ways to implement data reliability in the transport layer in the context of WSNs. The very nature of a WSN precludes the possibility of using TCP effectively as a means of reliable data transfer. A WSN's simple hardware, low power requirements, and lack of complexity means that a robust protocol like TCP would be over-utilizing what little resources are available. Instead, a simpler, more efficient protocol would be effective for WSNs, which are typically deployed and designed for a very focused purpose.

Prior research attempting to devise an efficient reliable data transfer protocol for WSNs has resulted in numerous ad hoc solutions. We studied the most widely documented two implementations: RMST and PSFQ. Both of these implementations focus on in-network reliability vs. TCP's end to end reliability. In-network reliability is the preferable solution for WSN due to the reduction of re-transmissions and requests between two nodes in the event of data loss. This helps to significantly reduce unnecessary communication which would otherwise drain the power on these nodes.

However, even these schemes have their flaws. PSFQ requires that all nodes be high powered and able to buffer all incoming messages. This results in an overly excessive

distribution of nodes that require both more power and more hardware. RMST provides no guarantee of latency or delivery order. In light of these downfalls, RWDT attempts to fuse the best concepts of each of these implementations, along with other concepts of reliable data transfer. RWDT reduces the PSFQ implementation to a few select seed nodes which provide control and routing for communication throughout the WSN. The slave nodes, or sensors, have been reduced to being simply information broadcasters. Once they are connected with their parent seed nodes, they are simply responsible for only reporting data.

Even so, RWDT, like most other WSN protocol implementations, is not flawless. WSNs are inherently a compromised solution. Their size, mobility, stealth, and flexibility come at the cost of limited power, limited computation power, and limited robustness. In the end, the goal is to find a solution that is able to balance these limitations to create a protocol well suited for any particular application.

RWDT's distributed model is a compromise as well. In cases where seed nodes have died or gone offline, that particular geographical region may have a loss of data. While this is highly undesirable, consider the other alternative. Imagine a model in which one main node collects the data from all the network nodes. This centralizes operations and maybe simplifies protocol as well, but should the main node go offline, the entire network becomes useless. That single point of failure because a critical liability.

It should be noted that RWDT is but one of many possible solutions. Whether or not it is the best compromise is subject to debate, but within the context of our research it appears to be the solution most easily adaptable to multiple applications. Many possible points of flexibility are also noted in this design, such as the replacement of RDT 3.0 with a smarter model such as Go-Back-N or Selective Repeat.

Lastly, within the scope of this project, we assumed that WSNs must maintain a strict protocol stack, much like the Internet does. However, WSNs typically have a very focused intent and application. The Internet benefits from this layer separation for maintainability and interoperability reasons. For WSNs more focused purpose, it is not inconceivable that interweaving layer functionalities could help simplify hardware and software functionality across the network. This is a topic noteworthy of further research in the field of WSNs.

Reference

[1] T. Braun, T. Voigt, A. Durkels, "Energy-Efficient TCP Operation in Wireless Sensor Networks, " Institute of Computer Science and Applied Mathematics, University of Bern, Switzerland. 2005.

[2] S. Kim, R. Fonseca, D. Culler, "Reliable Transfer on Wireless Sensor Networks," University of California at Berkeley. Berkeley California. 2004.

[3] X. Luo, K. Zheng, Y. Pan, Z. Wu, "A TCP/IP implementation for wireless sensor networks," in IEEE International Conference on Systems, Man and Cybermetrics," Zhejiang Province, China. 2004.

[4] S.-J. Park, R. Vedantham, R. Sivakumar, and I. F. Akyildiz, "A scalable approach for reliable downstream data delivery in wireless sensor networks," in *Proc. Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'01)*, Tokyo, Japan, May 2004.

[5] Ramesh. S, "A Protocol Architecture for Wireless Sensor Networks," School of Computing, University of Utah. Salt Lake City, Utah.

[6] F. Stann and J. Heidemann, "RMST: Reliable data transport in sensor networks," in *Proc. 1st IEEE Intl. Workshop on Sensor Network Protocols and Applications (SNPA)*, Anchorage, Alaska, May 2003.

[7] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy, "PSFQ: A reliable transport protocol for wireless sensor networks," *Proc. First ACM Intl. Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, 2002.