Reliable Multihop Bulk Transfer Service for Wireless Sensor Networks

Péter Völgyesi, András Nádas, Ákos Lédeczi
Institute for Software Integrated Systems, Vanderbilt University
Nashville, TN, USA
{peter.volgyesi, andras.nadas, akos.ledeczi}@vanderbilt.edu

Károly Molnár
Embedded Information Technology Research Group,
Hungarian Academy of Sciences – Budapest University of Technology and Economics
Budapest, Hungary
kmolnar@mit.bme.hu

Abstract

Multihop message routing is a well studied and widely documented area of research in wireless sensor networks (WSN), where the dynamic and lossy nature of the wireless medium and severe resource constraints pose major challenges. The vast majority of related research focuses on reliable and power-efficient transfer of small amounts of data, such as low frequency sensor readings or event detections. However, in a few but notable WSN applications, reliable transfer of mass data is essential. The paper describes an efficient multihop bulk transfer service along with a complete sensor network application utilizing it for on-demand image transfers. The paper focuses on the unique problems of the service, such as resource allocation, flow control and mobility throughout the modeling, simulation and implementation phases. Models of the protocol have been built and simulated in a probabilistic wireless network simulator. The prototype implementation targets TinyOS, a well-known WSN operating system.

1. Introduction

Classic wireless sensor network applications usually involve many-to-one periodic [4] or event-triggered [9] data collection services. These services focus on the power-efficient and reliable transfer of relatively small amount of data through unreliable wireless links and nodes with severe resource constraints. Duty cycling, smart scheduling and in-network data aggregation are well known techniques to improve power efficiency, while redundant links and packet retransmissions can increase the reliability of such services.

However, there are a few but interesting sensor applications which depend on a mass data transfer service. Notable examples are imaging sensors that can easily capture large amount of data, acoustic beamforming applications, where raw samples need to be transmitted to a centralized station. Even in those applications where the sensors can locally process and can extract features from the captured data the capability of transferring unprocessed sensor readings is essential in the development and test phases [9]. There are several unique problems that need to be addressed in a mass data transfer service. Because of the limited and shared communication bandwidth, redundant packets—from the same node or along alternative paths—are prohibitive in these scenarios. Instead, the protocol has to provide built-in mechanisms to increase the probability of successful packet transmissions. Potential solutions include resource (channel) arbitration and continuous link status estimation. The limited storage capabilities of the intermediate nodes allow to store only a tiny fraction of the data at a time, thus effective end-to-end and link-by-link flow control mechanisms must be employed to prevent the accidental loss of data.

As the application context of our bulk transfer service we consider a sensor network utilizing both imaging and non-imaging sensors, sensorless router nodes and one or more base stations as they are shown in Figure 1. Sensors, routers and base stations can be dynamically added, removed or moved within the network and these changes should be detected and reported at the base stations with limited delay (few seconds). Changes in the network topology due to node failures or fading communication links must be transparent to the user. Event detections originating at non-imaging sensors (PIR sensors, accelerometers, magnetometers, microphones and pressure sensors) must be forwarded to every base station in the network. Also, each individual node should be accessible from each base station all the time for modifying operational parameters and to query the status of the sensor node. The operator at the base station should be able to request moderately large (10-30 kbytes) images from any of the imaging sensors and the multihop network should forward these images to the the base station with minimum delay and low packet loss. During the image transfer the mobility of the base station can be constrained, however the station should be able to move without any restrictions between transfers.

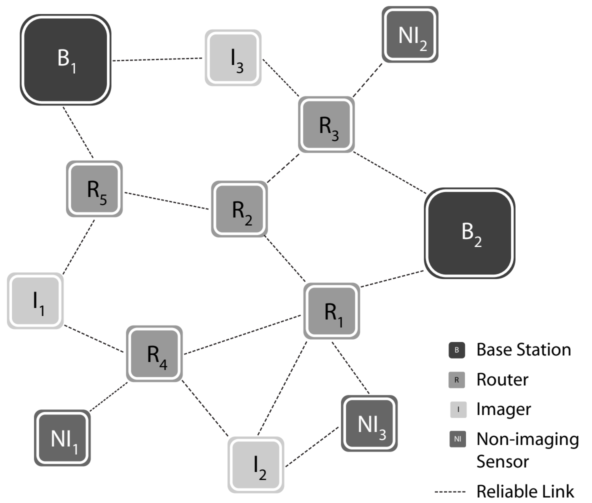


Figure 1. Wireless image sensor application. Images are transferred to one or more base stations on-demand through an ad-hoc wireless network.

The sensor application utilizes TinyOS, a well-known WSN operating system [3] running on MicaZ nodes [8] with low power (1 mW) IEEE 802.15.4 compliant radios operating in the 2.4Ghz ISM frequency band. Imaging and non-imaging sensor functions are implemented on sensorboard add-ons connected to the MicaZ node through a UART communication interface. XBow Stargate single board computers—also equipped with MicaZ nodes—carry out the base station tasks and connect the MicaZ network to WiFi-enabled PDAs.

In this paper, we explore the unique problems of a bulk transfer service throughout the modeling, simulation and implementation phases. In Section 2 we give a brief overview of related work. Section 3 presents the mechanisms and architecture of the bulk transfer service. We also investigate an efficient link estimator and neighborhood management service, which is essential to the bulk transfer protocol. As a first step towards implementation in Section 4 we build and evaluate a simulation model of the service using Prowler, a probabilistic wireless network simulator. Section 5 presents our TinyOS-based implementation and briefly describes the most important network management and non-imaging data services. We evaluate the performance and reliability of the service in Section 6.

2. Related Work

There are several well studied and documented routing schemes and media access protocols exist in wireless sensor networks. One of the most influential results on our work was published by Woo et al. [13], which aims at an accurate link estimation method in dynamically changing ad-hoc topologies. Our connectivity measures build on this method, however—as it is discussed later in the paper—we decided to employ active ping messages for neighborhood monitoring. In their work they also explore different routing policies (shortest path, minimum transmission, broadcast, destination sequenced distance vector) on top of the link estimator service, but their focus is on a many-to-one, periodic data collection scenario.

In the TinyOS community [12] is one of the most prominent and early study on media access mechanisms and transmission control. They propose an adaptive rate-control mechanism for energy efficient media access with fair bandwidth allocation. Again, their work aims at relatively sparse network traffic. In a closely related paper Polastre et al. [7] describe B-MAC, the prevailing media access scheme in TinyOS. They argue that a flexible and on-the-fly reconfigurable media access protocol—providing bidirectional application interfaces—can result in better overall performance. The prototype implementation of our bulk transfer service was built on top of this infrastructure, where the advantages of their approach were apparent.

Some alternative MAC protocol offerings have been based on the RTS-CTS scheme inspired by IEEE 802.11 wireless LANs [2]. S-MAC [14] provides this mechanism as a general purpose MAC layer, although it is more sensitive to changing network conditions and relies on more complicated mechanisms (eg.: synchronization, scheduling) then B-MAC.

PSFQ (Pump Slowly,Fetch Quickly) is a reliable transport protocol with hop-by-hop acknowledgements suitable for sensor network reprogramming. However, in this application the packets are originated at the base station and disseminated to all the nodes in the network. Straw (Scalable Thin and Rapid Amassment Without loss) is a reliable data collection service developed at UC Berkeley. Straw was built on Drip and Drain [11] and it draws the complexity to the receiver (PC), while senders (wireless nodes) are kept simple and light-weight. Although the service is available in the TinyOS source tree, the results are not yet published.

3. Bulk Transfer Service

One of the most significant differences between sparse message routing and bulk transfer is the cost of redundant messages. These messages can greatly improve the reliability of the former one without any significant performance loss, but can drastically degrade the throughput in the latter case. On the other hand, administrative and control messages are superfluous to protect few data packets, but fairly lightweight for longer packet bursts. For the above reasons we decided to employ a *Request-to-Send* (RST) *Clear-to-Send* (CTS) control scheme as it is shown in Figure 2. This is a simplified version of a the approach in 802.11 wireless LANs [2]. For similar reasons WiFi networks selectively employ this handshaking mechanism for protecting longer bursts or larger packets. According to the scheme, large data files are packetized, then the packets are sent through the network in bursts, where the size of a burst depends on the storage capabilities of the nodes. A network node can either be *idling*, *collecting* packets in a burst from another node or from the sensorboard, or *transmitting* previously collected packets towards the base station. A node never accepts packets from more than one bursts at a time, however, multiple concurrent transfers can pass through the node on a burst-by-burst basis. Before the node enters the transmit state, it has to send a request packet (RTS) to the next node in the routing path and wait for an explicit acknowledgement (CTS). If the CTS packet does not arrive—for several possible reasons: the original RTS packet was lost, the receiver was already in *collect* or *transmit* states, or the CTS packet was lost—the sender node tries to retransmit its request for a few times with random backoff delays. Also, if another potential sender node overhears an RTS or CTS or burst data packet from or to another sender, it automatically delays its RTS request or discards a positive CTS acknowledgement. This handshaking protocol provides simple but efficient flow-control on a link-by-link basis, detects asymmetric or weak links and effectively avoids hidden terminal problems. Also, as it is shown in Figure 2, if the underlying MAC layer supports it—as it is the case in the B-MAC implementation in TinyOS[7]—the initial backoff time can and should be lowered (or eliminated) within the burst. Since we expected very low packet loss within the handshake protected bursts and did not want to interfere with the rapid bursts, we decided to employ a negative acknowledgement-based end-to-end protocol for requesting missing/lost packets.

We chose a simple spanning tree protocol for building routing paths, which avoids redundant links and obtains short routes with low hop counts. To provide a robust routing service and to support mobility requirements, base stations build up unique single-use routing trees for each image request. Since the request itself has to be propagated throughout the network anyway, this does not cause significant performance loss. The base station starts the tree building process by broadcasting a routing beacon with the hop count set to zero and advertising the base station as the immediate parent. Any node overhearing this beacon message stores the new information in its routing table and retransmits the beacon after increasing the hop count and designating itself as the new parent. To support multiple base stations and concurrent transfers, each beacon message contains a tree identifier generated by the root (base station). Later on, incoming packet bursts and the associated RTS/CTS messages will refer to the same tree identifier. A simplified routing table is shown in Table 1. This table describes three routes (identified as 0x56, 0x88 and 0x89). In the first tree (0x56) the address of the intermediate parent is 13 and the current node is 2 hops away from the base station with address 1. The *Lock* attribute is set if the current node received or transmitted data packets in the given tree recently, thereby protecting active trees from being evicted from the table.

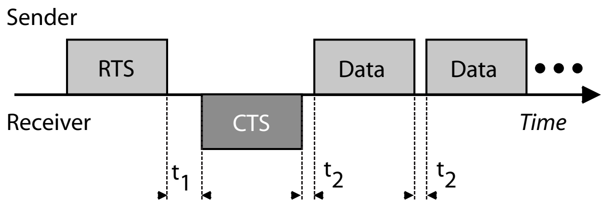


Figure 2. Request-to-Send/Clear-to-Send handshake protects longer packet bursts

Tree ID	Root	Parent	Hop Count	Lock
0x56	1	13	2	0
0x88	2	21	4	0
0x89	2	22	5	1

Table 1. Routing table with spanning tree information

Although spanning trees provide a simple and efficient routing infrastructure, one should carefully consider their weaknesses in low-power wireless environments with inherently lossy and asymmetric communication channels. The previously described handshake protocol can detect weak links in an existing tree, but cannot prevent such a tree from forming. Therefore, we have enhanced the original tree building protocol to accept a potential parent only if the link between the current node and the new parent is ”reliable”. This simple addition to the spanning tree protocol needs a new network service that maintains the neighborhood connectivity information. We decided to collect link statistics with active *ping* messages. Every node periodically transmits a short broadcast message and waits for replies. The initial frequency of these messages—in the first minute after power on or reset—is relatively high to provide fast startup. Also, the intervals between the ping messages are randomized to avoid unfair schedules. Each node maintains a neighborhood statistics table (see Table 2) and increases the *Reply Count* attribute upon receiving a reply from one of its neighbors (new neighbors are added to the table on-demand).

Neighbor	Reply Count	Score
13	9	80
21	7	74
22	8	95
12	2	15

Table 2. Link reliability database

After several (N) rounds the node updates the link reliability scores by incorporating and resetting the Reply Count values in the previous score using and exponentially averaging low-pass filter:

$$score[k] = \alpha * score[k-1] + (1-\alpha) * \frac{100 * reply\_count}{N} \tag{1}$$

In this manner the link monitor service assigns a reliability score (0-100) to all of the neighbors, which can further be used to decide if a potential parent should be accepted in the spanning tree—eg. if its score is above an arbitrarily chosen minimum score threshold. Note, that our metric is solely based on the count of successful packet transmissions, since techniques based on signal strength measurements may result in inferior link quality indicators [1].

Our link estimator can detect asymmetric (because it builds statistics from reply messages) and weak links. Although active ping messages cause additional network traffic, they can identify and discard overloaded nodes as parents (because of dropped packets in the radio stack) even if the physical radio link is reliable towards them. Passive link monitors cannot provide such features.

4. Protocol Modeling and Simulation

Before implementing the protocol and related services in nesC on the motes, we evaluated them in simulation utilizing realistic radio models. The probabilistic wireless network simulator (Prowler) [10] is an event-driven tool that simulates the nondeterministic nature of the communication channel and the low-level communication protocol of the wireless sensor nodes. To produce replicable results while testing the application, Prowler can be set to operate in deterministic mode also. It can incorporate arbitrary number of nodes on arbitrary and even dynamic topology. Prowler models all the important aspects of the communication channel and the application. The tool is implemented in MATLAB, thus it provides a fast and easy way to prototype applications, and has nice visualization capabilities.

The nondeterministic nature of the radio propagation is characterized by a probabilistic radio channel model. A simplified, but accurate model is used to describe the operation of the Medium Access Control (MAC) layer. The applications interact with the MAC layer through a set of events and commands just like in actual TinyOS applications.

The radio propagation model determines the RF signal strength at a particular point in the space for all transmitters in the system. Based on this information the signal reception conditions at the receivers can be evaluated and collisions can be detected. The signal strength from the transmitter to a receiver is determined by a deterministic propagation function (modeling the decay of signal strength with distance), and by random disturbances (modeling the fading effect, the time-varying nature of the signal strength, and other miscellaneous transmission effects.)

The MAC layer communication is modeled by a simplified event channel. When the application initiates a packet transmission, the MAC layer checks if the channel is idle after a random time interval. If not, it continues the idle checking until the channel becomes idle. Before each check it waits for random backoff time. When the channel is found idle, the transmission begins and after a constant time period simulating the transmission time, the application receives an event indicating that the packet has been sent. After the reception of a packet on the receivers side, the application receives another event signaling packet reception or collision depending on the success of the transmission.

Similarly to the real TinyOS framework, Prowler applications are event-based. The simulator signals important events for the application code, such as initialization completed, packet sent, packet received, packet collided and clock ticked. The application in turn can initiate actions such as set clock and send packet. These can cause further events. Several debugging/visualization tools are also available, including switching mote LEDs on/off, drawing lines and arrows, and printing text messages.

Since Prowler was originally targeted at Berkeley Mica and Mica2 motes, we had to extend the radio propagation and MAC layer behavioral models with the characteristics of the the Chipcon CC2420 radio chip employed by MicaZs. We conducted several real-world measurements in different environments (indoor, outdoor, unobstructed and multipath) to infer reasonable values for the fading parameters. The signal strength (P_{rx}) from the transmitter to a receiver is modeled by the following random function:

$$P_{rx}(i,j) = \frac{P_{tx}(i,j)}{1+d^{\gamma}} * [1+\alpha(d)] * [1+\beta(t)] \quad (2)$$

where P_{tx} is the transmit power and d is the distance. Disturbances in distance (eg. multipath effects) and time are modeled by random variables α and β with standard distribution. We consider the packet transmission successful, if the Signal to Interference and Noise ratio ($SINR$) is above a minimum threshold:

$$SINR = \frac{P_{rx}(i,j)}{\sigma^2 + \sum_{k \neq j} P_r x(i,k)} \quad (3)$$

where σ is an arbitrary noise variance parameter. Table 3 contains the empirically chosen values for the parameters in this radio model. MicaZ specific MAC parameter values (bit time, backoff intervals, etc.) were also added to the model.

Parameter	Value	Parameter	Value
α (fading)	0.3	β (multipath)	0.02
σ (rx noise)	0.05	$SINR_{min}$	2.0
γ (decay)	2.0		

Table 3. Radio propagation parameters in Prowler for MicaZ radios

After preparing Prowler for simulating MicaZ-based networks, we created a prototype implementation of the image transfer application in MATLAB, focusing primarily on the bulk transfer and closely related services. One of the most important lessons that we have learnt here was the importance of redundant (automatically repeated) spanning tree beacons for increasing the reliability of the spanning tree formation process.

5. Implementation

According to the application requirements, we had to address the reliable routing problem of non-imaging data as well. Furthermore, these packets might need to be forwarded to multiple base stations at the same time and the routing service should rapidly adapt to changes in the network topology. For the above reasons we decided to utilize a broadcast-based flooding protocol for non-imaging data. Although it may generate unnecessary traffic in certain areas of the network, it promptly adapts to the changing topology—since no routing information is maintained by the nodes. Since these tradeoffs cannot be completely evaluated and compared in the design and early implementation phases, we were prepared for possible policy changes in this routing logic. Our previous work, the Directed Flood Routing Framework (DFRF) [5] enabled us to develop interchangeable policy alternatives without modifying other parts of the application.

The general architecture of the routing framework is shown in Figure 3. At the heart of the framework is the routing engine—responsible for message buffer maintenance and automatic packet aggregation, for communicating with the radio stack and the applications and for driving the policy plug-ins. Policies are state machines that describe which packets need to be rebroadcasted by the node and when. We managed to reuse two existing policy plug-ins, the broadcast policy for sparse non-imaging packets and a gradient convergecast policy—where intermediate nodes rebroadcast a data packet zero, one or more times until it is received from a node "closer" to the root—for network management functions.

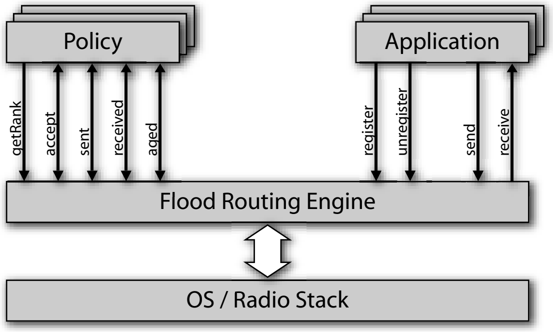


Figure 3. Flood-routing framework components

It seemed to be an obvious choice to implement the bulk transfer protocol as a policy plug-in for DFRF and indeed the first incarnation of the service was implemented this way. After the first rudimentary tests we realized that the unique requirements and non-flooding behavior of the bulk transfer service necessitates modifications to the DFRF interfaces and to some of the internal buffer management algorithms (most of these changes were needed to expose the flow-control mechanism to the application level and for conducting burst transmissions more efficiently). Since many of these changes were not harmonious with the original concepts of the flood routing framework, we finally implemented the service as a standalone set of TinyOS components (see Figure 4. The first of these components, *LinkMonitor* collects, maintains and provides neighborhood connectivity statistics. It maintains a table of up to 16 neighbors and broadcasts ping messages in 20 second intervals with up to 1 second randomization (in the initial rapid discovery phase ping messages are sent in $250ms$-$500ms$ intervals). After 10 ping messages it updates the neighborhood scores according to Eq. 1. The *RoutingTable* component is responsible for building and maintaining spanning tree information. It can store up to 16 spanning tree entries on each network node. Since reliable spanning tree formation is vital for the bulk transfer service, each node repeats the beacon message twice (with a random delay of up to $125ms$). A potential parent is accepted if its score—provided by the *LinkMonitor* service—is above 75, an empirically obtained threshold. The locking interval is set to 2 seconds. The RTS/CTS handshake protocol was implemented in *FlowControl*, a separate TinyOS component. Overheard RTS, CTS or burst packets cause a random backoff between $8ms - 24ms$ before entering into the *transmit* state. If the channel is considered clear the component sends an RTS packet and waits for an explicit CTS acknowledgement with a $60ms$ timeout. The component gives up on the request and signals failure after more then 5 backoff or 3 timeout events, whichever comes first. Finally, the *BulkRouting* component integrates the previous modules, provides the application interface and maintains the burst buffers. The current MicaZ-specific implementation handles data packets with the maximum length of 45 bytes while a single burst contains up to 18 packets.

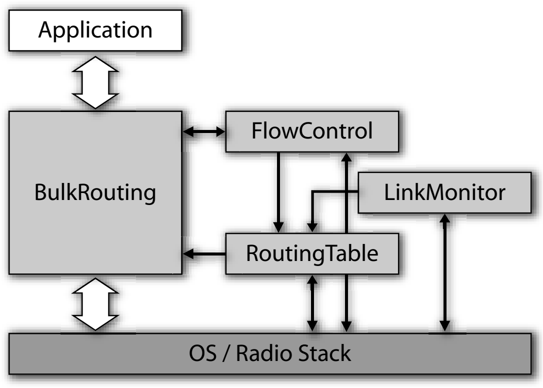


Figure 4. Component architecture of the Bulk Transfer Service

Rich network management facilities proved to be essential not only in the protocol development and test cycles, but in the final application deployment as well. We decided to implement these features early in the development cycle—even before the managed services were ready. We successfully utilized several of our existing TinyOS and Java based utilities—which are part of the standard TinyOS distribution—for network management. *RemoteControl* is a generic service for sending simple commands and/or parameter values to other services or applications running on the nodes. We integrated this tool into the bulk transfer service components to provide access to key parameters and data structures (eg. parent and root addresses in the spanning tree, backoff delays and timeout values). The DFRF framework was also utilized with the convergecast *Gradient Policy* for downloading complete routing tables or link statistic databases to the management station. Its performance characteristics and complexity place this policy between the broadcast-based flooding and the spanning tree based bulk transfer protocol, which makes it perfectly suitable for transferring moderately sized tables (3-4 network packets each). Optimal timing parameters are the cornerstones of the bulk transfer service. Therefore, these values were inferred in real-world test scenarios. For these we used an accurate and mature time synchronization and message time stamping service [6]. The above services and tools are integrated and coherently presented on the network management PC within the *MessageCenter* Java-based framework.

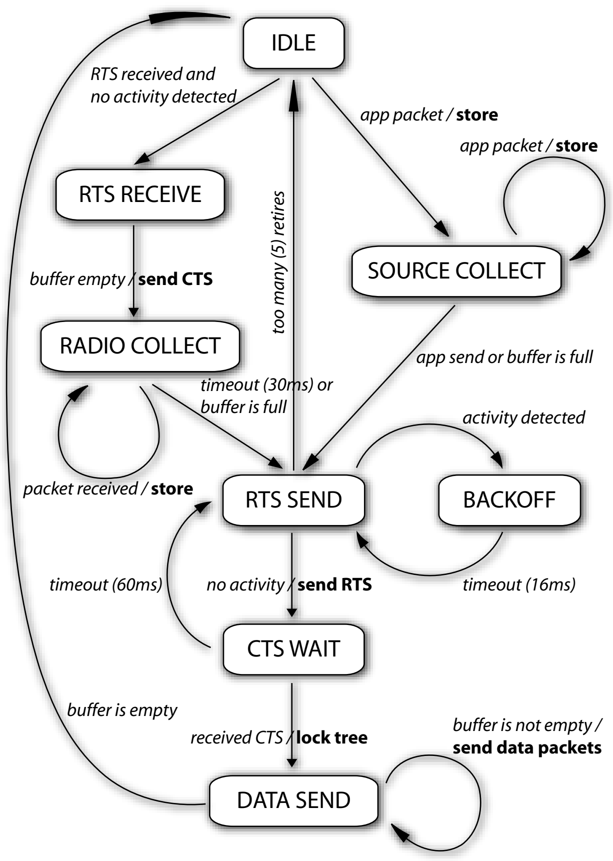


Figure 5. State machine of the Bulk Transfer Protocol

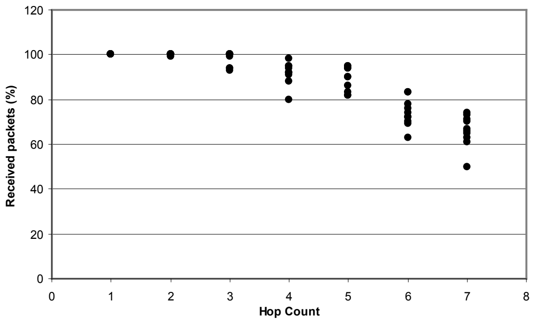
Although in the final application different types of real sensors and PDA gateways were deployed, we had to create an environment with simulated sensor and gateway functions to start the software development before the hardware devices were built and tested. These sensor and gateway simulators were also implemented as *MessageCenter* plugins and were running on PCs connected to one or more MicaZ nodes through UART communication links. Because in low hop-count networks the serial interface turned out to be the real bottleneck, these simulators enabled us to realize the importance of a solid flow-control mechanism very early in the development and to notify the hardware developers in time to incorporate the flow-control mechanism in the real sensors as well.

6. Conclusions

We have evaluated the performance of the bulk transfer service in different deployments and traffic loads. The results in a particularly congested and severe multipath scenario are shown in Figure 6. In this setup the nodes were deployed on the ground with 5-10 meter spacing and multihop routing was assured by a small modification to the *RoutingTable* component. Two different physical topologies were used: first the sensors were deployed on a single line, next they were (almost) randomly scattered in a larger room. Note, that software enforced spanning trees may degrade the overall performance of the service, since they result in denser topologies, then were actually needed to route between the end points. We used the Java-based imaging sensor and gateway simulators to transfer 10KB jpeg image files (286 radio packets with 35 byte effective payload) through multiple hops and did not employ any packet aknowledgement/retransimission scheme for calculating the successful packet reception ratio. Within the first three hops the protocol seemed to be extremely reliable and it remained very usable to five hops. Above this hope count, potential applications need to employ more sophisticated retransmission protocols. The transmission time degraded gracefully (each hop added 1-2 seconds delay) as we increased the number of hops, especially if we consider that the majority of the nodes were in the same collision domain.

The main lesson we learned from our experiences in building a reliable bulk transfer service for wireless sensor networks is that the simple collision avoidance provided by the MAC layer is not enough. With the RTS/CTS handshaking we effectively ”reserve” the channel for the duration of a burst. On the other hand, this has significantly less overhead and it is much more flexible than something like circuit switching.

To be able to handle the unreliable and dynamic nature of the wireless medium as well as to support mobility, the continuous link quality monitoring service and the on-demand



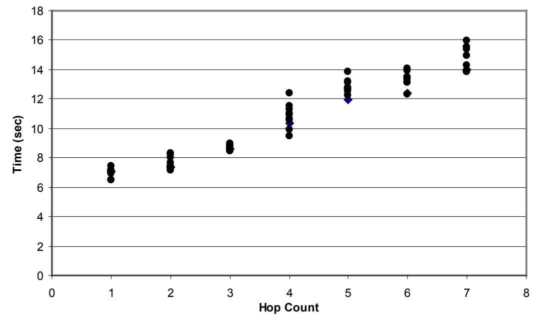


Figure 6. Successful packet transmission ratio and transfer time with different hop counts in a multipath and congested scenario

rapid spanning tree formation are essential in minimizing packet-loss. The final key component in achieving close to 100 % reliability is the NACK-based end-to-end flow control scheme. The protocol can adapt to dynamically changing ad-hoc topologies extremely well between transfer sessions (spanning trees are built on-demand before each transfer), however—in its current form—it cannot tolerate extreme changes within the sessions. This poses an important constraint on the maximum practical size of the bulk data.

As part of an experiment we ported the bulk transfer service to Berkeley mica2 radios. Even though this needed the replacement of the entire radio stack below the service, we only had to modify the timing parameters, but the rest of the source code remained exactly the same. We believe that the service can be easily implemented on other platforms and can provide an often neglected functionality in sensor networks.

References

[1] D. S. J. De Couto, D. Aguayo, B. A. Chambers, and R. Morris. Performance of multihop wireless networks: Shortest path is not enough. In *Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I)*, Princeton, New Jersey, October 2002. ACM SIGCOMM.

[2] M. Gast. *802.11 Wireless Networks: The Definitive Guide*. O'Reilly, 2005.

[3] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. *SIGOPS Oper. Syst. Rev.*, 34(5):93–104, 2000.

[4] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, Sept. 2002.

[5] M. Maroti. Directed flood-routing framework for wireless sensor networks. In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 99–114, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[6] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49, New York, NY, USA, 2004. ACM Press.

[7] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA, 2004. ACM Press.

[8] J. Polastre, R. Szewczyk, C. Sharp, and D. Culler. The mote revolution: Low power wireless sensor network devices. In *Hot Chips 16: A Symposium on High Performance Chips*, Stanford, California, USA, Aug. 2004.

[9] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12, New York, NY, USA, 2004. ACM Press.

[10] G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi. Simulation-based optimization of communication protocols for large-scale wireless sensor networks. In *IEEE Aerospace Conference (CDROM)*, 2003.

[11] G. Tolle and D. Culler. Design of an application-cooperative management system for wireless sensor networks. In *EWSN '05: Proceedings of the 2nd European Workshop on Wireless Sensor Networks*, 2005.

[12] A. Woo and D. E. Culler. A transmission control scheme for media access in sensor networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 221–235, New York, NY, USA, 2001. ACM Press.

[13] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27, New York, NY, USA, 2003. ACM Press.

[14] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1567–1576, New York, NY, USA, June 2002. USC/Information Sciences Institute, IEEE.