Data Transport Reliability in

Wireless Sensor Networks —

A Survey of Issues and Solutions

Andreas Willig, Holger Karl

Abstract

Reliable data transport is an important facet of dependability and quality of service in wireless sensor networks. This paper gives an introduction to the reliable data transport problem and surveys protocols and approaches for this protocol, often developed for particular applications to reflect the application-specific dependability requirements. A joint characteristic of many of the discussed protocols is that they combine mechanisms from several layers to achieve their reliability goals while being energy-efficient. This very need to be energy-efficient precludes Internet-style approaches to reliability – handle it in the end system – and necessitates in-network solutions.

I. Introduction

The specific application, requirements, and techniques of wireless sensor networks – among others, large node number, constrained (energy) resources, data-centric networking, in-network processing – are by now commonplace in the literature. Less well considered is the appropriate notion of reliability:

- Is there any chance to actually detect the phenomena the network is supposed to detect? Are there enough sensors of the right modalities present? Is the area of interest sufficiently covered by sensor nodes? This is the subject of the *coverage and deployment problem.*

- The sensors are cheap and their readings can thus be noisy, giving rise to the *information accuracy problem*. The standard approach is redundancy, i.e. to deliver multiple sensor readings and to improve the signal-to-noise ratio by properly combining these.

Andreas Willig is with the Hasso-Plattner-Institute of Software Systems Technology, University of Potsdam.

Holger Karl is head of the Computer Networks Group, University of Paderborn

This article is based on [1, Chap. 13]

- Even when an event has been reliably detected, this information must be transported over multiple hops towards special sink or gateway nodes and further to the user. This is one instance of the *reliable data transport problem.*

This paper focuses on the reliable data transport problem, for discussion of coverage issues we refer the reader to [2], [3], [4], [5], [6]. The information accuracy problem is beyond the scope of this paper.

Unlike in the Internet, reliability cannot be provided predominantly by the end systems and using transport protocols. Rather, the inherent tradeoffs between reliability, quality of the provided information, and constrained energy requires cross-layer solutions. Applying TCP, for example, to achieve reliability is hampered by several mismatches between TCP and wireless sensor networks (WSN) – unique node addresses vs. data-centric, difficulties to support in-network processing when using destination-terminal-addressed TCP connections, TCP's potential needless repetition of all packets when some losses might well be tolerated, per-packet overhead even with header compression, and TCP's general problems when running over wireless links. TCP might be made more suitable by caching mechanisms [7], [8]; code complexity, on the other hand, seems not to be problematic [9], [10], [11], [12].

This paper is structured as follows. An introduction to the reliable data transport problem is given in the next Section II. The delivery of single packets, blocks of packets, or infinite streams of packets is discussed in Sections III, IV and V, respectively. The paper is concluded in Section VI.

II. RELIABLE DATA TRANSPORT

The problem of achieving reliable transmission between remote nodes over multiple hops despite channel errors, collisions or congestion has at least the following dimensions:

- *Single packet vs. block of packets vs. stream of packets*: the cases of delivering only a single packet on the one hand and of delivering a number or even an infinite stream of packets on the other hand differ substantially in the protocol mechanisms usable in either case. Reliable delivery of single packets can be important for example for highly aggregated data, reliable delivery of blocks is required for applications like disseminating new code or new queries into the network (re-tasking, see [13]). Periodic data reporting is the primary example for streams of packets.

- *Guaranteed vs. stochastic delivery*: Some applications require guaranteed delivery. Examples are: (i) reporting of very important events from sensors to a sink node, (ii) the distribution of new code or queries from the sink node to sensors (re-tasking, see [13]), or (iii) handing over the target state in a tracking application between nodes close to the target trajectory (see for example [14], [15]). Other situations might well tolerate a certain degree of losses. For example, when many sensors

transmit strongly correlated sensor readings, occasional loss is tolerable. One way to specify the tolerable amount of losses is to prescribe a *delivery probability*. In general, the higher the desired delivery probability, the higher are the energy costs needed to achieve this.

- *Sensors-to-sink vs. sink-to-sensors vs. local sensor-to-sensor*: as opposed to other types of networks communication in sensor networks does not take place between arbitrary nodes, but is either from (groups of) sensors to a single or a few sink nodes, from a sink to (groups of) sensors or locally between (groups of) sensors when these run collaborative signal processing algorithms.

There is no single protocol covering all the points in this design space (TCP is not doing this either), but several solutions have been developed for single points or small point sets.

III. SINGLE PACKET DELIVERY

Single packet delivery is important for example when a sensor node wants to deliver aggregated data to a sink node. Most of the literature focuses on the case of stochastic guarantees and, accordingly, the most important performance measure is the *packet delivery probability* (PDP).

A. Approaches using a single path

Taking all the one-hop physical layer mechanisms like FEC for granted, the prime mechanisms to ensure reliability are retransmissions and the usage of multiple packets. Let us start with retransmissions.

Three issues have to be resolved: (i) who detects losses and what are the indicators used; (ii) who requests retransmissions; and (iii) who actually carries out these retransmissions? In single packet delivery the data packet can get lost and the transmitting node has to use timers and retransmissions; the receiver uses acknowledgments.

There is more flexibility in case of block or stream delivery. For example, it is possible to let the receiver detect losses (e.g. by checking for holes in the sequence number space) and request retransmission of missing packets by using *negative acknowledgement* (NACK) packets. If additionally NACKs are understood as carrying *implicit acknowledgements* then there is no necessity to send positive acknowledgements for every packet, thus saving lots of energy.

For single hop delivery, two standard positive acknowledgment approaches are the following:

- MAC-layer retransmissions: when a node on the path forwards the data packet, it expects to receive a MAC-layer acknowledgment. Setting timers is easy, since only single-hop propagation delays and packet processing times have to be considered. Typically, the transmitter makes a bounded number
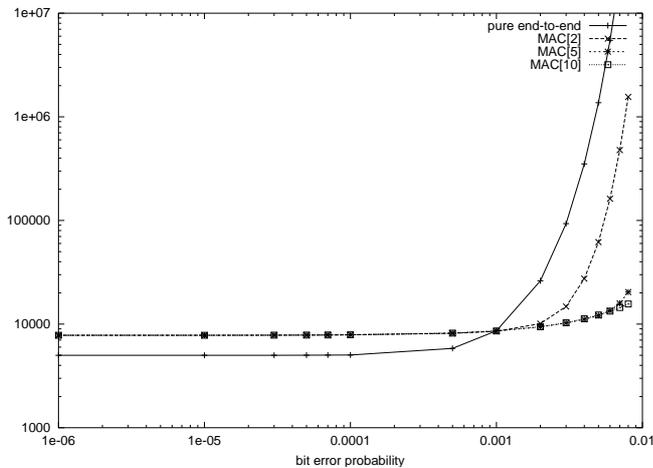
Fig. 1. Comparing expected costs for pure end-to-end acknowledgements vs. end-to-end acknowledgements plus k trials at the MAC layer (denoted as MAC$[k]$) for varying bit error rate p and $n = 10$ hops

of trials to successfully forward the packet and drops it after this number has been exhausted. However, for small data packets the acknowledgements create significant overhead which is invariably expended even on exceptionally good channels.

- End-to-end retransmissions: the source node needs to buffer the packet until an acknowledgement from the sink node arrives. Again, the number of trials made by the source node is typically bounded. However, setting timers in this case is much harder, since reasonable guesses would need knowledge on the number of hops, the per-hop delay and the current cross-traffic. End-to-end retransmissions can be combined with MAC-layer retransmissions. Theoretically, when the number of MAC-layer retransmissions is unbounded, the source node can free its buffer as soon as the packet is delivered successfully to the next hop.

Putting aside problems with setting timers, the most energy-efficient choice between these schemes depends on the error conditions. To illustrate this, we compare in Figure 1 the expected energy costs for the case of pure end-to-end retransmissions with schemes combining end-to-end and MAC-layer retransmissions with k trials. We have chosen a scenario with $n = 10$ hops between source and sink and a simple channel error model, which flips all bits independently at the same probability p (the "bit error rate") [1, Chap. 13]. The energy consumption model is such that transmission and reception of packets incur the same energy costs [16] [1, Chap. 2]. At all other times the nodes are sleeping, consuming no energy. Only data and acknowledgement packets are counted. The following things are noteworthy:

- For very good channels (low bit error rate) it is best to do without MAC-layer acknowledgements.

Beyond a certain point, however, the pure end-to-end scheme is overly wasteful. A similar result holds when the bit error rate is kept fixed (e.g. at 10^{-3}) and the number n of hops is varied: for small numbers of hops it is better to not use MAC-layer acknowledgements, but beyond a threshold they are indispensable. To make optimal usage of energy resources, the actually chosen scheme should be decided dynamically.

- The nodes along the path expend their energy unevenly. Specifically, the source node and likely the first few intermediate nodes have to work upon *every* end-to-end retransmission, even if the preceding trial failed close to the sink node. However, as the number of allowed MAC-layer trials increases (and thus the probability that the packet takes a hop successfully), the energy expenditure among nodes becomes more even.

The HHR approach (Hop-by-Hop Reliability) described in [17] relies on sending multiple copies of the same packet; unicast by each node back to back to the next upstream node. The required number of copies is determined from a locally estimated packet error rate, the desired packet delivery probability and the hop-distance to the sink. Alternatively, packets are repeated until a local acknowledgment has been received (HHRA). In comparison, there is a threshold bit error rate below which HHR is better than HHRA and beyond which the reverse behavior can be observed. Both schemes work well for channels with independent errors, on typical wireless channels with bursty errors (for example caused by multipath fading), however, they are sub-optimal. Specifically, for HHR the multiple copies are simply wasted when all copies are transmitted within the same channel deep fade and during good channel periods, a single or a few packets would likely suffice.

In summary, as soon as the error conditions degrade, MAC-layer retransmissions are an effective means to reduce the energy required to reach a target packet delivery probability. However, when the effort required to reach the target delivery probability shall be kept at minimum level, nodes need extra information like for example their hop-distance to the intended receiver or local error estimates. Obtaining these information incurs extra energy costs.

B. Approaches using multiple paths

A first approach to leverage multiple paths is to set them up in advance (either node-disjoint ones or braided paths) and declare one of them the default route. Once problems occur, another route is used [18] or the route is repaired locally by a rerouting scheme [19]. However, in either case extra route maintenance is required which does not necessarily pay off in single packet delivery applications.

Other approaches send not only a single packet over *one* of the paths but transmit multiple packets

over multiple paths in parallel. In the ReInForM scheme [20], multiple copies of the same packet are transmitted over randomly chosen routes. Specifically, it is assumed that a packet is destined to a sink node and that each node knows its hop distance to the sink as well as the hop distances of all its immediate neighbors. Packet duplication can occur at every intermediate node, not only at the source node. An intermediate node has to decide two things: the number of copies to create and the upstream nodes to which the packet is actually forwarded. With respect to the latter choice, ReInForM prefers nodes which are really closer to the sink but otherwise the choice is random. This distributes the load over many nodes and avoids quick depletion of nodes along a "good" route. The number of duplicates is determined from the locally estimated error rate, the hop distance to the sink and the target delivery probability.

Another possibility using multiple copies extends the HHR and HHRA protocols by exploiting the broadcast property of wireless media [17]. Unlike the point-to-point based HHR, in Hop-by-Hop Broadcast (HHB) a broadcast packet is considered successful if *any* of the sender's neighbors (towards the destination) forwards the packet further. To avoid degenerating into flooding, a neighbor forwards the packet only with probability p; p is chosen (by the sender, using the number of its neighbors) such that on average only a single copy is forwarded.

The HHBA scheme (Hop-by-Hop Broadcast with Acknowledgements) uses the same idea, however, similar to the HHRA scheme there is a gap between the packet copies. Any upstream node receiving the packet sends an acknowledgement (slightly randomized to avoid collisions) and the forwarding node x stops transmission of further copies as soon as such an acknowledgement is received.

In a further scheme [21], the source node adds a number of redundancy bits to the original data. The resulting increased data block is fragmented and each fragment is transmitted on another path. The coding is chosen such that a subset of these fragments is sufficient to reconstruct the original data. Similar *parity packet schemes* have been used in other contexts, too [22].

C. The case of multiple receivers

So far, all schemes try to deliver a single packet to a *single* receiver. The problem of delivering a single packet to a set of receivers is much harder. One particular problem is that positive acknowledgements would lead to the *ack implosion* problem known from multicast protocols [23]. One option is reliable MAC-layer broadcast [24], another one is the WFP approach (Wait for First Packet) taken in the GARUDA project [25] for reliably delivering the first packet of a block of packets from the sink to all sensors. This approach rests on the transceivers ability to generate short pulses of high energy at the transmitting side

and the ability to distinguish these pulses from normal data transmissions based on their energy profile on the receiving side. The sink sends these pulses periodically. The neighboring nodes start generating pulses by themselves once they hear the sinks pulses. Repeating this, after some time the whole network will be pulsing and the sink transmits the data packet. A sinks neighbor receiving the data packet stops pulsing and also transmits the data packet. This process continues and the data packet propagates through the network. Any node (including the sink) having the data packet and still hearing a pulsing neighbor retransmits the packet. This way, continuing pulses functions as an *implicit nack*. The pulsing interval has to be carefully adjusted to avoid too frequent collisions with data packets.

D. Summary

First, there is a general trend that increasing the target delivery probability in a fixed network scenario leads to increased energy costs. Similarly, increasing the hop distance between source and sink node while keeping the target delivery probability fixed also increases the overall energy expenditure. It depends on the chosen scheme how this energy expenditure is distributed over the nodes.

IV. PACKET BLOCK DELIVERY

Block transfers occur when large amounts of data (e.g., code updates) have to be transported. One important feature of such a block transfer is that NACKs can be used. This potentially reduces the number of acknowledgement packets.

A NACK can be regarded as a retransmission request issued by the receiver. When an intermediate node caches the segments, it can serve such a request as well as the original source node could but with the benefit that the NACK and the following retransmitted segment do not need to travel the whole distance between source and sink node. Such a node is also called a *recovery server* [26], [25]. In an extreme case, all nodes in the network spend some buffer for caching.

We discuss three schemes incorporating these ideas.

A. PSFQ: Pump Slowly, Fetch Quickly

The PSFQ protocol presented in [13] is designed to deliver a number of segments from a single source node to a subset of receiver nodes or even to all nodes within a sensor net. It provides guaranteed delivery, e.g., for code updates.

The protocol consists of three basic primitives: a *pump* operation, a *fetch* operation and a *report* operation. By the pumping operation, the sink node transmits all the segments making up the block one

by one, using MAC-layer broadcasts. The time T_{min} between the different segments is comparably large and each segment is equipped with a sequence number. All other nodes behave as follows:

- When a node A receives a new segment not yet seen, it stores it in an internal cache. When the segment has already been received before, the new segment is simply dropped.

- When the new segment is received in-sequence, node A waits for some random time and forwards it further. However, forwarding is suppressed when A finds that four or more of its neighbors have already forwarded the same segment, since the expected additional coverage achieved by A forwarding the segment tends to be small.

- When the packet is received out-of sequence, it is also stored, but instead of forwarding it, the node requests immediate retransmission of the missing segments from any upstream neighbor using a NACK message indicating the missing segment(s). As soon as the node receives the missing segments, it starts forwarding the segments in-sequence in the pumping mode, i.e. with long delays in between.

The decision to forward packets only in-sequence has the advantage that loss events do not propagate: Suppose that node A pumps packets $x_1, x_2, x_3, \ldots, x_n$. Node A's downstream neighbour B has received and forwarded packet x_i and afterwards receives packet x_{i+2}. Node B triggers a fetch operation. If B were to forward x_{i+2} further to some downstream (w.r.t. B) node C, C would trigger a fetch operation, too, which is likely useless and a waste of energy.

The fetch operation corresponds to a NACK or a retransmission request and is triggered by missing sequence number. When segments from the end of the block are missing, there is no higher sequence number and this method of detecting losses fails. To attack this problem, a node A computes from the last seen segment number and the knowledge of T_{min} some estimate on when the missing segments *should* have arrived. If nothing is received by that time, node A proactively triggers the fetch operation by sending the NACK packet. If the upstream neighbors do not possess the missing segments, they forward the NACK further, until it eventually reaches a node having the missing segments.

The NACK packets themselves are broadcasted and *any* upstream neighbor having some of the missing segments is invited to respond. To avoid collisions among them, they introduce random delays before sending their answer.

The report operation is requested from the sink node. The most distant nodes (as indicated by a TTL field in the packet) issue report packets indicating their own address and the received/missing segments. This way the sink can judge the progress of the code block dissemination.

B. GARUDA

The scheme developed in the GARUDA project [25] addresses a similar problem as PSFQ, namely the reliable transfer of block data from a sink to all sensors or a significant part of the network. GARUDA uses a NACK-based scheme and additionally takes great care that the *first* first packet of a block is reliably delivered to all sensors (discussed in Section III-C). This solves the problem of NACK-based schemes that a receiver needs to receive at least one packet from the block to detect losses of further packets at all.

GARUDA constructs an approximation to the minimum dominating set of the sensor network topology and the members of this set (called *core members*) act as recovery servers for downstream core members and neighboring non-core members. Only those nodes are candidates for the core that have a hop distance to the sink being an integer multiple of three. A candidate core member refrains from becoming a core member when there are enough core members in its neighborhood. On the other hand, non-core members having no core member in their range can request a candidate core member to really become a core member. All core members know at least one upstream (i.e. closer to the sink) core member from which they request retransmissions.

The reliable delivery of a block of data from the sink proceeds in two steps: first within the core, afterwards the non-core nodes fetch missing data from their associated core members. GARUDA is based on out-of-order delivery. A core member x requests missing segments from its upstream core member y, but does this only when x knows that the missing segment is indeed available at y. To achieve this, node y includes into every forwarded packet a bitmap indicating the segments that y already has, and x can use this knowledge to suppress NACKs for packets missing at y. A non-core member a associated to x suppresses all retransmission requests until x has all the segments present, indicated by a full bitmap.

C. RMST: Reliable Multi-Segment Transport

The RMST scheme [7] adds reliable data transfer to directed diffusion [27].

RMST is designed for delivering larger blocks of data in multiple segments from a source node to a sink node. This is for example required when time series data has to be transmitted. RMST combines several mechanisms to enforce reliability:

- MAC-layer retransmissions.
- In RMST's *cached mode* the sink node and all intermediate nodes on an enforced path cache segments and check the cache periodically for missing segments. When a node detects missing segments, it generates a NACK message which travels back to the source along the reinforced path. The first node

A having missing segments in its cache forwards them again towards the sink (and thus towards the requesting node). If *A* can retrieve all requested segments from its cache, then *A* drops the NACK packet, otherwise it is forwarded further upstream. Both the segments and the NACK packets are represented in terms of attributes, to be compatible with directed diffusion. In the *noncached mode* of RMST only the sink node has such a cache but not the intermediate nodes; therefore, NACK's travel back to the source node (which clearly also needs to cache the segments).

- On the application layer redundancy is used: the source sends out the whole data block periodically until the sink explicitly unsubscribes.

- By frequently repeating interest propagation, dissemination of exploratory events and subsequent establishment of (new) reinforced routes some resilience against node failures is achieved.

By investigating different combinations of the above mechanisms for their total number of bytes (data plus overhead) needed to transmit 50 segments of 100 bytes size, it showed up that MAC-layer retransmissions are helpful in case of higher packet loss rates, but interestingly using the cached mode *without* MAC-layer retransmissions (and thus without MAC layer overhead like acknowledgements or RTS/CTS handshakes) is the cheapest approach (given that *all* intermediate nodes cache segments).

D. Summary

The discussed schemes reveal that for the case of block transfer the usage of NACK packets and of caching data within the network is beneficial in terms of energy. RMST has shown that the utility of additional MAC layer delays is disputable, at least when *all* intermediate nodes in the network have sufficient memory to cache all segments. We suspect that MAC layer acknowledgements can bring real benefits when only a fraction of intermediate nodes can cache segments and NACK packets and retransmissions have to travel longer ways.

V. PACKET STREAM DELIVERY

Some sensor network applications require the sensor nodes to generate and report their data periodically. One important reliability target in such a setup is to ensure that the sink receives a sufficient number of packets per unit time, for example to achieve a desired information accuracy. Since many environmental processes vary only slowly, repeated sensor readings of the same or neighbored sensors are often correlated and accordingly some lost packets are acceptable. Therefore, the key mechanism to ensure delivery of the desired number of packets at the sink node are not retransmissions, but instead to control either the

packet generation rate of the sensor nodes or alternatively the number of nodes generating packets at a fixed rate. We discuss two mechanisms in this section, one for each of these "control knobs".

Controlling the packet transmission rate is intimately related to congestion control issues [28], [29], [30]. The adverse effects of congestion on the overall energy consumption and information accuracy have been shown in [28]: dropping packets is a waste of energy and can lead to throughput reduction, i.e. to a reduction of the number of packets delivered at sink nodes. In [29] the CODA congestion control framework is presented which is composed out of: (i) a localized congestion detection mechanism based on nodes observing their buffer occupancy and the load on the channel, (ii) a local back-pressure mechanism for short-term congestion handling, and (iii) a rate-regulation mechanism by which the sinks control the source nodes on a longer-term basis. A method based on explicit bandwidth allocation to different packet classes is discussed in [30].

A. ESRT: Event to Sink Reliable Transport

The ESRT protocol works by adjusting the reporting sensors packet generation rate such that it stays in a region where sufficient numbers of packets arrive at the sink without producing congestion [31]. It is assumed that the sink requires this minimum number of packets to achieve a desired information quality.

The situation considered by the algorithm is that of a single sink node to which all sensor nodes direct their readings. The sink node is not energy-constrained and can transmit with sufficient power to reach all the sensors. It uses this ability to *control* the rate f_n by which sensors generate data packets in the n-th round of the algorithm. The control strategy is based on a certain relationship between the generation rate f_n on the one hand and the observed sink quality (given as the rate of delivered packets per unit time) and congestion state on the other hand. Specifically, the following regimes can be distinguished:

- For very low generation rates there is no congestion and insufficient quality.
- When increasing the generation rates, the desired quality is reached to within some fraction ϵ without causing congestion. Stated differently: the network is uncongested and the sink receives just the right number of packets to achieve the desired quality, not much more or less. This is the *target region*.
- When increasing the generation rates further more packets than needed are delivered without causing congestion.
- Another increase in generation rate starts to decrease the number of delivered packets, because congestion starts to build up and packets are dropped. There is one region with congestion but still delivering sufficient quality, but the quality drops below the required level when the rates are increased further.

The sink node collects congestion signs and observes the rate of incoming packets for a certain time, called a *round*. Based on this information it determines the current regime, computes a new desired generation rate for the next round (which can be larger, smaller or equal to the current generation rate) and broadcasts this to all sensor nodes. The control strategy strives to reach the target region. The congestion state is detected by sensors from their local buffer occupancy, taking the current occupancy and the growth trend of buffer occupancy with respect to previous rounds into account. Upon congestion detection the sensor node sets a congestion notification bit in outgoing packets. The sink infers a congestion state when any incoming packet has this bit set.

Under the assumption that in the non-congested regime there is a linear relationship between the reporting rate and the number of packets received at the sink per unit time, it can be shown that the protocol always converges to the target region, with the convergence speed depending on ϵ. The protocol does not require the sink or the sensor nodes to have global knowledge like for example the current number of available sensor nodes. A disadvantage is that *all* sensor nodes are controlled at once, treating interesting regions (where faster rates are appropriate) or regions with higher node density in the same way as uninteresting regions or regions with low node density.

B. A Gur game algorithm

Instead of controlling the rate by which nodes generate packets, the round-based algorithm [32] controls the number of active nodes in a probabilistic way. Again, it is assumed that there is a single sink node controlling a number of sensors and the sink requires some minimum number k^* of packets during a round to achieve a desired information quality. This is easy to achieve using a broadcast channel, knowledge about the number of sensors N, and a probabilistic decision rule in each node (send with probability k^*/N). But how to do this without knowing N?

One interesting way is based on the Gur game. In short, the Gur game assumes N independent players voting "yes" or "no" in a round. A referee counts the k "yes" votes, determines a reward probability $r = r(k)$, and broadcasts r. Each player rewards itself with probability r. Voting and $r(k)$ have to be chosen to obtain k^* answers. The idea is to have nodes drift towards a settled opinion on voting yes or no based on r – with probability r, a node becomes more fixed on its voting behavior, with $1 - r$ it will lean more towards switching. Using a specific reward function [32] depending on k^*, the system will converge to k^* yes and $N - k^*$ no votes (or, in the case of WSNs, does not answer at all).

VI. Conclusions

Reliability in sensor networks is multi-faceted, and reliable data transport, a very important and interesting issue and the focus of this paper, is one aspect. The reliable data transport problem itself has also many faces, ranging from single packet delivery to delivery of periodic streams. Many of the protocols discussed in this paper attack the reliability problem by combining mechanisms on several layers, from the MAC layer up to the application layer. This is different from the Internet-way of thinking about reliability, but necessary when energy is at premium. It is also fair to say that reliability issues have so far attracted less attention in the research community than, say, MAC or routing protocols. There is accordingly plenty of room for interesting research, like for example: (i) design and evaluation of further mechanisms for improving reliability, taking the complex behavior of wireless channels into account; (ii) adaptively controlling the actual mix of mechanisms (FEC vs. MAC-layer ARQ vs. NACKs vs. . . .) depending on locally estimated error rates or on global knowledge of current error situations ("network weather"); (iii) experimental studies regarding reliability and energy-efficiency in real sensor networks (like for example [33]); and (iv) consideration of timing-aspects [34]. It will be interesting to follow the development of end-to-end reliability solutions in the realm of sensor networks.

References

[1] H. Karl and A. Willig, *Architectures and Protocols for Wireless Sensor Networks*. Chichester: John Wiley & Sons, 2005, in preparation.

[2] B. Liu and D. Towsley, "A study on the coverage of large-scale sensor networks," in *Proc. 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'04)*, 2004.

[3] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak, "Exposure in wireless ad-hoc sensor networks," in *Proc. 7th Annual International Conference on Mobile Computing and Networking (MobiCom '01)*, July 2001, pp. 139–150.

[4] X.-Y. Li, P.-J. Wan, and O. Frieder, "Coverage in wireless ad hoc sensor networks," *IEEE Transactions on Computers*, vol. 52, no. 6, June 2003.

[5] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," *MONET - Mobile Networks and Applications*, 2004, to appear.

[6] H. Koskinen, "On the coverage of a random sensor network in a bounded domain," in *Proc. 16th ITC Specialist Seminar*, 2004, pp. 11–18.

[7] 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.

[8] A. Dunkels, T. Voigt, J. Alonso, and H. Ritter, "Distributed tcp caching for wireless sensor networks," in *Proc. Third Mediterranean ad Hoc Networking Conference (MedHocNet)*, June 2004.

[9] A. Dunkels, "Full tcp/ip for 8-bit architectures," in *Proc. First international conference on mobile applications, systems and services (MOBISYS 2003)*, San Francisco, CA, May 2003.

[10] A. Ramakrishnan, "16-bit embedded web server," in *Proc. ISA/IEEE Sensors for Industry Conference*, 2004, pp. 187–193.

[11] T. Lin, H. Zhao, J. Wang, G. Han, and J. Wang, "An embedded web server for equipments," in *Proc. 7th International Symposium on Parallel Architectures, Algorithms and Networks*, May 2004, pp. 345–350.

[12] J. Riihijärvi, P. Mähönen, M. J. Saaranen, J. Roivainen, and J.-P. Soininen, "Providing network connectivity for small appliances: a functionally minimized embedded web server," *IEEE Communications Magazine*, vol. 39, no. 10, pp. 74–79, Oct. 2001.

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

[14] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, "Distributed Target Classification and Tracking in Sensor Networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1163–1171, Aug. 2003.

[15] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration," *IEEE Signal Processing Magazine*, pp. 61–72, Mar. 2002.

[16] J. L. Hill and D. E. Culler, "MICA: A wireless platform for deeply embedded computing," *IEEE Micro*, vol. 22, no. 6, pp. 12–24, Dec. 2002.

[17] B. Deb, S. Bhatnagar, and B. Nath, "Information assurance in sensor networks," in *Proc. 2nd ACM Intl. Workshop on Wireless Sensor Networks and Applications (WSNA)*, San Diego, CA, Sept. 2003.

[18] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks," *Mobile Computing and Communications Review (MC2R)*, vol. 1, no. 2, 2002.

[19] D. Tian and N. D. Georganas, "Energy efficient routing with guaranteed delivery in wireless sensor networks," in *Proc. IEEE Wireless Communications and Networking Conference 2003 (WCNC'03)*, Institute of Electrical and Electronics Engineers. New Orleans, USA: IEEE Press, Mar. 2003.

[20] B. Deb, S. Bhatnagar, and B. Nath, "Reinform: Reliable information forwarding using multiple paths in sensor networks," in *Proc. 28th Annual IEEE Conference on Local Computer Networks (LCN 2003)*, Bonn, Germany, Oct. 2003, to appear.

[21] S. Dulmann, T. Nieberg, J. Wu, and P. Havinga, "Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, Mar. 2003.

[22] G. Carle and E. W. Biersack, "Survey of error recovery techniques for ip-based audio-visual multicast applications," *IEEE Network Magazine*, vol. 11, no. 6, pp. 24–36, Nov. 1997.

[23] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 784–803, 1997.

[24] K. Tang and M. Gerla, "MAC Reliable Broadcast in Ad Hoc Networks," in *Proc. IEEE Military Communications Conference, 2001 (MILCOM 2001)*, Oct. 2001, pp. 1008–1013.

[25] 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.

[26] S.-J. Park and R. Sivakumar, "Poster: Sink-to-sensors reliability in sensor networks," in *Proc. 4th ACM Intl. Symp. on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, Annapolis, MD, June 2003.

[27] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, Feb. 2003.

[28] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "Infrastructure tradeoffs for sensor networks," in *Proc. 1st ACM Intl. Workshop on Sensor Networks and Applications (WSNA)*, Atlanta, GA, Sept. 2002.

[29] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "Coda: Congestion detection and avoidance in sensor networks," in *Proc. First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, Los Angeles, CA, Nov. 2003, pp. 266–279.

[30] B. Hull, K. Jamieson, and H. Balakrishnan, "Poster abstract: Bandwidth management in wireless sensor networks," in *Proc. 1st Intl. Conf. on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, CA, Nov. 2003, pp. 306–307.

[31] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz, "Esrt: Event-to-sink reliable transport in wireless sensor networks," in *Proc. ACM MOBIHOC 2003*, Association of Computing Machinery.   Annapolis, Maryland: ACM Press, June 2003.

[32] R. Iyer and L. Kleinrock, "Qos control for sensor networks," in *Proc. ICC'03*, Anchorage, Alaska, May 2003, pp. 517–521.

[33] J. M. Reason and J. M. Rabaey, "A study of energy consumption and reliability in a multi-hop sensor network," *ACM Mobile Computing and Communications Review*, vol. 8, no. 1, pp. 84–97, Jan. 2004.

[34] J. A. Stankovic, T. F. Abdelzaher, C. Lu, L. Sha, and J. C. Hou, "Real-Time Communication and Coordination in Embedded Sensor Networks," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 1002–1022, July 2003.