Data Transport Control in Wireless Sensor Networks

Hongwei Zhang

Department of Computer Science
Wayne State University
Detroit, Michigan 48202, USA
hzhang@cs.wayne.edu
http://www.cs.wayne.edu/~hzhang

Vinayak Naik

Center for Embedded Networked Sensing
University of California – Los Angeles
Los Angeles, California 90095, USA
naik@cens.ucla.edu
http://www.lecs.cs.ucla.edu/~naik

Abstract

Dynamics of wireless communication, resource constraints, and application diversity pose significant challenges to data transport control in wireless sensor networks. In this chapter, we examine the issue of data transport control in the context of two typical communication patterns in wireless sensor networks: convergecast and broadcast. We study the similarity and differences of data transport control in convergecast and broadcast, we discuss existing convergecast and broadcast protocols, and we present open issues for data transport control in wireless sensor networks.

1 Introduction

Wireless sensor networks are increasingly innovating the way we interact with the physical world, and they tend to have a broad range of applications in science (e.g., ecology), engineering (e.g., industrial control), and our daily life (e.g., healthcare). Spatially distributed, sensor nodes coordinate with one another

through messaging passing. Two typical messaging passing tasks in wireless sensor networks are convergecast and broadcast. Convergecast enables a sink node to collect information (e.g., event detection) from multiple spatially distributed nodes, and broadcast enables a node to disseminate data (e.g., a new sensor node program) from itself to all the other nodes in the network.

Even though message passing has been studied extensively in traditional networks such as the Internet and wireless networks, wireless sensor networks bring unique challenges to the design of message passing services due to the complex properties of wireless communication, resource constrains, and application diversity. Among other tasks, data transport control is an important, challenging task in sensor networks. Moreover, data transport control differs in different message passing tasks. For instance, fairness is an important issue in convergecast but not in broadcast; broadcast tends to require 100% reliability in most cases (e.g., in sensor network reprogramming), but reliability requirements may vary significantly in different convergecast scenarios; the source node of broadcast is a single node which may serve as a single-point-of-control in broadcast, yet the sources nodes in convergecast are usually spatially distributed.

In this chapter, we examine in detail the data transport control issues in convergecast and broadcast in Sections 2 and 3 respectively. The discussion of broadcast is from the perspective of sensor network reprogramming since it is one of the most commonly uses broadcast services in sensor networks. We make concluding remarks in Section 4.

2 Data transport control in convergecast

In this section, we first review the basic issues and approaches in data transport control for convergecast, and then give a detailed treatment of the protocol Reliable Bursty Convergecast (RBC) [Zhang et al. 2007].

2.1 Introduction

In convergecast, multiple source nodes need to report data to a sink node, creating the funneling effect where the traffic load increases as the distance to the sink node decreases. One consequence of the funneling effect is network congestion where packet queues overflow because packets arrive at nodes faster than what the nodes can transmit. The funneling effect also increases channel contention and thus the probability of packet loss as a result of increased collision probability. To ensure reliable data transport in convergecast, therefore, two basic issues are congestion control and error control. Besides reliable data transport, another issue is to ensure fairness in delivering data from different source nodes. Fairness in data

delivery is important because, otherwise, the sink node cannot detect or observe the phenomena happening in regions whose sensing packets experience significant loss. The research community has proposed different approaches to address the congestion control, error control, and fairness control issues in convergecast, and we discuss a few representative mechanisms in the next section.

2.2 Background

For congestion control, Wan et al. proposed the protocol CODA (for *Congestion Detection and Avoidance*) [Wan et al. 2003]. In CODA, a node monitors both its queue length and the channel load condition (e.g., the number of packets transmitted in a short interval) to detect any potential network congestion in its local neighborhood. A node declares the network as being congested when the queue length and/or channel load condition exceed certain threshold values. Once a node detects network congestion, it can use two complementary approaches to ameliorate congestion: open-loop, hop-by-hop congestion control, and closed-loop, end-to-end congestion control. In open-loop, hop-by-hop congestion control, a node having detected congestion will inform the corresponding transmitting nodes of the congestion; these transmitting nodes will reduce their transmitting rates accordingly and then propagate this "congestion" information backward along the direction toward the traffic sources, creating the diffusing "backpressure" so that the sources will eventually reduce their traffic generation rates too. In open-loop, end-to-end congestion control, the sink coordinates with the sources to regulate the traffic generation rates at different sources.

In CODA, Wan et al. did not differentiate between the congestion within a node and that in wireless transmission. To address this issue, Ee and Bajcsy [Ee and Bajcsy 2004] proposed a system where congestion within a node and congestion in wireless communication are treated separated. In [Ee and Bajcsy 2004], congestion within a node are detected by monitoring its queue length, and the congestion is dealt with by an open-loop, hop-by-hop control mechanism similar to that in CODA; congestion in wireless communication (also commonly referred to as *channel contention*) is addressed by letting nodes randomly backoff at the timescale of application transmission interval rather than at the timescale of radio transmission rate. Besides congestion control, Ee and Bajcsy also proposed a rate-based mechanism to ensure fairness in packet delivery. Corroborating several observations in [Wan et al. 2003] and [Ee and Bajcsy 2004], Hull et al. [Hull et al. 2004] studied the effectiveness of different congestion and fairness control mechanisms, and they found out that 1) hop-by-hop flow control is effective for all types of workloads and utilization levels, and 2) rate limiting is particularly effective in achieving fairness.

Focusing on the reliability of delivering information related to an event, ESRT [Sankarasubramanjam et al. 2003] controls congestion based on the relationship

between event reliability and source report frequency. More specifically, the sink node continuously measures the event reliability, and decides on the source report frequency accordingly; the sources will generate reports based on the frequency-feedback from the sink node to avoid congestion in the network.

For reliable packet delivery in sensor networks, Stann and Heidermann [Stann and Heidermann 2003] studied the benefit of hop-by-hop error control and recovery compared with end-to-end error control. For instance, Figure X.1 shows the number of transmissions required to send 10 packets across 10 hops in hop-by-hop and end-to-end error control respectively. We see that hop-by-hop error control significantly reduces the number of transmissions required for reliable data delivery.

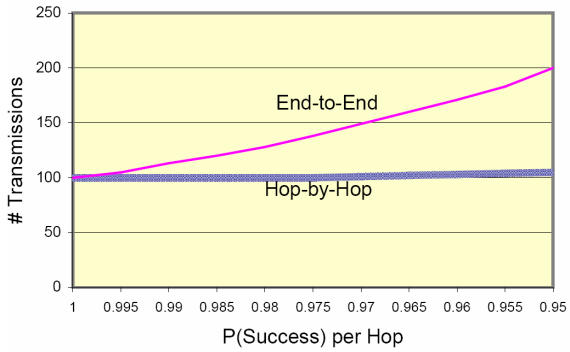

Figure X.1: Number of transmissions required to send 10 packets across 10 hops.

For reliable, real-time packet delivery in bursty convergecast where a huge burst of data need to be delivered from multiple source nodes to a sink node, Zhang et al. [Zhang et al. 2007] proposed the protocol *Reliable Bursty Convergecast* (RBC). RBC addresses the challenge of reliable, real-time error control in the presence of high channel contention and collision. To improve channel utilization and to reduce ack-loss, RBC uses a window-less block acknowledgment scheme that guarantees continuous packet forwarding and replicates the acknowledgment for a packet; to alleviate retransmission-incurred channel contention, RBC employs differentiated contention control. Moreover, RBC uses mechanisms to handle varying ack-delay and to reduce delay in timer-based retransmissions. We elaborate on the protocol RBC in the next section.

2.3 RBC: Reliable Bursty Convergecast

A typical application of wireless sensor networks is to monitor an environment (be it an agricultural field or a classified area) for events that are of interest to the users. Usually, the events are rare. Yet when an event occurs, a large burst of pack-

ets is often generated that needs to be transported reliably and in real-time to a base station.

One exemplary event-driven application is demonstrated in the DARPA NEST field experiment ``A Line in the Sand'' (simply called *Lites* hereafter) [Arora et al. 2004]. In Lites, a typical event generates up to 100 packets within a few seconds and the packets need to be transported from different network locations to a base station, over multi-hop routes.

The high-volume bursty traffic in event-driven applications poses special challenges for reliable and real-time packet delivery. The large number of packets generated within a short period leads to high degree of channel contention and thus a high probability of packet collision. The situation is further exacerbated by the fact that packets travel over multi-hop routes: Firstly, the total number of packets competing for channel access is increased by a factor of the average hop-count of network routes; Secondly, the probability of packet collision increases in multi-hop networks due to problems such as hidden-terminals. Consequently, packets are lost with high probability in bursty convergecast. For example, with the default radio stack of TinyOS [TinyOS], around 50% of packets are lost for most events in Lites.

For real-time packet delivery, hop-by-hop packet recovery is usually preferred over end-to-end recovery [Stann and Heidermann 2003]; and this is especially the case when 100% packet delivery is not required (for instance, for bursty convergecast in sensor networks). Nevertheless, existing hop-by-hop control mechanisms do not work well in bursty convergecast. Via experiments with a testbed of 49 MICA2 motes and with traffic traces of Lites, Zhang et al. [Zhang et al. 2007] observed that the commonly used link-layer error control mechanisms do not significantly improve and can even degenerate packet delivery reliability. For example, when packets are retransmitted up to twice at each hop, the overall packet delivery ratio increases by only 6.15%; and when the number of retransmissions increases, the packet delivery ratio actually decreases, by 11.33%.

One issue with existing hop-by-hop control mechanisms is that they do not schedule packet retransmissions appropriately; as a result, retransmitted packets further increase the channel contention and cause more packet loss. Moreover, due to in-order packet delivery and conservative retransmission timers, packet delivery can be significantly delayed in existing hop-by-hop mechanisms, which leads to packet backlogging and reduction in network throughput.

On the other hand, the new network and application models of bursty convergecast in sensor networks offer unique opportunities for reliable and real-time transport control:

- Firstly, the broadcast nature of wireless channels enables a node to determine, by snooping the channel, whether its packets are received and forwarded by its neighbors.

- Secondly, time synchronization and the fact that data packets are timestamped relieve transport layer from the constraint of in-order packet delivery, since applications can determine the order of packets by their timestamps.

Therefore, techniques that take advantage of these opportunities and meet the challenges of reliable and real-time bursty convergecast are desired.

Zhang et al. [Zhang et al. 2007] studied the limitations of two commonly used hop-by-hop packet recovery schemes in bursty convergecast. They discovered that the lack of retransmission scheduling in both schemes makes retransmission-based packet recovery ineffective in the case of bursty convergecast. Moreover, in-order packet delivery makes the communication channel under-utilized in the presence of packet- and ack-loss. To address the challenges, they designed protocol RBC (for *Reliable Bursty Convergecast*). Taking advantage of the unique sensor network models, RBC features the following mechanisms:

- To improve channel utilization, RBC uses a window-less block acknowledgment scheme that enables continuous packet forwarding in the presence of packet- and ack-loss. The block acknowledgment also reduces the probability of ack-loss, by replicating the acknowledgment for a received packet.
- To ameliorate retransmission-incurred channel contention, RBC introduces differentiated contention control, which ranks nodes by their queuing conditions as well as the number of times that the enqueued packets have been transmitted. A node ranked the highest within its neighborhood accesses the channel first.

In the rest of this section, we examine in more detail the shortcomings of the existing error control mechanisms and how RBC addressed these shortcomings.

2.3.1 Performance with existing error control mechanisms

Two widely used hop-by-hop packet recovery mechanisms in sensor networks are synchronous explicit ack and stop-and-wait implicit ack. Zhang et al. [Zhang et al. 2007] studied their performance in bursty convergecast, and we discuss their findings as follows.

2.3.1.1 Synchronous explicit ack (SEA)

In SEA, a receiver switches to transmit-mode and sends back the acknowledgment immediately after receiving a packet; the sender immediately retransmits a packet if the corresponding ack is not received after certain constant time. Zhang et al. studied the performance of SEA when used with B-MAC [Polastre et al. 2004] and S-MAC [Ye et al. 2002]. B-MAC uses the mechanism of CSMA/CA (carrier

sense multiple access with collision avoidance) to control channel access; S-MAC uses CSMA/CA too, but it also employs RTS-CTS handshake to reduce the impact of hidden terminals.

SEA with B-MAC. The event reliability, the average packet delivery delay, as well as the event goodput is shown in Table X.1, where RT stands for the maximum number of retransmissions for each packet at each hop (e.g., RT = 0 means that packets are not retransmitted), ER stands for Event Reliability; PD stands for packet delivery latency; EG stands for event goodput [Zhang et al. 2007].

Metrics	RT = 0	RT = 1	RT = 2
ER (%)	51.05	54.74	54.63
PD (seconds)	0.21	0.25	0.26
EG (packets/sec)	4.01	4.05	3.63

Table X.1: SEA with B-MAC in Lites trace.

Table X.1 shows that when packets are retransmitted, the event reliability increases slightly (i.e., by up to 3.69%). Nevertheless, the maximum reliability is still only 54.74%, and, even worse, the event reliability as well as goodput decreases when the maximum number of retransmissions increases from 1 to 2.

SEA with S-MAC. Unlike B-MAC, S-MAC uses RTS-CTS handshake for unicast transmissions, which reduces packet collisions. The performance data for S-MAC is shown in Table X.2.

Metrics	RT = 0	RT = 1	RT = 2
ER (%)	72.6	74.79	70.1
PD (seconds)	0.17	0.183	0.182
EG (packets/sec)	5.01	4.68	4.37

Table X.2: SEA with S-MAC in Lites trace

Compared with B-MAC, RTS-CTS handshake improves the event reliability by about 20% in S-MAC. Yet packet retransmissions still do not significantly improve the event reliability and can even decrease the reliability.

Analysis. We can see that the reason why retransmission does not significantly improve --- and can even degenerate --- communication reliability is that, in SEA, lost packets are retransmitted while new packets are generated and forwarded, thus retransmissions, when not scheduled appropriately, only increase channel contention and cause more packet collision.[1] The situation is further exacerbated by ack-loss (with a probability as high as 10.29%), since ack-loss causes unnecessary retransmission of packets that have been received. To make retransmission effective in improving reliability, therefore, we need a retransmission scheduling mechanism that ameliorates retransmission-incurred channel contention.

[1] This is not the case in wireline networks and is due to the nature of wireless communications.

2.3.1.2 Stop-and-wait implicit ack (SWIA)

SWIA takes advantage of the fact that every node, except for the base station, forwards the packet it receives and the forwarded packet can act as the acknowledgment to the sender at the previous hop [Maroti 2004]. In SWIA, the sender of a packet snoops the channel to check whether the packet is forwarded within certain constant threshold time; the sender regards the packet as received if it is forwarded within the threshold time, otherwise the packet is regarded as lost. The advantage of SWIA is that acknowledgment comes for free except for the limited control information piggybacked in data packets. The performance results for SWIA is shown in Table X.3.

Metrics	RT = 0	RT = 1	RT = 2
ER (%)	43.09	31.76	46.5
PD (seconds)	0.35	8.81	18.77
EG (packets/sec)	3.48	2.58	1.41

Table X.3: SWIA with B-MAC in Lites trace

We can see that the maximum event reliability in SWIA is only 46.5%, and that the reliability decreases significantly when packets are retransmitted at most once at each hop. When packets are retransmitted up to twice at each hop, the packet delivery delay increases, and the event goodput decreases significantly despite the slightly increased reliability.

Analysis. The above phenomena are due to the following reasons. First, the length of data packets is increased by the piggybacked control information in SWIA, thus the ack-loss probability increases (as high as 18.39% in our experiments), which in turn increases unnecessary retransmissions. Second, most packets are queued upon reception and thus their forwarding is delayed. As a result, the piggybacked acknowledgments are delayed and the corresponding packets are retransmitted unnecessarily. Third, once a packet is waiting to be acknowledged, all the packets arriving later cannot be forwarded even if the communication channel is free. Therefore, channel utilization as well as system throughput decreases, and network queuing as well as packet delivery delay increases. Fourth, as in SEA, lack of retransmission scheduling allows retransmissions, be it necessary or unnecessary, to cause more channel contention and packet loss.

To address the limitations of SEA and SWIA in bursty convergecast, Zhang et al. [Zhang et al. 2007] designed protocol RBC. In RBC, a window-less block acknowledgment scheme is designed to increase channel utilization and to reduce the probability of ack-loss; a distributed contention control scheme is also designed to schedule packet retransmissions and to reduce the contention between newly generated and retransmitted packets. Given that the number of packets competing for channel access is less in implicit-ack based schemes than in ex-

plicit-ack based schemes, Zhang et al. designed RBC based on the paradigm of implicit-ack (i.e., piggybacking control information in data packets).

2.3.2 Windowless acknowledgment

In traditional block acknowledgment [Brown et al. 1989], a sliding-window is used for both duplicate detection and in-order packet delivery.[2] The sliding-window reduces network throughput once a packet is sent but remains unacknowledged (since the sender can only send up to its window size once a packet is unacknowledged), and in-order delivery increases packet delivery delay once a packet is lost (since the lost packet delays the delivery of every packet behind it). Therefore, the sliding-window based block acknowledgment scheme does not apply to bursty convergecast, given the real-time requirement of the latter.

To address the constraints of traditional block acknowledgment in the presence of unreliable links, RBC takes advantage of the fact that in-order delivery is not required in bursty convergecast. Without considering the order of packet delivery, we only need to detect whether a sequence of packets is received without loss in the middle and whether a received packet is a duplicate of a previously received one. To this end, a *window-less block acknowledgment* scheme is designed to ensure continuous packet forwarding irrespective of the underlying link unreliability as well as the resulting packet- and ack-loss. For clarity of presentation, we consider an arbitrary pair of nodes S and R where S is the sender and R is the receiver.

Window-less queue management. The sender S organizes its packet queue as (M+2) linked lists, as shown in Figure X.2, where M is the maximum number of retransmissions at each hop. For convenience, we call the linked lists *virtual queues*, denoted as Q_0, …, Q_{M+1}. The virtual queues are ranked such that a virtual queue Q_k ranks higher than Q_j if $k < j$.

Virtual queues Q_0, Q_1, …, and Q_M buffer packets waiting to be sent or to be acknowledged, and Q_{M+1} collects the list of free queue buffers. The virtual queues are maintained as follows:

- When a new packet arrives at S to be sent, S detaches the head buffer of Q_{M+1}, if any, stores the packet into the queue buffer, and attaches the queue buffer to the tail of Q_0.
- Packets stored in a virtual queue Q_k ($k > 0$) will not be sent unless Q_{k-1} is empty; packets in the same virtual queue are sent in FIFO order.

[2] Note that SWIA is a special type of block acknowledgment where the window size is 1.

- After a packet in a virtual queue Q_k ($k \geq 0$) is sent, the corresponding queue buffer is moved to the tail of Q_{k+1}, unless the packet has been retransmitted M times in which case the queue buffer is moved to the tail of Q_{M+1}.
- When a packet is acknowledged to have been received, the buffer holding the packet is released and moved to the tail of Q_{M+1}.

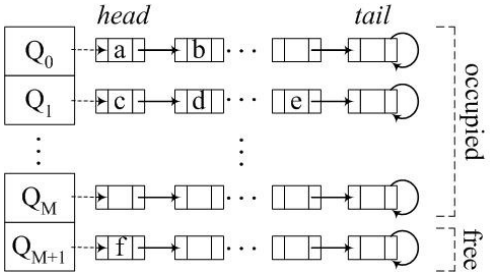

Figure X.2: Virtual queues at a node

The above rules help identify the relative freshness of packets at a node (which is used in the differentiated contention control in Section 2.3.3); they also help maintain without using sliding windows the order in which unacknowledged packets have been sent, providing the basis for window-less block acknowledgment. Moreover, newly arrived packets can be sent immediately without waiting for the previously sent packets to be acknowledged, which enables continuous packet forwarding in the presence of packet- and ack-loss.

Block acknowledgment & reduced ack-loss. Each queue buffer at S has an ID that is unique at S. When S sends a packet to the receiver R, S attaches the ID of the buffer holding the packet as well as the ID of the buffer holding the packet to be sent next. In Figure X.2, for example, when S sends the packet in buffer a, S attaches the values a and b. Given the queue maintenance procedure, if the buffer holding the packet being sent is the tail of Q_0 or the head of a virtual queue other than Q_0, S also attaches the ID of the head buffer of Q_{M+1}, if any, since one or more new packets may arrive before the next enqueued packet is sent in which case the newly arrived packet(s) will be sent first. For example, when the packet in buffer c of Figure X.2 is sent, S attaches the values c, d, and f.

When the receiver R receives a packet p_0 from S, R learns the ID n' of the buffer holding the next packet to be sent by S. When R receives a packet p_n from S next time, R checks whether p_n is from buffer n' at S: if p_n is from buffer n', R knows that there is no packet loss between receiving p_0 and p_n from S; otherwise, R detects that some packets are lost between p_0 and p_n.

For each maximal sequence of packets p_k, …, $p_{k'}$ from S that are received at R without any loss in the middle, R attaches to packet $p_{k'}$ the 2-tuple $<q_k, q_{k'}>$, where q_k and $q_{k'}$ are the IDs of the buffers storing p_k and $p_{k'}$ at S. We call $<q_k, q_{k'}>$ the *block acknowledgment* for packets p_k, …, $p_{k'}$. When S snoops the forwarded

packet $p_{k'}$ later, S learns that all the packets sent between p_k and $p_{k'}$ have been received by R. Then S releases the buffers holding these packets. For example, if S snoops a block acknowledgment <c, e> when its queue state is as shown in Figure X.2, S knows that all the packets in buffers between c and e in Q_1 have been received, and S releases buffers between c and e, including c and e.

One delicate detail in processing the block acknowledgment $<q_k, q_{k'}>$ is that after releasing buffer q_k, S will maintain a mapping $q_k \leftrightarrow q_{k''}$, where $q_{k''}$ is the buffer holding the packet sent (or to be sent next) after that in $q_{k'}$. When S snoops another block acknowledgment $<q_k, q_n>$ later, S knows, by $q_k \leftrightarrow q_{k''}$, that packets sent between those in buffers $q_{k''}$ and q_n have been received by R; then S releases the buffers holding these packets, and S resets the mapping to $q_k \leftrightarrow q_{n''}$, where $q_{n''}$ is the buffer holding the packet sent (or to be sent next) after that in q_n. S maintains the mapping for q_k until S receives a block-NACK [n', n) or a block acknowledgment $<q, q'>$ where $q \neq q_k$, in which case S maintains the mapping for n or q respectively. Via the buffer pointer mapped as above, the node S can process the incoming block acknowledgments and block-NACKs. For convenience, we call the buffer being mapped to the *anchor* of block acknowledgments. In the examples discussed above, buffers $q_{k''}$ and $q_{n''}$ have been anchors once. We also call the packet in an anchor buffer an *anchor packet*.

In the above block acknowledgment scheme, the acknowledgment for a received packet is piggybacked onto the packet itself as well as the packets that are received consecutively after the packet without any loss in the middle. Therefore, the acknowledgment is replicated and the probability for it to be lost decreases significantly.

Duplicate detection & obsolete-ack filtering. Since it is impossible to completely prevent ack-loss in lossy communication channels, packets whose acknowledgments are lost will be retransmitted unnecessarily. Therefore, it is necessary that duplicate packets be detected and dropped.

To enable duplicate detection, the sender S maintains a counter for each queue buffer, whose value is incremented by one each time a new packet is stored in the buffer. When S sends a packet, it attaches the current value of the corresponding buffer counter. For each buffer q at S, the receiver R maintains the counter value c_q piggybacked in the last packet from the buffer. When R receives another packet from the buffer q later, R checks whether the counter value piggybacked in the packet equals to c_q: if they are equal, R knows that the packet is a duplicate and drops it; otherwise R regards the packet as a new one and accepts it. The duplicate detection is local in the sense that it only requires information local to each queue buffer instead of imposing any rule involving different buffers (such as in sliding-window) that can degenerate system performance.

For the correctness of the above duplicate detection mechanism, we only need to choose the domain size C for the counter value such that the probability of losing C packets in succession is negligible. For example, for the high per-hop packet loss probability 22.7% in the case of Lites trace, C could still be as small as 7,

since the probability of losing 7 packets in succession is only 0.003%. (Given the small domain size for the counter value as well as the usually small queue size at each node, the duplicate detection mechanism does not consume much memory. For example, it only takes 36 bytes in the case of Lites.)

In addition to duplicate detection, we also use buffer counter to filter out obsolete acknowledgment. Despite the low probability, packet forwarding at R may be severely delayed, such that the queue buffers signified in a block acknowledgment have been reused by S to hold packets arriving later. To deal with this, R attaches to each forwarded packet the ID as well as the counter value of the buffer holding the packet at S originally; when S snoops a packet forwarded by R, S checks whether the piggybacked counter value equals to the current value of the corresponding buffer: if they are equal, S regards as valid the piggybacked block acknowledgment; otherwise, S regards the block acknowledgment as obsolete and ignores it.

Aggregated-ack at the base station. In sensor networks, the base station usually forwards all the packets it receives to an external network. As a result, the children of the base station (i.e., the nodes that forward packets directly to the base station) are unable to snoop the packets the base station forwards, and the base station has to explicitly acknowledge the packets it receives. To reduce channel contention, the base station aggregates several acknowledgments, for packets received consecutively in a short period of time, into a single packet and broadcasts the packet to its children. Accordingly, the children of the base station adapt their control parameters to the way the base station handles acknowledgments.

2.3.3 Differentiated contention control

In wireless sensor networks where per-hop connectivity is reliable, most packet losses are due to collision in the presence of severe channel contention. To enable reliable packet delivery, lost packets need to be retransmitted. Nevertheless, packet retransmission may cause more channel contention and packet loss, thus degenerating communication reliability. Also, there exist unnecessary retransmissions due to ack-loss, which only increase channel contention and reduce communication reliability. Therefore, it is desirable to schedule packet retransmissions such that they do not interfere with transmissions of other packets.

The way the virtual queues are maintained in our window-less block acknowledgment scheme facilitates the retransmission scheduling, since packets are automatically grouped together by different virtual queues. Packets in higher-ranked virtual queues have been transmitted less number of times, and the probability that the receiver has already received the packets in higher-ranked virtual queues is lower (e.g., 0 for packets in Q_0). Therefore, we rank packets by the rank of the virtual queues holding the packets, and higher-ranked packets have higher-priority

in accessing the communication channel. By this rule, packets that have been transmitted less number of times will be (re)transmitted earlier than those that have been transmitted more, and interference between packets of different ranks is reduced.

Window-less block acknowledgment already handles packet differentiation and scheduling within a node, thus we only need a mechanism that schedules packet transmission across different nodes. To reduce interference between packets of the same rank and to balance network queuing as well as channel contention across nodes, inter-node packet scheduling also takes into account the number of packets of a certain rank so that nodes having more such packets transmit earlier.

To implement the above concepts, we define the rank rank(j) of a node j as <M-k, $|Q_k|$, ID(j)>, where Q_k is the highest-ranked non-empty virtual queue at j, $|Q_k|$ is the number of packets in Q_k, and ID(j) is the ID of j. rank(j) is defined such that 1) the first field guarantees that packets having been transmitted fewer number of times will be (re)transmitted earlier, 2) the second field ensures that nodes having more packets enqueued get chances to transmit earlier, and 3) the third field is to break ties in the first two fields. A node with a larger rank value ranks higher. Then, the distributed transmission scheduling works as follows:

- Each node piggybacks its rank to the data packets it sends out.
- Upon snooping or receiving a packet, a node j compares its rank with that of the packet sender k. j will change its behavior only if k ranks higher than j, in which case j will not send any packet in the following $w(j, k) \times T_{pkt}$ time. T_{pkt} is the time taken to transmit a packet at the MAC layer, and w(j, k) = 4 - i, when rank(j) and rank(k) differ at the i-th element of the 3-tuple ranks. w(j, k) is defined such that the probability of all waiting nodes starting their transmissions simultaneously is reduced, and that higher-ranked nodes tend to wait for shorter time. T_{pkt} is estimated by the method of *Exponentially Weighted Moving Average* (EWMA).
- If a sending node j detects that it will not send its next packet within T_{pkt} time (i.e., when j knows that, after the current packet transmission, it will rank lower than another node), j signifies this by marking the packet being sent, so that the nodes overhearing the packet will skip j in the contention control. (This mechanism reduces the probability of idle waiting, where the channel is free but no packet is sent.)

2.3.4 Experimental results

Table X.4 shows the performance results of RBC, and we can observe the following properties of RBC:

Metrics	RT = 0	RT = 1	RT = 2
ER (%)	56.21	83.16	95.26
PD (seconds)	0.21	1.18	1.72
EG (packets/sec)	4.28	5.72	6.37

Table X.4: RBC in Lites trace

- The event reliability keeps increasing, in a significant manner, as the number of retransmissions increases. The increased reliability mainly attributes to reduced unnecessary retransmissions (by reduced ack loss and adaptive retransmission timer) and retransmission scheduling.
- Compared with SWIA which is also based on implicit-ack, RBC reduces packet delivery delay significantly. This mainly attributes to the ability of continuous packet forwarding in the presence of packet- and ack-loss and the reduction in timer-incurred delay.
- The rate of packet reception at the base station and the event goodput keep increasing as the number of retransmissions increases. When packets are retransmitted up to twice at each hop, the event goodput reaches 6.37 packets/second, quite close to the optimal goodput — 6.66 packets/second — for Lites trace.

Compared with SWIA, RBC improves reliability by a factor of 2.05 and reduces average packet delivery delay by a factor of 10.91. Compared to SEA with B-MAC (simply referred to as SEA hereafter), RBC improves reliability by a factor of 1.74, but the average packet delivery delay increases by a factor of 6.61 in RBC. Interestingly, however, RBC still improves the event goodput by a factor of 1.75 when compared with SEA. The reason is that, in RBC, lost packets are retransmitted and delivered after those packets that are generated later but transmitted less number of times. Therefore, the delivery delay for lost packets increases, which increases the average packet delivery delay, without degenerating the system goodput. The observation shows that, due to the unique application models in sensor networks, metrics evaluating aggregate system behaviors (such as the event goodput) tend to be of more relevance than metrics evaluating unit behaviors (such as the delay in delivering each individual packet).

3 Data transport control in reprogramming

In this section, we discuss the basic issues and approaches in data transport control for the purpose of sensor network reprogramming.

3.1 Introduction

The large scale of deployments of the wireless networks of embedded devices demand an ability to reprogram the nodes in the field, possibly over multiple hops. Since a program has to reach in entirety, the reprogramming service has to deliver data with 100% reliability. Therefore, the need for a reprogramming service translates into a problem of reliable dissemination of bulk data in wireless networks of embedded devices. Designing such a service is a challenging problem due to the limited energy, memory, and lossy wireless links.

3.2 Background

Early work in wireless networks showed that simple retransmissions of broadcast messages leads to the broadcast storm problem, where redundancy, contention, and collision impair the ability to perform well. The naïve approach of simple retransmission is not reliable and fast. Hence, a more intricate handling of the transmissions in the space and time is needed.

The problem of sending a new program, which typically consists of many packets, is different from that of sending a command, which typically consists of a few packets. The probability of contention and collisions is more in the case of sending a new program. Further, optimizing latency of transmission is an important concern for the reprogramming case. Although broadcasting a few packets has its own research challenges, we will only focus on broadcasting a large number packets due to the space constraints. Hence, the solutions for disseminating less data are not suitable for the reprogramming. In this section, we will look at the state-of-the-art reprogramming services for the wireless networks of embedded devices.

3.3 Challenges

The primary challenges in the problem of reprogramming are as follows.

- 100% reliability: the lossy links commonly found in the wireless networks of embedded devices makes the problem of providing 100% reliability hard.
- Energy consumption: the battery-powered nature of the embedded devices necessitates that the energy consumption has to be minimized. The operations for Mica mote in the decreasing order of energy consumed are as follows: EEPROM write 16-bytes, transmit a packet, receive a packet, idle listen for 1-milisecond, and EEPROM read 16-bytes [Mainwaring et al. 2002, Hui and Culler 2004].

- Time to reprogram the entire network: since the primary objective of the networks is sensing, it is desirable to minimize the time required to reprogram the network.
- Memory consumption: since the size a program could be larger than that of the available RAM, the broadcast service must be scalable in terms memory consumption.

All of the above mentioned challenges differentiate the problem of reliable dissemination in wireless embedded devices from that in wireless networks of PCs.

3.4 Techniques of reprogramming

We enumerate the commonly used techniques to address the above-mentioned challenges. These will help practitioners to understand the working of the state-of-the-art reprogramming services. The practitioners can use them to tune the performance of the existing reprogramming services or develop new services of their own.

- 100% reliability:
  - Hop-by-hop recovery: Given the lossy nature of the network, the recovery of the lost packets is done in a hop-by-hop manner. This reduces the number of transmissions as compared to that of end-to-end recovery.
  - Sender selection and suppression: The goal of the sender selection and suppression technique is to ensure that at most one node broadcast the data in a radio range. An example of the selection criteria to select a node that has larger number of potential receivers [Kulkarni and Wang 2005]. If a node sending data messages also overhears data messages from other nodes at the same time, it suppresses its transmissions based on any rule that uniquely determines an order among the nodes [Hui and Culler 2004]. An example of such rule is the IDs of the nodes. The sender selection and suppression reduces the number of collisions.
  - Time Division Multiple Access (TDMA): At the link layer, CSMA/CA protocol results in lower latency when the number of nodes, simultaneously transmitting in a neighborhood, is less. As the number of nodes increases, the number of backoffs increases. Also, the number of collisions due to the hidden terminal effect increases, thereby resulting in more retransmissions and hence more latency. One way to deal with the increased number of simultaneous transmissions is to use TDMA, where each node is allocated a time-slot to transmit [Naik et al. 2007, Kulkarni and Aramugam 2004]. The TDMA schedule is computed to guarantee that no two nodes within collision range from each other

transmit at the same time. The collision range is approximately equal to the twice the transmission range.

- Use of implicit ACK and NACK-based explicit ACK: The use of TDMA creates a lower bound on the latency to hear from each of the transmitter in a node's range. In simpler words, a node knows when the other nodes in its neighborhood will transmit [Naik et al. 2007, Kulkarni and Aramugam 2004]. This property enables a node to use implicit acknowledgement to detect message loss. The advantage of implicit acknowledgement over that of explicit is that the implicit acknowledgement reduces the number of message transmissions and hence more energy efficient. However, implicit ACK requires the sender to maintain state about the receivers. The amount of state grows linearly in terms of the number of receivers. Therefore, a more scalable approach is to use NACK-based recovery, where a receiver reports the sequence numbers of the lost packets to the sender and the sender rebroadcasts the requested packets [Kulkarni and Wang 2005].

- Energy consumption:
  - Sender selection and suppression: Use of sender selection and suppression reduces the number of concurrent transmissions in a neighborhood, thereby reducing the number of collisions.
  - Load balancing while selecting senders: An unfair sender selection process will tax a sender with transmissions and sapping its energy. A fair sender selection process takes into consideration the remaining energy at the node [Kulkarni and Wang 2005].
  - Duty cycling of radio: If a technique of sender selection and suppression is employed, a node that loses in the selection and suppression round, can chose to switch off it radio [Kulkarni and Arumugam 2004]. The use of sender selection and suppression gives an opportunity to the non-selected and suppressed nodes to switch off their radios while the selected sender is transmitting.
  - Minimum Connected Dominating Set (MCDS) for selecting senders: The transmission of radio messages consume significant amount of energy. Reducing the number of senders will save energy. A constraint to the problem of selecting senders is that all the nodes must receive the entire program. If we induce a graph over the wireless network, the problem of selecting a minimum set of senders is equivalent to that of finding a minimum connected dominating set (MCDS) of the induced graph [Naik et al. 2007]. A formal definition of MCDS is given in the section titled Terminologies/Keywords, here we give an example as shown in Figure X.3.

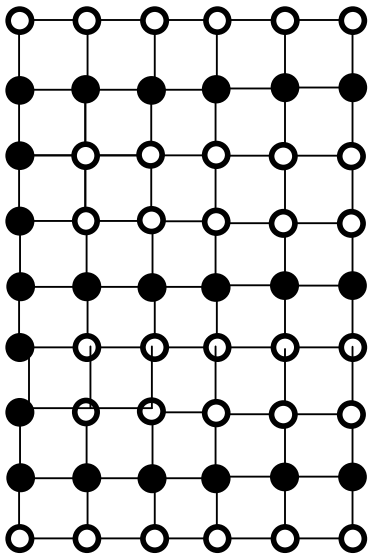

Figure X.3: MCDS of a network. The circles represent nodes. A line between two nodes means that those two nodes can communicate with each other. The filled-in circles represent MCDS.

- Time to reprogram the entire network:
  - Pipelining of messages over multiple hops: Although a node has not received an entire program, it can become a sender and start sending packets. This policy results in pipeline of transmissions, thereby reducing the latency [Hui and Culler 2004, Naik et al. 2007,Kulkarni and Arumugam 2004]. However, the flipside of the policy can be hidden terminal effect if care is not taken to ensure that two nodes within colliding range are simultaneously transmitting.
  - Transmitting as fast as possible: The use of sender selection and suppression technique enables a sender to transmit data packets at the fastest rate [Hui and Culler 2004].

- Memory consumption:
  - Forward/download phase in MNP: The size of a program could be far more than that of RAM. In fact, it could even be infeasible to save a bitmap, where a bit is allocated for each packet in the program, in the RAM. Two ways to deal with the small size of RAM is to either use a window-based recovery or save the information about the lost packets in the EEPROM instead of RAM. In window-based recovery, a sender does not transmit a new packet unless all the packets in the last window are successfully received. In the latter option of using EEPROM, al-

though the size is not a problem, the latency become an issue since read and write access to the EEPROM is slower than that of RAM. One way to expedite access to the lost packets is to maintain a linked list connecting the slots of the lost packets [Kulkarni and Wang 2005]. This way, it is not necessary to traverse the entire list of packets to search for the lost packets.

– Segmentation of the program: The entire program could be divided into packets and a fixed N number of buffers could be allocated in the RAM, where the size of the buffer is same as that of a packet [Hui and Culler 2004]. Since a node does have all of the packets in its RAM, the node can respond quickly only to those retransmit requests, for which the packets are in its RAM.

Although we have classified techniques depending upon with of the 4 distinct challenges do they address, the classification is not disjoint. For example, sender selection technique not only reduces collisions and improves reliability, but it also saves energy by reducing number of transmissions.

In Table X.6, we summarize which of the above-mentioned techniques the commonly used reprogramming services employ. The first row lists the 4 challenges and the first column lists the name of the services.

4 Thoughts for practitioners

Convergecast. In RBC, the tolerance of out-of-order packet delivery enables the design of windowless block acknowledgment. In general, network protocol design tends to be application-specific in wireless sensor networks, and we should pay attention to the application properties in designing or choosing network protocols. For instance, open-loop, hop-by-hop control is more appropriate for transient congestion, and closed-loop, end-to-end control is better for persistent congestion. Moreover, end-to-end error control may well be necessary to ensure 100% data delivery.

Reprogramming. The reprogramming services can be categorized into two broad classes, which are (a) ad-hoc and (b) structured, depending upon their approaches to selecting senders. The structured approach induces a graph over the network of nodes, where each node is a vertex and there is an edge between two vertices if the two corresponding nodes can communicate with each other. It then uses this graph

	Reliability	Energy efficiency	Latency	Memory consumption
Deluge [Hui and Culler 2004]	Sender selection and suppression, NACK-based	Sender selection and suppression	Pipelining while forwarding packets	Dividing the program into pages and packets

	recovery			
Infuse [Kulkarni and Arumugam 2004]	TDMA, Implicit ACK	Duty cycling of radio	Pipelining while forwarding packets	Window based recovery
MNP [Kulkarni and Wang 2005]	Sender selection and suppression, NACK-based recovery	Load balancing while selecting senders	---	Use of linked list in EEPROM to keep records of lost packets
Sprinkler [Naik et al. 2007]	TDMA, Implicit ACK	Use of MCDS while selecting senders	Pipelining while forwarding packets	---

Table X.6: Techniques used by the well known reprogramming services

to select senders. The ad-hoc services do not induce such a graph. While Deluge and MNP fall under the ad-hoc category, Infuse and Sprinkler fall under the structured category.

The benefit of using a structured approach is that computing a MCDS and a TDMA schedule requires fewer control messages than that of ad-hoc approach. An intuitive reason behind this is that position of a node in a graph can be used to decide whether the node becomes a sender or not. However, inducing a graph over nodes can be complex given the highly varying nature of the wireless links. For example, Infuse and Sprinkler assume that the distance between the nodes is an indication of the quality of the link between them and rely on location of the nodes to induce a graph. In practice, the assumption may not hold and could degrade the performance of the protocol.

5 Directions for future research

Convergecast. Despite the fact that many data transport control mechanisms have been proposed for convergecast in wireless sensor networks, how to effectively ensure application-specific QoS remains an open issue. Much work is also needed to address the interaction between QoS provisioning and in-network processing in wireless sensor networks, since QoS provisioning affects the spatial and temporal data flow in the network, which in turns affects the effectiveness of in-network processing and thus messaging efficiency and reliability. Network coding tends to be an effective approach to improving the efficiency and QoS messaging in wire-

less networks, and it is worthwhile to explore how to apply network coding for the purpose of reliable, efficient data transport in sensor network convergecast.

Reprogramming. In the case of ad-hoc reprogramming services, the contention in the wireless medium may increase as the density of the network increases. For example, in Deluge the nodes advertise themselves as senders after they have received a page. In a dense network, these advertisements cause contention [Hui and Culler 2004]. Although knowledge about density of the network will eliminate this problem, such knowledge needs partial information about the graphical topology of the network and hence violates the philosophy of the ad-hoc approach. Therefore, future research is needed to suppress the advertisements to avoid contention.

The structured approaches, such as Infuse and Sprinkler, assume location information at each node to induce a graph [Naik et al. 2007, Kulkarni and Arumugam 2004]. Briefly, the idea is to use the position of nodes in the graph to decide whether two nodes are within contention range of each other and then compute MCDS and TDMA schedule. However, localization in wireless embedded networks is a hard problem in itself. Most of the state-of-the-art localization techniques demand special acoustic or ultrasonic hardware, which may not be available in all the embedded devices. Hence, computing MCDS and TDMA schedule without depending on a localization service demands further research.

6 Conclusions

We have reviewed the challenges and approaches for data transport control in sensor network convergecast and reprogramming-oriented broadcast. We have also seen the difference (in both challenges and solution methods) in data transport control for convergecast and broadcast. For guaranteed QoS and efficiency in convergecast and broadcast, we have also presented the important open problems in data transport control. In general, how to design application- and task-specific data transport control mechanisms remains an interesting, open problem.

References

Anish Arora, Prabal Dutta et al (2004). A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking**.** Computer Networks (Elsevier) 46(5):605-634

Cheng Tien Ee and Ruzena Bajcsy (2004). Congestion Control and Fairness for Many-to-One Routing in Sensor Networks. ACM SenSys

Geoffrey Brown, Mohamed Gouda, Raymond Miller (1989). Block Acknowledgment: Redesigning the Window Protocol. ACM SIGCOMM

Jonathan Hui, David Culler (2004). The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. ACM SenSys

Bret Hull, Kyle Jamieson, Hari Balakrishnan (2004). Mitigating Congestion in Wireless Sensor Networks. ACM SenSys

Sandeep Kulkarni, Mahesh Arumugam (2004). Infuse: A TDMA Based Data Dissemination Protocol for Sensor Networks. Technical Report MSU-CSE-04-46, Michigan State University

Sandeep Kulkarni, Limin Wang (2005). MNP: Multihop Network Reprogramming Service for Sensor Networks, IEEE ICDCS

Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, John Anderson (2002). Wireless Sensor Networks for Habitat Monitoring. First ACM International Workshop on Wireless Sensor Networks and Applications

Miklos Maroti (2004). The Directed Flood Routing Framework. Technical report, Vanderbilt University, ISIS-04-502

Vinayak Naik, Anish Arora, Prasun Sinha, Hongwei Zhang (2007). Sprinkler: A Reliable and Energy Efficient Data Dissemination Service for Extreme Scale Wireless Networks of Embedded Devices, IEEE Transactions on Mobile Computing, 6(7)

Joseph Polastre, Jason Hill, David Culler (2004). Versatile Low Power Media Access for Wireless Sensor Networks. ACM SenSys

Yogesh Sankarasubramanjam, Ozgur B. Akan, Ian F. Akyildiz (2003). ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. ACM MobiHoc

TinyOS. http://www.tinyos.net/

Chieh-Yih Wan, Shane B. Eisenman, Andrew Campbell (2003). CODA: Congestion Detection and Avoidance in Sensor Networks. ACM SenSys

Wei Ye, John Heidemann, Deborah. Estrin (2002). An Energy-Efficient MAC Protocol for Wireless Sensor Networks. IEEE INFOCOM

Hongwei Zhang, Anish Arora, Young-Ri Choi, Mohamed Gouda (2007). Reliable Bursty Convergecast in Wireless Sensor Network**.** Computer Communications (Elsevier) 30(13)

Terminologies/keywords:

1. Convergecast: the transport of packets from multiple spatially distributed sensor nodes to a common sink node.
2. Congestion control: control the packet generation rate at the sources and intermediate nodes to avoid over-utilizing the network in terms of the node packet buffers and wireless channels.
3. Error control: detect and recover transmission errors in the network which are caused by packet transmission collision and other factors.
4. Fairness: equality of different nodes in accessing network resources (e.g., wireless transmission bandwidth).
5. Windowless block acknowledgment: the block acknowledgment scheme that enables continuous data transmission irrespective of packet- and ack-loss without any constraint as imposed by the sliding window size in traditional window-based block acknowledgment mechanisms.
6. MCDS: A dominating set (DS) of a graph G = (V,E) is a subset of V′ of V such that every vertex $v \in V$ is either in V′ or adjacent to some member of V′. A minimum connected dominating set (MCDS) is a connected dominating set of minimum cardinality.
7. TDMA-based transmission: A time division multiple access (TDMA)-based transmission is a scheme where in each node is given a schedule, such that neither two adjacent nodes nor two nodes sharing a same adjacent node transmit at the same time.
8. Pipelining of transmissions: The process of pipelining of transmission is composed of simultaneous transmissions of packets by nodes in time.
9. Implicit ACK: Implicit ACK is an acknowledgement scheme, where a sender infers that (a) a packet has been successfully received if it overhears the forwarding of the packet by the sender's successor nodes and (b) otherwise if it does not overhear the forwarding.
10. NACK-based recovery: If a node recovers a lost packet by explicitly asking the sender to retransmit the lost packet then such a recovery is called as NACK-based recovery.

Exercises

1. What are the basic issues in data transport control in sensor network convergecast?
2. How is ESRT different from protocols such as CODA?
3. Study the paper on RBC [Zhang et al. 2007], and discuss the respective roles of window-less block acknowledgment and distributed contention control in improving the reliability and goodput of convergecast?
4. Analyze the ack-loss probability in RBC.
5. RBC has focuses on windowless block acknowledgment and distributed contention control. But queue may still overflow without careful flow control. Please design a flow control mechanism to work with RBC.
6. What is the broadcast storm problem?
7. What are the challenges in the reliable broadcast in wireless networks of embedded devices?
8. When is TDMA faster than CSMA/CA in terms of latency?
9. What is MCDS and how is it useful for the reliable broadcast in wireless networks of embedded devices?
10. What are the two categories of the reliable reprogramming services and what are differences between them?

Solutions to exercises

1. Congestion control, error control, and fairness control
2. ESRT focuses on the collective reliability in delivering information related to an event, yet CODA focuses more on the congestion control issue with respect individual source nodes.
3. To understand the individual impact of window-less block acknowledgment and differentiated contention control in RBC, Zhang et al. [Zhang et al. 2007] evaluated the performance of RBC without using differentiated contention control (i.e., RBC with window-less block acknowledgment only), and the results are shown in the following table

Metrics	RT = 0	RT = 1	RT = 2
ER (%)	54.90	77.19	82.29
PD (seconds)	0.22	1.12	1.52
EG (packets/sec)	4.04	4.13	4.12

Comparing the above results with Table X.4, we can observe the following:

- Differentiated contention control improves packet delivery performance even when there is no retransmission (i.e., RT = 0). This is because the contention control reduces channel contention by prioritizing channel access according to the degree of queue accumulation at different nodes.
- Without differentiated contention control, packet delivery reliability also improves significantly when RT (maximum number of per-hop retransmissions) increases from 0 to 1, but the improvement becomes far less when RT increases from 1 to 2. This is because differentiated contention control plays an increasingly important role when RT (thus channel contention) increases.

Comparing the above results with Tables 1 and 3, we see that, with window-less block acknowledgment alone, RBC significantly improves the packet delivery performance of SEA and SWIA. The reasons are as follows:

- Compared with SEA, the channel contention is less in window-less block acknowledgment because no explicit acknowledgment packet is generated (thus reducing the number of packets in the network). Moreover, the intra-node packet prioritization (via the queue management) in window-less block acknowledgment also improves the packet delivery reliability.
- Compared with SWIA, window-less block acknowledgment improves packet delivery reliability by reducing ack-loss probability (and thus reducing unnecessary packet retransmissions) and employing intra-node packet prioritization. Window-less block acknowledgment also significantly reduces packet delivery delay by careful timer management and by enabling continuous packet transmission in the presence of packet- and ack-loss.

4. For convenience, we define the following notations:

   - p: the probability of losing a single (data) packet;
   - N: the number of packets received in succession without any loss in the middle;
   - N': the number of packets lost in succession;
   - B: the number of packets received in succession without any loss in the middle, after a packet is already received;
   - A: the number of times that the acknowledgment for a packet is received at the sender.

   Assuming that packet losses are independent of one another, we have the probability mass functions for random variables N and N' as follows.

   $P[N = k] = p(1-p)^k$
   $P[N' = k] = (1-p)p^k$

   In RBC, when a packet m is received at a receiver R, the acknowledgment for m can reach back to the sender S in two ways: S snoops m when it is forwarded by R later, with probability P_{self}; or S does not snoop m but snoops a packet whose block acknowledgment acknowledges the reception of m, with probability P_{ba}. Therefore, the probability P_{rbc} of S receiving the acknowledgment for m can be derived as follows:

$$\begin{aligned}
P_{self} &= 1-p \\
P_{ba} &= p\sum_{k=0}^{\infty} P[B=k]P[A \geq 1|B=k] \\
&= p\sum_{k=0}^{\infty} P[B=k](1-P[A=0|B=k]) \\
&= p\sum_{k=0}^{\infty} P[N=k+1](1-P[N'=k]) \\
&= \frac{p(1-3p+4p^2-2p^3)}{1-p+p^2} \\
P_{rbc} &= P_{self} + P_{ba} \\
&= 1-p+\frac{p(1-3p+4p^2-2p^3)}{1-p+p^2}
\end{aligned}$$

   Then, the probability $P_{rbc'}$ of losing the acknowledgment for a packet in RBC is $1\text{-}P_{rbc}$.

   In the case of Lites trace and implicit-ack, p = 22.7%. Therefore $P_{rbc'}$ = 8.89%, reducing the ack-loss probability of SWIA by a factor of 2.07.

5. In the presence of high traffic load, the packet queue at a node may accumulate and overflow if the corresponding senders transmit too many packets in a short time. This issue can be avoided by a simple hop-by-hop flow control mechanism as follows:

   - When forwarding packets, a node piggybacks the number of free queue buffers at its place.
   - Whenever a sender S detects that the number L_r of free queue buffers at the receiver R is below a threshold L, S will stop sending any packet in the fol-

lowing $(L-L_r)\times d_{e,R}$ time. L is a constant chosen such that the probability of losing L packets in succession is negligible (by which the sender will not fail to detect the congestion state at the receiver), and $d_{e,R}$ is the average interval between R releasing one buffer and the next one while there are packets enqueued at R. (R estimates $d_{e,R}$ by the method of EWMA.)

- After learning the number L_r of free buffers at the receiver R each time, the sender S will send at most L_r packets to R in the following $L_r \times d_{e,R}$ time unless S snoops another packet forwarded by R.
- To help relieve queue congestion, the nodes having less than L queue buffers are not subject to the differentiated contention control.

6. Simple retransmission of packets for forwarding the packets results in contention and collisions and impairs the reliability in a wireless network. Such a phenomenon is called as the broadcast storm problem.
7. The challenges in the reliable broadcast in wireless network networks of embedded devices are providing 100% reliability, minimizing energy consumption, minimizing latency, and minimizing memory usage.
8. CSMA/CA protocol results in lower latency when the number of nodes, simultaneously transmitting in a neighborhood, is less. As the number of nodes increases, the number of backoffs increases. Also, the number of collisions due to the hidden terminal effect increases, thereby resulting in more retransmissions and hence more latency. A minimal TDMA schedule, which designed by taking into consideration the density of nodes in a neighborhood, has lower latency than that of CSMA/CA.
9. A dominating set (DS) of a graph G = (V,E) is a subset of V′ of V such that every vertex $v \in V$ is either in V′ or adjacent to some member of V′. A minimum connected dominating set (MCDS) is a connected dominating set of minimum cardinality. A use of nodes in the MCDS of a network results in minimizing the number of transmitters, thereby saving the energy.
10. The two categories of the reliable reprogramming services are ad-hoc and structured-based. The structured approach induces a graph over the network of nodes, where each node is a vertex and there is an edge between two vertices if the two corresponding nodes can communicate with each other. The ad-hoc services do not induce a graph over the network.