Application-adaptive Messaging in Sensor Networks

Hongwei Zhang[†], Loren J. Rittle[‡], Anish Arora[†]
Technical report: OSU-CISRC-6/06-TR63
The Ohio State University

***Abstract*—We present an architecture *Sensornet Messaging Architecture* (SMA) for providing messaging services that self-adapt to application properties (e.g., QoS requirements and traffic patterns) in wireless sensor networks. In SMA, we decompose messaging into three sub-components: traffic-adaptive link estimation and routing, application-adaptive structuring, and application-adaptive scheduling. Taking packet packing (i.e., aggregating shorter packets into longer ones) as an example of in-network processing, we propose an algorithm that schedules packet transmissions to improve the achievable in-network processing while satisfying application QoS requirements at the same time. We evaluate our design by both simulation and experimentation with Tmote Sky sensor nodes, and we find that our approach significantly improves energy efficiency and messaging reliability.**

***Keywords*—Wireless sensor networks, application-adaptive messaging, architecture, algorithm, structuring, scheduling, packet packing**

1 Introduction

Wireless sensor networks, which we refer to as *sensornet*s hereafter, consist of nodes that can sense, compute, communicate, and potentially control [4]. With their unique capabilities in observing and controlling the physical world, sensornets have a broad range of potential applications in science (e.g., ecology and seismology), engineering (e.g., industrial control and precision agriculture), and our daily life (e.g., traffic control and health care). The broad application domains diversify sensornet systems in many ways, such as their traffic patterns and requirements on quality of service (QoS). For instance, in data-collection systems such as those for ecological study, application data are usually generated periodically, and the applications can tolerate certain degree of loss and delay in data delivery; yet in emergency-detection systems such as those for intruder detection, data are generated only when rare and emergent events occur, but the data need to be delivered reliably and in real time. Along with opportunities, application diversity poses substantial challenges to the design of efficient, dependable, and scalable messaging services in sensornets.

The past years of experience in building sensornet systems have shown that messaging services designed for one class of applications may not apply to another. For instance, the default TinyOS messaging stack, which was designed mainly for data-collection sensornets, did not work well and led to severe packet loss in event-detection sensornets such as those demonstrated in the sensornet field projects *A Line in the Sand* [5] and *ExScal* [6]. Therefore, we need to design messaging services according to the unique application properties, as also observed in [42].

In providing application-specific messaging services, the current practice of systems engineering in sensornets is to design a messaging stack for each individual application. Consequently, a variety of messaging stacks have been developed in the past few years [8, 30, 43], but they rarely interoperate with one another. Clearly, this monolithic approach is inefficient and unscalable in the long run for several reasons. Firstly, the number of applications will increase (and potentially in a significant manner) as sensornets evolve, but we cannot afford to develop individual messaging stack for each new application. Secondly, we may not know the exact requirements or properties of the applications in hand when we are designing messaging services, especially when application properties change temporally; this makes it difficult if not impossible to customize messaging services before application deployment. Therefore, we need an approach of designing messaging services that does not require building the whole stack from scratch each time a new application emerges, and that does not assume knowing application properties before deployment.

To fulfill the above objective, we need a messaging architecture that identifies the common components across diversified applications, so that these components can be reused from one application to another. Then, for messaging components to be reusable across diversified applications, we need to design messaging algorithms that automatically adapt to application properties on the fly. This demand for a unified architecture and automatic application adaptation also arises when we want to shield the complexity of messaging from application developers [37]. Having a unified architecture and application-adaptive messaging services will facilitate designing a unified interface between the messaging layers and applications, so that application developers do not need to understand the details of the underlying messaging services, and they only need to provide a few high level application-specific parameters such as their QoS requirements for messaging.

Contributions of the paper. To accommodate diversity in sensornet applications, we propose the *Sensornet Messaging Architecture* (SMA) in which we adopt two levels of abstraction:

- At the lower level, we identify the component *traffic-adaptive link estimation and routing* (TLR) that is responsible for precisely estimating wireless link properties (e.g., reliability) according to application traffic patterns. TLR is generic to all sensornet applications and can be performed automatically without explicit input from applications.
- At the higher level, we identify the components *application-adaptive structuring* (AST) and *application-adaptive scheduling* (ASC) to support functionalities (e.g., in-network processing and QoS) that are tightly coupled

[†] Hongwei Zhang and Anish Arora are with the Department of Computer Science and Engineering, The Ohio State University, USA. E-mail: {zhangho, anish}@cse.ohio-state.edu.
[‡] Loren J. Rittle is with the Pervasive Platforms and Architectures Lab, Motorola Labs, USA. E-mail: ljrittle@motorola.com.

with applications. AST and ASC incorporate application-specific properties (e.g., methods of in-network processing and QoS requirements) in forming messaging structures and in scheduling packet transmissions respectively.

Taking packet packing (i.e., aggregating shorter packets into longer ones) as an example of in-network processing, we study the ASC component in detail. We propose to schedule packet transmissions so that the *utility* of a transmission (e.g., degree of in-network aggregation) is maximized. To this end. we propose a distributed algorithm in which a node dynamically estimates the potential utility of transmitting a packet and decides when to transmit so that the utility is maximized while satisfying certain end-to-end timeliness guarantees on data delivery. This algorithmic framework is generically applicable to other in-network processing methods.

We evaluate our design via both simulation and experimentation with 18 Tmote Sky sensor nodes. We find that our approach significantly improves energy efficiency and messaging reliability. For instance, the energy efficiency is improved by a factor up to 3.22, and the reliability is improved by 12.92%.

Considering the diversities in both applications and technologies, defining messaging architecture is usually a challenging and long-running process. Thus we do not expect to be definitive and exhaustive in this paper, instead we hope to draw more attention to the architectural and algorithmic aspects of application-adaptive messaging by presenting the initial thoughts on the messaging architecture and some benefits of application-adaptive messaging.

Organization of the paper. We present the messaging architecture SMA in Section 2, and we design the algorithm for application-adaptive scheduling in Section 3. Then, we evaluate our design in Section 4, and we discuss related work in Section 5. We make concluding remarks and discuss open issues in Section 6.

2 SMA: an architecture for application-adaptive messaging

In this section, we first review the basic functions of sensornet messaging, based upon which we identify the common messaging components and design the architecture SMA.

As in the case for the Internet, the objective of messaging in sensornets is to deliver data from their sources to their destinations. To this end, the basic tasks of messaging are, given certain QoS constraints (e.g., reliability and latency) on data delivery, choose the route(s) from every source to the corresponding destination(s) and schedule packet flow along the route(s). Unlike wired networks, however, wireless communication, constrained resources, and application diversity in sensornets introduce new challenges to the design of messaging services.

Nodes communicate with one another via wireless links in sensornets, yet wireless links are subject to the impact of a variety of factors such as fading, multi-path, environmental noise, and co-channel interference. Consequently, the properties of wireless links (e.g., reliability) are dynamic and assume complex spatial and temporal patterns [48, 46]. Therefore, one foundational component of sensornet messaging is precisely estimating wireless link properties and then finding routes of high quality links to deliver data traffic. Given that data traffic pattern affects wireless link properties due to interference among simultaneous transmissions [46], link estimation and routing should be able to take into account the impact of application data traffic, and we call this basic messaging component *traffic-adaptive link estimation and routing (TLR)*.

With the basic communication structure provided by the TLR component, another important task of messaging is to adapt the structure and data transmission schedules according to application properties such as in-network processing and QoS requirements. Given the resource constraints in sensornets, application data may be processed in the network before it reaches the final destination to improve resource utilization. For instance, data arriving from different sources may be compressed at an intermediate node before it is forwarded further. Given that messaging determines the spatial and temporal flow of application data and that data items from different sources can be processed together only if they meet somewhere in the network, messaging significantly affects the degree of processing achievable in the network. It is therefore desirable that messaging consider in-network processing when deciding how to form the messaging structure and how to schedule data transmissions. In addition, messaging should also consider application QoS requirements (e.g., reliability and latency in packet delivery), because messaging structure and transmission schedule determine the QoS experienced by application traffic[21, 39, 18]. In-network processing and QoS requirements tend to be tightly coupled with applications, thus we call the structuring and scheduling in messaging *application-adaptive structuring (AST)* and *application-adaptive scheduling (ASC)* respectively.

These messaging components are coupled with applications in different ways and in different degrees, so we adopt two levels of abstraction in designing the architecture for application-adaptive messaging. The architecture, SMA (for *Sensornet Messaging Architecture*), is shown in Figure 1. At the lower level, traffic-adaptive link estimation and routing (TLR) interacts directly with the link layer to estimate link properties and to form the basic routing structure in a traffic-adaptive manner. TLR can be performed without explicit input from applications, and TLR does not directly interface with applications. At the higher level, both application-adaptive structuring (AST) and application-adaptive scheduling (ASC) need input from applications, thus AST and

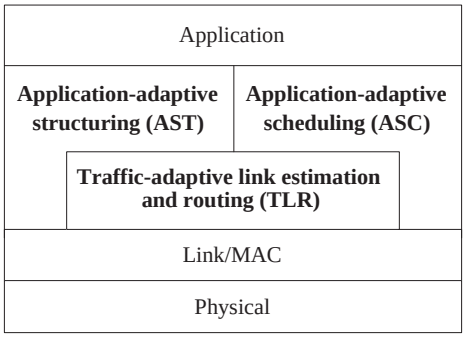


Figure 1: SMA: an architecture for application-adaptive messaging

ASC interface directly with applications. Besides interacting with TLR, AST and ASC may need to directly interact with link layer to perform tasks such as adjusting radio transmission power level and fetching link-layer acknowledgment to a packet transmission. In the architecture, the link and physical layers support higher-layer messaging tasks (i.e., TLR, AST, and ASC) by providing the capability of communication within one-hop neighborhoods.

In what follows, we elaborate on the individual components of SMA.

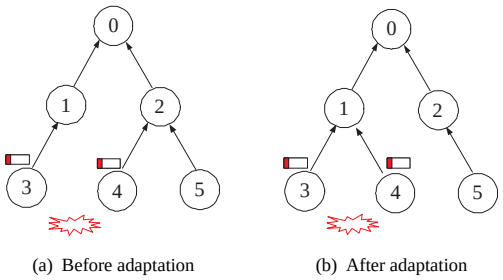
Traffic-adaptive link estimation and routing (TLR). To estimate wireless link properties, one approach is to use beacon packets as the basis of link estimation. That is, neighbors exchange broadcast beacons, and they estimate broadcast link properties based on the quality of receiving one another's beacons (e.g., the ratio of beacons successfully received, or the RSSI/LQI of packet reception); then, neighbors estimate unicast link properties based on those of broadcast, since data are usually transmitted via unicast. This approach of beacon-based link estimation has been used in several routing protocols including ETX [43, 10]. Nevertheless, we find that there are two major drawbacks of beacon-based link estimation. Firstly, it is hard to build high-fidelity models for temporal correlations in link properties [41, 40, 22], thus most existing routing protocols do not consider temporal link properties and assume independent bit error or packet loss. Consequently, significant estimation error can be incurred as we showed in [46]. Secondly, even if we could precisely estimate unicast link properties, the estimated values may only reflect unicast properties in the absence rather than in the presence of data traffic, since network traffic pattern affects link properties due to interference. This is especially the case in event-detection applications, where events are usually rare (e.g., one event per day) and tend to last for only a short time at each network location (e.g., less than 20 seconds). Therefore, beacon-based link estimation cannot precisely estimate link properties in a traffic-adaptive manner.

To address the limitations of beacon-based link estimation, Zhang et al. [46] proposed the routing protocol *Learn on the Fly* (LOF) that estimates unicast link properties via MAC feedback [1] for data transmissions themselves without using beacons. Since MAC feedback reflects in-situ network condition in the presence of application traffic, link estimation in LOF is traffic-adaptive. LOF also addresses the challenges of data-driven link estimation to routing protocol design, such as uneven link sampling (i.e., the quality of a link is not sampled unless the link is used in data forwarding). It has been shown that, compared with beacon-based link estimation and routing, LOF improves both the reliability and energy efficiency in data delivery. More importantly, LOF quickly adapts to changing traffic patterns, and this is achieved without any explicit input from applications. Thus LOF can serve as an instantiation of the TLR component.

The TLR component provides the basic service of automatically adapting link estimation and routing structure according to application traffic patterns. TLR also exposes its knowledge of link and route properties (such as end-to-end packet delivery latency) to higher level components AST and ASC, so that AST and ASC can optimize the degree of in-network processing while providing the required QoS in delivering individual pieces of application data.

Application-adaptive structuring (AST). One example of application-adaptive structuring is to adjust messaging structure according to application QoS requirements. For instance, radio transmission power level determines the communication range of each node and the connectivity of a network. Accordingly, transmission power level affects the number of routing hops between any pairs of source and destination and thus packet delivery latency. Transmission power level also determines the interference range of packet transmissions, and thus it affects packet delivery reliability. Therefore, radio transmission power level (and thus messaging structure) can be adapted to satisfy specific application QoS requirements, and Kawadia and Kumar have studied this in [21].

Besides QoS-oriented structuring, another example of application-adaptive structuring is to adjust messaging structure according to the opportunities of in-network processing. Messaging structure determines how data flows spatially, and thus affects the degree of in-network processing achievable. For instance, as shown in Figure 2(a), nodes 3 and 4 detect the same



(a) Before adaptation (b) After adaptation

Figure 2: Example of application-adaptive structuring

event simultaneously. But the detection packets generated by nodes 3 and 4 cannot be aggregated in the network, since they follow different routes to the destination node 0. On the other hand, if node 4 can detect the correlation between its own packet and that generated by node 3, node 4 can change its next-hop forwarder to node 1, as shown in Figure 2(b). Then the packets generated by nodes 3 and 4 can meet at node 1, and be aggregated before being forwarded to the destination node 0.

In general, to improve the degree of in-network processing, a node should consider the achievable in-network processing when choosing the next-hop forwarder. One way to realize this objective is to adapt existing routing metrics. For each neighbor k, a node j estimates the utility $u_{j,k}$ of forwarding packets to k, where the utility is defined as the reduction in messaging cost (e.g., number of transmissions) if j's packets are aggregated with k's packets. Then, if the cost of messaging via k without aggregation is $c_{j,k}$, the associated messaging cost $c'_{j,k}$ can be adjusted as follows (to reflect the utility of in-network processing):

$$c'_{j,k} = c_{j,k} - u_{j,k}$$

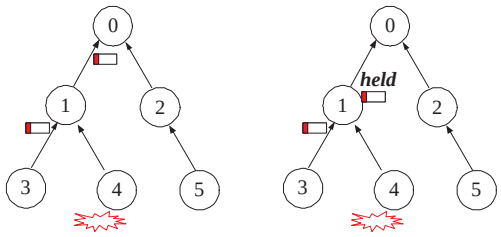
Accordingly, a neighbor with the lowest adjusted messaging cost is selected as the next-hop forwarder.

[1]The MAC feedback for a unicast transmission includes whether the transmission has succeeded and how many times the packet has been retransmitted at the MAC layer.

Since QoS requirements and in-network processing vary from one application to another, AST needs input (e.g., QoS specification and utility of in-network processing) from applications, and it needs to interface with applications directly.

Application-adaptive scheduling (ASC). One example of application-adaptive scheduling is to schedule packet transmissions to satisfy certain application QoS requirements. To improve packet delivery reliability, for instance, lost packets can be retransmitted. But packet retransmission consumes energy, and not every sensornet application needs 100% packet delivery rate. Therefore, the number of retransmissions can be adapted to provide different end-to-end packet delivery rates while minimizing the total number of packet transmissions [5]. To provide differentiated timeliness guarantee on packet delivery latency, we can also introduce priority in transmission scheduling such that urgent packets have high priority of being transmitted [45]. Similarly, data streams from different applications can be ranked so that transmission scheduling ensures differentiated end-to-end throughput to different applications [13].

Besides QoS-oriented scheduling, another example of application-adaptive scheduling is to schedule packet transmissions according to the opportunities of in-network processing. Given a formed messaging structure, transmission scheduling determines how data flows along the structure temporally and thus the degree of in-network processing achievable. To give an example, let us look at Figure 3(a). Suppose node 4 detects an



(a) Before adaptation (b) After adaptation

Figure 3: Example of application-adaptive scheduling

event earlier than node 3 does. Then the detection packet from node 4 can reach node 1 earlier than the packet from node 3. If node 1 immediately forwards the packet from node 4 after receiving it, then the packet from node 4 cannot be aggregated with that from node 3, since the packet from node 4 has already left node 1 when the packet from node 3 reaches node 1. On the other hand, if node 1 is aware of the correlation between packets from nodes 3 and 4, then node 1 can hold the packet from 4 after receiving it (as shown in Figure 3(b)). Accordingly, the packet from node 3 can meet with that from node 4, and these packets can be aggregated before being forwarded.

In general, a node should consider both application QoS requirements and the potential in-network processing when scheduling data transmissions, so that application QoS requirements are better satisfied and the degree of in-network processing is improved. Given that in-network processing and QoS requirements are application specific, ASC needs to directly interface with applications to fetch input on parameters such as QoS specification and the utility of in-network processing.

In the remainder of this paper, we discuss ASC in detail using packet-packing (i.e., aggregating shorter packets into longer ones) as an example of in-network processing. Detailed study of AST is a part of our future work.

Remark. It is desirable that the components TLR, AST, and ASC be deployed all together to achieve the maximal network performance. That said, the three components can also be deployed in an incremental manner while maintaining the benefits of each individual component, as we will show in Section 4.

3 Application-adaptive scheduling

In this section, we study application-adaptive scheduling in the context of packet packing. We first discuss the concept and benefits of packet packing (i.e., aggregating shorter packets into longer ones), then we design a scheduling algorithm that improves the degree of in-network packet packing while satisfying application-specific QoS requirements. We also discuss related implementation issues.

3.1 Packet packing

In sensornets, an information unit (e.g., a report after an event detection) from each sensor is usually short (e.g., less than 10 bytes [5]), and the header overhead of each packet is relatively high (e.g., up to 31 bytes at the MAC layer of IEEE 802.15.4). Fortunately, the maximum size of MAC payload is usually much longer than that of each information unit (e.g., 102 bytes per MAC frame in 802.15.4). Therefore, the MAC frame format allows for aggregating several short information units into a single MAC frame, which we refer to as *packet packing* hereafter. Having several information units share the overhead[2] of a packet (or frame) transmission, packet packing reduces the amortized overhead of transmitting each information unit. Packet packing also reduces the number of packets contending for channel access, hence it reduces the probability of packet collision and improves information delivery reliability, as we will show in Section 4.

While aggregating short information units reduces the overhead of transmitting each information unit, it increases the length of packets being transmitted. Given that packet delivery rate of a wireless link decreases as packet length increases, a long packet with aggregated information units may be retransmitted more often, for reliable data delivery, than the short packets without aggregation. To understand whether packet packing is still beneficial in the presence of lossy wireless links, therefore, we need to understand whether the increased packet loss rate overshadows the benefits of packet packing. To this end, we mathematically analyze the issue as follows.

Feasibility analysis. For the sake of simplicity, we assume in this section that packet transmissions are independent, and we validate the benefit of packet packing in Section 4.2 where temporal correlations in packet transmissions are considered. We

[2]The overhead includes not only the number of header bytes transmitted but also the energy taken to wake up radios, since radios may well be in low-power sleeping state in sensornets.

also define the following notations:

l_1	:	payload length of an unpacked packet, i.e., the length of a single information unit;
p_1	:	delivery rate of an unpacked packet;
k	:	packing ratio, i.e., the ratio of the payload length of a packed packet to that of an unpacked packet;
h	:	the ratio of header length to payload length in an unpacked packet;
C_0	:	overhead of transmitting a packet.

Then, for a packed packet with packing ratio k, the ratio of the overall length of the packed packet to that of an unpacked packet is $\frac{kl_1+hl_1}{l_1+hl_1}$. Thus, the delivery rate p_k of the packed packet can be calculated as follows:

$$p_k = p_1^{\frac{kl_1+hl_1}{l_1+hl_1}} = p_1^{\frac{k+h}{1+h}}$$

To reflect the overhead of transmitting a packet pkt over a wireless link, we define the *amortized cost* (AC) of transmitting pkt as follows:

$$AC_{pkt} = \frac{C_0}{len_{pkt}} \times ETX_{pkt} \quad (1)$$

where len_{pkt} is the payload length of pkt, and ETX_{pkt} is the expected number of transmissions taken to successfully deliver pkt over the wireless link. Given that the expected number of transmissions to successfully deliver a packet with packing ratio k is $\frac{1}{p_k}$, the amortized cost of transmitting a packet with packing ratio k, denoted by AC_k, can be calculated as follows:

$$AC_k = \frac{C_0}{kl_1} \times \frac{1}{p_k} = \frac{C_0}{kl_1 p_k}$$

Since an unpacked packet has a packing ratio of 1, the amortized cost of transmitting an unpacked packet is AC_1, that is, $\frac{C_0}{l_1 p_1}$.

For a given packing ratio k, the ratio R_k of AC_1 to AC_k reflects whether packet packing is beneficial, that is, packet packing is beneficial if $R_k > 1$. Precisely, $R.k$ is calculated as follows:

$$R_k = \frac{AC_1}{AC_k} = kp_1^{\frac{k-1}{1+h}}$$

In a typical sensornet system [5, 6], the ratio h of header length to that of a single information unit is around 3, and the packing ratio can be up to 12. For $h = 3$, Figure 4 shows R_k as a function of p_1 and k, when $h = 3$. From the figure, we can see that packet packing reduces the amortized cost of packet transmission as long as the link reliability is no less than 40%, which is usually the case in practice (e.g., link reliability was $\tilde{7}5\%$ even in heavily loaded sensornet systems [5, 6]). We also see that, if link reliability is greater than 67%, the amortized cost of packet transmission always decreases as the packing ratio increases. Since link reliability is usually greater than 67% in practice, we can always try to maximize the packing ratio so that the amortized cost of packet transmission is reduced.

Remarks. The above analysis focuses on a single link, but the observations easily carry over to multi-hop networks since link reliability p_1 reflects the impact of channel fading and collision even in the case of multi-hop networks.[3] The analysis has not considered the benefits (e.g., fewer number of packet collisions) of reduced channel contention as a result of packet packing (which reduces the number of packets contending for channel access). We will study the impact of these factors by experimentation with real-world sensor nodes in Section 4.2.

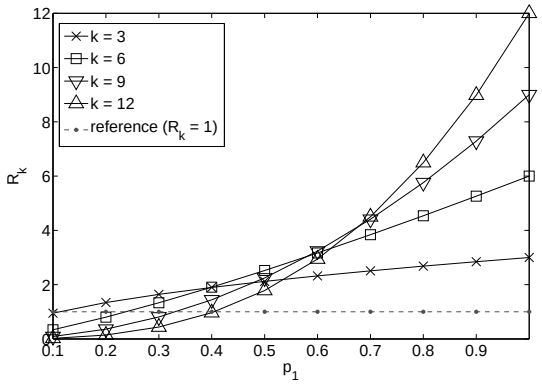


Figure 4: $R_k = \frac{AC_1}{AC_k}$

3.2 Packing-oriented scheduling

In this section, we design a scheduling algorithm to facilitate in-network packet packing. According to the analysis in Section 3.1, the amortized cost of packet transmission decreases as the packing ratio increases. Therefore, the objective of packing-oriented scheduling is to schedule packet transmissions such that as many short packets as possible are packed into long packets, by which the amortized overhead of packet transmission is reduced. To reflect the overhead of a packet transmission tx, we define the *amortized cost* (AC) of the transmission as $\frac{C_0}{L_{tx}}$, where L_{tx} is the payload length of the packet being transmitted. Then, we can define the *utility* of a scheduling action (i.e., transmit or hold a packet) as the expected reduction in the amortized cost of packet transmissions in the network. Accordingly, whether a short packet should be held at or be immediately transmitted from a node to its parent depends on the utility of locally holding the packet and the utility of transmitting the packet.

Since locally holding a packet increases the delay in delivering the packet, the scheduling algorithm should not hold a packet too long to violate the timeliness requirement of information delivery specified by the application. Therefore, both the timeliness requirement of information delivery and application traffic pattern (e.g., spatial and temporal distributions of data packets) affect packet transmission scheduling in a network. Since the timeliness requirement and the traffic pattern vary from one application to another and are usually unknown beforehand, the scheduling algorithm should adapt to the timeliness requirement and traffic pattern on the fly.

In what follows, we first discuss how to calculate the utilities of holding and transmitting a packet in an application-adaptive

[3]Note that the increased per-packet transmission time as a result of increased packet length will not cause more collision, since the time taken to transmit a packet (e.g., 4 milliseconds) is usually much less than the inter-packet interval (e.g., usually at least a few seconds).

manner, then we present a scheduling rule that improves the overall utility.

3.2.1 Utility calculation

For convenience, we define the following notations:

L	:	maximum payload length per packet;
$ETX.j$	:	expected number of transmissions taken to transport a packet from node j to its destination;
$p.j$	:	the parent or next-hop of a node j in the routing tree;
$ETX_0.j$	:	expected number of transmissions taken to transport a packet from node j to $p.j$.

(For simplicity of presentation, we only consider the case where every packet needs to delivered to be the base station of a sensornet [5]. The algorithm discussed in this paper is readily applicable to the case where there are multiple base stations.)

The utilities of holding and transmitting a packet pkt at a node j depend on the following parameters related to traffic patterns:

- With respect to j itself and its children:

t_l	:	expected time to receive another packet pkt' from a child or locally from an upper layer;
s_l	:	expected payload size of pkt'.

- With respect to the parent of j:

t_p	:	expected time till the parent transmits another packet pkt'' that does not contain information units generated or forwarded by j itself;
s_p	:	expected payload size of pkt''.

The utilities of holding and transmitting a packet pkt also depend on the following constraints posed by application QoS requirement and wireless communication:

- Grace period t_f for delivering pkt: the maximum allowable latency in delivering pkt minus the expected time taken to transport pkt from j to its destination without being held at any intermediate node along the route.

  If $t_f \leq 0$, pk should be transmitted immediately to minimize the extra delivery latency.
- Spare packet space s_f of pkt: the maximum allowable payload length per packet minus the current payload length of pkt.

  Parameter s_f and the size of the packets coming next from an upper layer at j or from j's children determine how much pkt will be packed and thus the potential utility of locally holding pkt.

Then, the utilities of holding and transmitting a packet are calculated as follows.

Utility of holding a packet. When a node j holds a packet pkt, pkt can be packed with packets from j's children or from an upper layer at j. Therefore, the utility of holding pkt at j is the expected reduction in the amortized cost of transmitting pkt after packing pkt. The utility depends on (a) the expected number of packets that j will receive within t_f time (either from a child or locally from an upper layer), and (b) the expected payload size s_l of these packets. Given that the expected inter-packet interval is t_l, the expected number of packets to be received at j within t_f time is $\frac{t_f}{t_l}$. Thus, the expected overall size $\mathcal{S}'_l$ of the payload to be received within t_f time is calculated as follows:

$$\mathcal{S}'_l = \frac{t_f}{t_l} s_l$$

Given the spare space s_f in the packet pkt, the expected size $\mathcal{S}_l$ of the payload that can be packed into pkt is calculated as follows:

$$\mathcal{S}_l = \min\{\mathcal{S}'_l, s_f\} = \min\{\frac{t_f}{t_l} s_l, s_f\}$$

Therefore, the expected amortized cost AC_l of transporting the packet to the destination after anticipated packing is calculated as follows:

$$AC_l = \frac{C_0}{(L - s_f) + \mathcal{S}_l} ETX.j$$

where $(L - s_f)$ is the payload length of pkt before packing.

Since the amortized cost AC'_l of transporting pkt without the anticipated packing is calculated as

$$AC'_l = \frac{C_0}{L - s_f} ETX.j$$

the utility U_l of holding pkt is calculated as follows:

$$\begin{aligned} U_l &= AC'_l - AC_l \\ &= \frac{C_0 \mathcal{S}_l}{(L-s_f)(L-s_f+\mathcal{S}_l)} ETX.j \\ &= \frac{C_0 \min\{\frac{t_f}{t_l} s_l, s_f\}}{(L-s_f)(L-s_f+\min\{\frac{t_f}{t_l} s_l, s_f\})} ETX.j \end{aligned} \tag{2}$$

Utility of immediately transmitting a packet. If node j transmits the packet pkt immediately to its parent $p.j$ when pkt is not yet fully packed, j pays the cost of transmitting a non-fully-packed packet. Yet the payload carried by pkt can be used to pack the packets that $p.j$ has received from its children other than j. Therefore, the utility of j transmitting a non-fully-packed packet pkt comes from the expected reduction in the amortized cost of packet transmissions at $p.j$ as a result of receiving the payload that pkt carries.

When j transmits pkt to $p.j$, the grace period of pkt at $p.j$ is still t_f, the expected number of packets that do not contain information units from j and can be packed with pkt at $p.j$ is $\frac{t_f}{t_p}$. Given the limited payload that pkt carries, it may happen that not all the packets to be transmitted at $p.j$ get packed (to full) via the payload from pkt. Accordingly, the utility U_p of immediately transmitting pkt is calculated as follows:

- If *all the parent packets get packed to full* via payload from pkt, i.e., $\frac{t_f}{t_p}(L - s_p) \leq L - s_f$:

  For each of such parent packet, the utility U' (or reduction in amortized cost) is calculated as follows:

  $$\begin{aligned} U' &= \frac{C_0}{s_p} ETX.(p.j) - \frac{C_0}{L} ETX.(p.j) \\ &= \frac{C_0(L-s_p)}{s_p L} ETX.(p.j) \end{aligned}$$

  Then, the overall utility U'_p is calculated as follows:

  $$U'_p = \frac{t_f}{t_p} U' = \frac{t_f}{t_p} \frac{C_0(L - s_p)}{s_p L} ETX.(p.j) \tag{3}$$

where $\frac{t_f}{t_p}$ is the expected number of packets that do not contain information units from j and can be packed with pkt.

- If *not all the parent packets get packed to full* via payload from pkt, i.e., $\frac{t_f}{t_p}(L - s_p) > L - s_f$:

  In this case, $\lfloor\frac{L-s_f}{L-s_p}\rfloor$ number of packets are packed to full, and the corresponding utility is $\lfloor\frac{L-s_f}{L-s_p}\rfloor\frac{C_0(L-s_p)}{s_pL}ETX.(p.j)$ (by Equation 3). In addition, there is a packet that gets partially packed via $\text{mod}(L - s_f, s_p)$ length of payload from pkt, and the corresponding utility is $\frac{C_0\text{mod}(L-s_f,s_p)}{(L-s_p)(L-s_p+\text{mod}(L-s_f,s_p))}ETX.(p.j)$ (by Equation 2). Therefore, the overall utility U_p'' is the summation of the above two terms as follows:

$$U_p'' = (\lfloor\frac{L-s_f}{L-s_p}\rfloor\frac{C_0(L-s_p)}{s_pL} + \frac{C_0\text{mod}(L-s_f,s_p)}{(L-s_p)(L-s_p+\text{mod}(L-s_f,s_p))})\times ETX.(p.j) \quad (4)$$

While immediately transmitting pkt to $p.j$ brings with it the utility discussed above, immediate transmission pays the cost C_p of transmitting a packet that is not full yet. According to the concept of amortized cost of packet transmission, C_p is calculated as follows:

$$\begin{aligned} C_p &= \frac{C_0}{L-s_f}ETX_0.j - \frac{C_0}{L}ETX_0.j \\ &= \frac{C_0 s_f}{(L-s_f)L}ETX_0.j \end{aligned} \quad (5)$$

Therefore, the utility U_p of immediately transmitting pkt to $p.j$ is calculated as follows:

$$U_p = \begin{cases} U_p' - C_p & \text{if } \frac{t_f}{t_p}(L - s_p) \le (L - s_f) \\ U_p'' - C_p & \text{otherwise} \end{cases} \quad (6)$$

where U_p', U_p'', and C_p are defined in Equations 3, 4, and 5 respectively.

3.2.2 Scheduling rule

To reduce the amortized cost of packet transmission, the objective of packing-oriented scheduling is to maximize the utility of transmission scheduling (including the utilities of transmitting and holding packets). Since we mainly focus on demonstrating the feasibility and benefits of application-adaptivity in messaging in this paper, we only study a greedy algorithm where each node tries to maximize the local utility of scheduling each packet transmission, and we relegate the design of globally optimal algorithm as a part of our future work.

Given a packet to be scheduled for transmission, if the probability that the packet is immediately transmitted is P_t ($0 \le P_t \le 1$), then the expected utility $U_t(P_t)$ is calculated as follows:

$$\begin{aligned} U_t(P_t) &= P_t \times U_p + (1 - P_t)U_l \\ &= U_l + P_t(U_p - U_l) \end{aligned} \quad (7)$$

where U_p and U_l are the utilities of immediately transmitting and locally holding the packet respectively. To maximize U_t, P_t should be set according to the following rule:

$$P_t = \begin{cases} 1 & \text{if } U_p > U_l \\ 0 & \text{otherwise} \end{cases}$$

That is, *the packet should be immediately transmitted if the utility of immediate transmission is greater than that of locally holding the packet*.

Remarks. The framework designed for packing-oriented scheduling is readily applicable to other in-network processing methods such as data compression, since the impact of in-network processing (no matter how it is achieved) can be modeled by the concept of *utility*. Detailed discussion of this, however, is beyond the scope of this paper.

3.2.3 Implementation

From the discussion in Section 3.2.1, a node j needs to obtain the following parameters when calculating the utilities of holding and transmitting a packet:

- On messaging structure: $ETX.j$, $p.j$, and $ETX_0.j$;
- On traffic pattern: t_l, s_l, t_p, s_p, and L.

Parameters related to messaging structure can be provided by component TLR or AST depending on the software architecture in a given system platform. On parameters related to traffic pattern, j can estimate by itself the parameters t_l and s_l, and L is readily available and fixed for each specific platform. To enable each node j to obtain parameters t_p and s_p, every node k in the network estimates the expected interval $t.k$ in transmitting two consecutive packets at k itself and the expected size $s.k$ of these packets. Then, every node k shares with its neighbors the parameters $t.k$ and $s.k$ by piggybacking these information onto data packets or other control packets in the network. When a node j overhears parameter $t.(p.j)$ and $s.(p.j)$ from its parent $p.j$, j can approximate t_p and s_p with $\frac{t.(p.j)\times t.j\times s.(p.j)}{t.j\times s.(p.j)-t.(p.j)\times s.k}$ and $s.(p.j)$ respectively. The derivation is as follows.

Approximation of t_p and s_p: Since information units generated or forwarded by the children of node $p.j$ are treated in the same manner (without considering where they are from), the expected size of the packet being transmitted by $p.j$ does not depend on whether the packet contains information units generated or forwarded by j. Thus, j can simply regard $s.(p.j)$ as s_p, the expected size of the packet transmitted by $p.j$ that does not contain information units coming from j.

Now we derive t_p as follows. Since the amount of payload transmitted by $p.j$ per unit time is $\frac{1}{t.(p.j)}s.(p.j)$ and the amount of payload transmitted by j is $\frac{1}{t.j}s.j$ per unit time, the amount of payload l_p that are transmitted by $p.j$ but are not from j per unit time is calculated as: $l_p = \frac{s.(p.j)}{t.(p.j)} - \frac{s.j}{t.j}$. Thus, the expected rate r_p that $p.j$ transmits packets that do not contain information units from j is calculated as: $r_p = l_p/s.(p.j) = \frac{1}{t.(p.j)} - \frac{s.j}{t.j\times s.(p.j)}$. Therefore, the expected interval t_p between $p.j$ transmitting two consecutive packets that do not contain information units from j is as follows: $t_p = \frac{1}{r_p} = \frac{t.(p.j)\times t.j\times s.(p.j)}{t.j\times s.(p.j)-t.(p.j)\times s.j}$.

□

4 Performance evaluation

We have implemented packing-oriented scheduling in TinyOS [1]. The implementation takes 40 bytes of RAM (plus the memory required for regular packet buffers) and 4,814 bytes of ROM.

To evaluate the performance of packing-oriented scheduling, we use the routing component MintRoute [43] that is readily available in TinyOS to form the routing structure. We are currently implementing the routing protocol *Learn on the Fly* (LOF) [46] in TinyOS, to provide the service for traffic-adaptive link estimation and routing (TLR). Packing-oriented scheduling is readily interoperable with LOF, but the detailed study is beyond the scope of this paper.

To understand the benefits of application-adaptive packet packing, we implement and compare the performance of the following messaging methods:

- *noPacking*: packets are delivered without being packed in the network.
- *simplePacking*: packets are packed if they are in the same queue, but there is not packing-oriented scheduling.
- *intelliPacking*: schedule packet transmissions so that packets are packed as much as possible while satisfying application requirement on the timeliness of information delivery, i.e., employ packing-oriented scheduling as discussed in Section 3.2).

In evaluating messaging performance, we consider the case of convergecast where every information unit is transported to a singe destination — the base station which acts as the interface between a sensornet and the rest of the world. For each method, its performance is evaluated according to the following metrics:

- *Packing ratio*: the average number of information units within each transmitted packet.
- *Energy efficiency*: the number of packet transmissions and receptions required to deliver a single information unit to the base station.
- *Information delivery reliability*: the ratio of the number of unique information units received at the base station to the number of unique information units generated in the network.

(Note: we do not compare the information delivery latency among the aforementioned messaging methods since they all satisfy the timeliness requirement specified by the application layer in our study.)

In what follows, we first evaluate the performance of different messaging methods via simulation. Then we evaluate their performance via experimentation with Tmote Sky sensor nodes [2], to corroborate our observations in simulation.

4.1 Simulation study

We use the simulator TOSSIM [24] that comes with TinyOS. In the simulation, 100 nodes are deployed in a 10×10 grid where each node can reliably communicate with nodes 3 grid-hops away. The traffic pattern is such that the base station is at one corner of the grid, and nodes in the farthest 4×4 subgrid from the base station periodically generate information units, with the interval between two consecutive information units uniformly distributed between 5 seconds and 15 seconds. The length of each information unit is 16 bytes, including information such as the node ID and timestamp at the source.

In the simulation, we first study a typical scenario where the maximum payload length is 102 bytes, and the application QoS requirement is specified such that the maximum allowable latency in delivering information units is 10 seconds [5, 6]. Then we study the impact that the maximum allowable information delivery latency and payload length have on the performance of intelliPacking. (We have also studied the impact of traffic load, and we found that increased traffic load has similar effect as that of increased allowable information delivery latency.)

A typical scenario. For the scenario where the maximum payload length is 102 bytes and the maximum allowable information delivery latency is 10 seconds, Figure 5 shows the packing

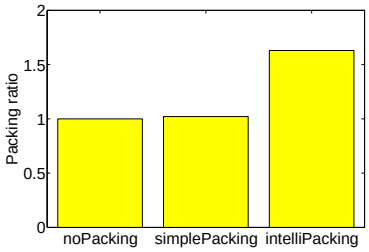
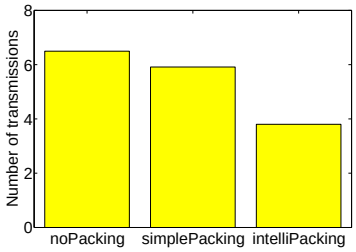


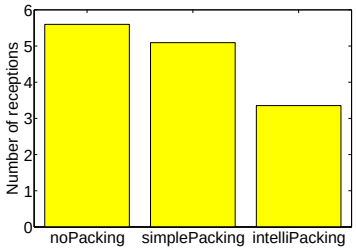
Figure 5: Packing ratio

ratio in the three messaging methods. The packing ratio is 1, 1.02, and 1.63 for noPacking, simplePacking, and intelliPacking respectively. We can see that, compared with simplePacking, intelliPacking significantly improves the packing ratio. This is because intelliPacking dynamically estimates traffic pattern and schedules packet transmissions so that the degree of in-network packet packing is improved.

As a result of the improved in-network packet packing, intelliPacking also improves energy efficiency in delivering information, as shown in Figure 6 Compared with noPacking, intelli-



(a) Number of transmissions



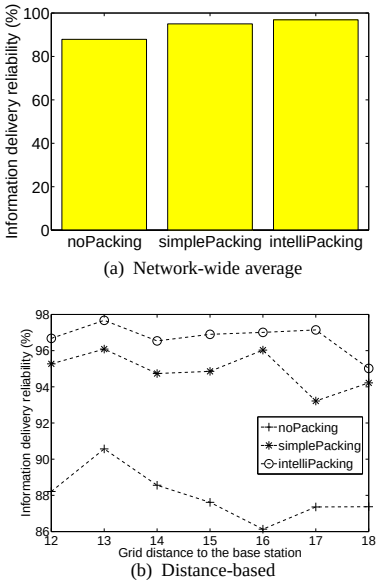
(b) Number of receptions

Figure 6: Average number of transmissions and receptions per information unit received

Packing reduces the average number of transmissions and receptions required for delivering an information unit by a factor of 2.33 and 2.35 respectively; compared with simplePacking, intelliPacking also reduces the average number of transmissions and

receptions required for delivering an information unit by a factor of 1.59 and 1.56 respectively.

Since intelliPacking reduces the number of packet transmissions, it reduces the degree of channel contention in the network and thus improves reliability in delivering information, as shown in Figure 7 which presents the network-wide average information delivery reliability, as well as the average reliability based on the distance (measured in grid-hops) from the source to the base station. Compared with noPacking and simplePacking, intelliPacking improves reliability by 8.98% and 1.87% respectively. (We will see a little bit later that the improvement in information delivery reliability is even much higher in real-world hardware based experiments.)



(a) Network-wide average

(b) Distance-based

Figure 7: Information delivery reliability

Impact of maximum allowable latency. To study the impact of application properties on intelliPacking, we vary the maximum allowable latency in information delivery from 3 seconds to 25 seconds and measure the corresponding performance of intelliPacking.

Figure 8 shows how the packing ratio increases as the maximum allowable information delivery latency increases. As the allowable latency increases from 5 seconds to 25 seconds, the packing ratio increases from 1.09 to 3.17 and by a factor of 2.9.

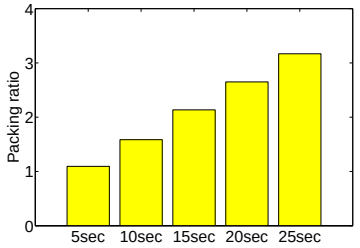
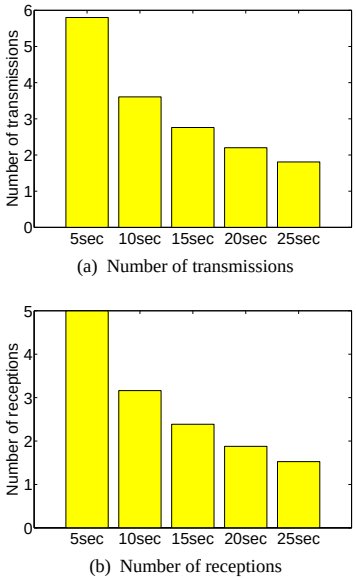


Figure 8: Packing ratio

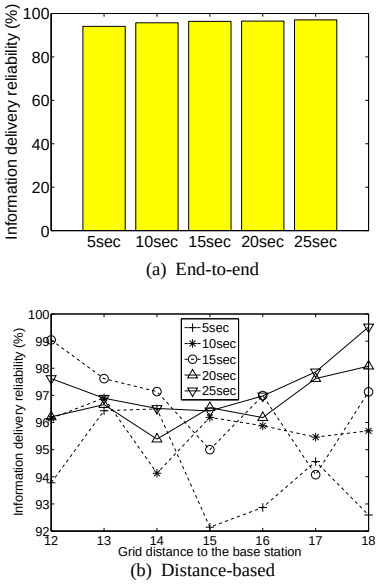
As the packing ratio increases, the energy efficiency increases by a factor up to 3.27, as shown in Figure 9. In the mean time, the messaging reliability increases by a factor up to 3% as the channel contention decreases due to increased packing ratio. This is shown in Figure 10.



(a) Number of transmissions

(b) Number of receptions

Figure 9: Average number of transmissions and receptions per information unit received



(a) End-to-end

(b) Distance-based

Figure 10: Information delivery reliability

Impact of maximum payload length. Since the maximum packet payload length determines the maximum number of information units that can be packed into a single packet, it affects the degree of in-network packet packing. Setting the maximum information delivery latency to the typical value of 10 seconds, we vary the maximum payload length from 40 bytes to 104 bytes and measure the corresponding performance of intelliPacking. Figures 11, 12, and 13 present the data on packing ratio, energy efficiency, and reliability respectively. From these figures, we see that increased maximum payload length improves the over-

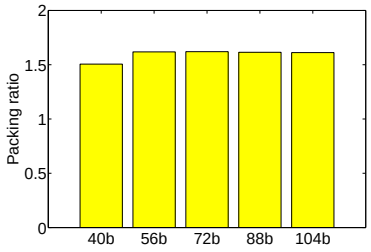
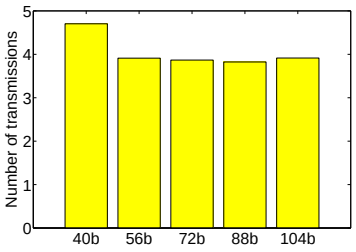
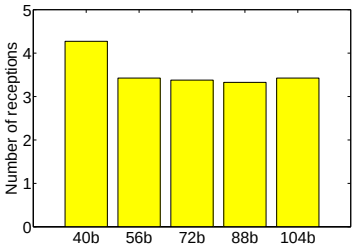


Figure 11: Packing ratio

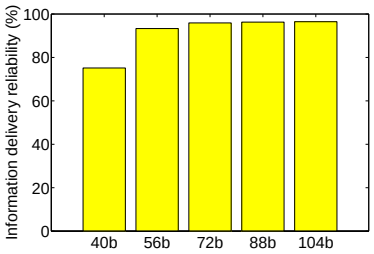


(a) Number of transmissions

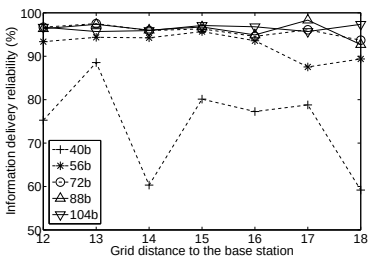


(b) Number of receptions

Figure 12: Average number of transmissions and receptions per information unit received



(a) End-to-end



(b) Distance-based

Figure 13: Information delivery reliability

all network performance, even though the improvement is not prominent given the tight information delivery latency (which in turns bounds from above the packing ratio). Note that, when the maximum payload length is 40 bytes, each packet can carry at most 2 information units, and a packet is immediately transmitted or forwarded after each packing. As a result, compared with scenarios of longer payload length, packet transmissions are more bursty (and less spread temporally) when the maximum payload length is 40 bytes, and thus the messaging reliability is relatively lower.

Having studied the benefits and the influence factors of application-adaptive in-network packet packing in simulation, we corroborate our findings via experimentation in the next subsection.

4.2 Experimental study

To understand the performance of the different messaging methods in real-world environment, we evaluate their performance via experimentation with Tmote Sky sensor nodes. These sensor nodes use CC2420 radios which are compatible with IEEE 802.15.4 standard. We deploy 18 Tmote Sky sensor nodes in a 3×6 grid, with every two closest nodes separated by 1.5 feet. The sensor grid is placed in an office environment as shown in Figure 14. By experimenting with real-world radios and environment, we can capture the impact of channel fading and channel contention, as well as the impact of temporal link properties (which did not discuss in Section 3.1).

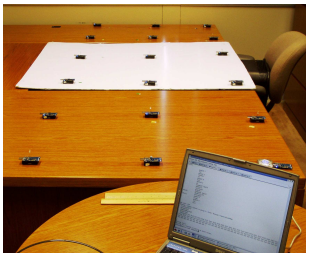


Figure 14: Tmote Sky sensor node grid

We set the transmission power level of the sensor nodes to be 2 (out of a range from 1 to 31) such that every node can reliably communicates with its immediate grid-neighbors. Similar to the typical scenario studied in simulation, the base station is at one corner of the grid, and nodes in the farthest 3×3 subgrid from the base station periodically generate information units, with the inter-unit interval uniformly distributed between 5 seconds and 15 seconds. The length of each information unit is 16 bytes, the maximum payload length is 102 bytes, and the maximum allowable information delivery latency is 10 seconds.

Figure 18 shows the packing ratio in different messaging methods. Compared with noPacking and simplePacking, intelliPacking improves the packing ratio by a factor of 5.25 and 3.5 respectively.

Accordingly, intelliPacking significantly improves the energy efficiency, as shown in Figure 19. Compared with noPacking and simplePacking, intelliPacking reduces the number of transmissions required for delivering an information unit by a factor of

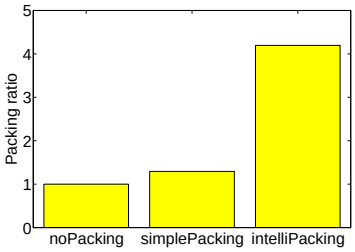
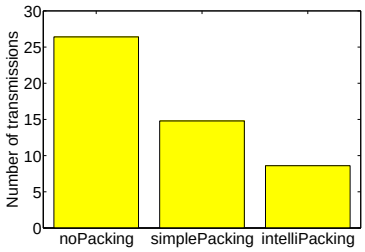
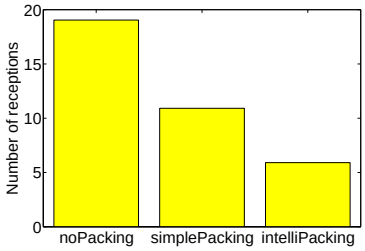


Figure 15: Packing ratio

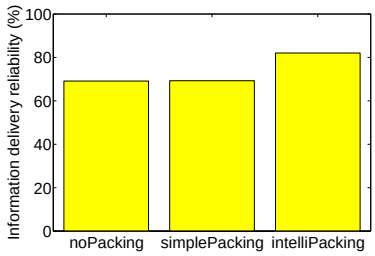


(a) Number of transmissions

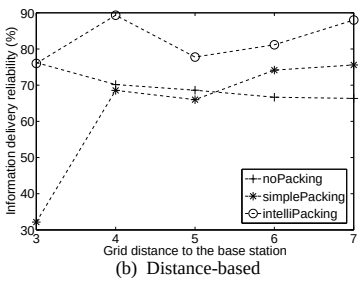


(b) Number of receptions

Figure 16: Average number of transmissions and receptions per information unit received



(a) End-to-end



(b) Distance-based

Figure 17: Information delivery reliability

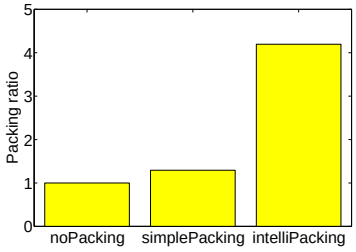
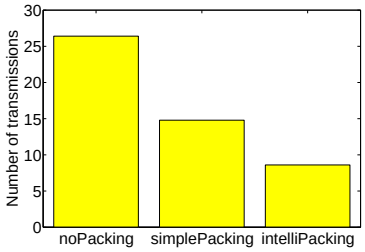
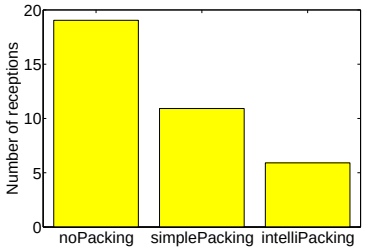


Figure 18: Packing ratio



(a) Number of transmissions



(b) Number of receptions

Figure 19: Average number of transmissions and receptions per information unit received

3.07 and 1.71 respectively, and intelliPacking reduces the number of receptions per information unit received by a factor of 3.22 and 1.85 respectively.

Because intelliPacking reduces the number of packet transmissions in the network, it reduces the degree of channel contention. Accordingly, it improves messaging reliability as shown in Figure 17. Compared with noPacking and simplePacking, intelliPacking improves the average messaging reliability by 12.92% and 12.77% respectively.

From the above study, we see that the experiments corroborate our observations in simulation, even strengthening the observations by showing higher degree of improvement in packing ratio, energy efficiency, and information delivery reliability.

5 Related work

In the Internet, the concepts of Application Oriented Networking (AON) [3] and Application-driven Networking [19, 15] have been being explored to enable coordination among disparate applications, to enforce application-specific policies, to improve visibility of information flow, and to enhance application optimization and QoS. While these concepts are generic enough to be applied to wireless sensor networks, the techniques employed

in and the problems faced by the Internet are quite different from those in sensor networks, due to the differences in both technologies and application domains. For instance, the extreme resource (e.g., computation, communication, and energy) constraints are unique to sensor networks and are not the major issues in the Internet.

For customized resource provisioning to different applications, Darwin [7] has been proposed for run-time resource management in the Internet. To match application requirements to communication protocols, DANCE [36] has been proposed to provide a service-oriented view to communication services, thus to avoid the inflexibility that results from a fixed binding between an application and a specific protocol stack. Active networks [38] have also been proposed to enable functionalities such as application-specific multicast, information fusion, and other services leveraging network-based computing and storage. Sensornet protocol SP [35] provides a unifying link layer abstraction for sensornets. Focusing on the interface between link and higher layers, SP is complementary to our focus on application-adaptivity and higher layer architecture issues. While [35] focuses on designing the single narrow waist to support all kinds of higher layer protocols, it is not our purpose to argue that there exists a single higher layer messaging architecture. Yet we believe it is still desirable to identify the common architecture for typical communication patterns (e.g., convergecast and broadcast), and our work in this paper focuses on the architecture for application-adaptive messaging from the perspective of convergecast.

Woo et al. [42] discussed networking support for query processing in sensor networks. Issues such as query-oriented routing, efficient rendezvous for storage and correlation, and unified in-network system have been discussed. While focusing on query processing, [42] does not concentrate on the architectural and algorithmic issues to support a broader range of applications such as distributed signal processing and computing.

To adapt communication protocols to changing network conditions and application requirements, Impala [27] used Adaptation Finite State Machine (AFSM) to control the adaptation of communication protocols. To provide application-specific QoS in ubiquitous environments, Nahrstedt et al. [31] proposed a framework for QoS specification and compilation, QoS setup, and QoS adaptation. Our work complements those in [27] and [31] by focusing on issues such as application-adaptive link estimation, structuring, and scheduling which are autonomous without human in the loop.

Mechanisms have been proposed in [9] and [26] for directing data queries to where information is via information-directed routing. [46] has proposed using data traffic itself to estimate wireless link qualities so that routes can be chosen according to the changing network conditions when applications change. Our work complements [26] and [46] by considering the architectural issues in application-adaptive messaging as well as the algorithmic issues in application-adaptive structuring and scheduling.

Query processing in sensornets has drawn a lot of attention recently [28, 44, 29, 32, 12, 11, 25, 17]. TinyDB [28] and Cougar [44] are two exemplary sensornet database systems which regard data collection as a database query process and then design mechanisms (such as semantic query forwarding and in-network aggregation) to execute the query efficiently. Mechanisms for efficient and robust in-network aggregation for query processing have also been proposed in [29] and [32]. Nonetheless, existing work in sensornet query processing has not focused on the QoS requirement of different applications, nor did they focus on the generic application-adaptive messaging in sensornets.

Unlike query-oriented data modeling and data processing, another aspect in modeling data in sensor networks is to investigate the correlation among them and then take advantage of the correlation in reducing the cost of data collection [23, 34, 33]. To this end, [23], [34], and [33] studied the problem of finding the best aggregation tree given the data sources and the correlation structure between the data sources. Our work complements [23], [34], and [33] by not assuming the knowledge of data sources and their correlations, such that the algorithms are more generically applicable. Also, we study the general architecture for application-adaptive messaging, which is not the focus of the above work. [14] studied scheduling issues in structure-free data aggregation, which is complementary to our focus on structure based messaging.

As a simple form of data aggregation, packet packing has also been studied in [20] and [47], where several short packets are packed into a long data packet if they meet at some node. As they focus on the impact of *simplePacking*, however, [20] and [47] did not study the problem of adapting packing policies to application QoS requirements. Nagle's algorithm [16] is also used in TCP to pack short data segments into longer ones, but it was not designed to be application-adaptive either.

6 Concluding remarks

We have identified the common components of application-adaptive messaging in sensornets, and accordingly proposed an architecture SMA that adopts two levels of abstraction: traffic-adaptive link estimation and routing at the lower level, and application-adaptive structuring and scheduling at the higher level. Taking packet packing as an example of in-network processing, we studied application-adaptive scheduling in detail. Based on the concept of *scheduling utility*, the algorithmic framework for packing-oriented scheduling is generically applicable to other in-network processing methods. Through simulation and experimentation, we have shown that our design improves both the energy efficiency and the reliability in sensornet messaging.

While we have validated SMA and application-adaptive scheduling from the perspective of packet packing, we believe our effort is only the first step toward the unified architecture for application-adaptive messaging in sensornets. As applications evolve, we hope to enrich our design by taking into account other in-network processing methods (e.g., information fusion) and application requirements (e.g., packet delivery reliability), and by studying the effectiveness of SMA for typical communication patterns and application scenarios in sensornets. Another important issue in sensornets is power management, and it has significant implications to the design of sensornet architecture and algorithms. Detailed study of this aspect is a part of the future work too.

Acknowledgment

We thank Oleg Andric, Bogdan Carbunar, Judy Fu, Steve Gilbert, Chen Jia, Jason LeBrun, Nitya Narasimhan, Daniel Stewart, Bryan Thale, Venu Vasudevan, and Yang Yu for their comments on this paper.

References

[1] TinyOS. http://www.tinyos.net/.

[2] Tmote sky sensor node. http://www.moteiv.com/.

[3] Application-oriented networking. http://www.cisco.com/, 2005.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks (Elsevier)*, 38(4):393–422, 2002.

[5] A. Arora and et al. A Line in the Sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks (Elsevier)*, 46(5), 2004.

[6] A. Arora, R. Ramnath, E. Ertin, P. Sinha, S. Bapat, V. Naik, V. Kulathumani, H. Zhang, H. Cao, M. Sridhara, S. Kumar, N. Seddon, C. Anderson, T. Herman, N. Trivedi, C. Zhang, M. Gouda, Y. R. Choi, M. Nesterenko, R. Shah, S. Kulkarni, M. Aramugam, L. Wang, D. Culler, P. Dutta, C. Sharp, G. Tolle, M. Grimmer, B. Ferriera, and K. Parker. Exscal: Elements of an extrem scale wireless sensor network. In *IEEE RTCSA*, 2005.

[7] P. Chandra, Y. hua Chu, A. Fisher, J. Gao, C. Kosak, T. E. Ng, P. Steenkiste, E. Takahashi, and H. Zhang. Darwin: Customizable resource management for value-added network services. In *ICNP*, 1998.

[8] Y.-R. Choi, M. Gouda, H. Zhang, and A. Arora. Stabilization of grid routing in sensor networks. *AIAA Journal of Aerospace Computing, Information, and Communication*, to appear.

[9] M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. Technical Report P2001-10113, PARC, 2001.

[10] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *ACM MobiCom*, pages 134–146, 2003.

[11] A. Deshpande, C. Guestrin, W. Hong, and S. Madden. Exploiting correlated attributes in acquisitional query processing. Technical report, Intel Research - Berkeley, 2004.

[12] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, 2004.

[13] C. T. Ee and R. Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *ACM SenSys*, pages 148–161, 2004.

[14] K.-W. Fan, S. Liu, and P. Sinha. On the potential of structure-free data aggregation in sensor networks. In *IEEE INFOCOM*, 2006.

[15] J. Follows and D. Straeten. *Application-Driven Networking: Concepts and Architecture for Policy-based Systems*. IBM, December 1999.

[16] M. G. Gouda. *Elements of Network Protocol Design*. John Wiley and Sons, 1998.

[17] J. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Toward sophisticated sensing with queries. In *IPSN*, 2003.

[18] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating congestion in wireless sensor networks. In *ACM SenSys*, pages 134–147, 2004.

[19] IBM. *Application Driven Networking: Class of Service in IP, Ethernet and ATM Networks*. IBM, August 1999.

[20] A. Jain, M. Gruteser, M. Neufeld, and D. Grunwald. Benefits of packet aggregation in ad-hoc wireless network. Technical Report CU-CS-960-03, University of Colorado at Boulder, August 2003.

[21] V. Kawadia and P. R. Kumar. Principles and protocols for power control in ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 23(5):76–88, 2005.

[22] A. Konrad, B. Zhao, and A. Joseph. A markov-based channel model algorithm for wireless networks. *Wireless Networks*, 9:189–199, 2003.

[23] R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul, and U. Ramachandran. DFuse: A framework for distributed data fusion. In *ACM SenSys*, 2003.

[24] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *ACM SenSys*, pages 126–137, 2003.

[25] X. Li, Y. J. Kim, R. Govindan, and W. Hong. Multi-dimensional range queries in sensor networks. In *ACM SenSys*, 2003.

[26] J. Liu, F. Zhao, and D. Petrovic. Information-directed routing in ad hoc sensor networks. In *ACM WSNA*, 2003.

[27] T. Liu and M. Martonosi. Impala: A middleware system for managing autonomic, parallel sensor systems. In *ACM PPoPP*, 2003.

[28] S. Madden, M. Franklin, and J. Hellerstein. TinyDB: An acquisitional query processing system for sensor systems. In *ACM Transactions on Database Systems*, 2004.

[29] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[30] M. Maroti. The directed flood routing framework. In *Technical report, Vanderbilt University, ISIS-04-502*, 2004.

[31] K. Nahrstedt, D. Xu, D. Wichadakul, and B. Li. Qos-aware middleware for ubiquitous and heterogeneous environments. *IEEE Communications Magazine*, 2001.

[32] S. Nath, P. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *ACM SenSys*, 2004.

[33] D. Petrove, R. Shah, K. Ramchandran, and J. Rabaey. Data funneling: Routing with aggregation and compression for wireless sensor networks. In *ICC Worshops*, 2003.

[34] P. Pietzuch. Path optimization in stream-based overlay networks. Technical Report TR-26-04, Harvard University, 2004.

[35] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica. A unifying link abstraction for wireless sensor networks. In *ACM SenSys*, pages 76–89, 2005.

[36] B. Reuther, D. Henrici, and M. Hillenbrand. DANCE: Dynamic application oriented network services. In *EUROMICRO*, 2004.

[37] L. Rittle, V. Vasudevan, N. Narasimhan, and C. Jia. MUSE: Middleware for using sensors effectively. In *INSS*, 2005.

[38] D. L. Tennenhouse and D. J. Wetherall. Towards an ative network architecture. *Computer Communication Review*, 26(2), 1996.

[39] C. Wan, S. Eisenman, and A. Campbell. CODA: Congestion detection and avoidance in sensor networks. In *ACM SenSys*, pages 266–279, 2003.

[40] H. S. Wang and N. Moayeri. Finite-state markov channel - a useful model for radio communication channels. *IEEE Transactions on Vehicular Technology*, 44(1):163–171, 1995.

[41] A. Willig. A new class of packet- and bit-level models for wireless channels. In *IEEE PIMRC*, 2002.

[42] A. Woo, S. Madden, and R. Govindan. Networking support for query processing in sensor networks. *Communications of the ACM*, 47(6):47–52, 2004.

[43] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *ACM SenSys*, pages 14–27, 2003.

[44] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. In *ACM SIGMOD*, 2002.

[45] H. Zhang, A. Arora, Y. R. Choi, and M. Gouda. Reliable bursty convergecast in wireless sensor networks. In *ACM MobiHoc*, 2005.

[46] H. Zhang, A. Arora, and P. Sinha. Learn on the fly: Data-driven link estimation and routing in sensor network backbones. In *IEEE INFOCOM*, 2006.

[47] Y. Zhang and Q. Huang. Coordinated convergecast in wireless sensor networks. In *IEEE Milcom*, 2005.

[48] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *ACM SenSys*, pages 1–13, 2003.