Flow Control Mechanisms for the Bundle Protocol in IEEE 802.15.4 Low-power Networks

Wolf-Bastian Pöttner and Lars Wolf
Institute of Operating Systems and Computer Networks
Technische Universität Braunschweig, Braunschweig, Germany
[poettner|wolf]@ibr.cs.tu-bs.de

ABSTRACT

Wireless Sensor Networks (WSN) are networks formed by wireless sensor nodes that generally feature a low-power microcontroller and an IEEE 802.15.4 radio. Most Delay Tolerant Wireless Sensor Networks (DTWSN) in literature use proprietary protocols that are specifically designed for a single purpose. In this paper we explore, how the Bundle Protocol (BP) can be used on top of the IEEE 802.15.4 PHY and MAC layers, while avoiding overhead for additional layers in between. To pursue this, a Convergence Layer (CL) has to realize certain tasks that usually are dealt with in layers 3 and 4 of the OSI 7-layer protocol stack. We argue that flow control has to be an integral ingredient of the CL and compare the performance of four different mechanisms using our BP implementation for Contiki.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Store and forward networks*

Keywords

Bundle Protocol, DTWSN, WSN, IEEE 802.15.4, Flow Control, Convergence Layer, low-power

1. INTRODUCTION

Delay Tolerant Networks (DTNs) has been accepted by the networking community as a way to deal with intermittently connected networks. Although a number of different protocols and solutions exist, the Bundle Protocol (BP) [10] can be considered the de-facto standard for delay tolerant communication. For different underlying network technologies, the BP uses so-called Convergence Layers (CLs) to safely transport bundles from one host to another in a single hop.

According to the BP specification, the duties of a CL are sending bundles to a subset of nodes within reach and delivering incoming bundles to the BP agent, whereas each CL is specific for a network technology. Several CL specifications and implementations for various network technologies exist. A prominent example is the TCP Convergence Layer (TCPCL) [1] based on TCP/IP.

IEEE 802.15.4 [3] is a Physical (PHY)- and Medium Access Control (MAC)-layer standard for Wireless Personal Area Networks (WPANs) which is designed to be energy efficient and enables wireless communication of low-power embedded devices with up to 250 kBit/s. While IEEE 802.15.4 communication stacks for standard PCs are still in the early stages most wireless sensor nodes (T-Mote Sky [7], INGA, etc.) feature an IEEE 802.15.4-compliant transceiver.

For efficiency reasons, we aim at using a CL to bridge the gap between the IEEE 802.15.4 PHY- and MAC layer and the BP agent while avoiding layers 3 and 4 of the protocol stack. To function correctly, the CL has to realize certain features that are normally located on these layers. In this paper, we focus on flow control mechanisms for hop-by-hop bundle transmission between two neighbouring nodes.

Such flow control mechanism has to make sure, that fast senders do not overrun slow receivers, while maintaining a high throughput and a low energy consumption at the same time. While a high throughput allows to make efficient use of potentially short contacts, a low energy consumption is important for long battery life. This means, that unnecessary transmissions and retransmissions should be avoided and that the signaling overhead has to be kept as low as possible. Finally, the mechanism must not contradict the IEEE 802.15.4 standard and should be practically implementable on today's off-the-shelf hardware.

In the remainder of this paper, we show our contribution in the following structure: We begin with related research in Section 2 and outline our motivation and background in Section 3. We present four flow control mechanisms in Section 4 and compare them using experimental results in Section 5. Finally, we conclude the paper in the Section 6.

2. RELATED WORK

Several publications have used DTN techniques in Wireless Sensor Networks (WSN) context, without using the standardized BP. ZebraNet [4] aims at tracking wildlife in Kenya, Seal-2-Seal [5] tracks contacts between wild animals and LUSTER [11] aims at monitoring environmental parameters to be used by ecologists. All those projects use a proprietary delay tolerant communication protocol and demonstrate the need for delay tolerant communication in WSNs.

Two published DTN implementations for WSN nodes exist. DTNLite [8] implements DTN concepts on TinyOS, but does not use the BP. ContikiDTN [6] uses the BP over TCPCL in IEEE 802.15.4 networks. To the best of our knowledge, no existing solution for using the BP in IEEE

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CHANTS'12, August 22, 2012, Istanbul, Turkey.
Copyright 2012 ACM 978-1-4503-1284-4/12/08 ...$15.00.

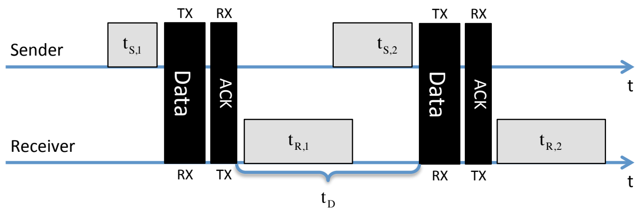


Figure 1: Timing of two nodes exchanging bundles.

802.15.4 networks without the need for existing layers 3 and 4 is available. The DTN2 reference implementation includes an Ethernet-based CL, that enables wired communication between neighbouring nodes without using IP and/or TCP. DTN2 also includes a Bluetooth-based CL and an AX.25-based CL that both avoid using existing layers 3 and 4.

From our literature study we conclude, there is a need for delay-tolerant communication in WSNs. We also see, that approaches to use the BP over IEEE 802.15.4 network exist. However, those approaches rely on TCP/IP for communication, which incurs a high overhead both in terms of communication but also program memory on the nodes. While the BP has already been used directly on top of layer 2, this was not the case for IEEE 802.15.4-based networks.

3. BACKGROUND

While IEEE 802.15.4 transceivers can be attached to a multitude of hardware platforms, we orient ourselves on the lowest common denominator. In most cases, these are the microcontroller platforms which are the basis for popular WSN nodes such as the Tmote Sky. Those systems feature a CPU with 4 - 16 MHz clock, 10 - 16 kB RAM, 48 - 128 kB program memory and an IEEE 802.15.4 transceiver. Such nodes are usually battery-powered, so that energy is scarce and has to be preserved.

We assume, that bundles are stored in persistent flash memory. Many WSN nodes are equipped with flash memory and use flash file systems to store multiple files onto the flash. Flash access produces certain delays, that are hardware dependent. The flash chip on the T-Mote sky takes $0.5 - 5\ ms$ to write a flash page and $\ll 1ms$ to read a page. Hence, we assume that the time to write a bundle is varying over time and dependent on the size of a bundle and is orders of magnitude longer than the time to read a bundle. Additional overhead is introduced by flash file systems.

The IEEE 802.15.4 standard uses CSMA-CA for medium access. Furthermore, the standard uses stop-and-wait for frame transmissions between two nodes with explicit Acknowledgement (ACK) frames for each data frame. The specification defines different Inter Frame Spaces (IFSs) depending on the length of a data frame to give the receiver appropriate time to process the frame. Since the standard does not include measures for flow control, a sender has to assume that the receiver has processed incoming frames after the IFS has elapsed. However, with a short IFS of 192 μs and the long IFS of 640 μs, this is orders of magnitude from the time that is needed to store a bundle into flash. To overcome this problem, the receiver might buffer bundles in RAM and use idle times to write the cache into flash. However, with RAM sizes in the order of some kB, few bundles can be buffered. These points make it clear, that a flow control mechanism for the IEEE 802.15.4 CL is necessary.

In the remainder of the paper we use the term bundle as placeholder for information that is transferred over the IEEE 802.15.4 link, whereas the same techniques are applicable to fragments of larger bundles.

For our approach we assume, that nodes can decide on a per-frame basis, whether a link-layer ACK should be sent or not. Furthermore, we assume, that nodes can piggyback data onto link-layer ACKs at no cost. These are reasonable assumptions as we have shown in [9].

3.1 Problem Statement

In Figure 1 we see one node sending two bundles to another node. The sender node takes a certain time $t_{S,1}$ to read the outgoing bundle from storage, do internal processing and to copy the data into the buffer of the radio. Afterwards, the MAC layer decides when to actually transmit the frame. If packet transmission is successful, the receiver sends a link-layer ACK frame and starts to read the bundle from the radio buffer to store it into the local storage within time $t_{R,1}$. In this example, the sender sends a second bundle after a certain interval, whereas the time between end of the previous ACK transmission and the beginning of the next data frame is denoted as t_D. As mentioned earlier, we have to assume that the processing and storage time at the sender t_S as well as the respective time at the receiver t_R is variable over time.

In order to give the receiver enough time to process incoming segments, the following conditions must hold: $t_D \geq t_R$ and $t_D \geq t_S$. Altogether, the best throughput can be achieved when $t_{D,i} = \max(t_{R,i}, t_{S,i+1})$. The goal of flow control is to minimize t_D given the stated constraints. A sender is able to predict or measure its own delay t_S, but information about t_R is needed to adapt t_D and to achieve optimal throughput.

4. FLOW CONTROL MECHANISMS

In this section, we discuss four flow control algorithms to achieve the stated goals.

4.1 Fixed Delay per Neighbour

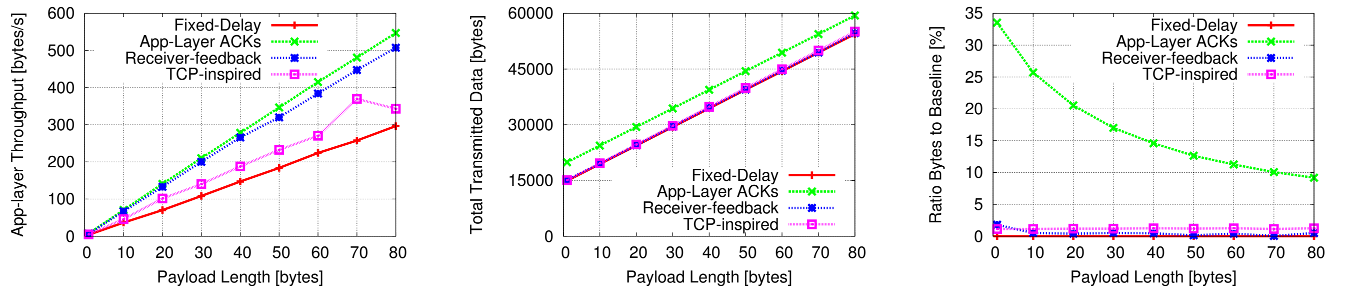
The sender uses a fixed t_D for each receiver and sends segments at a constant rate. This approach has a fixed and limited throughput that is determined by the choice of t_D. However, with a t_D that is high enough, this approach is the optimal approach in terms of energy consumption, because only data segments are transmitted. Due to the time variability of t_R, the delay has to be chosen in a conservative way to be able to handle worst-case situations: $t_D = \max(t_{D,i})$.

4.2 Application-Layer ACKs

This mechanism uses a dual-ACK strategy on the link- and the application-layer. On the link-layer, the sender sends out a frame and waits for the link-layer ACK to signal successful reception of the frame. If no such ACK is received within 864 μs (defined in the IEEE 802.15.4 specification), the sender can quickly retransmit the frame without loosing too much time. When the receiver has successfully processed the contents of the frame, an additional application-layer ACK frame is sent to trigger the sender to transmit the next data segment. Optionally, the receiver can transmit a NACK frame to reject a segment and to trigger the sender to transmit the next data segment. We expect this approach to achieve a significant throughput improvement compared to the previous approach but also significantly increased energy consumption due to higher signaling overhead.

4.3 Receiver-Feedback

In this approach, the sender sends a segment to the receiver. The receiver then estimates the time t_R it will take to process and store this bundle and piggybacks the estimation onto the link-layer ACK that is sent to the sender. To



(a) Application-Layer throughput when exchanging 500 bundles with varying bundle payload sizes and flow control approaches.

(b) Total amount of transferred bytes (in both directions) for the exchange of 500 bundles for varying bundle payload sizes and flow control approaches.

(c) Total amount of transferred bytes (in both directions) compared to the fixed-delay approach.

Figure 2: Performance Evaluation of Flow Control approaches.

be able to piggyback information onto the link-layer ACKs, those have to be generated in software. Furthermore, the sender estimates its own processing time t_S for the next bundle and starts preparing transmission of the next segment at $T + t_R - t_S$, whereas T is the current time. This approach does not need additional signaling overhead but allows for variable times of t_R and t_S. It is expected to realize the highest throughput, whereas the energy consumption should be comparable to the first approach, since no additional bytes are transmitted.

4.4 TCP-inspired

The approach is inspired by the TCP congestion control, in which the rate is increased as long as link-layer ACK frames are received and decreased if ACKs are missing to detect network overload. The receiver node has to selectively acknowledge incoming frames to make sure, that all acknowledged frames will be processed and to avoid overrunning internal buffers.

We use an initial interval t_D and a threshold t_{th}. The sender sends out a frame towards the receiver. If a link-layer ACK is received, the sender multiplies t_D with factor f_D in case that t_D is above the threshold t_{th}. Otherwise, t_D is decremented for every link-layer ACK. If one link-layer ACK is lost, t_{th} is set to the current t_D multiplied with the factor f_L and t_D is set to the new threshold t_{th}.

We expect, that this approach will expose a slightly higher energy consumption than the first, because a certain amount of frames have to be retransmitted when t_D is decreased too far. Furthermore, we expect a lower throughput compared to the previous approach due to frames that will be lost whenever t_D goes below t_R.

5. EVALUATION

In order to evaluate the suitability of the flow control approaches for the IEEE 802.15.4 CL, we measure the performance and the power consumption of the algorithms on real nodes. Performance is mainly expressed in application-layer throughput, but also in the number of transmitted (and received) bundles per second. Furthermore, potentially lost bundles are also relevant to ensure reliable data delivery.

5.1 Setup

In this evaluation, we have used two TMote Sky sensor nodes running Contiki OS.The sender sends 500 bundles to a receiver over the wireless link using the IEEE 802.15.4 frame format and PHY and Clear Channel Assessment (CCA) to determine if the channel is free. We have chosen this simple MAC to provide general results, that are not specific for a particular MAC. We assume, that the results likewise apply to other MACs. To exclude influences of the environment, all presented results are the arithmetic mean of 4 measurement runs. We have measured using varying bundle payload sizes between 1 and 80 bytes, whereas the actual size of the radio frame depends on the BP overhead and the variable length of the Self-Delimiting Numeric Values (SDNVs).

We have implemented the flow control mechanisms for our BP implementation μDTN, including software-generated ACKs using the CC2420 radio chip. Furthermore, the nodes use Contiki's COFFEE [12] for persistent storage. For the feedback-based flow control approach, sender and receiver have measured their respective processing time per bundle and used an exponentially weighted moving average $t_{R,i} = \alpha \cdot t_{R,new} + (1 - \alpha) \cdot t_{R,i-1}$ with an α of 0.4 to calculate the expected time t_R for the following bundles. We have empirically determined the value for α in several experiments. Our estimation approach is simplified but acceptable for our experiment, since the major contributor to t_R is the storage component which exposes a linearly rising write time per bundle.

5.2 Metrics

We measure the throughput at the receiver ignoring lost or duplicate bundles and we only count the payload of bundles to obtain application-layer throughput values. For the power consumption, we have used Contiki's software-based Energy Estimation [2] mechanism to verify a strong correlation between transmit power consumption and transmitted bytes. We therefore measure transmitted bytes and only count bytes that are related to bundle transmission and ignore other frames such as discovery. We count the full IEEE 802.15.4 frame length but we exclude the IEEE 802.15.4 link-layer ACK frames, which would add another 3 bytes to each unicast transmission.

5.3 Performance Results

In general, none of the presented approaches permanently lost frames. In Figure 2a we see the application-layer throughput that can be achieved when exchanging bundles with the presented flow control approaches. The fixed-delay approach has the lowest throughput with a constant linear increase for larger payload sizes. This is caused by the conservative choice of $t_D = 250ms$, that was necessary to cope even with worst-case situations. The TCP-inspired approach reaches a higher throughput, that is less stable and shows a minor anomaly for bundles with 70 bytes payload. We have used an initial $t_D = 250ms$ with a threshold of $t_{th} = 100ms$. The decrease factor was $f_D = 0.5$ and the factor used to determine the new threshold after a missing ACK was $f_L = 2$. Below the threshold, t_D was decremented by 0.244 ms per ACK. The receiver-feedback approach also shows a linearly increasing throughput, while the application-layer ACK based approach reaches the highest throughput of all.

In general, the results of the different approaches are close to our expectations. The steady linear increase of all throughput measurements is caused by the fact, that the rate of bundles per second is independent of the bundle payload size. With a fixed rate of bundles, increasing payload sizes lead to increased throughput. This phenomenon is caused by the file system performance. We have measured between 1.4 and 8.3 ms for opening an existing file and between 8.5 and 225.8 ms for creating a new file. Since the receiver node has to create a new file for every incoming bundle, COFFEEs performance effectively limits the achievable throughput. The time for writing and reading bytes is negligible compared to those times. This also explains, why the overall throughput seems low compared to the 250 kBit/s of IEEE 802.15.4.

Based on Section 4.3 we have expected the receiver-feedback approach to expose the highest throughput of all. Upon further investigation, we have found two reasons, why the throughput is slightly lower than the application-layer ACK approach. On the one hand, the estimation of t_R has to be as accurate as possible to allow back to back packet transmissions. Our EWMA approach is always a bit behind reality and looses some frames in critical situations that have to be retransmitted. On the other hand, the feedback from the receiver to the sender has to be accurate. Unfortunately, the feedback mechanism allows only one byte to be send back, which is not enough for a high resolution timestamp. Those two factors combined lead to the throughput that is behind our expectations.

5.4 Power Consumption Results

As a representative for the power consumption we have measured the transmitted bytes as shown in Figure 2b. As expected, the fixed-delay approach transmits the lowest number of bytes and can be seen as the baseline here. However, also the TCP-inspired and the receiver-feedback approach show a similar number of transmitted bytes. This means, that both approaches do not introduce additional signaling overhead and avoid retransmitting bundles. While this is expected for the receiver-feedback approach, it is surprising for the TCP-inspired version. The application-layer ACK approach shows a constant offset in the amount of transmitted bytes due to the additional application-layer ACK frames that are transmitted.

In Figure 2c we show the overhead ratio of the four approaches compared to the fixed-delay version (normalized to 0 %). We see here, that in fact the receiver-feedback approach only has marginal additional overhead, whereas the TCP-inspired approach is slightly higher. This was expected, since at some point the TCP approach looses segments, since t_D has been decreased too far. The application-layer ACK approach shows a decreasing overhead ratio, that is caused by the fact, that the amount of user data is increased while the overhead stays constant.

6. CONCLUSIONS

In this paper we have introduced and compared flow control approaches for a BP CL for IEEE 802.15.4-based wireless networks. We have compared the transmitted bytes (as representative for power consumption) and throughput to avoid overrunning slow receivers.

The TCP-inspired flow control approach has a good power consumption but only mediocre throughput results. The application-layer ACK approach has the highest throughput, but also the highest power consumption. The receiver-feedback approach is the best trade-off between power and throughput because the power consumption is similar to the baseline, while the performance is almost similar to the application-layer ACK approach.

Nevertheless, the receiver-feedback and the TCP-inspired approach have to be implemented for each radio chip in a platform specific way. Therefore, we propose to use the application-layer ACK approach as default and the receiver-feedback approach for systems that support it.

7. REFERENCES

[1] M. Demmer and J. Ott. Delay Tolerant Networking TCP Convergence Layer Protocol. *IETF Draft*, 2008.

[2] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proc. of EmNets '07*, pages 28–32, New York, NY, USA, 2007. ACM.

[3] IEEE. 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). *IEEE Standard for Information technology*, 2007.

[4] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. *SIGOPS Oper. Syst. Rev.*, 36(5):96–107, Oct. 2002.

[5] A. Lindgren, C. Mascolo, M. Lonergan, and B. McConnell. Seal-2-Seal: A delay-tolerant protocol for contact logging in wildlife monitoring sensor networks. In *Proc. of MASS 2008*, 29 2008-oct. 2 2008.

[6] M. Loubser. Delay Tolerant Networking for Sensor Networks. Master's thesis, Swedish Institute of Computer Science, 2005.

[7] Moteiv Corporation. Tmote Sky : Datasheet, 2006. `http://www.snm.ethz.ch/pub/uploads/Projects/tmote_sky_datasheet.pdf`.

[8] R. Patra and S. Nedevschi. DTNLite: A Reliable Data Transfer Architecture for Sensor Networks. Technical Report CS294–1, Berkeley, 2003.

[9] W.-B. Pöttner, S. Schildt, D. Meyer, and L. C. Wolf. Piggy-Backing link quality measurements to IEEE 802.15.4 acknowledgements. In *Proc. of WiSARN-FALL 2011*, Valencia, Spain, October 2011.

[10] K. Scott and S. Burleigh. Bundle Protocol Specification. RFC 5050 (Experimental), Nov. 2007.

[11] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter. LUSTER: wireless sensor network for environmental research. In *Proc. of SenSys '07*, pages 103–116. ACM, 2007.

[12] N. Tsiftes, A. Dunkels, Z. He, and T. Voigt. Enabling Large-Scale Storage in Sensor Networks with the Coffee File System. In *Proc. of IPSN 2009*, San Francisco, USA, Apr. 2009.