Data Elevators: Applying the Bundle Protocol in Delay Tolerant Wireless Sensor Networks

Wolf-Bastian Pöttner, Felix Büsching, Georg von Zengen, and Lars Wolf
Technische Universität Braunschweig
Institute of Operating Systems and Computer Networks (IBR)
Mühlenpfordtstr. 23, 38106 Braunschweig
Email: [poettner | buesching | vonzeng | wolf]@ibr.cs.tu-bs.de

***Abstract*—Delay Tolerant Networking (DTN) enables transfer of data where conventional network protocols fail to deliver data because no continuous end-to-end connectivity is available. While the Bundle Protocol (BP) has been established as the standard DTN protocol in many application areas, Wireless Sensor Networks (WSN) often use proprietary protocols with a subset of the BP features.**

In this paper we use an exemplary application to demonstrate how the BP can be beneficial for many WSN-based projects. We show, how low-power sensor nodes can transport bundles by exploiting existing movement in the environment. More importantly we show, how 8-bit WSN nodes can seamlessly interact with standard BP implementations running on standard PCs.

Our application, a sensor that is installed on the roof of a 15-story building, is using an elevator to transport bundles carrying measured values to our lab. We analytically compare the BP to existing protocols for WSNs, evaluate our application scenario and give insight into principal limitations.

I. Introduction

In many areas of communication, a protocol that is designed specifically for a single purpose is better suited for that single application than a more general standardized protocol. Especially in extremely resource-constrained networks of sensor nodes, this has led to the development of a multitude of different communication protocols and specialized mechanisms that are all well suited for only a small number of applications. In contrast, we use the Internet Protocol (IP) in conjunction with TCP or UDP for almost all applications in the Internet, mainly because these protocols are "good enough" and the effort of developing a specialized solution is not justified by the benefits of such.

In general, standard protocols have the disadvantage that they are not optimized for a specific use-case and, therefore, have a higher overhead compared to specialized protocols. On the upside however, using standard protocols significantly lowers the entry barrier to set up a network, because implementations are readily available. That also allows seamless integration of sensor networks using standard protocols into existing back-end networks, since protocol stacks for those systems exist as well. Furthermore, standard protocols usually have already solved common problems and researchers can start with application design, without having to implement the next purpose-built communication protocol.

As of today, IP can be considered the de facto standard for communication in Wireless Sensor Networks (WSN) when seamless integration with existing backend networks (that are usually based on IP) is required [1]. However, a significant amount of application scenarios for WSNs do not offer continuous end-to-end connectivity, but have to rely on store-and-forward network techniques to deal with intermittent connectivity. Furthermore, our experience has shown, that in many applications, the WSN is not isolated, but data is transported to central point(s) and fed into back-end systems. Delay tolerant networks (DTN) have been widely accepted as a way to deal with intermittent connectivity and huge delays. The Bundle Protocol (BP) can be considered the de facto standard for DTNs [2].

In this paper, we claim that the BP should be the de facto standard for delay tolerant communication not only in DTNs, but also in Delay Tolerant Wireless Sensor Networks (DT-WSN). While this incurs manageable overhead (see Section III-B), it also allows for seamless integration with existing BP-based DTNs (see Section III-A). We show, that tiny sensor nodes can handle the processing burden and that the communication overhead is in the same dimension as existing, non-delay-tolerant protocols. We further show our Data Elevator application in which we have used our BP implementation for common sensor node platforms in conjunction with a BP implementation for PCs to monitor the outside temperature. Data Elevators is a store-carry-and-forward scenario, in which an elevator is used as a data mule to transport data from the roof into our laboratory. We evaluate our application scenario and provide insight into the network capacity that can be achieved using an elevator. Finally, we argue that the same implementation can be beneficial for existing DTN projects in the literature.

The technical key contributions of this paper are:

- A BP Convergence Layer for IEEE 802.15.4-based networks that negates the need for an overlay network.
- A hardware independent design of μDTN, our BP implementation, for heterogeneous wireless sensor networks.
- A BP implementation for 8- and 16-bit Microcontrollers.
- Interoperability with BP-compliant standard DTN implementations for PCs.
- Network capacity model for DT-WSNs.
- Evaluation of the Data Elevator use case.

The remainder of the paper is structured as follows: In section II related publications regarding DTN in general, application scenarios for DTN in WSNs, and existing related implementations is given. Section III outlines the general benefits of the BP and discusses why it is also suitable for WSNs from both, a technical and an application driven, points of view. Our implementation of the BP for WSNs is briefly explained in section IV. In section V the "Data Elevator", an application for Disruption Tolerant Networking is presented and evaluated in section VI. We conclude the paper in section VII.

II. Related Work

In this section we discuss related research efforts. We first concentrate on conceptual publications within the scope of DT-WSNs. We then look at application scenarios, which require Delay Tolerant Networking and could benefit from our work. Finally, we discuss implementations of DTN concepts for sensor nodes and explain why our approach is different.

A. Conceptual Work

Shah et al. [3] exploit mobility of objects to transport data in sparse sensor networks by using the objects as data mules. The paper introduces a three-tier architecture in which fixed sensor nodes sample data, mobile mules transport data and fixed access points with WAN connectivity receive data and transport it into a data warehouse. Fall [4] has published the idea of delay tolerant networking as a way to deal with intermittent connectivity. The paper states, that sensor networks use a yet-to-be-standardized sensor transport protocol and it implies that communication with sensor network devices will be handled over a serial port. The poster [5] outlines, how the concepts of DTN can be beneficial for WSNs. Special emphasis is put on the fact, that a standardized architecture is necessary to avoid a separate implementation of a subset of DTN features for each individual application. Furthermore, the poster proposes to use the usual bundle layer stack on sensor nodes, thereby forming an overlay network and using existing routing and transport protocols. Cerf et al. [6] have published a specification for the architecture of DTNs. This document can be considered the basis of many DTN solutions, especially the BP that implements a significant set of the specified functionality. In 2007, Scott and Burleigh [7] have published the "Bundle Protocol Specification" as a standard protocol to be used in DTNs. While the document specifies data formats and various other protocol details, the BP is intended to be used as an overlay network on top of existing protocol stacks. The RFC explicitly lists "Sensor network nodes" as implementation target but states that routing is towards a hard-coded default route and that protocol logic should be realized in ASICs.

We conclude, that a variety of different DTN solutions exist. However, all of them either see the BP as an application-layer protocol and form an overlay network or do not use a standardized protocol at all. In contrast, our approach is to apply the standardized BP in WSNs and avoid the overhead of existing layers 3 and 4.

B. Application Scenarios

As a multitude of application scenarios for DT-WSNs exist we only give an outline of some of the major contributions as examples for a whole class of use cases.

The aim of ZebraNet [8] is to track wildlife in Kenya by attaching GPS-equipped nodes to animals. Nodes wirelessly exchange data to allow researches to recover as many information pieces as possible by discovering only a subset of the animals. The authors state that delay of data is irrelevant but a high delivery rate of samples is important. The goal of SeNDT [9] is to monitor water quality and noise emission using sensor nodes and to use a store-and-forward network to collect data. SeNDT uses an implementation of the Licklider Transmission Protocol (LTP) on top of an IP stack and is implemented for the Linux operating system. LUSTER [10] aims at monitoring environmental parameters to be used by ecologists and uses a multi-layer approach with two separate elements related to our work. In the sensor network, LUSTER uses an overhearing-based logging system in which log nodes overhear radio transmission and save data to persistent local storage. Furthermore, LUSTER uses a DTN approach for backend connectivity over intermittent links, which allows a back-end to query nodes for data that was lost during the downtime of the link. Seal-2-Seal [11] is a delay tolerant protocol to log contacts between animals in the wild. It has been implemented for Contiki and evaluated in simulations. Vineyard Computing [12] presents a WSN, in which sensors on a vineyard periodically sample temperature and data mules (i.e. workers, dogs, etc.) carry nodes to collect the data.

From this overview we learn, that the necessity for delay tolerant communication in WSNs exists. However, all of the presented approaches are either based on non-standardized protocols that are purpose-built or are implemented using significantly more powerful PCs running the Linux operating system.

C. DTN Implementations

For computers running the Linux operating system, a number of different open source BP implementations such as IBR-DTN [13], DTN2[1] and ION[2] exist. While the individual goals and specific features differ, they all share in common that the target platform has to be significantly more powerful than a tiny sensor node.

DTNLite [14] is an implementation of the concepts of DTN for TinyOS. Data is stored in non-volatile memory on nodes and is transported in a store-and-forward manner towards a single sink, while custody transfer is used to support end-to-end reliability. The implementation supports convergence layers and relies on the Matchbox file system to store data on nodes. DTNLite forms an overlay network and relies on existing multi-hop routing protocols, but does not implement

[1] http://sourceforge.net/projects/dtn/
[2] https://ion.ocp.ohiou.edu/

the BP. Furthermore, the authors assume that all packets share the same destination.

ContikiDTN [15] is an implementation of the BP for Contiki. The authors claim interoperability with the DTN2 reference implementation and ContikiDTN implements the TCP Convergence Layer using Contiki's uIP stack. The author has provided simulated evaluation of ContikiDTN, but source code is not available. In contrast to our approach, ContikiDTN forms a BP overlay network and hence carries the overhead of using the BP in a TCP data stream.

SCAR [16] is a context-aware routing protocol for delay-tolerant store-and-forward sensor networks. Nodes distribute sampled information to neighbors that are considered the "best carriers" for information towards the sink, taking into account location, mobility and battery level. While SCAR is designed to transport general-purpose application data, it does not use a standardized protocol and can be considered a routing protocol in terms of BP.

III. Bundle Protocol in IEEE 802.15.4-based Wireless Networks

In this section we provide and overview over the technical features of the BP in general, outline the communication and computational overhead and present our network capacity model for DT-WSNs.

A. Technical Perspective

The BP is an address-centric, message-based experimental protocol for environments with intermittent connectivity, high bit-error rates and large delays. Designed to form an application-layer overlay network over existing internets, it enables end-to-end connectivity where no continuous end-to-end connection between two endpoints ever exists. The protocol data units (PDU) are called bundles and the protocol supports hop-by-hop and end-to-end reliable data transmission using custody transfer. Bundles have a lifetime and will be deleted by all nodes after expiration. Furthermore, BP stores bundles on nodes when they can currently not be routed, making use of scheduled or opportunistic contacts. Bundles can consist of a Primary Bundle Block (PBB, i.e. a header), a Bundle Payload Block and multiple extension blocks. Although the BP is designed to form an overlay network, bundles still have to be routed from node to node. While routing is an essential part of a DTN, there is no single routing protocol associated with the BP. As for MANETs, a multitude of routing approaches exist that are compatible with the BP.

The BP is well suited to be used in resource-constrained DT-WSNs, because the protocol specification is very flexible and allows to be used in a variety of use cases. Furthermore, binary header fields are encoded using Self-Delimiting Numeric Values (SDNV) that are of variable length and always as small as possible (on a per-byte basis) for their current content. To be able to communicate over various underlying network technologies, BP uses so-called Convergence Layers (CL). CLs are responsible to send and receive bundles to and from nodes within range, while the communication on the lower-layers is dependent on the employed protocols. To avoid the overhead of addressing nodes via string-based Endpoint Identifiers (EID), Compressed Bundle Header Encoding [17] (CBHE) has been specified. CBHE addresses nodes via SDNVs and is well suited for sensor networks, because the length of the addressing fields adapts to the address of the node.

The strongest assumption of the BP is, that all nodes have access to accurate real world time. While this may be true for traditional DTNs, wireless sensor nodes typically do not have synchronized clocks. The Bundle Age Block [18] can be used in situations in which nodes to not have access to a timing source and allows to track the age of bundles and expire them, after the lifetime has elapsed.

In the remainder of this paper, we use the term BP for the BP using bundles with a CBHE-encoded Primary Bundle Block and including the Bundle Age Block.

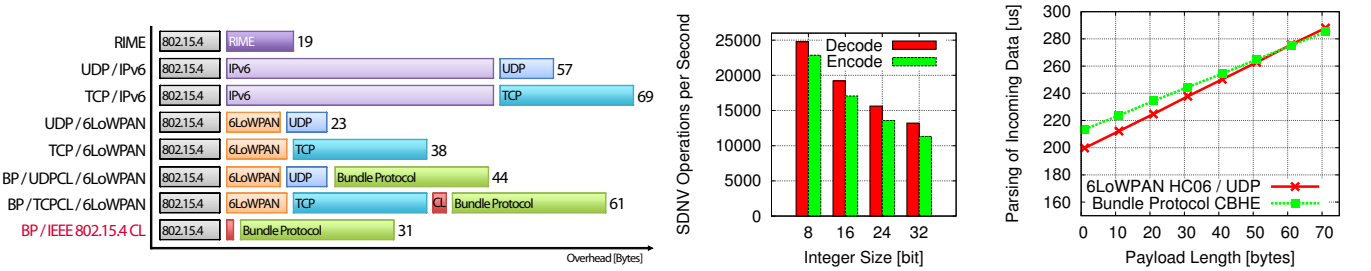
B. Communication Overhead

In many WSNs, IEEE 802.15.4 [19] is used for wireless communication, offering a maximum frame length of 127 bytes. Since each transmitted bit costs precious energy, minimizing the overhead of communication protocols saves energy and storage and decreases the amount of PDU that have to be fragmented.

In order to compare the overhead of different communication protocols for WSNs, we have looked at a typical use case. We assume unicast communication between two nodes over multiple hops using the IEEE 802.15.4 short addressing mode. We assume payloads that do not require fragmentation of PDUs and neglect overhead produced by routing protocols. While we originally intended to compare the overhead of existing DTN solutions for WSNs, we were unable to get ahold of source code and/or specification documents. Consequently, we are comparing commonly used WSN protocols that are freely available.

Figure 1a shows a comparison of the header length for various communication protocols in WSNs, whereas header length translates into protocol overhead in the scope of this comparison. With the settings mentioned above, IEEE 802.15.4 has a fixed header length of 9 bytes. Since all protocols mentioned in the following reside inside an IEEE 802.15.4 frame, this overhead applies to all those protocols. The RIME communication stack is an isolated solution, being available only in Contiki, thus, it is listed here as baseline for sensor network protocols and does not offer backend connectivity. In multi-hop mode, RIME has a protocol header of 19 bytes including the IEEE 802.15.4 frame.

Uncompressed IPv6 [20] has a fixed header length of 40 bytes and while UDP adds another 8 bytes of overhead TCP adds 20 bytes. 6LoWPAN [1] allows to compress the IPv6 and UDP header depending on the employed features and compression mechanism. Using the IPHC compression method, the length of the IPv6 and UDP header can be reduced to 14 bytes in total for our scenario plus 9 bytes for the IEEE 802.15.4 header. Since no header compression for TCP headers



(a) Protocol Overhead (Less is Better).

(b) SDNV En- and Decoding Operations on INGA per Second (More is Better).

(c) Time to parse one incoming PDU (Less is Better).

Figure 1: Computational Overhead of the Bundle Protocol compared to 6LoWPAN.

is specified in 6LoWPAN, the total length of 6LoWPAN, TCP and IEEE 802.15.4 header is 38 bytes.

The length of the Primary Bundle Block and the Payload Block Header in the BP using CBHE is 21 bytes. When bundles are transferred via UDP or TCP in 6LoWPAN, the total length of headers including the IEEE 802.15.4 header is 44 bytes for UDP and 61 bytes for TCP CL. When transmitting bundles inside IEEE 802.15.4 frames without using 6LoWPAN or Rime, a 1 byte header for multiplexing has to be added. This brings the total header length for BP inside IEEE 802.15.4 frames to 31 bytes.

Based on these numbers, we can see that the combination of UDP / 6LoWPAN offers the lowest overhead of protocols with backend connectivity. Nevertheless, BP with the IEEE 802.15.4 CL has a slightly higher overhead but offers additional functionality in the light of network disruptions. While application-layer DTN using UDP over 6LoWPAN is very well possible, this would incur additional overhead. Furthermore, header compression techniques similar to those in 6LoWPAN can bring down the overhead of the BP to comparable numbers.

C. Computational Overhead

Full-featured protocols from the PC world are usually complex and require significant effort to parse on tiny nodes. While IP and 6LoWPAN have been widely accepted as being manageable with today's nodes, the BP poses new computational challenges. For example, header fields are encoded as SDNVs, which require extensive bit shift operations to reconstruct the original number. Furthermore, the header is of variable length and parsing is not as straightforward as for the original IP specification. Since the speed of parsing protocol messages is implementation dependent, we show exemplary, that the BP is manageable to today's sensor nodes.

To evaluate the overhead of SDNV encoding and decoding, we have used INGA [21] nodes to measure how long the respective operations take. While our implementation is not optimized yet, the numbers are given here just to show an impression of the speed. Figure 1b shows the number of encode and decode operations that can be performed per second. As expected, the duration depends on the length of the en- or decoded data, whereas longer numbers take longer to process. We can see, that between 11316 and 24774 operations can be performed per second. Even when processing all 16 SDNVs of a bundle, this would still allow for 707 bundles per second, which is more than IEEE 802.15.4 can handle.

Furthermore, we have compared the time it takes to parse an incoming PDU. This time is important, since it limits the maximum throughput that can be achieved. We have compared 6LoWPAN/UDP in Contiki's default settings using HC06 header compression with CBHE encoded BP. Again, we want to give an impression of the speed. Figure 1c shows the time it takes to process incoming PDUs of varying payload sizes. We can see here, that 6LoWPAN is slightly faster than BP for smaller payload sizes but reaches the same speed for larger payloads.

We can conclude that today's sensor nodes are able to handle the BP and that from a computational perspective BP is comparable to 6LoWPAN.

D. Network Capacity Model

Implementation details aside, the capacity of a DTN mainly depends on the minimum capacity of all involved links. The capacity of a link in a opportunistic DTN is a function of the duration and frequency of contacts between nodes and the throughput. Since the amount of data that can be exchanged between nodes usually is fixed per time interval, longer contacts mean more exchanged data. Furthermore, a higher number of contacts also increases the capacity, because with more contacts, data can be exchanged more often. In effect, the data path between individual hops of a DTN can be modeled as a bottleneck that can only transmit a certain amount of bundles per time. Our model consists of a sender and a receiver node with limited storages and certain link capacities and is expressed in the following paragraph.

$$C_{i,j} = Contacts_i \cdot Duration_i \cdot BundleRate_i \quad (1)$$

$$T_i = min(S_{Send,i}, C_{i,j}) \quad (2)$$

$$S_{Send,i} = min(S_{Send,i-1} - T_{i-1} + N_i, S_{Cap,Send}) \quad (3)$$

$$S_{Recv,i} = min(S_{Recv,i-1} + T_i, S_{Cap,Recv} \quad (4)$$

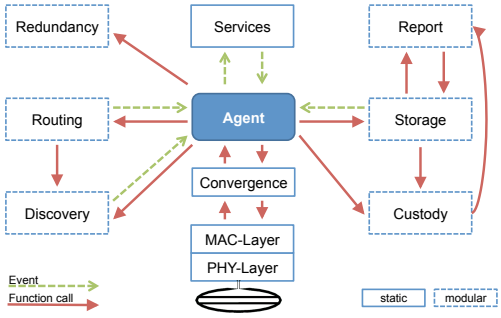


Figure 2: Block diagram of μDTN's architecture.

The capacity $C_{i,j}$ of a link j in equation (1) is the total number of bundles that can be transferred in time interval i. $C_{i,j}$ can be calculated based on the number and duration of contacts as well as the rate at which bundles can be exchanged. The amount of bundles that can be transferred (T_i in equation (2)) in time interval i is the minimum of the link capacity and the number of bundles waiting in the storage of the sender $S_{Send,i}$. $S_{Send,i}$ in equation (3) can be calculated using $S_{Send,i-1}$, the bundles that have been transmitted during the last interval T_{i-1} and the bundles that have been created in the current interval N_i. Since the storage of the sender is limited, $S_{Send,i}$ is the minimum of the theoretical amount of bundles in storage and the real storage capacity $S_{Cap,Send}$. The bundles in the storage of the receiver $S_{Recv,i}$ in equation (4) depend on $S_{Recv,i-1}$ and the number of transmitted bundles T_i, but cannot be higher than the storage capacity of the receiver $S_{Cap,Recv}$.

The presented model can be used to dimension a DT-WSN will be shown in Section VI-B. It provides insight into the amount of storage that is necessary to support a specific sample interval assuming a certain contact pattern. Furthermore, it allows dimensioning the network in a way, that the expected bundle loss stays within application requirements.

IV. Bundle Protocol Implementation for WSNs

To use the BP in WSNs, we need respective implementations for a sensor network platform on the one hand and for a PC BP implementation on the other hand.

A. μDTN: Bundle Protocol Implementation for Contiki

While the overall concept applies for any 8- or 16- bit embedded system, our implementation μDTN is designed and implemented for Contiki OS which supports a significant number of hardware platforms. The implementation has been successfully tested on INGA and TelosB, two completely different hardware architectures. μDTN uses a modular architecture which allows configuring different implementations of the various components during compile time. This makes the design flexible and efficient at the same time to cope with the tightly limited resources of wireless sensor nodes. During the design and implementation process, we have given memory efficiency a higher priority than the actual speed because RAM is a limited resource on most of the WSN target platforms.

μDTN is split up into different modules as shown in Figure 2. The "Agent" is the central entity and cannot be exchanged while other modules may have different implementations.

1) Discovery Module: The discovery module is responsible to discover neighboring nodes within range. We have adapted IP Neighbor Discovery [22] (IPND) because it is flexible and standardized. The discovery module periodically sends out beacon frames. Upon reception of a beacon from another node, the agent is notified. Nodes time out after 5 consecutive beacons have been lost.

2) Routing Module: The routing module is responsible to decide which bundles should be forwarded to which neighbor in range. While the selection of a routing protocol is application-dependent, we have implemented a modified version of flooding because it is robust and reliable. The module creates n copies of a bundle and sends those to all neighboring nodes except the originator of a bundle, the neighbor from which the bundle has been received, and nodes to which the bundle has already been delivered.

3) Storage Module: The storage module is responsible to store a bundle on a local medium and to read back the bundle upon request by the agent. μDTN can use the flash memory of nodes to persistently store bundles or store bundles in RAM. While the former allows bundles to survive periods in which nodes are switched off, the latter is faster and allows higher throughputs.

4) IEEE 802.15.4 Convergence Layer: The BP Specification demands that the BP is used as an overlay network, thereby relying on existing layers 1 to 4. This allows to bypass nodes that are not equipped with a BP implementation, but also incurs a significant overhead as seen in Section III-B. Since we assume that all nodes in a WSN run the same software, we insert a 1 byte header and transmit bundles directly inside the payload of IEEE 802.15.4 MAC frames. The header consists of 2 bits for dispatching, 2 bits for multiplexing and 4 bits for segment sequence numbers. To handle bundles larger than a single frame, our approach is to segment those bundles and reassemble them hop-by-hop.

Due to space limitations, a detailed description of μDTN, including all modules and a substantial general evaluation has to be omitted here. More details can be found in [23].

B. IEEE 802.15.4 Convergence Layer for IBR-DTN

To allow for seamless backend connectivity of our BP-based DT-WSN, we need a BP implementation for PCs using the same CL. Hence, we have implemented a CL for IBR-DTN using the linux-zigbee[3] IEEE 802.15.4 stack. This stack supports various hardware platforms, whereas we have used the *Ben WPAN*[4] USB stick on x86 machines and iMote 2 nodes running Linux.

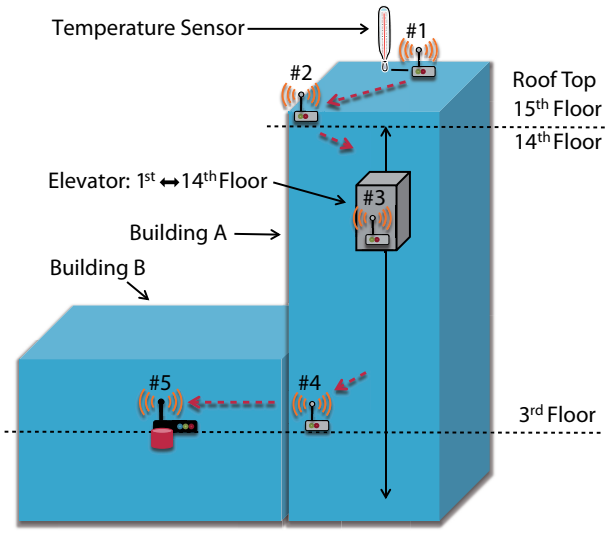


Figure 3: ”Data Elevator“ use case: Temperature data from the rooftop is transported wirelessly and physical via a multi-hop delay tolerant wireless sensor network (DT-WSN), utilizing an elevator as data mule to bridge the gap between 15th and 3rd floor.

V. Application Scenario Description

We claim that many DT-WSN scenarios would benefit from our solution to apply the BP to WSNs. To give an example of an application, which can be evaluated with reasonable effort, we implemented the following exemplary use case.

The Data Elevator is a monitoring scenario in which we measure the ambient temperature on the rooftop of a 15-story university building. The temperature data is intended to be further processed and displayed in our laboratory which is located in the 3rd floor of an adjacent building. For such monitoring applications, installing cables through the building is out of the question and single-hop radio communication through 13 ceilings of armored concrete using low-power radios is impossible. Installing antennas on the outer face of the building may be possible, but comes at high costs and significant effort. Thus, we use a multi-hop DT-WSN to transport data from the roof utilizing the movement of an elevator.

Since air temperature usually changes slowly, a sampling interval of 5 minutes is adequate. No real-time transmission of sensor readings is necessary, since temperature readings are mainly needed for statistics. While even significant delays are acceptable, 100 % of the sensor readings have to reach the sink to have a complete data set to allow short- and long-term trend analysis.

To implement the monitoring scenario, we have installed 5 nodes as indicated in Figure 3. On the rooftop of building A we have node #1 which is equipped with a temperature sensor and sampling data every 5 minutes. Nodes #2 to #4 are data relays,

[3]http://sourceforge.net/apps/trac/linux-zigbee/
[4]http://en.qi-hardware.com/wiki/Ben_WPAN

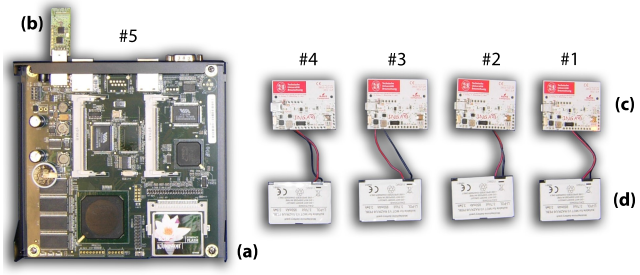


Figure 4: Hardware setup for the real world evaluation of the ”Data Elevator“ scenario: An ALIX board (a) with an IEEE802.15.4 ATUSB stick (b) acts as sink. Four *INGA* nodes (c) with external LiPo batteries (d) form the DT-WSN.

located on the 15th floor near the staircase, in the elevator cabin and on the 3rd floor next to the elevator. Node #5 is the sink running on an x86 PC. It is located in our laboratory on the 3rd floor in the adjacent building B. Incoming bundles are written to a database, from where the data is further processed and visualized.

We have used flood routing (see Section IV-A2) in this experiment, where the intuitive data path is from node #1 via nodes #2, #3, and #4 to node #5. However, on several instances we have observed direct communication between nodes #1 and #3, and nodes #3 and #5.

A. Implementation of the Data Elevator

The heterogeneous system setup consists of an *ALIX 2D2*[5] system board, and several INGA sensor nodes as to be seen in figure 4. The ALIX board is an x86 system (500 MHz AMD Geode LX800 CPU), equipped with 256 MByte SDRAM, running IBR-DTN on Debian Linux and acts as sink node in our experiment. While the connection to the backbone network is established via one of the integrated Ethernet ports, the wireless sensor network is tethered through a *Ben WPAN* IEEE 802.15.4 stick, based on an ATmega and communicating via Atmels AT86RF231 radio transceiver.

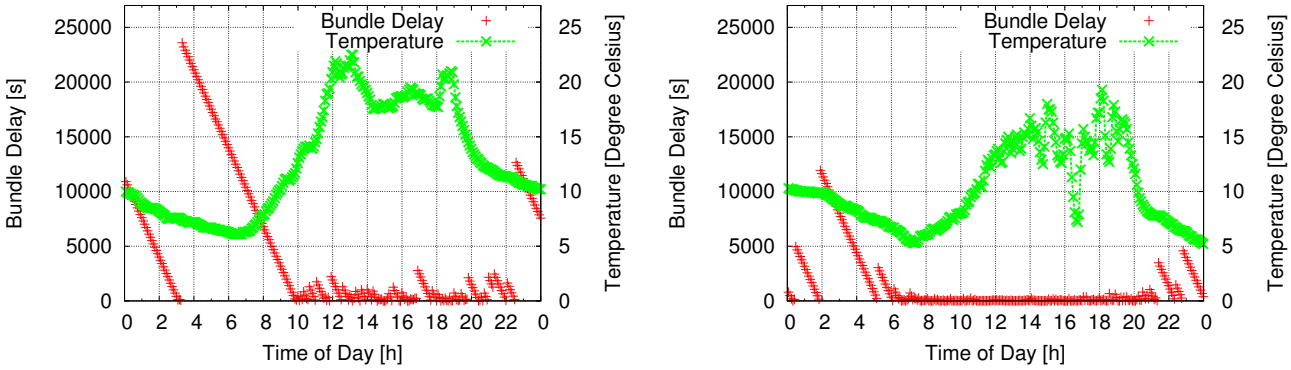
The remaining part of the WSN consists of four INGA nodes, which are also based on Atmels 8-bit ATmega architecture and communicate via an AT86RF231 radio transceiver. All INGA nodes are powered by rechargable LiPo batteries. Besides other sensors, INGA comes with a digital onboard pressure sensor (Bosch BMP085), which integrates a precise temperature sensor (16 bit resolution) we utilized for the temperature measurements on the rooftop.

The temperature of node #1 is sampled every 5 minutes; for each measurement a bundle of 9 bytes is generated that is addressed towards sink node #5. Nodes #2 to #4 act as DTN nodes and mainly store and forward bundles. The RAM-based storage module can hold up to 100 bundles of measurement data.

B. Additional Concepts

In scenarios with a higher data rate or long lasting disruptions the storage memory (either RAM or flash) may become

[5]http://pcengines.ch/alix2d2.htm/



(a) Weekend (b) Workday

Figure 5: Ambient Temperature and Bundle Delay over Time of Day on 2012-04-14 (a) and 2012-04-16 (b).

a limiting factor. In this case, the sampling rate should be adapted to the remaining storage space to avoid the loss of data. Furthermore, application specific changes in sampling rate are possible that reduce the rate, for example, over night, when less samples may be needed. Additionally, the overall system behavior (including radio duty-cycling and discovery) can change controlled by battery level or remaining system energy. Many scenarios transport data to the sink and then run processing algorithms to extract relevant pieces of information. For such scenarios, in-network data aggregation can help to distribute processing in the network and to reduce the amount of bundles that are transported.

VI. Evaluation

To evaluate the performance and potential of BP-based DTWSNs, we have performed an experimental evaluation. On the one hand, we want to assess what performance we can achieve in the Data Elevator scenario. According to our application requirements, performance primarily means bundle delivery rate and bundle delay. On the other hand, we want to see what the potential of such a data mule scenario is. Therefore, we have looked at the network capacity in the second part of the evaluation.

We deployed the system as described in the previous section. The sensor on the rooftop was covered in a waterproof box that was sometimes (depending on solar altitude) directly exposed to the sun. After system startup we performed a manual time synchronization with the accuracy of about 1 second by generating a time-sync bundle containing the local timestamp of the node. In the current configuration, Contiki is using INGA's internal oscillator and not the more precise external clock crystal for its clock, which may induce a drift of the nodes clocks. To compensate for that possible drift we generate a second time-sync bundle at the end of the measurements and calculated the offset, which is subtracted in all presented results.

To gather data regarding the contact time and duration every node sends additional statistical bundles of 46 byte every 4 hours. We evaluated the system for two whole days (24 hours each) and present the evaluation data for a weekend day (Saturday, 2012-04-14) and a working day (Monday, 2012-04-16). As expected, all bundles that have been generated were successfully delivered to the sink node.

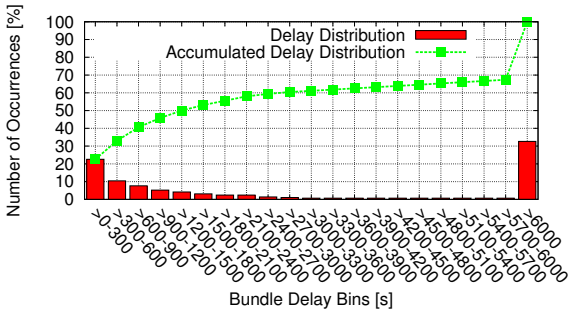
A. Evaluation of the Data Elevator

In Figure 5 temperature and bundle delay are displayed over the time of the day of the two 24-hour measurements. Bundle delay is defined as the time between generation of a bundle (i.e. sampling temperature) and arrival at the sink. The assumption that a sampling interval of 5 minutes is sufficient for ambient outside temperature data is evident. Due to the fact that there is only one working elevator for the whole building, which is in use nearly non-stop on a working day during daytime, it is also not surprising that the delay at this time of the day is low. More interesting regarding the capability to handle huge delays is the nighttime on weekends, where delays of more than 6 hours (21 600 s) occur.

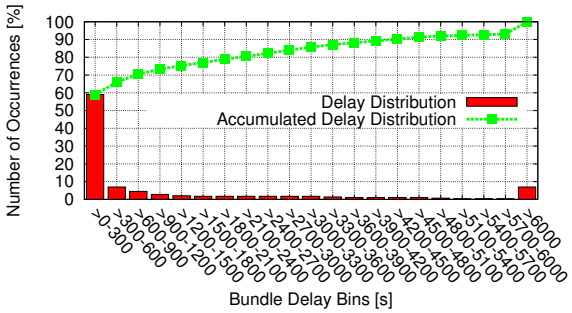
The maximum bundle delay is 11 930 s (3.3 hours) on the workday and 23 591 s (6.6 hours) on the weekend day. In the interval between 8 am and 8 pm, the maximum delays are 667 s during the workday and 6492 s on the weekend. On the weekend, we have observed a total number of 80 undelivered bundles (including statistical bundles) being stored in RAM of Node #2. With a storage capacity of 100 bundles, this is only 20 bundles short of overrunning buffers.

Figure 6 shows a histogram of the bundle delay and the cumulative distribution function (CDF) for both evaluation days. The figure shows, that on the working day 59.03 % of all bundles have been delivered within 5 minutes. A total of 27.09 % of the bundles experienced delays between 300 and 6000 s, whereas 6.94 % of the bundles were delayed by more than 6000 s. For the weekend day, 50 % of the bundles have been delivered within 1500 s of their creation. However, 32.64 % of the bundles experienced a delay of more than 6000 s.

Bundle delay is a function of the movement of the elevator. On the weekend, the elevator is used less frequently and the bundle delay is increased significantly. Capacity of bundle



(a) Weekend



(b) Workday

Figure 6: Bundle Delay Histogram and CDF on 2012-04-14 (a) and 2012-04-16 (b).

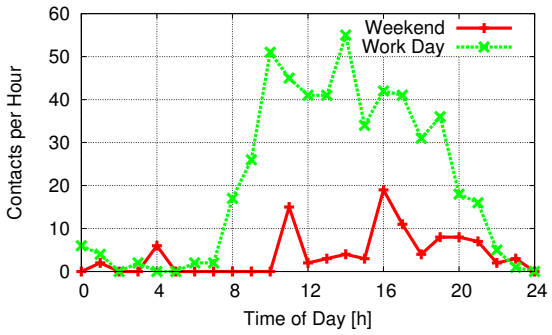


Figure 7: Elevator activity derived from the number of contacts between node #2 and #3 (15th floor).

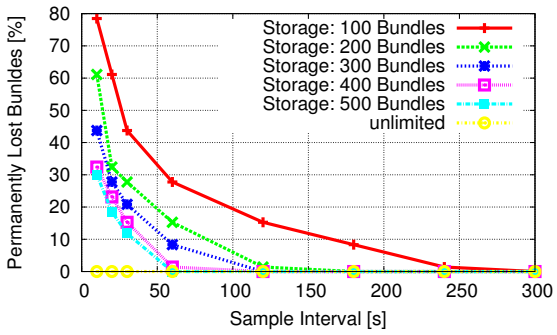


Figure 8: Calculated permanently lost bundles for varying sample intervals and storage capacities on the weekend.

storage on nodes has to be provisioned for the worst-case delays that will happen, otherwise bundles may get lost.

In Figure 7 the activity of the elevator of both evaluation days is shown. This data was derived from the statistical bundles that recorded the number of contacts per hour between the elevator node and node #2 in the 15th floor. As the elevator usually is located at the lower levels of the building due to its programming to return to the first floor after idling for a certain time, the link in the 15th floor is the bottleneck in the communication path. The recorded frequency distribution correlates with the delivery delay observed in Figure 5; as expected the highest frequency occurs during daytime on a working day and the lowest at night on a weekend. We also observed that the duration of a contact had no influence on the success of forwarding bundles, as all contacts were long enough to transmit the complete storage in all cases.

We can conclude, that reliably transporting temperature readings from the rooftop using the elevator works as expected. During our experiment, 100 % of the bundles have been delivered to the sink using the multi-hop DT-WSN.

B. Evaluation of Network Capacity

In Section III-D we have introduced our model for the capacity of a DTN. The model depends on the number and duration of contacts, the throughput per time interval, the bundle generation frequency and the storage capacities. To evaluate the capacity of our Data Elevator scenario, we have modeled the network as a 4 hop (5 node) network. With μDTN we have measured a hop-to-hop bundle throughput of 72 bundles/s which we have used for our model. Furthermore, we have collected information from all nodes regarding the average number and duration of contacts per hour. This gives us a good basis to apply our model to our network on an hourly basis as shown in Figure 8. We have used contact traces from the weekend in order to model the worst-case behavior of the network.

The Figure shows the number of bundles that would have been permanently lost due to storage overruns (in contrast to bundles that have not yet been delivered) over different sample intervals for varying storage capacities. For the settings from our experiment (100 bundles in storage, 300 s sample interval), the model produces a bundle loss of 0 % as expected. However, with a sample interval of 240 s the storage capacity of 100 bundles is not sufficient and the network would produce a bundle loss of 1.39 %. Increasing the storage capacity to 500 bundles (which is easily possible using flash-based storage) would allow to decrease the sample interval to 60 s without loosing bundles. With a hypothetical unlimited storage, sample intervals of as low as 10 s can be used without bundle loss.

It is interesting to note, that in all of the above-mentioned situations bundles would have been lost due to overrunning storages and not due to short contacts. In fact, we either observed long contacts (during daytime) or no contacts at all

(during night time).

VII. Conclusions

In this paper we have shown, that the BP is suitable for DT-WSNs even on low-power 8- and 16-bit node platforms, such as common low-power WSN nodes.

From a network architecture perspective, isolated solutions are problematic whereas (protocol-wise) homogeneous networks are more desirable. Since most DT-WSNs are not islands from a functional perspective, using the BP in backends as well as in the WSN is the logical consequence for realizing an integrated solution. Different to what the BP specification says, the BP can as well be used as layers 3 and 4 of the protocol stack, whereas an overlay network is not required. This allows using the BP with a simple CL directly inside IEEE 802.15.4 radio frames, thereby significantly reducing overhead.

We have shown, that the communication overhead of the BP is in the same dimension as existing non-delay-tolerant communication protocols but offers the additional functionality of efficiently handling network disruptions. Furthermore, the BP is comparable to those existing protocols from the computational-effort perspective, which makes it usable even on tiny sensor nodes. To provide insight into the capacity of a DT-WSN, we have developed an analytical network capacity model, which calculates the theoretical capacity based on contact traces, storage capacities and bundle generation intervals.

μDTN is our standard-compliant, full-featured BP implementation for Contiki, which is interoperable with BP-compliant DTN implementations on Linux, such as IBR-DTN using the IEEE 802.15.4 CL. It has been successfully tested on INGA and TelosB nodes and provides interoperability of heterogeneous hardware platforms.

We have deployed μDTN in the "Data Elevator" use case, where temperature data is transported to our backend system in a multi-hop DT-WSN using an elevator as data mule. In the evaluation we have shown, that 100 % of the bundles have been delivered successfully, whereas the delay depends on the elevator activity. Even battery-powered wireless sensor nodes can handle multiple hours of network disruptions using RAM-based storage. With the use of flash-based storage, the nodes are able to handle disruptions of several weeks.

Since the BP is the de facto standard in DTN communication, it is the first choice to be used in WSN scenarios that require delay-tolerant communication and backend connectivity. Our results show, that most application scenarios discussed in this paper would benefit from using the BP, since its advantages outweigh the drawbacks. We expect, that the same applies for many existing and future DT-WSN applications.

Acknowledgment

The Lower Saxony research network "Design of Environments for Ageing" acknowledges the support of the Lower Saxony Ministry of Science and Culture through the "Niedersächsisches Vorab" grant programme (grant ZN 2701).

References

[1] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," RFC 6282 (Proposed Standard), Internet Engineering Task Force, Sep. 2011.

[2] E. Davies, "DTN—The state of the art," 2009. [Online]. Available: http://www.n4c.eu/Download/n4c-wp2-012-state-of-the-art-101.pdf

[3] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: modeling a three-tier architecture for sparse sensor networks," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, may. 2003, pp. 30 – 41.

[4] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '03. ACM, 2003, pp. 27–34.

[5] M. Ho and K. Fall, "Poster: Delay Tolerant Networking for Sensor Networks," in *In Proceedings of Sensor and Ad Hoc Communications and Networks (SECON)*, 2004.

[6] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-Tolerant Networking Architecture," RFC 4838 (Informational), Internet Engineering Task Force, Apr. 2007.

[7] K. Scott and S. Burleigh, "Bundle Protocol Specification," RFC 5050 (Experimental), Internet Engineering Task Force, Nov. 2007.

[8] 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.*, vol. 36, no. 5, pp. 96–107, Oct. 2002.

[9] S. Farrell, V. Cahill, D. Geraghty, I. Humphreys, and P. McDonald, "When TCP Breaks: Delay- and Disruption- Tolerant Networking," *Internet Computing, IEEE*, vol. 10, no. 4, pp. 72 –78, july-aug. 2006.

[10] 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 *Proceedings of the 5th international conference on Embedded networked sensor systems*, ser. SenSys '07. ACM, 2007, pp. 103–116.

[11] 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 *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*, 29 2008-oct. 2 2008.

[12] J. Burrell, T. Brooke, and R. Beckwith, "Vineyard computing: sensor networks in agricultural production," *Pervasive Computing, IEEE*, vol. 3, no. 1, pp. 38 – 45, jan. 2004.

[13] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf, "IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation," *Electronic Communications of the EASST*, vol. 37, Jan 2011.

[14] R. Patra and S. Nedevschi, "DTNLite: A Reliable Data Transfer Architecture for Sensor Networks," Berkeley, Tech. Rep. CS294–1, 2003.

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

[16] B. Pasztor, M. Musolesi, and C. Mascolo, "Opportunistic Mobile Sensor Data Collection with SCAR," in *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, oct. 2007.

[17] S. Burleigh, "Compressed Bundle Header Encoding (CBHE)," RFC 6260 (Experimental), Internet Engineering Task Force, May 2011.

[18] D. Brown, S. Farrell, and S. Burleigh, "DTN Bundle Age Block for Expiration without UTC," *IRTF Draft*, 2010. [Online]. Available: http://tools.ietf.org/pdf/draft-irtf-dtnrg-bundle-age-block-01.pdf

[19] The Institute of Electrical and Electronics Engineers, Inc. (2006, Sep.) Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). IEEE 802.15.4-2006.

[20] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460 (Draft Standard), Internet Engineering Task Force, Dec. 1998, updated by RFCs 5095, 5722, 5871, 6437, 6564.

[21] F. Büsching, U. Kulau, and L. Wolf, "Demo: INGA - An Inexpensive Node for General Applications," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '11. Seattle, WA, USA: ACM, 2011.

[22] D. Ellard and D. Ellard, "DTN IP Neighbor Discovery (IPND)," *IRTF Draft*, 2010. [Online]. Available: http://http://tools.ietf.org/html/draft-irtf-dtnrg-ipnd-01

[23] G. von Zengen, F. Büsching, W.-B. Pöttner, and L. Wolf, "An Overview of μDTN: Unifying DTNs and WSNs," in *Proceedings of the 11th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze" (FGSN)*, 2012.