An Empirical Performance Comparison of DTN Bundle Protocol Implementations

Wolf-Bastian Pöttner Johannes Morgenroth Sebastian Schildt Lars Wolf

IBR, Technische Universität Braunschweig
Mühlenpfordstraße 23, Braunschweig, Germany
[poettner|morgenroth|schildt|wolf]@ibr.cs.tu-bs.de

ABSTRACT

In recent years, Delay Tolerant Networking (DTN) has received a lot of interest from the networking community. Today the Bundle Protocol (RFC 5050) is the standard communication protocol in DTNs and three major implementations are available. Since DTN is still a young research area, specifications as well as implementations have not yet reached the same state of maturity as e.g. TCP protocol stacks.

As of now no quantitative analysis of the different implementation's performance or a structured evaluation of interoperability has been undertaken. In this paper we present an interoperability check and perform an extensive quantitative performance analysis of the three main Bundle Protocol implementations.

While the overall results show that all implementations provide some baseline compatibility with each other, the stability and achieved performance under stress situations varies widely between implementations.

1. INTRODUCTION

Implementing Delay Tolerant Networking (DTN) as a way to cope with intermittently connected networks has gained a lot of interest in the research community lately. The standard DTN protocol is the Bundle Protocol defined in RFC 5050 [12]. Today several Bundle Protocol implementations exist, focusing on different use cases.

As of now, no quantitative analysis of the different implementations's performance or a structured evaluation of interoperability has been undertaken. In this paper we measure and compare the performance of the three major Bundle Protocol implementations: DTN2, IBR-DTN and ION. We also report compatibility and interoperability results between the different implementations.

When implementing a new protocol stack for data transmission, the possible throughput is one of the most important quantities to determine. Especially in DTNs, with short contacts between the nodes, the utilization of the available bandwidth determines the efficiency of the whole network. The performance of DTN setups is often analyzed only theoretically through the use of specialised simulators such as The ONE [7]. As these simulators only use approximations of the Bundle Protocol, the observed performance might not be in line with the performance achieved by real DTN implementations in a deployed application.

E. Oliver and H. Falaki [9] have built a testbed for DTN2 evaluation on low-power, low-cost computers. Their performance analysis considers wired and wireless links between four nodes in the experiments and discloses a correlation between the size of a bundle and the possible throughput.

A short comparison of the throughput of ION to the DTN2 implementation is done in [1]. The authors report bandwidths of over 800 MBit/s with a RAM based setup, but omit many important details of the measurement setup.

In [11] IBR-DTN is introduced and some basic tests are presented comparing the performance of the IBR-DTN implementation to DTN2.

The Jet Propulsion Laboratory evaluated the readiness of ION for space missions in the DINET Experiment [13]. During a period of 27 days, some 300 images were transmitted from JPL nodes located on the earth to a spacecraft in the orbit. Then they were automatically forwarded from the spacecraft back to the JPL nodes.

As of now, the performance of the three main Bundle Protocol implementations has only been measured punctually and due to different setups the results are not directly comparable. There has not been a comprehensive comparison of the three implementations with a well defined experimental setup. In this paper, we systematically investigate the performance over the omnipresent TCP convergence layer.

The remainder of this document is organized as follows: Section 2 introduces the three Bundle Protocol implementations and sheds some light on the specialities of each implementation relevant to our experiments. In section 3 we evaluate the network performance of the implementations and examine the interoperability when using the implementations in heterogeneous environments. Finally, in section 4 we summarize our findings and conclude.

2. THE THREE BUNDLE PROTOCOL IMPLEMENTATIONS

This paper evaluates the three major Bundle Protocol implementations: DTN2, IBR-DTN and ION. Since these implementations focus on different applications, certain design decisions for routing, storage or API differ widely. The following section briefly introduces the implementations and outlines notable design features.

In this paper we use the term "disk" storage to refer to bundles being stored persistently on a medium such as a hard drive or flash storage while we refer to storage relying on volatile system memory such as RAM as "memory" storage.

2.1 DTN2

DTN2 [3] is the reference implementation of the Bundle Protocol by the Delay Tolerant Networking Research Group (DTNRG). It provides a flexible framework for DTN related experiments and can be configured and managed by a TCL console and configuration files. Extensions for routing, storage and convergence layers are easily attachable through XML interfaces.

DTN2 features two different built-in modules for bundle storage: A memory based storage and a disk based storage relying on the BerkeleyDB library.

2.2 IBR-DTN

IBR-DTN [11] is a lightweight, modular and highly portable Bundle Protocol implementation designed for embedded systems. The included DTN daemon provides different routing schemes as well as a discovery mechanism.

The IBR-DTN implementation has several flavors of disk storage. Since it has to deal with potentially large blocks as part of a bundle, a BLOB management unit provides a transparent access to container objects which store the data in memory or on disk based on the runtime configuration. These containers are used to store all potentially large blocks (e.g. payload blocks) and avoid the usage of RAM for raw data, if configured properly. Bundle objects composed of several blocks are stored in a bundle storage module for later usage. By default all bundles are held non-persistent in data structures and depending on the BLOB configuration raw data is stored on disk or in memory. If the storage is configured to use disk storage for bundles, all bundle objects are serialized into files and the corresponding object is destroyed, freeing the RAM occupied by this bundle until it is needed again. In this case the store mechanism is persistent and survives reboots or powerfails.

2.3 ION

Interplanetary Overlay Network (ION) [1] is a Bundle Protocol implementation from the Jet Propulsion Laboratory (JPL) specifically developed to run in robotic spacecrafts on a Real-time operating systems (RTOS). However, besides VxWorks, it also runs in x86 Linux distributions.

ION is designed much like a database and the bundle storage is based on the Simple Data Recorder (SDR), an component that already runs in spacecrafts. It can be configured to store data on disk, in memory or to use both media. The SDR supports a transaction mechanism that ensures database integrity in case of failing database operations. ION allows to configure, which combination of storage mechanisms should be used.

The whole system is based on a shared-memory concept that is also used to communicate with sender and receiver processes. Instead of supporting discovery of neighbors, it relies on scheduled wireless contacts. It is optimized for links with small available bandwidths and supports Compressed Bundle Header Encoding [2] (CBHE) for the Bundle Protocol Headers as well as the Licklider Transmission Protocol [10] (LTP). The authors state, that the main purpose of ION is to use CBHE and the LTP, whereas this paper focuses on standard Bundle Protocol with TCPCL to enable comparability and interoperability with DTN2 and IBR-DTN.

3. EVALUATION

In networks with intermittent connections the throughput is especially important, as the goal is to take as much benefit from transient connections as possible. In this evaluation we have concentrated on the throughput that can be achieved in different scenarios. We have evaluated the raw throughput using a GBit LAN connection, the throughput in a bandwidth-limited opportunistic scenario, the possible performance that can be achieved when using different implementations as sender and receiver and finally the effect of the TCPCL segment length onto the throughput.

3.1 Experimental Setup

All experiments were conducted in a controlled environment. As target for all implementations we used computers equipped with an Athlon II X4 IV 2.8 GHz including 4 GiB RAM running Ubuntu Linux. The computers are connected using 1 GBit Ethernet. For the basic throughput tests we did not simulate bandwidth constrained links or disruption as our goal was to test the raw performance of DTN implementations rather than their functionality in disrupted networks.

On the PCs we used different physical Ethernet ports for the Bundle Protocol traffic and the traffic used for controlling and monitoring the experiments. The GBit NICs for Bundle Protocol traffic used a Realtek (RTL-8111 / 8168B) controller. The raw TCP throughput achieved by this setup is ∼940 MBit/s which has been

measured using iperf[1]. In this paper we used DTN2 version 2.70, IBR-DTN version 0.6.2 and ION version 2.3.0. All implementations have been measured using their respective default configuration unless otherwise noted.

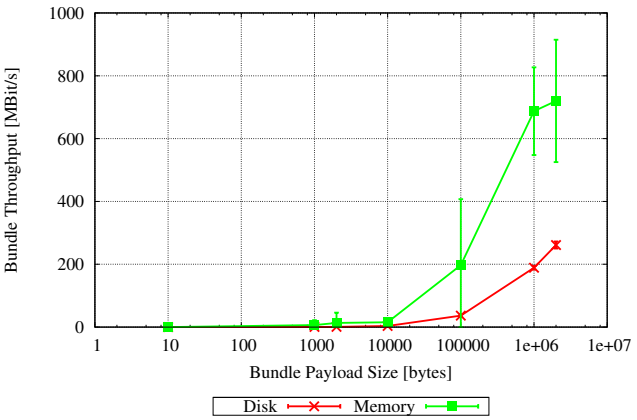
3.2 Network Throughput

In this experiment we measured the network performance of the different implementations. We measured throughput between two nodes that are connected via GBit Ethernet. While a GBit link might be uncommon for typical DTN applications it shows the daemons ability to saturate a given link. Failing to reach high bandwidths in this experiment indicates that bundle processing overhead in a given daemon might be too high. This will not only pose a problem with a high bandwidth link, but it could also preclude a resource constrained node to saturate a lower bandwidth link.

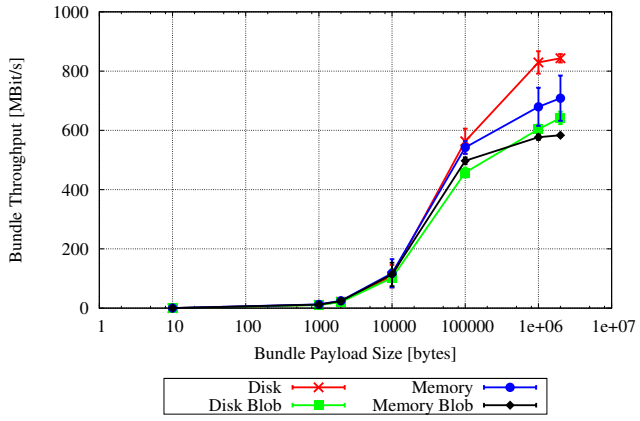
On the sender node we injected 1000 bundles for the receiver. After the bundles had been received by the sending daemon we opened the connection and measured the time until all bundles have arrived at the receiver. For each payload size we performed 10 runs. The plots in Figure 1 show the average of all ten runs as well as the standard deviation. The bandwidth plotted is application layer bandwidth, i.e. it only considers the size of the payload, not protocol overhead. The daemons have been restarted after performing the 10 runs for each payload size, to prevent any influence that old runs might have on the the following measurements. This experiment uses the TCP Convergence Layer.

It can be seen, that DTN2 and IBR-DTN come near to the theoretical limit of the path for large bundle sizes with IBR-DTN reaching 850 MBit/s and DTN2 790 MBit/s bundle layer throughput. In this test ION is not competitive as irrespective of the chosen storage backend it never reaches speeds in excess of 100 MBit/s. In fact all storages perform very similar for ION, which indicates that the responsible bottleneck is not located in a specific storage module.

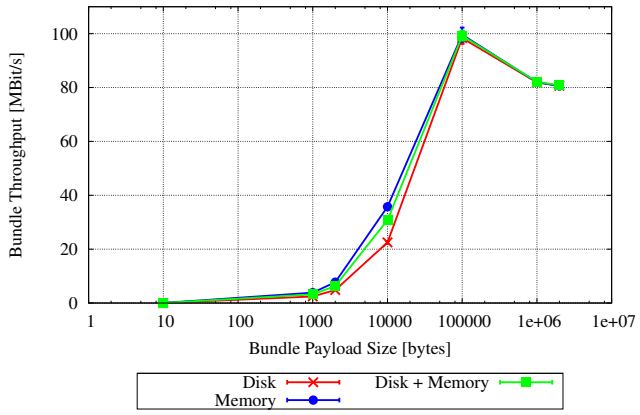
For small bundle sizes DTN2's (fig. 1a) disk and memory storage achieve the same throughput, which indicates that the throughput is bounded by processing overhead. For bundles $\geq$ 10 kByte DTN2's disk storage achieves lower performance than memory storage, which indicates that for these sizes the throughput of the storage engine limits performance. The variances for DTN2's memory storage are extremely high. Upon further investigation, we discovered that after each run using memory storage DTN2 gets gradually slower. For a payload size of 2 MByte[2], the throughput for memory storage varies from 890.98 MBit/s to 574.60 MBit/s with a decreasing trend.



(a) DTN2 Throughput



(b) IBR-DTN Throughput



(c) ION Throughput (with transactions)

Figure 1: Network throughput for different bundles sizes, storage backends and bundle protocol implementations.

IBR-DTN (fig. 1b) shows a similar behavior where the throughput increases with larger bundle sizes, however in absolute terms throughput is always significantly higher than DTN2's. In IBR-DTN the performance

[1] `http://iperf.sourceforge.net/`

[2] 1 MByte $\doteq 10^6$ bytes and 1 kByte $\doteq 10^3$ bytes

is more consistent between different storage backends, while DTN2 performance is much higher when using the memory-based storage compared to its disk based storage.

Interestingly, IBR-DTN shows higher throughputs for disk-based storage compared to memory storage. After further investigation this turned out to be caused by the respective implementation of the storage modules. Disk-based storage uses multiple threads and makes good use of the multiple CPU cores in our test machines while memory-based storage only uses one thread and subsequently only one core.

3.3 Opportunistic Throughput

The goal of the previous scenario was a clean throughput test to reveal the maximum performance of each implementation. In this second scenario the goal is to get realistic throughput values in a network with opportunistic contacts. Based on a scenario for public transport systems published in [8], we have two fixed nodes which are out of range of each other and a mobile node that commutes between them. The timing is based on measurements published in [5]. In each cycle, the mobile node first makes contact with the sender for 72 s, then has no contact for 30 s to make contact with the receiver for 72 s afterwards. After another 30 s without contact, the cycle starts again with the whole experiment lasting 30 minutes. The contacts are simulated using iptables. We limited the TCP throughput to 12.7 MBit/s based on the results presented in [5] using a Token Bucket Filter (tbf) based on the Linux Classless Queuing Disciplines (qdisc). The bundle creation speed is fixed and ensures that always enough bundles for the next contact are present on the sender node. Under optimal conditions this setup would allow transmitting a total of 1000.59 MByte of raw TCP data in 30 minutes.

The hardware used is the same as described in section 3.1. Each experiment is repeated five times for different bundle sizes with DTN2 and IBR-DTN using the memory storage backends. ION is not able to compete in this test, since it does not support opportunistic contacts. We used flooding as routing module for DTN2 and epidemic routing for IBR-DTN, since they are virtually identical in functionality. The total amount of payload that could be transferred is listed in table 1, while the values represent the average and the standard deviation of 5 measurement runs.

The results show, that in these tests DTN2 slightly outperforms IBR-DTN, with the difference being more pronounced for smaller bundle sizes. The slightly lower performance of IBR-DTN was expected, since the memory-based storage produces a lower throughput as already stated in Figure 1b. Also, IBR-DTN has a higher processing overhead when the storage subsystem keeps track of a high number of bundles. Another influencing factor is the fact that DTN2 supports reactive bundle fragmentation that is helpful whenever a bundle has been partially transferred. IBR-DTN does not yet support this feature, so that incomplete bundles are dropped and have to be completely retransmitted upon the next contact.

	250 kB	**500 kB**	**1 MB**	**2 MB**
DTN2	918.2 MB ±4.18 MB	917.3 MB ±2.28 MB	917.6 MB ±2.30 MB	908 MB ±6.63 MB
IBR-DTN	841.3 MB ±2.96 MB	913.1 MB ±0.65 MB	912.6 MB ±1.52 MB	902 MB ±0.00 MB

Table 1: Amount of transmitted data and standard deviation in the opportunistic scenario

3.4 Interoperability

Interoperability is an important issue as the DTN community is rather small and the Bundle Protocol quite young. Not only can incompatibilities expose bugs in the respective daemons but more importantly they might also highlight ambiguities or undefined areas in the Bundle Protocol specification. For the interoperability tests we check whether the daemons can discover each other, which is important for applications relying on opportunistic contacts. However, the most important thing is Bundle Protocol compatibility: If all daemons adhere to the Bundle Protocol [12] and TCPCL [4] specification they should be able to exchange bundles. For the working combinations we will measure the data throughput to give an estimate of the performance that can be expected in heterogeneous environments.

3.4.1 Discovery

Typically, discovery mechanisms are used to detect an available link between two daemons. The IPND draft [6] is a specification for such a mechanism. However, DTN2 implements a proprietary IP-Discovery mechanism using TCP or UDP announcements for IPv4 address/port pairs while IBR-DTN supports IPND in versions 0 and 1 in addition to the DTN2 IP-Discovery mechanism. The ION implementation is strictly focussed on scheduled contacts and does not possess a discovery mechanism.

To prove the interoperability between the three implementations, we used a setup with the standard TCP convergence layer and did a simple test by forwarding a single bundle from one implementation to another using default settings. Before forwarding the bundles, the daemons had to discover each other and setup a TCP connection. We have found out that DTN2 and IBR-DTN are basically compliant and can dynamically discover each other. However, configuration is necessary to specify the common discovery mechanism and UDP port to use.

TX \ RX	DTN2	IBR-DTN	ION
DTN2	197.231 MBit/s	193.202 MBit/s	69.5134 MBit/s
IBR-DTN	677.153 MBit/s	542.337 MBit/s	73.612 MBit/s
ION	106.132 MBit/s	103.955 MBit/s	99.6257 MBit/s

(a) 100 KBytes payload size

TX \ RX	DTN2	IBR-DTN	ION
DTN2	687.329 MBit/s	635.088 MBit/s	80.997 MBit/s
IBR-DTN	881.463 MBit/s	679.45 MBit/s	80.718 MBit/s
ION	85.425 MBit/s	85.313 MBit/s	81.961 MBit/s

(b) 1 MBytes payload size

Table 2: Average Interoperability Throughputs

3.4.2 *Throughput*

To determine, how the interaction between different implementations impacts throughput, we conducted some additional performance tests. We created 1000 bundles on the sender node and then measured the time it takes to transfer these bundles to the receiver node. We used the TCPCL in its respective default configuration and opted to use memory storage to preclude any performance impacts due to disk caching and similar issues. We set up static routing and opened and closed connection between daemons using either built-in methods such as a command line interface or an admin interface (ION, DTN2) or iptables (IBR-DTN). We ran these measurements for all 9 possible pairs of implementations and the results are stated in table 2a for 100 KByte payload size and in table 2b for 1 MByte payload with the respective maximum value marked in bold. Each test was performed 10 times and the tables show the average throughput of these runs.

The results for DTN2 show that it can send 100 KByte bundles with up to 196.231 MBit/s, while it can receive such bundles with up to 677.153 MBit/s. Also, DTN2 can send 1 MByte bundles with up to 687.329 MBit/s while it can receive them with up to 881.465 MBit/s. This allows the conclusion that DTN2 has a bottleneck in the sending components that limits the amount of bundles that can be processed per time interval.

IBR-DTN is able to send 100 KByte bundles with up to 677.152 MBit/s while it can receive such bundles with up to 542.337 MBit/s. For 1 MByte bundles, IBR-DTN can transmit with up to 881.463 MBit/s and receive with up to 679.45 MBit/s. This leads to the conclusion that IBR-DTN has a bottleneck in the receiving component limiting the throughput of this implementation.

Finally, ION transmitted bundles of 100 KByte with up to 106.132 MBit/s while it received such bundles with up to 73.612 MBit/s. Also, ION transmitted bundles of 1 MByte with up to 85.425 MBit/s and received such bundles with up to 81.961 MBit/s. These very low values for ION imply that it has a bottleneck effectively limiting the throughput that can be achieved when transmitting or receiving bundles. The numbers also show, that ION seems to have a significant overhead per byte that is transmitted which explains why larger bundles (with less overhead) reach a lower throughput than smaller bundles. Overall ION shows a performance that is not competitive with either DTN2 or IBR-DTN. This may be explained by the fact that ION is optimized for small bundles sizes and for the use of LTP and CBHE, two techniques which we did not use in our tests to preserve comparability of the results.

An interesting result of the interoperability throughput tests is, that for both payload sizes, the fastest sender and receiver combination is IBR-DTN and DTN2. Intuitively one would have expected to find the highest performance between a sender and receiver of the same implementation. However, this confirms the result that DTN2 is restrained by a bottleneck in the sending component while IBR-DTN is restrained by a bottleneck in the receiving component. It can also be concluded, that for bundle transfer between the same implementation DTN2 reaches the highest throughput for 1 MByte payload size, slightly ahead of IBR-DTN with ION far below. However, for 100 KByte bundles, IBR-DTN significantly outperforms the two competitors.

3.5 TCP Segment Length

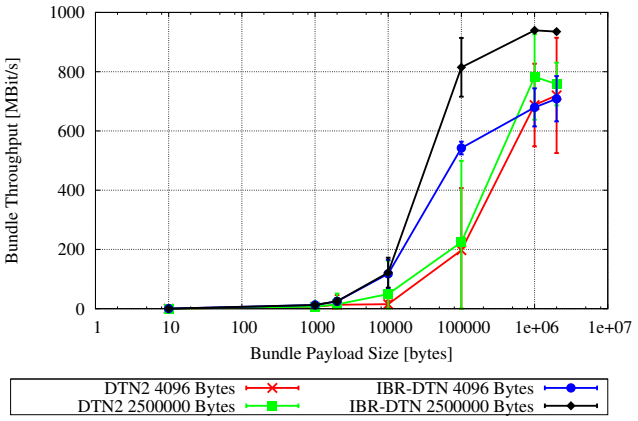


Figure 2: Network throughput with different TCPCL segment lengths.

The TCPCL [4] specification defines that bundles may be transmitted in multiple segments with a separate acknowledgement for each segment. While ION always transmits one bundle in a single segment, DTN2 and IBR-DTN use a default segment length of 4096 Bytes. Modifying this segment length reduces the overhead of the TCPCL and may lead to higher performance.

We conducted throughput measurements similar to those stated in Section 3.2 using the default segment

length and a value of 2.5 MBytes which will transport each bundle in a single segment. Figure 2 plots the average throughput and the standard deviation of 10 measurement runs per data point. While no difference can be observed for bundles sizes $\leq$ 1000 Bytes, significantly improved throughput is visible for bundles sizes $\geq$ 10 KBytes. For a bundle payload size of 1 MByte, increasing the TCPCL segment length has improved the throughput for DTN2 by 94.91 MBit/s while IBR-DTN's throughput was improved by 260.08 MBit/s.

The results show, that modifying the TCP segment length to fit the specific application of a delay tolerant network can significantly improve throughput.

4. CONCLUSIONS

To the best of our knowledge, we have performed the first extensive study of performance and interoperability of the three relevant Bundle Protocol implementations. We have looked at the throughput that can be achieved using TCPCL as well as the effects of the storage systems available for the implementations. Further, we have investigated which performance can be achieved in an opportunistic scenario and how the different implementations interact with each other. Finally, we have evaluated the effects of the TCP segment length onto throughput.

Generally the three implementations are interoperable. The achievable performance depends on a significant number of factors and cannot be predicted easily. This becomes especially clear when looking at the result that the fastest way of transmitting bundles is to use IBR-DTN as sender and DTN2 as receiver.

One of the major influencing factors is the underlying storage system that effectively limits the achievable throughput for links with high bandwidth. Disk storage can be considered the default option for DTNs and is not only influenced by the throughput of the disk but also by the concrete implementation and caching mechanisms of the underlying operating system. Especially when the total amount of bundles exceeds the available RAM, this can be a bottleneck.

Evaluating the performance and interoperability of Bundle Protocol implementations is important for users and developers of DTNs alike. DTN users want to choose their implementation wisely based on their usage scenario. Developers want to tune their implementations to achieve the best usability for a specific target scenario.

Acknowledgements

This work has been partially supported by EFRE (European fund for regional development) project OPTraCom (W2-800-28895) and by the NTH School for IT Ecosystems.

5. REFERENCES

[1] S. Burleigh. Interplanetary Overlay Network: An Implementation of the DTN Bundle Protocol. pages 222–226, Jan. 2007.

[2] S. Burleigh. Compressed Bundle Header Encoding (CBHE). *IETF Draft*, Feb. 2011.

[3] M. Demmer, E. Brewer, K. Fall, S. Jain, M. Ho, and R. Patra. Implementing Delay Tolerant Networking. Technical report, IRB-TR-04-020, Dec. 2004.

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

[5] M. Doering, W.-B. Pöttner, T. Pögel, and L. Wolf. Impact of Radio Range on Contact Characteristics in Bus-based Delay Tolerant Networks. In *Eighth International Conference on Wireless On-Demand Network Systems and Services (WONS 2011)*, pages 195–202, Bardonecchia, Italy, Jan. 2011.

[6] D. Ellard and D. Brown. DTN IP Neighbor Discovery (IPND). *IETF Draft*, Mar. 2010.

[7] A. Keränen, J. Ott, and T. Kärkkäinen. The ONE simulator for DTN protocol evaluation. In *Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, pages 1–10, 2009.

[8] S. Lahde, M. Doering, W.-B. Pöttner, G. Lammert, and L. Wolf. A practical analysis of communication characteristics for mobile and distributed pollution measurements on the road. *Wireless Communications and Mobile Computing*, 7(10):1209–1218, Jan. 2007.

[9] E. Oliver and H. Falaki. Performance Evaluation and Analysis of Delay Tolerant Networking. In *Proceedings of the 1st International Workshop on System Evaluation for Mobile Platforms*, MobiEval '07, pages 1–6, New York, NY, USA, 2007. ACM.

[10] M. Ramadas, S. Burleigh, and S. Farrell. Licklider Transmission Protocol - Specification. RFC 5326 (Experimental), Sept. 2008.

[11] 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*, 37:1–11, Jan. 2011.

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

[13] J. Wyatt, S. Burleigh, R. Jones, L. Torgerson, and S. Wissler. Disruption Tolerant Networking Flight Validation Experiment on NASA's EPOXI Mission. *International Conference on Advances in Satellite and Space Communications*, pages 187–196, 2009.