Bundle Protocol Mail Convergence Layer

Leveraging legacy Internet infrastructure for DTNs

Sebastian Schildt Björn Gernert Lars Wolf

Institute for Operating Systems and Computer Networks
Technische Universität Braunschweig
Braunschweig, Germany
[schildt|gernert|wolf]@ibr.cs.tu-bs.de

ABSTRACT

In the past many research prototypes employed a DTN to provide mail services to remote areas. This work proposes the opposite: Using SMTP and IMAP to transport the Bundle Protocol (BP). Despite the obvious feasibility of such a scheme, we will show that the Mail Convergence Layer (MCL) provides advantages over other Convergence Layers and can qualitatively enhance the capabilities of a DTN.

Each DTN node has an associated mailbox that can be used to deposit bundles, if the node is absent. In this case a mail server effectively acts as an intermediate always-on DTN node without the need to deploy a BP router. We will present the motivation and the concept of the MCL as well as an open-source implementation.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Store and forward networks*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing protocols*

Keywords

Bundle Protocol, Convergence Layer, DTN, IMAP, SMTP

1. INTRODUCTION

To enable communication in various challenged networks, DTN technologies have been applied. While originally proposed for Interplanetary Networks (IPN), where traditional networks can not be used at all, in recent years DTN technologies have been applied to other types of networks where DTNs can leverage additional benefits. One important application area are opportunistic Pocket Switched Networks [9] consisting of mobile wireless devices such as smartphones.

The Bundle Protocol (BP), specified in RFC5050 [15] is a widely used DTN protocol. It implements the Store-Carry-and-Forward mechanism inherent to DTNs and additionally supports end-to-end acknowledgements on top of the hop-by-hop approach. There are various implementations available [14], some of which can run on mobile devices [1, 10].

CHANTS'13, September 30, 2013, Miami, Florida, USA.

ACM 978-1-4503-2363-5/13/09 ...$15.00.
http://dx.doi.org/10.1145/2505494.2505497.

While the quest for a DTN killer app for a general audience has been ongoing for quite some time [11], DTN aficionados are always eager to point out that one of the oldest and most integral parts of contemporary Internet has always been based on DTN principles: The Internet mail system based on SMTP. Other classical internet services with DTN qualities include UUCP (Unix to Unix Copy Protocol) or NNTP (Network News Transfer Protocol), however SMTP is the service that is most widely used until today. This has led to development of projects aiming to bring mail based Internet connectivity to remote rural communities using a DTN [1, 12]. In this work we will present the opposite approach: Using the Internet mail system to transport Bundles in a DTN system. As we will show, this has numerous benefits, especially for mobile users.

The contributions of this work are as follows:

- Discussion of current limits of BP connectivity in opportunistic networks.
- Introduction of the MCL specification and discussion of design decisions.
- Discussing the place of the MCL in terms of BP terminology and discussion of use cases that benefit from using the MCL.
- Introduction and performance evaluation of our open-source MCL implementation.

2. STATE OF THE ART

In an opportunistic scenario, where users may or may not be able to be online at the same time, there are several basic mechanisms that are applied in DTN networks to enable connectivity. In this section we will focus mainly on approaches that have been implemented in BP environments.

Local Discovery

For local contacts some sort of discovery is needed. In BP-based DTNs the IP Neighbor Discovery Protocol (IPND) [5] can be used. IBR-DTN [14] uses IPND, while for example the DTN2[1] BP implementation uses a similar, but in-

[1] `http://sourceforge.net/projects/dtn/`

compatible, local discovery mechanism. The problem is, that these discovery approaches are based on beaconing, which takes a lot of energy for transmitting and for listening. Many schemes have been proposed to allow transceiver duty-cycling while still being able to guarantee discovery even in the case of unsynchronized clocks [4, 2]. However, according to our knowledge, these more advanced approaches have not yet been implemented in any of the widespread BP implementations. In any case, no matter which local discovery method gets used, there is always a trade-off between energy usage and discovery latency.

Global Name Resolution

For contacts outside the local network, which are reachable via the Internet, BP-based DTNs can use the BitTorrent DHT to resolve EIDs and learn about available nodes [13]. However, for mobile nodes that frequently change networks, joining a DHT is not an ideal solution as network churn adversely affects the performance of a DHT. Furthermore, just like the local discovery approaches, keeping in contact with the DHT consumes energy.

DTN Routing

In case of a node that is temporarily not available, there exist various routing protocols that try to forward bundles "towards" the destination. In the case of Pocket Switched Networks, where knowledge about the network is of stochastic nature at best, common DTN routing protocols implement variants of flooding [17, 7] or gossiping [16]. This generates lots of bundle copies and transmissions, which in the case of mobile devices, also means wasted energy and storage. As an alternative to flooding protocols, link-state routing has been proposed and implemented for BP-based DTNs [3]. This is however no solution for highly mobile, opportunistic scenarios.

A suitable solution for these kinds of scenarios would be a number of well-known always-online Internet BP routers. Thus, in case of an unavailable destination, a node would only need to transmit the packet once to one of the routers in the DTN backbone. Once the destination comes online, it will connect to the backbone and receive its bundles. This is an approach that is taken by the "Cloud Uplink" function in the Android version of IBR-DTN [10]. The problem with this approach is, that for scalability it needs the deployment of a sufficient number of DTN nodes in the Internet, which adds additional investment and running costs to the deployment of mobile DTN applications.

We propose using Internet mail servers using the widespread SMTP and IMAP protocols as temporary storage for Bundles. A BP node is assigned an e-mail address, which can be considered a care-of address: If a destination node is not available, but the source is connected to the Internet, bundles will be send to the designated e-mail address assuming that a mail server should be available continuously. The mail server acts like an answering machine for a currently absent DTN node.

3. ARCHITECTURE

The goal of the MCL's architecture was to allow for easy parsing and to be compatible with standard e-mail servers. Therefore we have chosen to encode the fields from the primary bundle block as well as fields from the payload block (processing flags, length) directly into mail headers, while the individual blocks will be put into separate MIME attachments. This design choice allows a BP implementation to fetch the mail headers first, and based on this information an early decision can be made whether the encoded bundle is of interest to the node or not. Furthermore, while not investigated in this work, this would allow some lightweight SIEVE-based [8] routing modules, which can operate by just forwarding a mail without the need to parse or download the bundle.

```
Return-path: <sender@server>
Envelope-to: recv@server
Delivery-date: Wed, 23 Jan 2013 19:44:25 +0100
From: sender@server
To: recv@server
Subject: Bundle for mail://sender@server
Bundle-EMailCL-Version: 1
Bundle-Flags: 144
Bundle-Destination: dtn://some/eid
Bundle-Source: dtn://second/eid
Bundle-Report-To: dtn:none
Bundle-Custodian: dtn:none
Bundle-Creation-Time: 412281870
Bundle-Sequence-Number: 1
Bundle-Lifetime: 3600
Bundle-Payload-Flags: 8
Bundle-Payload-Block-Length: 35
Bundle-Payload-Data-Name: payload.data
Content-Type: multipart/mixed;
 boundary="=_f-20r0xUuORzjAo2CVz1bGFWJK1irHf4t+jNIoYURaTVkAY6"

This is a multi-part message in MIME format. Your mail reader does
not understand MIME message format.
--=_f-20r0xUuORzjAo2CVz1bGFWJK1irHf4t+jNIoYURaTVkAY6


--=_f-20r0xUuORzjAo2CVz1bGFWJK1irHf4t+jNIoYURaTVkAY6
Content-Type: application/octet-stream
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=payload.data

VGhvcm91Z2ggcmVhZGVyIGFjaGlldmVtZW50IHVubG9ja2VkIQ==
--=_f-20r0xUuORzjAo2CVz1bGFWJK1irHf4t+jNIoYURaTVkAY6--
```

Figure 1: MCL-Encoded Bundle Example

An example of a bundle encoded using the MCL can be seen in Figure 1. It shows a bundle from *dtn://second/eid* with mail address *sender@server* delivered to the node *dtn://some/eid* with mail address *recv@server*. The payload can be found in the attachment named "payload.data". The complete specification can be found in the Internet draft describing the MCL [6].

3.1 Propagation of Mail Addresses

While not relevant to the design of the MCL itself, there needs to be a mechanism how a node learns about the mail address of another node. In addition to static configuration the MCL implementation of IBR-DTN opts to extend the two standard discovery mechanisms: IPND [5] for local networks and BT-DHT [13] for Internet-scale node discovery. It seems that learning an e-mail address via IPND at first does not make sense, because once you discovered a node in the same subnet, a direct TCP connection is most likely the more efficient way to transfer bundles to that node. The rationale for this approach is, that the IBR-DTN neighborlist will retain the information about the e-mail address longer than other discovery information. Thus, after having seen a node once in the local neighborhood, the mail address is known and can be used as a fallback if there are bundles available for that node, but the node itself is not. The same is true for e-mail address information extracted from the DHT. The BT-DHT has the additional advantage that for around 30 min after a node's departure from the DHT the stored information can still be retrieved [13]. Another possibility is a specific EID scheme such as *email://* that encodes the mail address directly.

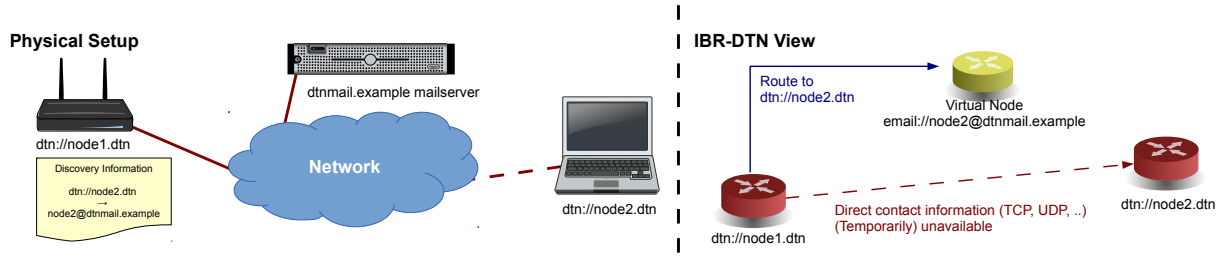


Figure 2: Internal Representation of MCL Contacts in IBR-DTN

3.2 Semantic Issues

Apart from the protocol issues there is the question, how to model the Internet mail system in terms of BP terminology. Is it considered an extension of a node and thus of no relevance to the view of the network, is it – being implemented as a convergence layer – just another communication channel such as TCP or IEEE 802.15.4 or is it an independent participant? To get a better grasp on this we compare it with the more standard TCP CL. In the BP you have only EIDs identifying a destination (most often equaling a node). Mechanisms like IPND, DHT or static configuration can be used to map convergence layer addresses (such as an IP) to an EID. Using the TCP CL, a direct communication line to the entity responsible for a given EID can be established. Bundles sent through the TCP CL to a node can be considered delivered or forwarded. Depending on the bundle the receiving node might acknowledge reception or even take custody of the transferred bundle. This is not the case for the MCL. The receiver might not retrieve its bundle in time before it is expired, or the mail might simply get lost because the destination mailbox is full or nonexisting due to outdated MCL addressing information. Therefore, marking a bundle as delivered when it is passed through the MCL to a destination's mail address might not be the optimal solution.

Our implementation opts to adopt the concept of a virtual node: When the mail address for a contact is available (through IPND, DHT or by receiving a bundle through the MCL), IBR-DTN adds a virtual node object representing the mail server and sets up a route to the destination through the virtual node. This situation is depicted in Figure 2, where node 1 has MCL contact information for the temporarily offline node 2. Using the convention of a virtual node, a bundle transferred through the MCL is not considered delivered, but only forwarded. After the destination receives the bundle from the IMAP server it can still send an ACK back or take custody if required, as these mechanisms also work across several hops in the BP. Furthermore, because the bundle is only forwarded, should the sender meet the destination later, before it had the opportunity to download the bundle from the mail server, the bundle will be delivered directly, thus ensuring the shortest possible delivery time. Once a node detects such an already received bundle on its IMAP server, it will delete that mail based on the header-information without wasting additional bandwidth by downloading it.

4. USAGE SCENARIOS

While the MCL is not intended to replace the more traditional CLs such as TCP, there are two use cases where the usage of the MCL is very suitable: 1. Mobile scenarios that require asynchronous communication, 2. Interfacing legacy systems that are not able to run a complete BP framework.

4.1 Mobile Scenarios

Consider a mobile DTN consisting of nodes that can join or leave the network and can come into each others range at any time. In such opportunistic networks transmitting data to the destination can be challenging. We consider a scenario where n_0 wants to transfer a bundle to a previously encountered node n_1. Figure 3 compares 4 different approaches for the transmission. In this scenario we consider 4 different times A,B,C and D when n_0 will go offline.

Static

This is the baseline scenario. It is based on the unrealistic assumption that n_0 knows exactly at which time n_1 will be available. Such hard-scheduled contacts are usually only available in IPN scenarios. In this baseline scenario transmission is always successful, except in situation A, where n_0 leaves the network before n_1 joins

IPND

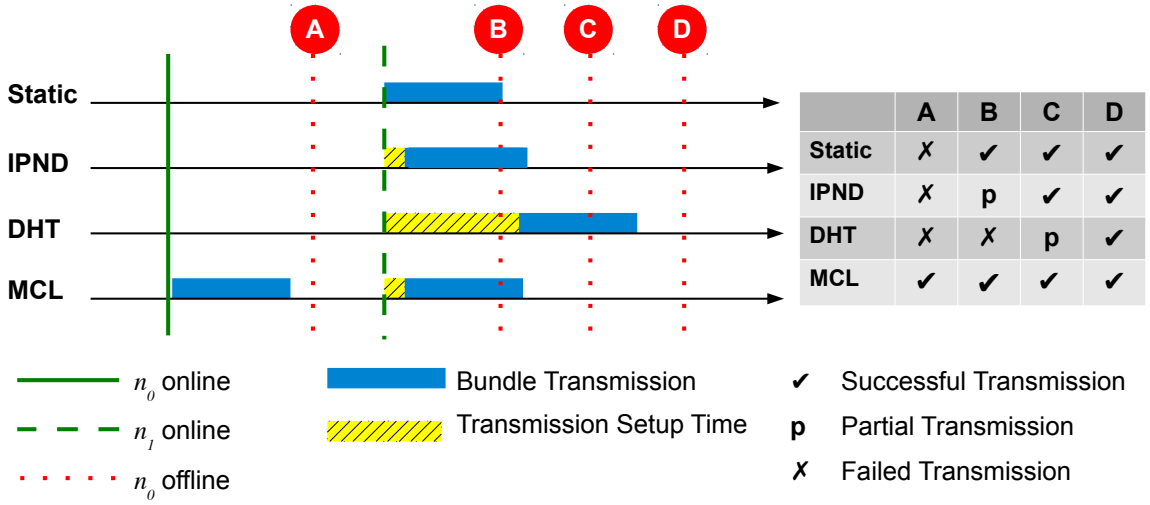
This is a very common scenario for opportunistic networks: Nodes employ some sort of discovery mechanism such as IPND [5], and once a suitable neighbor comes into communication range, data is exchanged. With any neighbor discovery mechanism there is a non-zero latency due to the tradeoff between energy efficiency and latency. In the example in situation B the data can not be transmitted fully due to the additional overhead.

DHT

For BP nodes with a stable Internet connection the BT-DHT naming service [13] is a good option for resolving EIDs. However, due to the high delays in building and joining the DHT, it is not the best choice in highly dynamic scenarios. In our example, once n_1 comes online it needs to associate itself with the DHT before it can announce its contact information, which can then subsequently be queried by n_0. This process can take up to several minutes [13]. In our scenario situation C is the first one that allows for partial transmission of the data and only situation D allows successful completion of the transfer.

MCL

The MCL is the only mechanism that enables complete transmission of the data in our scenario in all situations. As we assume n_0 already had contact to n_1 before, it knows n_1's MCL address. Therefore, n_0 can begin transmission to the



	A	B	C	D
Static	✗	✔	✔	✔
IPND	✗	p	✔	✔
DHT	✗	✗	p	✔
MCL	✔	✔	✔	✔

Figure 3: Mobile and Dynamic Scenarios

mail server immediately, and the data transfer succeeds even in situations were n_0 leaves the network before n_1 joins it.

There are more conceivable options where using MCL is the best situation: Mobile devices in cellular network suffer from highly varying degrees of bandwidth. With highly asynchronous bandwidths, where n_0's uplink bandwidth is very high compared to n_1's download bandwidth, direct transmission rate is limited to n_1's lower bandwidth. If either of the devices can not stay online long enough, the data can not be fully transferred. Using the MCL n_0 can make full use of its bandwidth, and allow n_1 to download the bundle anytime later, without the need of n_0 being online. Of course, the same arguments holds, if n_0 and n_1 are never online at the same time: The asynchronicity of the MCL allows communication between the nodes, with the mail servers acting as "lightweight" DTN Routers.

4.2 Interfacing Legacy Applications

When deploying a DTN system, it is quite likely that legacy applications need to be interfaced. Usually an application specific proxy incorporating a BP networking stack and application specific interfaces needs to be developed. When a full BP stack is not needed, you might opt to teach the legacy application a basic subset of the BP, just enough for the communication needs of the application. When developing the MCL implementation for IBR-DTN, we found the MIME-based MCL encoding to be much more accessible to third party tools than the usual on-wire format of the BP. For example, we whipped up a very small PHP library that could run on a webserver supporting PHP, generating and parsing syntactically valid MCL Bundles containing a single payload block. The generated bundles are fully MCL compliant and accepted by IBR-DTN. Using this code, a standard, cheap shared webhosting account providing PHP can be used to interface a BP based system without the need for any third party libraries that are usually not available on such machines.

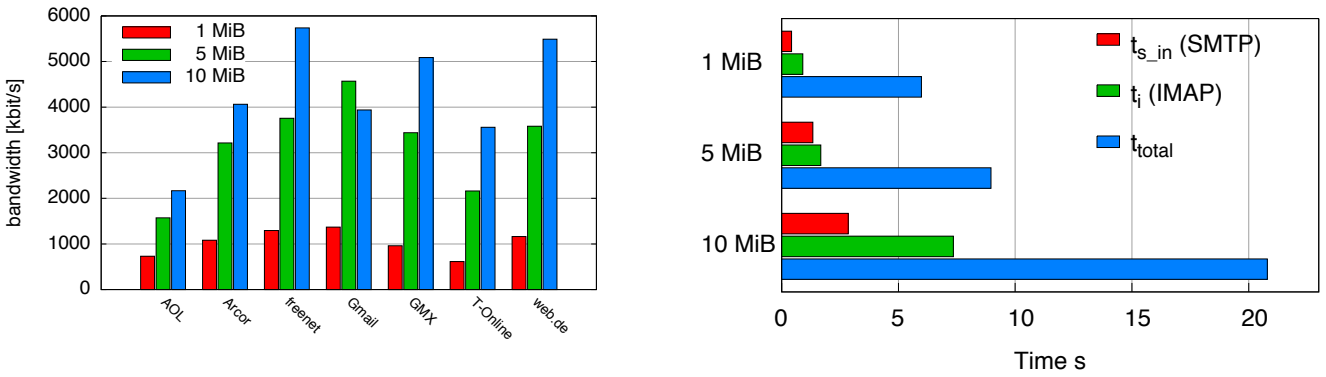
5. EVALUATION

As the MCL is reliant on adding various mail headers to an e-mail, we did test whether common mail providers will preserve the new headers when sending or receiving mail. We tested 7 popular freemail providers by adding the extra headers to a mail sent to or received by them. To emulate some more complex e-mail processing we configured a forwarding chain including 5 separate mail providers. In all these tests the injected mail headers have been preserved, which indicates that the MCL approach should work across most mail server setups.

All things being equal, the base performance of the MCL is somewhat lower than using the TCP CL, as the necessary Base-64 encoding required by the MIME standrd increases the transfer volume by around 33%. Generally, to minimize the transfer volume it is a good idea to compress the contents of a bundle. For this purpose IBR-DTN supports the transparent compression of bundle payloads indicated by an additional Block. This is however not standardized in RFC5050 and thus not compatible with other BP implementations. In the following experiments the bundle size is always the payload size of the bundle before it is encoded by the MCL and no further compression is applied. We performed two experiments to give an idea what kind of bandwidth and latency can be expected when using the MCL with common mail providers.

5.1 Throughput

We measured the throughput that can be achieved using common e-mail providers. In this test node n_0 sends a packet to n_1 using the MCL. n_1 has been configured to use different freemail providers, while n_0 always used the same mailserver, which was under our control. We set the bundle size to 1, 5 and 10 MiB, and performed 100 transmissions for each bundle size and mail provider. The results are shown in Figure 4(a). We conclude that generally larger bundles offer better bandwidth. The observed bandwidths range from 614 $kBit/s$ for T-Online and 1 MiB bundles to 5738 $kBit/s$ using freenet with 10 MiB bundles. Several factors contribute to the total observed bandwidth: In addition to the SMTP and IMAP transfer speeds there is always a fixed per-bundle overhead for initiating the connection to the SMTP and IMAP server and some lost time due to the polling interval, which was set to 5 seconds for



(a) With different freemail providers

(b) SMTP and IMAP transfer times

Figure 4: MCL Throughput Measurements

this experiment (the IMAP library used in our implementation does not yet support the IMAP IDLE feature). Many freemail providers limit the maximum attachment size, so sending arbitrarily large bundles is not possible (however, BP fragmentation can be used). Interestingly, Gmail shows consistently slower performance for 10 MiB bundles than for 5 MiB bundles.

The results shown in Figure 4(a) are the total bandwidth from n_0 to n_1. Keep in mind that transferring a bundle using the MCL includes at least two, more likely three, separate sequential transfers: First the sender has to upload the data to his mail server using SMTP (t_{s_in}). If the sender and receiver use different mail providers, the sender's mail server needs to forward the mail to the receivers SMTP server ($t_{s\_forward}$). After reception of a mail, before delivering it to a user's mailbox there will be some additional processing such as malware scans or antispam filtering (t_{proc}). After the mail has been delivered to the receivers mailbox, it needs to be downloaded using IMAP (t_i). Due to the nature of the Internet mail system all these transfers are sequential for one bundle, and the times need to be added. Additional delays are incurred by the processing and polling intervals of the MCL implementation (t_{MCL}). The total time to transmit a bundle using the MCL is $t_{total} = t_{s\_in} + t_{s\_forward} + t_{proc} + t_i + t_{MCL}$.

Figure 4(b) shows t_{s_in}, t_i as well as t_{total} for transferring bundles from our mail server to a MCL node using Gmail. The displayed values are the average of transferring 100 bundles. Obviously for t_{total} it holds $t_{total} > t_{s\_in} + t_i$. For 1 MiB bundles the SMTP transfer takes 427 ms, the IMAP transfer from Gmail takes 903 ms. In this case t_{total} is ≈ 5.98 s. In this case the overhead $t_{s\_forward} + t_{proc} + t_{MCL}$ is ≈ 4.65 s. For 5 and 10 MiB bundles the overhead is ≈ 5.95 s and ≈ 10.60 s respectively. The increase in overhead for 10 MiB bundles is larger than expected. This is in line with the results from Figure 4(a), where Gmail's performance suffers with larger bundles. Another factor limiting the achieved bandwidth when transferring larger bundles using Gmail is lower IMAP bandwidth for large mails. While 5 MiB mails could be downloaded with ≈ 24.5 $Mbit/s$ on average that speed plummeted to $\approx 11.1 Mbit/s$ for 10 MiB bundles.

While t_{total} might seem high, and the overall achieved bandwidth seems low, keep in mind that the sender will only see t_{s_in}. It can forward a bundle quickly. Similarly, a receiver only sees t_i: From its viewpoint the other components are just delay the bundle has accumulated in the DTN. Furthermore, in our MCL implementation the receiver will receive a bundle already forwarded through the MCL directly *if* the source is available, before it is downloading it from the IMAP server. Thus, *if* a bundle is received through the MCL it *was* the fastest possible path in the DTN. To get a better understanding of $t_{s\_forward}$ and t_{proc}, the following section will take a look at the latencies that can be achieved using the MCL.

5.2 Latency

In this test we measured the latency that can be achieved between two nodes, n_0 and n_1, using the MCL. Node n_0 always uses our own mail server, while n_1 uses various freemail providers. The `dtnping` tool was used to send 25 ping probes from n_0 to n_1 (the most restrictive freemail provider used for this test limited the number of mails that could be sent in a row to 25). The measured latencies consist of the latencies introduced by the MCL implementation and the processing time at the mail servers. The most relevant time for the MCL overhead is polling time: As the used IMAP library does no yet support the IMAP IDLE extension, we opted for a poll interval of 5 seconds in this experiment. Another delay comes from an optimization in our MCL implementation: To avoid repeated connections to a SMTP server, newly generated bundles will be enqueued for up to 10 seconds, before the mailserver is contacted and the bundles are transmitted. As these conditions are the same for every run, the measured data shows speed differences in the mail providers processing.

The results for this experiment can be seen in Figure 5. The average latency values range from 11 s for GMail and web.de to 177 s for freenet. The maximal observed time is 240 s (also freenet). The results demonstrate, that with common freemail providers real-world latencies between 10 to 25 s can be expected. However, as different mail provider optimize their system in different ways, and latencies are a very secondary concern in mail systems, delays can be much higher, as can be seen with freenet. As expected, the MCL is no fit for real-time applications, but with many providers the achieved latencies still allow using messaging applications without decreasing user comfort too much.

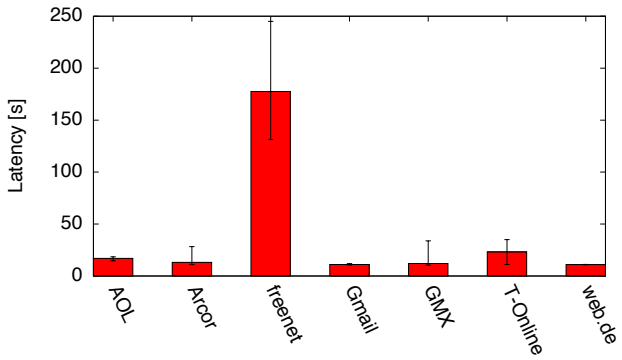


Figure 5: Latency results

6. CONCLUSIONS

We presented and evaluated a concept and implementation of a mail based BP convergence layer. Finally being able to use the Internet mail system – the only widespread Internet-based DTN communication system in existence and use today – as transport medium for the BP offers various advantages: As we have shown, the existing mail server infrastructure can be used to partially offset the need to invest into dedicated BP infrastructure by acting as lightweight always-on BP routers. Doing so is especially beneficial for mobile nodes with intermediate connections such as current mobile phones or tablets. By assigning e-mail addresses to such devices, they basically have a "forwarding address" that enables them to communicate asynchronously when their online times do not overlap.

The IBR-DTN MCL implementation integrates into the standard IPND and DHT discovery mechanisms and is open-source[2]. The published Internet draft, specifying the protocol in detail, should make third-party implementations straightforward. We hope that other BP implementations adopt the MCL and are ready to help.

Acknowledgment

We would like to thank Vincent Richards from Kisli SAS for granting us a VMIME license that will allow us to distribute – besides the source – binary IBR-DTN builds including the MCL implementation.

7. REFERENCES

[1] A. Azfar, J. Jiang, L. Shan, M. J. P. Marval, R. Yanggratoke, and S. Ahmed. ByteWalla : Delay Tolerant Networks on Android phones. Technical report, KTH Telecommunication Systems Laboratory, 2010.

[2] M. Bakht, M. Trower, and R. H. Kravets. Searchlight: Won't You Be My Neighbor? In *Proceedings of the 18th annual international conference on Mobile computing and networking - Mobicom '12*, page 185, New York, New York, USA, Aug. 2012. ACM Press.

[3] M. Demmer and K. Fall. DTLSR: Delay Tolerant Routing for Developing Regions. In *Proceedings of the 2007 workshop on Networked systems for developing regions - NSDR '07*, page 1, New York, New York, USA, Aug. 2007. ACM Press.

[4] P. Dutta and D. Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *Proceedings of the 6th ACM conference on Embedded network sensor systems - SenSys '08*, page 71, New York, New York, USA, Nov. 2008. ACM Press.

[5] D. Ellard and D. Brown. DTN IP Neighbor Discovery (IPND). *Internet-Draft*, 2010.

[6] B. Gernert and S. Schildt. Delay Tolerant Networking Email Convergence Layer Protocol. *Internet-Draft*, 2013.

[7] S. Grasic, E. Davies, A. Lindgren, and A. Doria. The evolution of a DTN routing protocol - PRoPHETv2. In *Proceedings of the 6th ACM workshop on Challenged networks - CHANTS '11*, page 27, New York, New York, USA, Sept. 2011. ACM Press.

[8] P. Guenther and T. Showalter. Sieve: An Email Filtering Language. RFC 5228 (Proposed Standard), Jan. 2008. Updated by RFCs 5229, 5429, 6785.

[9] P. Hui, A. Chaintreau, R. Gass, J. Scott, J. Crowcroft, and C. Diot. Pocket Switched Networking: Challenges, Feasibility and Implementation Issues. In I. Stavrakakis and M. Smirnov, editors, *Autonomic Communication*, volume 3854 of *Lecture Notes in Computer Science*, pages 1–12–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[10] J. Morgenroth, S. Schildt, and L. Wolf. Demo: A Bundle Protocol Implementation for Android Devices. In *Proceedings of the 18th annual international conference on Mobile computing and networking - Mobicom '12*, page 443, New York, New York, USA, Aug. 2012. ACM Press.

[11] J. Ott. 404 Not Found? – A Quest For DTN Applications. In *Proceedings of the third ACM international workshop on Mobile Opportunistic Networks - MobiOpp '12*, page 3, New York, New York, USA, Mar. 2012. ACM Press.

[12] A. Pentland, R. Fletcher, and A. Hasson. DakNet: Rethinking Connectivity in Developing Nations. *Computer*, 37(1):78–83, Jan. 2004.

[13] S. Schildt, T. Lorentzen, J. Morgenroth, W.-B. Pöttner, and L. Wolf. Free-riding the BitTorrent DHT to improve DTN connectivity. In *Proceedings of the seventh ACM international workshop on Challenged networks - CHANTS '12*, page 9, New York, New York, USA, Aug. 2012. ACM Press.

[14] 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.

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

[16] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and Focus: Efficient Mobility-Assisted Routing for Heterogeneous and Correlated Mobility. *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 79–85, 2007.

[17] A. Vahdat and D. Becker. Epidemic Routing for Partially-Connected Ad Hoc Networks. 2000.

[2] `https://github.com/ibrdtn/ibrdtn`