Delay Tolerant Network on smartphones: Applications for communication challenged areas

Hervé Ntareme
Royal Institute of Technology, KTH
Forum 120, 164 40 Kista
Sweden
+46 8 790 4248
ntareme@kth.se

Marco Zennaro
ICTP – International Centre for Theoretical Physics
Strada Costiera, 34151 Trieste, Italy
+39 328 1214733
mzennaro@ictp.it

Björn Pehrson
Royal Institute of Technology, KTH
Forum 120, 164 40 Kista
Sweden
+46 8 790 4284
bpehrson@kth.se

ABSTRACT
This paper discusses the Delay Tolerant Network (DTN) service and protocol stack and presents an implementation of it on the Android platform that is called "Bytewalla". It allows the use of Android phones for the physical transport of data between network nodes in areas where there are no other links available, or where existing links need to be avoided for security reasons or in case the Internet is shut down by a government authority like it happened in some Arab countries during the spring of 2011.
The implementation of a store and forward messaging application and a Sentinel Surveillance health-care application (SSA) that runs on top of Bytewalla are presented together with a few usage scenarios. Our conclusion is that the integration of DTN links in the general IP-network architecture on mobile phone platform is feasible and will make it easier to integrate DTN applications into communication-challenged areas. To our knowledge our implementation of the bundle protocol is the first on the Android platform.

Categories and Subject Descriptors
C.2.1 [**Network Architecture and Design**]: – Store and forward networks.

General Terms
Design, Reliability, Algorithms

Keywords
Delay Tolerant Networks, Android, Mobile phone

1. INTRODUCTION
A geographical area where there is a demand for communication services but no adequate supply is sometimes called "communication challenged area". The most common challenges include a lack of communication infrastructures in terms of wired or wireless links and reliable power supply. Moreover, traditional telecommunication operators hesitate to invest in such areas since they only see low revenues, high costs, high risks, and no profit.
In this paper, we discuss the Delay Tolerant Networking (DTN) approach [1] [2] [3] to deal with this challenge: To transfer data via mobile devices that are physically transported between nodes by extending the Internet Protocol suite with the Bundle Protocol [4]. Specifically we implemented the Bundle Protocol (BP) on the Android OS platform.
In some areas, it is cost effective, at least in the shorter term perspective, to organize such physical transport of data rather than to deploy a physical network infrastructure, such as optical fiber cables or broadband wireless links. Moreover, this can provide business opportunities that could attract local entrepreneurs.
Mobile phones are by far the most commonly available mobile device, also in developing regions. According to the International Telecommunication Union (ITU), the total number of mobile phone subscriptions in 2011 is more than five billions [5]. Additionally, smartphones are currently experiencing accelerating rates of adoption worldwide.
The Android platform was chosen to be used in our project due to its openness for application developers and increasing popularity. The applications we targeted are a general store-and-forward E-mail service, and a Sentinel Surveillance Application (SSA) to be used as a basis for health-care applications in our "ICT for Rural Development programme" [6].
The idea of using physical transport links to forward data between nodes in IP-networks is not new. In 2003, mobile Wi-Fi access points and servers were mounted on buses in rural India to transport emails between rural subscribers and the Internet [8]. Experiments with DTN in similar contexts has been explored in the "Sámi Network Connectivity" project, 2004-2006 [9] and in the "Networking for Communication Challenged Communities" project 2008-2011 [10].
The rest of this paper is organized as follows: In section 2, we discuss about the DTN protocols and services. In section 3, we provide a description of relevant parts of our implementation. In section 4, we briefly present the Bytewalla network setup. In section 5, we discuss two DTN applications that run on top of Bytewalla to be used as a proof of concept for developing extra DTN applications in the future. In section 6, we present our conclusions and future work.

2. DELAY TOLERANT SERVICES AND PROTOCOLS
Several evolving wireless networks such as terrestrial civilian networks connecting mobile wireless devices, including mobile phones, PDAs, or wireless sensor networks (in water or land) or space-networks, such as the InterPlaNetary (IPN) Internet Project [11] do not conform to the Internet's underlying assumptions which are: Continuous, Bidirectional End-to-End connection, short round trips and consistent symmetric data rates between source and destination and low error rates on each link [12]. Such networks are characterized with intermittent connectivity, long or variable delay, asymmetric data rates, and high error rates.

Therefore, connecting them to the Internet requires the intervention of a service that can translate between incompatible networks characteristics and which can provide a buffer for mismatched network delays. The Delay-Tolerant Networking (DTN) helps to address the above mentioned technical issues.

The DTN concept was first conceived within the Inter-Planetary Network Research Group charter (IPNRG) of the Internet Research Task Force (IRTF), to deal with the challenges in high delay environments. The DTN architecture defines an overlay layer on top of the TCP/IP architecture between the transport and the application layer of the network on which it is hosted. This layer forms an overlay that uses persistent storage to solve network interruption related problems and provides functionalities similar to the Internet layer described in the original ARPANET/Internet designs.

The overlay network approach is represented by the Bundle Protocol (BP) (RFC5050). The basic idea is that each packet transmitted is called a "bundle" and contains all of the signaling as well as the data required to transit the transport layer which is referred to as the bundle convergence layer. Therefore, the bundle architecture operates as an overlay network, whereby DTN nodes are identified by Endpoint Identifiers (EIDs), which are the bundling equivalent of addresses. Bundles are routed in a store and forward manner between participating nodes over varied network transport technologies (including both IP and non-IP based transports).

Other DTN protocols include the Licklider Transmission Protocol (LTP) which is a point-to-point protocol designed to be a potential convergence layer to support the bundle protocol, though it can also be used in other contexts. It is primarily designed for the true high-latency case of deep space communications; to be usable as a convergence layer for the bundle protocol. But it can also be used above traditional connectionless transport layer like UDP in terrestrial contexts, including sensor networks using data mules. [13].
DTN are frequently used in disaster relief missions, peace-keeping missions, and in vehicular networks. Moreover, NASA has tested DTN technology for spacecraft communications.

3. IMPLEMENTATION

3.1 Bytewalla network architecture

The Bytewalla network architecture consists of two networks which can be deployed to interoperate from two separate remote locations.

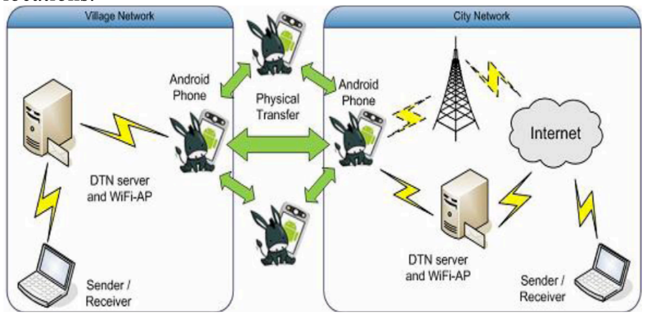


Figure 1: Bytewalla Network Architecture

A practical scenario would be to deploy one network in a rural village which lacks Internet connection and the other network in a city where there is broadband Internet connection as illustrated in Figure 1.

3.2 Software components overview

There are three software components in the implementation: DTNService, DTNManager, and DTNApps. These components interact with each other and with the Android TCP/IP stack. The interactions are illustrated in figure 2.

DTNService is a backend transparent application serving DTN communication. Therefore, even though the user decides to use other applications such as to make a phone call or read text messages, he can still send/receive DTN bundles. To be able to run in backend in the Android platform, this component is implemented as Android Service [14]. This component uses the TCP/IP stack of the Android platform to achieve network communication. Because DTNService is running in backend, there is a need for a user interface to interact with the service. This is the reason why the DTNManager module is needed. It is a front end application for the user to configure, monitor, and manage the DTNService module. This front end application is designed as an Android Activity [15].

DTNApps are the applications running on top of the DTNService. Two sample DTN applications developed are DTNSend and DTNReceive. DTNSend is a DTN application allowing users to send text messages over DTN. DTNReceive is a DTN application allowing users to receive text messages over DTN. Because both of them are front end applications similar to DTNManager, they are mapped to the Android activity.

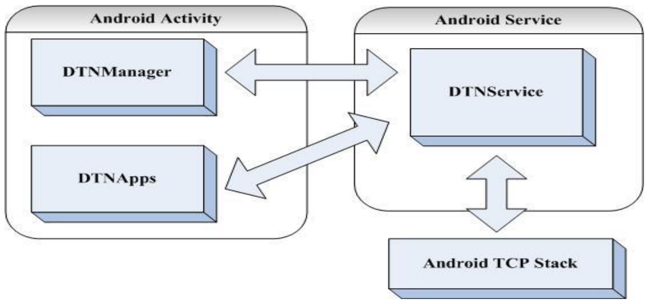


Figure 2: Software components overview

3.3 DTN internal design summary

The internal design of DTNService follows the design and working principles of DTN2 [16], which is the DTN reference implementation developed in C++ by the Delay Tolerant Network Research Group (DTNRG). Therefore, the design has similar characteristics as the reference implementation by which it follows a modular design. This allows the system to be upgraded easily by adding extra functionalities.

The design of DTNService is composed of nine modules which are: Bundle Daemon, Contact Manager, TCP Convergence Layer, Discovery, Persistent Storage, Registration, Bundle Router, Fragmentation manager and APILib. DTNService is an event driven system. There are several types of events, such as, bundle receiving event, bundle transmitted event, or contact initiation event. The system works according to the event handling functions defined in event handling components including Bundle Daemon, Bundle Router, and Contact Manager. The communication among the different modules is illustrated in figure 3. The arrows represent the communication between each module

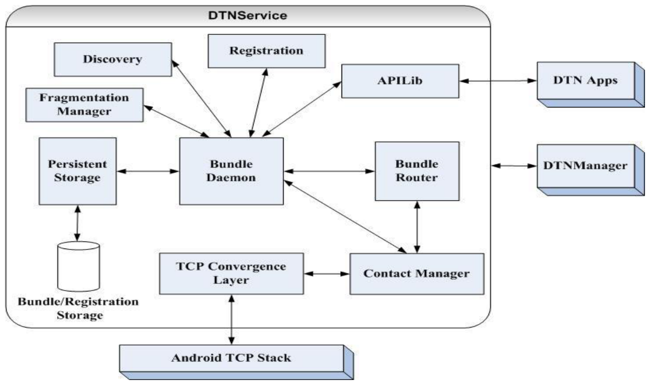


Figure 3: Internal design summary.

3.3.1 Bundle Daemon

Bundle daemon is the main event handler of the system. It is the central processing unit and responsible for communicating with other module for processing the bundle event. Every bundle event is checked first by the daemon. If the Bundle Daemon determines that other two event processing components including Bundle Router and Contact Manager should process the event as well, it forwards the event to these components. Otherwise, it removes it.

3.3.2 TCP Convergence layer

TCP Convergence Layer [17] is the transport mechanism over TCP that the DTN application uses to transmit bundles to a next hop.

3.3.3 Discovery

Discovery is the method by which other nodes can be aware of the "existence" of other DTN participant nodes and their addresses [18]. Discovery is based on the IP protocol; each node sends and listens IP UDP announcements "to discover" remote neighbors. Once a node is discovered, the Bundle Daemon together with the Contact Manager creates a link for that node.

3.3.4 Contact Manager

Contact Manager is the service in charge of detecting new opportunities of connections. Each opportunity of connection ("opportunistic link") with a neighbor ("contact") is under the control of Contact Manager which also does the scheduling of the links. One of the main tasks of this module is to manage the availability of links and contact; this is made by posting events in the Bundle Daemon. Other main tasks are to provide the contact information to the Bundle Router and the linkage to the underlying module "TCP Convergence Layer".

3.3.5 Persistence storage

Persistent storage is the storage mechanism that stores data objects on the disk. Persistent storage is a generic implementation so that it can store different types of objects in the database.

3.3.6 Registration

This module handles the specified registration created from the Bundle Daemon. Every bundle received by the daemon is checked with this module and if it is matched, it is delivered to the registration for further processing.

3.3.7 Bundle router

The Bundle router module is the main decision maker in regard to forwarding the bundles to the destination. In this implementation the routing is handled by the PRoPHET algorithm [19] which is used to make the route decisions for the outgoing bundles.

3.3.8 Fragmentation Manager

The Fragmentation manager module task is to fragment large bundles. It keeps the state of all the fragmented bundles and partially received bundles. Then, it reconstructs the bundle from the received fragments.

3.3.9 APILib

The APILib module provides an Application Programming Interface (API) to develop a DTN application on the Android platform. The API communicates with the bundle daemon module to achieve the API call. This is the channel that is used by other components such as DTNApps to access the DTNService module. This component is implemented by the Android Binder class of the Android APIs.

4. NETWORK SETUP

The network setup is accomplished in four steps which are:

a. Install the required software on the two remote servers:
   - o Ubuntu Linux and Oracle Berkeley DB
   - o OASYS software (Object-oriented Adaptors to SYStem interfaces), DHCP server and The DTN2 software.

b. Install the Bytewalla application on the Android phone [20].

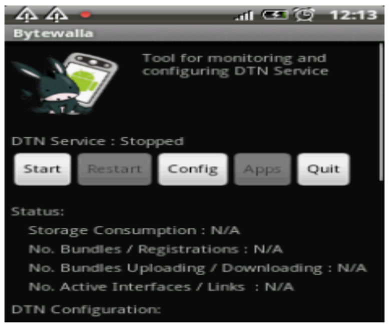
c. Install and configure WIFI access points on the two servers.

d. Configure the three nodes to send and receive DTN Bundles.

5. APPLICATIONS

5.1 The Android mobile phone application

The Bytewalla application has two applications to send and receive data bundles from inside the mobile application. Figure 4 illustrates the GUI of the application.



Tool for monitoring and configuring DTN Service

DTN Service : Stopped

Status:
Storage Consumption : N/A
No. Bundles / Registrations : N/A
No. Bundles Uploading / Downloading : N/A
No. Active Interfaces / Links : N/A
DTN Configuration:

Figure 4: Application example

The configuration part consists of four sections:

a. **Storage section:** In this section the user can define the type of the service to be used for storing the bundles which could be the SD card of the phone. Moreover, the amount of memory to be used can be set.

b. **Interfaces section:** This section refers to the listener interface of the application. It consists of three fields: ID, type of convergence layer and local port. The type of convergence layer used is TCP.

c. **Link section:** This section includes the information needed to start a connection to the DTN servers.

d. **Routing section:** Bytewalla supports both static and dynamic routing.

5.2 E-mail application

The E-Mail application is used to send E-mail messages from users who are based in a remote village without Internet connection

5.2.1 DTN E-mail integration

The servers receive emails from the clients and convert the emails into bundles and vice versa. The bundles are forwarded to the Android phones and transported physically from one location to another. They are delivered on the DTN server when a phone enters in contact with it.

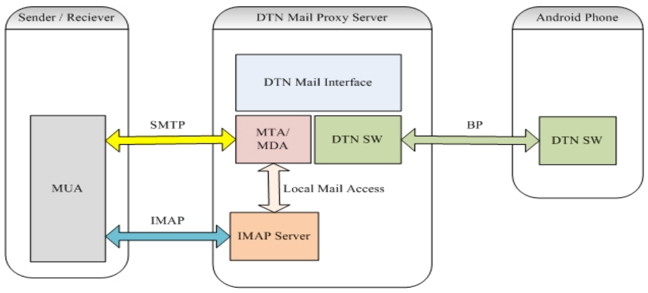


Figure 5: Village protocol deployment

Figure 5 describes the village protocol deployment which consists of a Sender/Receiver, a DTN Mail Proxy Server, and an Android phone. The DTN Mail Proxy Server in the village protocol contains a DTN mail Interface. The sender sends a mail through a MUA which communicate with the MTA/MDA by the SMTP protocol which in turn forwards the mail to the DTN mail Interface. Then, the mail is transferred to DTN software. Then, the DTN software sends the email to the Android phone's DTN software through the Bundle Protocol. The reverse procedure happens when a user in the village receives an E-mail. The IMAP server is used to deliver the Email to the village MUA. A DHCP server allocates dynamic IP addresses to the sender/receiver device and the Android phones.

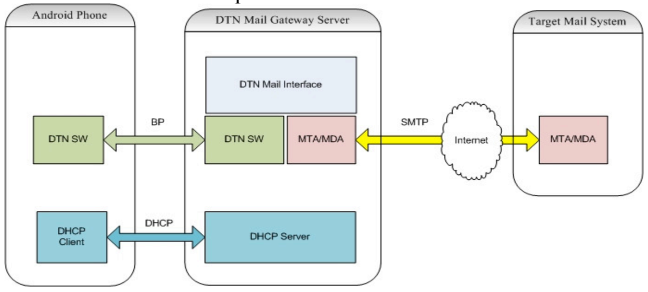


Figure 6: City protocol deployment

The city protocol is similar to the village protocol except that the MTA/MDA of both the DTN mail gateway server and target mail system communicates through the SMTP protocol over Internet both for sending and receiving emails. The communication between the DTN Mail gateway server and the Android phones are the same as in village protocol as it is described in figure 6.

5.3 Sentinel Surveillance Application (SSA)

The goal of the SSA is to provide a facility to report medical related data such as the level of the stock of medical drugs or number of patients in a village to a healthcare authority located in another remote region such as a city by using the Bytewalla DTN network. The SSA needs to reside on both sites; the city and the village, as records are maintained in the databases. It provides access to authenticated users to avoid illegal data manipulation at both sending and receiving sites.

It is a server side web based application which uses a set of basic open source services and software. The list below describes the services and software required in order to run the SSA.

a. *DTN Daemon:* It is based on DTN2 software.
b. *Database:* MySQL is used to store patient's records.
c. *Web Server:* Apache 2 web server
d. *Cron:* A Linux based scheduler
e. *Domain Name Service:* Bind9

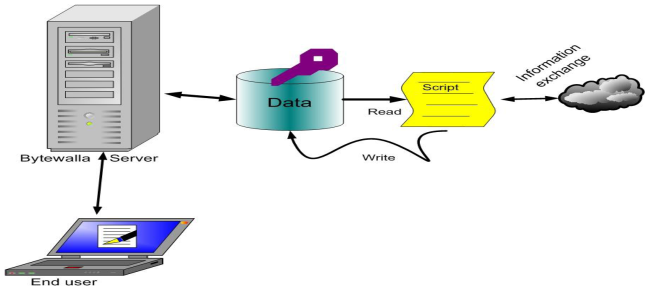


Figure 7: Architecture of the SSA application

The SSA consists of "SSA send" and "SSA receive" applications which are used to send and receive the records. The "SSA send" is executed when a user enters a record in the application. "SSA receive" is required to execute regularly in order to fetch the records. The design of SSA send is illustrated in the figure 8 and described below.

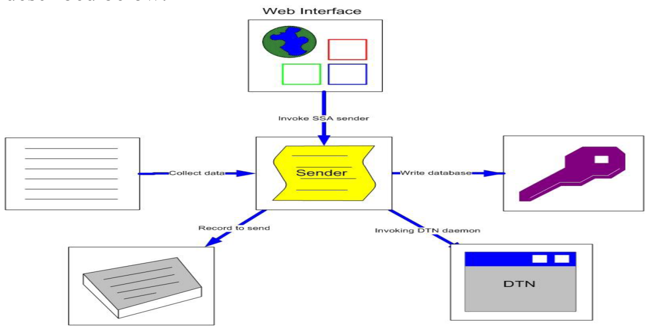


Figure 8: SSA sending process

a. A user interacts with the system using a web interface.
b. SSA sender collects data from the user and stores it in the database.
c. SSA sender writes the same record in a text file with a time stamp.
d. SSA sender transfers the text files to the DTN Daemon.
e. The DTN daemon builds the bundle out of the text file.
f. The bundles reside in a bundle data store and handed over to an Android phone when it passes nearby the site.

The design of the "SSA receive" application follows almost a similar process in receiving the records as described below and as illustrated in figure 9.

a. The Android phone transfers the bundles to the DTN daemon and the data is stored in DTN data store.
b. SSA receiver is invoked by Cron scheduler.
c. SSA receiver scans the DTN data store to fetch the records.
d. "SSA receiver" reads and stores the records in a database.
e. "SSA receiver" backs up the bundle received from the Bytewalla Android phone.
f. "SSA receiver" logs all the performed activities.

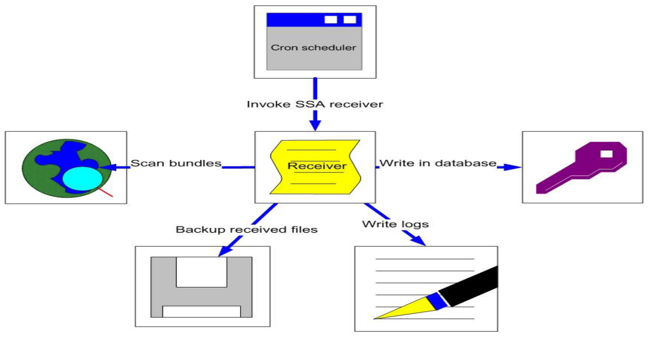
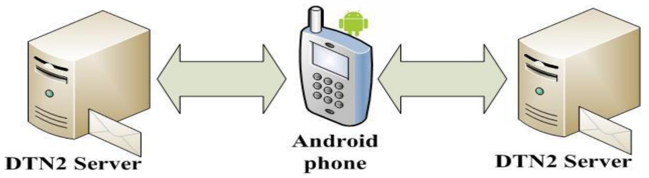


Figure 9: SSA reception process.

6. PERFORMANCE MEASUREMENTS

The test environment consists of two Ubuntu desktop servers which runs DTN2 and an HTC Tattoo Android phone as described in figure 5.



105 DTN bundles were generated to be routed between the two servers. 35 different sizes of bundles were generated with the initial bundle size measuring 100KB. Each of the remaining bundle size was incremented by 100KB. So the size of the last 3 bundles was 3.5MB.

6.1 Bandwidth analysis

The time for downloading 105 bundles is shown in figure 6. There was 189013.1 KB of data in total. In this case the bundles were downloaded from the server to the Android phone. The graph shows a linear increase in download time with the increase of bundle size.

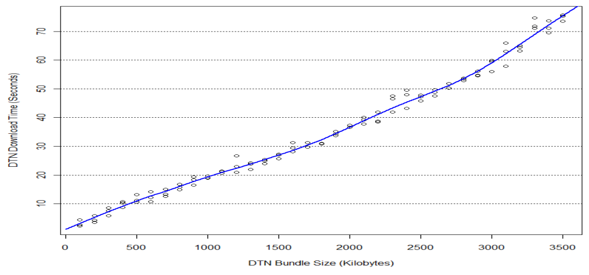


Fig 6: Download bandwidth

This follows a linear equation of the general form y=mx+b. This is expected because while downloading the bundles, there were no other download activities running which can consume the bandwidth. The average download bandwidth equals to the Total Bundle Size / Total Download Time, which is 51.36 KB/s**.**

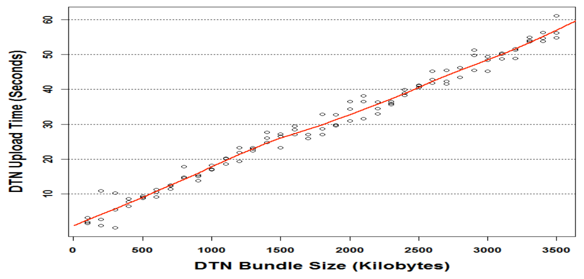


Fig 7: Upload bandwidth

The bandwidth analysis for upload time is shown in figure 7. The bundles were sent from the Android phone to the other server. As expected, the upload time increased linearly with the increase of bundle size. An average value for the upload bandwidth is as follows: The average *upload bandwidth* equals to the Total Bundle Size / Total Upload Time which is 60.29 KB/S.

A comparative study between the download and upload bandwidth is shown in figure 8. For small bundles of size 100 KB, the download and upload times are identical. Up to the bundle size of 1.5MB, there is a very small difference in upload and download time. After that, the difference increases as the download time increases. A significant observation made from this figure is that the download time is always higher than the upload time.

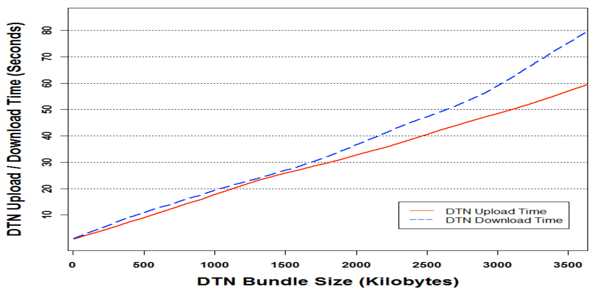


Fig 8: Comparison between download and upload bandwidth

6.2 Power consumption analysis

The battery consumption was recorded with a fully charged battery of HTC tattoo phone. The 105 bundles had a total size of 184.68 MB**.** Figure 9 shows the cumulative battery consumption for downloading the bundles. No other applications were running during the period the data were being recorded. This ensured the battery power was consumed only by the DTN software.

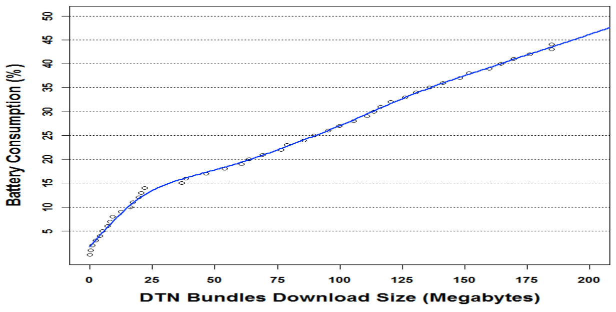


Fig 9: Cumulative battery consumption for download

The battery consumption rate was high for first 22 MB of data. But after that there was a drop in battery consumption till 75 MB size of data. Then, there is a short sharp increase of power consumption and a continued linear consumption. At the end 44% of the total battery charge was consumed for downloading 184.68

MB of data. The average battery consumption rate was equal to 26.20 mAH/MB.

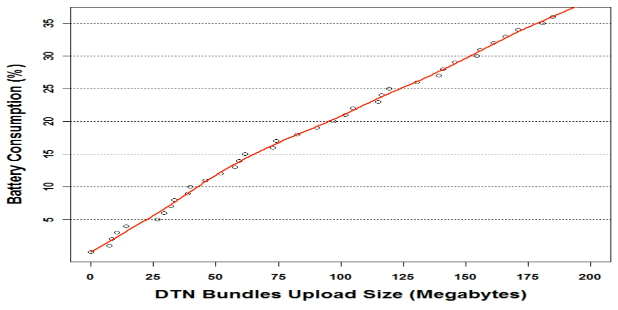


Fig 10: Cumulative battery consumption for upload

A comparative study is shown between the cumulative download and upload battery consumption in figure 11. The battery consumption for uploading data is always lower than the battery consumption for downloading data.

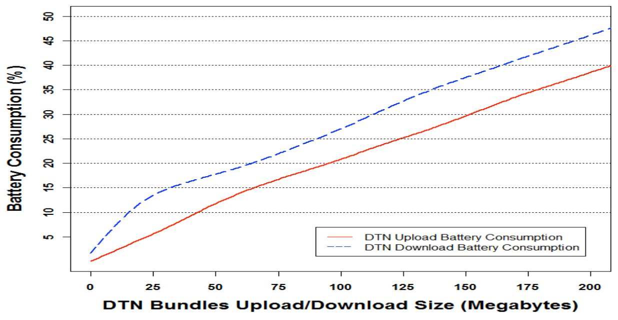


Fig 11: Comparison between upload and download battery consumption

7. CONCLUSIONS AND FUTURE WORK

Our approach allows the integration of DTN in the general mobile IP-network architecture which make it easier to extend delay-tolerant applications into communication-challenged areas. Our implementation follows the Bundle Protocol Specification – RFC 5050.

The future work will include adding more tools and applications on top of the Bytewalla network. Hence, we plan to integrate popular social networking applications such as Twitter and YouTube. A subscription service on top of which educational materials can be exchanged between communication challenged areas would be of great benefit to local communities. Moreover, we plan to use Bytewalla to tap up and transport sensor data from remotely located wireless sensor network for monitoring the quality of drinking water [7]. Finally, a business model could be elaborated to help local entrepreneurs to setup some businesses on top of the Bytewalla system.

8. ACKNOWLEDGMENTS

Our thanks to the Bytewalla 1 and 3 teams and Dr. Avri Doria for their great contributions.

9. REFERENCES

[1] K. Fall, S. Farrell, "DTN: An architectural retrospective", IEEE Journal on selected Areas in Common, Vol.26, no.5. pp. 828-826, June 2008.

[2] A. MacMahon, S. Farrell. "Delay-And Disruption-Tolerant Networking," IEEE Internet Computing, vol. 13, no. 6, pp. 82-87, Nov/Dec. 2009.

[3] V. Cerf, A Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H Weiss "Delay-Tolerant Networking Architecture", IETF RFC 4838, Apr. 2007.

[4] K. Scott and S. Burleigh, "Bundle Protocol Specification", IETF RFC 5050, Nov. 2007.

[5] International Telecommunication Union, Press release. http://www.itu.int/newsroom/press_releases/2010/06.html. Barcelona, 15 February 2010, last accessed on 2011-03-22.

[6] ICT4RD project: http://www.ict4rd.ne.tz/, last accessed 2011-03-25.

[7] M. Zennaro, et all: Water Quality Wireless Sensor Network (WQWSN): An Application to Water Quality Monitoring in Malawi. ICPP Workshops, Vienna, Austria, Sept 2009.

[8] A. Pentland, R. Fletcher, A. Hasson. Daknet: Rethinking Connectivity in Developing Nations. IEEE Computer Society Press, Los Alamitos, CA, USA. 2004.

[9] Sami Network connectivity Project. http://www.epractice.eu/cases/saminetwork, Last accessed 2011-03-24.

[10] N4C project, http://www.n4c.eu/. Last accessed 2011-03-22.

[11] InterPlanetary Internet Project, http://www.ipnsig.org/home.htm , last accessed on 2011-03-21.

[12] F. Warthman, Delay-Tolerant Networks (DTNs): A tutorial. Available at http://www.dtnrg.org/docs/tutorials/warthman-1.1.pdf, last accessed on 2011-03-25.

[13] Stephen F., Vinny C. Delay and Disruption-Tolerant Networking. Artech House. Norwood, MA. 2006.

[14] Android Developers,"Android Service", http://developer.android.com/reference/android/app/Service.html, last visited: 2011-03-25.

[15] Android Developers,"Android Activity", http://developer.android.com/reference/android/app/Activity.html, last accessed 2011-03-25.

[16] DTN2 Documentation, DTNRG, http://www.dtnrg.org/docs/code/DTN2/doc/manual/intro.html, last checked on 2011-03-22.

[17] M. Demmer, "Delay Tolerant Networking TCP Convergence Layer Protocol draft-irtfdtnrg-tcp-clayer-02.txt", http://tools.ietf.org/html/draft-irtf-dtnrg-tcp-clayer-02, last visited on 2011-06-20.

[18] D. Ellard and D. Brown, "DTN IP Neighbor Discovery (IPND)draft-irtf-dtnrg-ipnd-01", http://tools.ietf.org/html//draft-irtf-dtnrg-ipnd-01, Last accessed: 2011-06-20

[19] A. Lindgren, A. Doria, E. Davies, S. Grasic. Probabilistic Routing Protocol for Intermittently Connected Networks draft-irtf-dtnrg-prophet-09. http://tools.ietf.org/html/draft-irtf-dtnrg-prophet-09. Last accessed 2011-06-20

[20] Bytewalla software installation guide. http://www.tslab.ssvl.kth.se/csd/projects/092106/sites/default/files/Bytewalla_Installation_Guide.pdf, Last accessed 2011-06-20.