Network Emulation Testbed for DTN Applications and Protocols

Razvan Beuran
Hokuriku StarBED Technology Center,
National Institute of Information and Communications Technology,
Ishikawa, Japan
Email: razvan@nict.go.jp

Shinsuke Miwa
Hokuriku StarBED Technology Center,
National Institute of Information and Communications Technology,
Ishikawa, Japan
Email: danna@nict.go.jp

Yoichi Shinoda
Research Center For Advanced Computing Infrastructure,
Japan Advanced Institute of Science and Technology,
Ishikawa, Japan
Email: shinoda@jaist.ac.jp

***Abstract*—Wireless devices are widely used today to access the Internet, despite the intermittent network connectivity they often provide, especially in mobile circumstances. The paradigm of Delay/Disruption Tolerant Networks (DTN) can be applied in such cases to improve the user experience. In this paper we present a network testbed for DTN applications and protocols that we developed based on the generic-purpose wireless network emulation testbed named QOMB. Our testbed is intended for quantitative performance assessments of DTN application and protocol implementations in realistic scenarios. We illustrate the practicality of our emulation testbed through a series of experiments with the DTN2 and IBR-DTN implementations, focusing on mobility in urban environments. The scalability issues that we have identified for DTN2 emphasize the need to perform large-scale repeatable evaluations of DTN applications and protocols for functionality validation and performance optimization.**

I. Introduction

Given the advent of smartphones and tablets, Internet access is being done more and more often using such kind of wireless devices, many times in mobile scenarios. It is said that about 85% of the world population is covered by commercial wireless signals, providing a greater reach than the electrical grid [14]. It was also estimated that, with over 1.6 billion users, mobile Internet will exceed desktop Internet use by 2015 [13].

The Delay/Disruption Tolerant Networks (DTN) approach seems to be a promising solution for improving user experience in such scenarios with intermittent connectivity by ensuring a "smoother" network access. Although the DTN concept was first put forward in connection with deep-space communications, it was later extended to the more generic class of *challenged networks* [10].

The DTN paradigm can be applied to a wide range of network categories, such as mobile networks, ad hoc networks, or sensor networks. It is notable that the MobilityFirst project of the Future Internet Architecture program of NSF also includes robustness and usability among the high-level requirements of the mobile Internet, and proposes "Generalized DTN" (GDTN) as means to meet these requirements [20].

Although current DTN implementations do not necessarily answer all the needs of the Future Internet, their built-in robustness mechanisms and features such as tunneling make it possible to start deploying DTN protocols in order to provide better connectivity over intermittent network connections.

Nevertheless, a wide-scale deployment of DTN protocols and applications cannot be made without a proper evaluation of the corresponding implementations in realistic scenarios, so as to ensure that they perform as expected under various circumstances. The main requirements for making it possible to perform such quantitative assessments are:

- Ability to execute in a controlled manner the DTN applications and protocols under test;
- Ability to accurately reproduce the communication conditions between the DTN nodes;
- Ability to recreate scenarios that are meaningful and realistic, including with regard to node placement and mobility, the communication environment, etc.

In order to meet these requirements, we based our work on the generic-purpose wireless network emulation testbed named QOMB, developed at the Hokuriku StarBED Technology Center, National Institute of Information and Communications Technology, Japan [3]. The emulation approach used by QOMB permits to directly execute application and protocol implementations. Moreover, the support for various wireless communication models, as well as the versatile scenario definitions in QOMB made it possible to extend its functionality as required for various DTN-related experiments. We note that the QOMB testbed is open for researchers worldwide, and details about its utilization are provided on the following web page: `http://starbed.nict.go.jp/en/index.html`.

The main contributions of this paper are:

- Detail the functionality enhancements that made possible DTN emulation experiments on QOMB;
- Demonstrate the practicality of our testbed through the performance assessment of the DTN2 and IBR-DTN implementations in several use-case scenarios.

The remainder of this paper is organized as follows. In Section II we present an overview of QOMB and of its components. In Section III we detail the extensions of QOMB that were needed in order to make DTN emulation experiments possible. Then, we discuss a series of DTN experiments in which we illustrate the practicality of our testbed by evaluating the performance of the DTN2 and IBR-DTN implementations (Section IV). In Section V we introduce several related projects. The paper ends with conclusions and references.

978-1-4673-1017-8/13/$31.00 ©2013 IEEE

II. QOMB Overview

QOMB is a wireless network emulation testbed that was initially created for IEEE 802.11 network emulation. The general architecture of QOMB and utilization examples in this context were presented in [3]. The modular architecture of QOMB and of its components makes it possible to extend the testbed to other wireless network technologies and devices.

In this section we introduce the main components of QOMB, and the generic architecture for DTN emulation. In particular, the QOMB wireless network emulation testbed is realized by integrating the large-scale wired-network testbed that is StarBED (and its support tools, such as SpringOS) with the wireless network emulation set of tools that is QOMET.

A. StarBED

StarBED is a large-scale wired-network testbed at the Hokuriku StarBED Technology Center of the National Institute of Information and Communications Technology, located in Ishikawa, Japan [16]. With over 1100 interconnected PCs made available for experiments, users can perform a wide range of network experiments on StarBED, and it represents the physical infrastructure of QOMB. The experiment network in StarBED is completely separated from the control network so as to avoid interactions between these two networks.

SpringOS is the main experiment-support software tool for StarBED, and it allows users to easily perform complex experiments with a large number of hosts [16]. The most important functions of SpringOS are:

1) *Experiment preparation:* Configure the experiment hosts and network for experiment execution;
2) *Experiment execution*: Effectively carry out the experiment by executing the required commands on the experiment hosts.

SpringOS has a message-based client-server architecture. Thus, a module called "SpringOS master" is in charge of controlling the experiment execution on all the experiment hosts. This is done with the assistance of local modules, called "SpringOS client", executed on each experiment host.

B. QOMET

QOMET (Quality Observation and Mobility Experiment Tools) is a set of tools for wireless network emulation [3]. QOMET allows the definition of various complex scenarios, including experiments with node mobility and urban settings. QOMET includes several wireless propagation models, as well as mobility models, such as random walk and a behavioral mobility model. QOMET provides the necessary mechanisms for performing wireless network emulation in a distributed manner by reproducing the communication conditions between the emulated nodes that are part of an experiment. QOMET relies on the experiment management mechanisms of StarBED for its distributed execution. Thus, the integration of StarBED, SpringOS and QOMET effectively results in a wireless network emulation testbed that we call QOMB.

The most important components of QOMET that make wireless network emulation possible are:

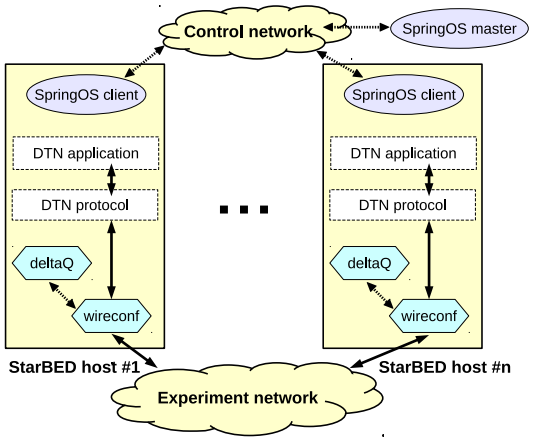

Fig. 1. Architecture of the QOMB-based DTN emulation framework.

1) deltaQ library: This library is in charge of computing the communication conditions between wireless nodes given a user-defined XML-based scenario. The scenario specifies the properties of the wireless nodes (position, parameters of the network technology that is utilized, mobility patterns, etc.), and of the environment in which they are placed (attenuation, shadowing, street and building structures, and so on). These properties are used to create a "virtual world" that corresponds to the user-defined scenario, in which the emulated wireless nodes move (virtually) and communicate with each other. The traffic exchanged is created by the real protocol and application implementations used in the experiments.

2) wireconf library: The communication conditions computed by the `deltaQ` library are recreated during the real-time experiment by the `wireconf` library. This library is in charge of controlling the communication conditions between the emulated wireless nodes in the experiment according to the user-defined scenario, by applying the corresponding network degradation (packet loss, delay, bandwidth limitation) to the traffic being generated and received by the nodes; more details on this functionality will be provided in Section III-A.

C. Emulation architecture

In Figure 1 we show the architecture of the QOMB-based network emulation framework used in our DTN experiments, created by combining the components mentioned so far. Note how SpringOS manages the experiment executed on the StarBED hosts via the global SpringOS master and the local SpringOS client modules. These operations are done through the control network in StarBED. SpringOS also plays a role in experiment preparation, as indicated in Section II-A.

The communication conditions between the emulated nodes are recreated in the experiment network by the `wireconf` library that interacts with the `deltaQ` library for condition computation purposes. The instances of the `wireconf` library on all the experiment hosts exchange information with each other in order to adjust the communication conditions in function of the continuously changing state of the emulated network, e.g., to account for the contention created by the wireless nodes as they use the communication channel.

The traffic through the emulated network is generated by

DTN applications, that in their turn run on top of the DTN protocol layer (see Section III-D for the supported protocols).

An experiment feature that was recently added to QOMB and that further differentiates it from other testbeds is the ability to conduct *hybrid experiments* in which real and emulated wireless nodes can be seamlessly integrated into the same network experiments [4]. In this context one can envisage using real wireless devices for the static nodes in a given DTN experiment scenario, thus achieving high fidelity, and use emulated wireless devices for the mobile nodes in the experiment, thus also attaining reproducibility and scale.

III. DTN EMULATION

Several new features and components were required in order to make DTN system emulation possible on QOMB.

A. Linux support

In the original version of QOMET the `wireconf` library was designed on top of the FreeBSD 5.4 ipfw2/dummynet system. However, many recent DTN implementations, such as DTN2, are mainly supported on Linux; thus, it was necessary to port the `wireconf` library to Linux.

For this purpose we took advantage of the fact that ipfw itself was also ported to Linux by its developers as of version ipfw3 [8]. Therefore, we made the necessary changes in the source code of the `wireconf` library so that it can be compiled and executed on top of ipfw3. The operating systems that we selected as target for experiment execution are Scientific Linux 6.0 and CentOS 6.0, which are both free variants of RedHat Enterprise Linux (RHEL) 6.0.

The main changes required in `wireconf` were:

1) Modify the interface used by `wireconf` to enforce the scenario communication conditions, so that it can drive the new ipfw3 module;
2) Modify the interface used by `wireconf` to obtain routing information from the kernel on our target RHEL-based operating systems.

The changes related to item (1) above implied adapting the `wireconf` code to take into account the changes in the ipfw3 source code compared to the previous version ipfw2. As for the interface with the kernel, some modifications were required since kernel headers and structures are different between FreeBSD 5.4 and RHEL 6.0. As RHEL 6 will receive support from its manufacturer at least until 2017, we don't envisage the need for porting the code again in the near future.

We stress in this context the fact that the other main component of QOMET, the `deltaQ` library, doesn't have similar limitations regarding execution, since it is mainly a computation engine. From start `deltaQ` has provided support both for FreeBSD and Linux, as well as for Windows.

B. Multi-interface support

In previous QOMET versions, the user was limited to defining a single wireless interface per node in the emulation scenario. However, modern wireless systems, such as smartphones, typically have multiple wireless interfaces, for instance GSM or LTE and WLAN. Therefore we added support for describing such experiments in QOMET.

This was accomplished by adding the "interface" element to the XML scenario description that is the input of the `deltaQ` library. This new element was introduced as a subcomponent of the "node" element that existed previously, and which used to implicitly define a single network interface. With the new "interface" element, users can define several network interfaces for each node in the scenario, each with its own properties (transmit power, receive sensitivity, etc.). During the experiment, the interface elements in QOMET are associated by `wireconf` to network interfaces of the experiment host according to user-defined settings.

A possible use of a multi-interface system in connection with DTN is that a certain application could leverage the presence of multiple interfaces in order to optimize its performance. For instance, by sending important traffic on links with low loss rate, and traffic that must be delivered quickly on links with low delay, an application could better cope with potential network disruptions. In this context we mention that, in addition to WLAN, we are currently adding WiMAX/LTE support in QOMET, thus making possible realistic experiments with such multi-interface wireless devices.

C. Fault injection

Assessing the behavior of DTN protocols and applications implies being able to create the necessary disruptions of the communication conditions. Until now, QOMET has relied on realistic scenarios in order to create meaningful communication conditions for the applications and protocols under test.

While we still believe that realistic scenarios are a fundamental requirement for a thorough evaluation of a network application or protocol, we acknowledge the fact that under certain circumstances — and in particular in the context of DTN — it is important to also be able to create disruptions in a controlled manner, even though they may not be entirely realistic. Such an approach allows to leverage the characteristics of emulation (repeatability, control, etc.) to test a certain system in simple network conditions, so as to assess its basic properties and/or debug its major issues [2].

We call *fault injection* the mechanism of introducing controlled disruptions in a network. This issue was analyzed in detail in [15]. We focus here on the features that were added to QOMET in order to support fault injection mechanisms that can be used in the context of DTN. Thus, we distinguish two categories of disruptions:

1) Communication environment disruptions;
2) Network condition disruptions.

Communication environment disruptions are created by the artificial injection of faults in the wireless communication environment between the emulated nodes. A typical example of such fault injection is the introduction of artificial electromagnetic noise in the emulated virtual world. As electromagnetic noise will interfere with the communication in the virtual environment, it will affect the network conditions in the experiment in an indirect and realistic manner; support for this feature was added to the `deltaQ` library. Electromagnetic noise can be controlled by the user in terms of position of the

noise source, its intensity, as well as the starting time and duration of noise generation, but the user has no direct control over the resulting communication conditions.

Network condition disruptions refer to the case when the user defines *explicitly* the network conditions between a certain pair of nodes, and the `deltaQ` and `wireconf` libraries recreate the specified conditions without performing any computation. Although this approach is not very realistic, it provides a good quantitative control over the network conditions, such as directly indicating the available bandwidth, delay and jitter, and/or loss rate. The user specifies in the QOMET scenario the interval during which a certain set of conditions should be applied, making it easy to describe the time-varying network conditions that characterize DTNs.

D. DTN protocols

The fact that our testbed supports Linux-based OSes means that most current DTN implementations can be compiled and executed, although minor changes may be required in some cases depending on the precise requirements of each DTN implementation in terms of libraries. This makes it straightforward to support both standard DTN implementations as well as any novel DTN implementation that may appear in connection with Future Internet research, such as the Generalized DTN of the MobilityFirst project [20].

The DTN implementations that we have used so far on our network emulation testbed are briefly described below:

1) DTN2: Reference implementation of the *bundle protocol*, which is a general overlay network protocol [19]. DTN2 represents the main focus of the implementation effort in the Delay Tolerant Networking Research Group (DTNRG) of the Internet Research Task Force (IRTF) [7]. We integrated in our testbed the latest version of DTN2, dtn-2.9.0, released in July 2012, and two preceding versions: dtn-2.8.0, released in August 2011, and dtn-2.7.0, released in February 2010.

The DTN2 implementation of the bundle protocol is represented by the module "dtnd" that must be running on all the nodes in the network. DTN2 also includes several sample applications. We used the following two: (i) *dtnping*, equivalent to the typical "ping" command; (ii) *dtnperf*, equivalent to the widely-used "iperf" command for network performance assessment.

To configure the dtnd module we used the default parameters, with a few exceptions. Thus, in order to establish links between nodes we used the built-in discovery protocol (with a "hello" message frequency of 5 s for most experiments); the links were constructed over TCP. We used two of the routing protocols built into dtnd: (i) *flood* that sends each message to all the known neighbors of a node; (ii) *dtlsr*, a delay-tolerant routing protocol similar to OLSR.

2) IBR-DTN: Lightweight bundle protocol implementation of the IBR group of the Technical University Braunschweig that mainly targets embedded systems [18].

We integrated version 0.6.5 of IBR-DTN from November 2011, which has a limited set of features in comparison with DTN2, as no implementation of "dtnperf" or "dtlsr" are included. However, IBR-DTN development is very active, and its latest version 0.8.0 (released in June 2012) includes new options for dtnping and new routing protocols, thus reducing the gap with respect to DTN2.

E. Discussion

Amongst the DTN features that we have discussed so far, the most challenging to consider is fault injection, since it is important to strike a balance in this context between user control and effect realism. Hence, we focused on the two categories of disruptions mentioned above, thus providing full flexibility for the user to choose the best approach for any given scenario.

A minor feature that we nevertheless consider important for enabling realistic experiments, including in the context of DTN, is the possibility to import motion trajectories into QOMET, such as those obtained from GPS devices during real-world experiments. Since there is no global standard way for storing GPS data, we instead implemented support for importing mobility data in the QualNet trace format, a flexible text format which can be easily produced from any kind of motion data, including from GPS traces. Moreover, many mobility generators, such as BonnMotion [1], support exporting mobility data in the QualNet format, hence they can be used to produce mobility traces for QOMET.

Another important feature in the context of WLANs, for which development is still ongoing, is the support for the emulation of Wi-Fi access point (AP) behavior. QOMET was so far used exclusively for ad hoc networking scenarios. However, in order to extend the range of possible experiments we are currently implementing support for AP-like functionality, mainly the association/disassociation process typical to AP-based communication.

IV. Experimental Results

The main focus of this paper is introducing the DTN emulation testbed that we designed and implemented. Therefore the experimental results presented in this section are simply examples that serve to illustrate the practicality of our testbed in this context. We shall not perform any evaluation of the core characteristics of the testbed, such as wireless network emulation and mobility, since such evaluations have already been done in previous publications [3].

A. Experiment overview

Our extensive evaluation of DTN2 and IBR-DTN leads us to the following general conclusions that demonstrate the need to perform repeatable experiments with DTN implementations:

1) Simple scenarios: With 2-3 nodes, both the DTN2 and IBR-DTN bundle protocol implementations behaved as expected. For dtnperf in DTN2 we noticed reasonable performance characteristics, especially for large bundle sizes. However, the RTT results shown by dtnping in DTN2 are one order of magnitude larger than the RTT measured with the dtnping command in IBR-DTN, pointing at significant overheads in DTN2. The performance of dtn-2.7.0 seems to exceed that of dtn-2.8.0, which is in its turn better than the performance of dtn-2.9.0.

TABLE I. SUCCESSFUL DTNPING REPLIES IN 26 AND 10-NODE EXPERIMENTS

Experiment type	DTN2 flood	IBR-DTN flood	DTN2 dtlsr
26 nodes (25 mobile)	6%	47%	28%
10 nodes (3 mobile)	42%	91%	93%

2) Larger scenarios: DTN2 performance degrades quickly with scale, leading to poor results for scenarios with as few as 26 nodes, even if only some of them are mobile. This low performance seems to be caused by performance bottlenecks that cause high CPU utilization on the participating DTN nodes even for relatively-low traffic loads. Results are much improved for IBR-DTN, including for large node counts, as Table I illustrates.

Results appear somewhat better for goodput measurements conducted using dtnperf in DTN2, but since we did no equivalent experiments with IBR-DTN it is not possible to make a comparative analysis.

3) Routing protocols: The implementation of flood routing in both DTN2 and IBR-DTN generally behaved as expected, having good performance in sparsely connected networks (hence for low overall network load), and poor performance in good connectivity conditions and with many traffic sources (hence with a high overall network load). On the other hand, while dtlsr in DTN2 had relatively good performance in good connectivity conditions, the performance was poorer than expected in sparse networks and in the presence of mobility, which leads us to believe that the performance of the dtlsr implementation itself is to be blamed in some of these circumstances.

B. Experiment example

We summarize here one of our experiments, with an urban environment including 5 mobile nodes of a total of 26 nodes (including a fixed one acting as a gateway); more detailed information is available in [5]. A snapshot of the scenario at time 100 s is shown in Figure 2. The 5 mobile nodes are #1, #3, #6, #18 and #22, and they move from the gateway to their respective destinations, whereas all the other nodes are placed from the beginning at their corresponding locations. The experiment area is of about 400x300 m. Each experiment run lasted for 10 minutes. We used either flood or dtlsr as routing protocols for DTN2 and flood for IBR-DTN. Other conditions were: transmit power 10 dBm (802.11b) and attenuation 3.32. The interval between dtnping requests was 10 s for DTN2 and 1 s for IBR-DTN (not configurable).

We conducted four types of such experiments with dtnping:

- *5 mobile*: All the 5 mobile nodes and the gateway send dtnping towards the gateway GW0;
- *1 mobile*: Only the mobile node #1 and the gateway send dtnping towards the gateway GW0;
- *5 fixed*: A total of 5 fixed nodes, #8, #11, #15, #17 and #20, and the gateway send dtnping towards GW0;
- *1 fixed*: Only fixed node #8 and the gateway send dtnping towards the gateway GW0.

The percentage of successful dtnping replies for these experiments is shown in Figure 3. We observe that DTN2 flood

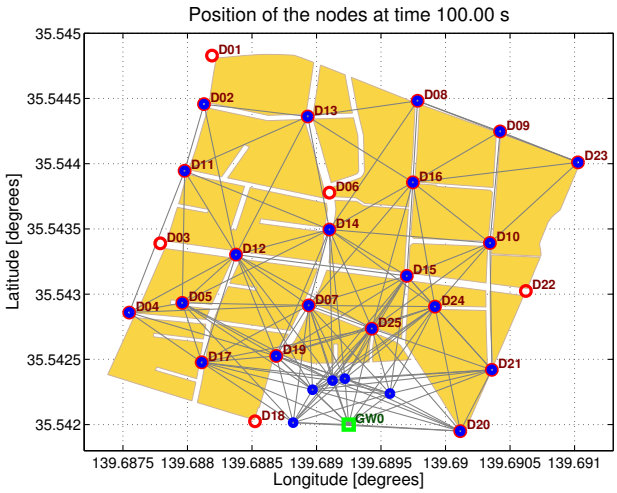

Fig. 2. Urban mobility scenario for DTN emulation experiments with 5 mobile nodes out of 26.

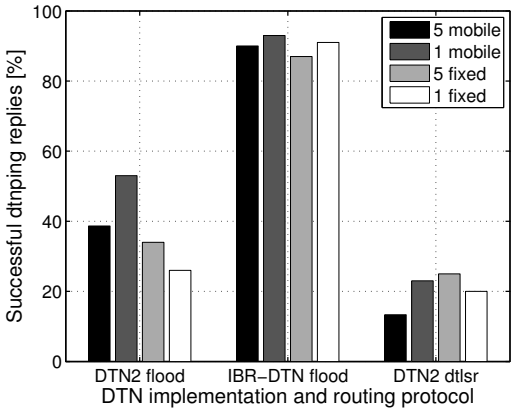

Fig. 3. Successful dtnping replies in DTN emulation experiments with 5 mobile nodes out of 26.

results are better than DTN2 dtlsr because of the higher success probability given by the flooding mechanism. Overall, DTN2 flood results are between 20% and 50%, whereas DTN2 dtlsr are between 10% and 30%. For comparison, in exactly the same circumstances and with a 10 times higher data rate per sending node IBR-DTN has a success rate of around 90% in all the tested scenarios.

A preliminary investigation showed that the bundle processing overhead in DTN2 is rather high, and the gateway, which is the destination of the dtnping request becomes quickly overloaded, with CPU utilization around 100% even on the Intel Pentium 4 3.2 GHz CPUs that we used. The difference with respect to IBR-DTN emphasizes the fact that there is still room for performance optimization of DTN2, and we believe it should be the focus of the next iterations of its development.

V. RELATED WORK

To the best of our knowledge, no equivalent to our DTN emulation testbed exists. However, below we present some related projects and research work.

DTN implementations have been previously evaluated, but only at a small scale. For instance, in [9] two scenarios with one sender, one receiver and up to 4 hops are compared through emulation experiments done on Emulab. The work in [17] uses 4 real wireless nodes for the evaluation, and includes a low-level performance analysis.

On the other hand, there is a considerable number of DTN evaluations at large-scale through simulation, such as [12], which used a realistic scenario with 600 vehicles. By contrast, our testbed allows to assess performance of real DTN implementations in large-scale scenarios, hence leads to results that have a direct practical application.

A DTN testbed using real nodes is DTN-Bone, described as an "effort to establish a worldwide collection of nodes running DTN bundle agents and applications" [6]. This testbed currently connects around 9 institutions, but makes available only a little more than a dozen nodes. Hence, we position DTN-Bone more as an inter-operability testbed (given that 5 different implementations of DTN are being run on it), rather than a testbed for DTN performance evaluation. The DTN testbed presented in [11] also includes only 12 geographically-spread nodes. Although not a DTN testbed *per se*, but a more generic WLAN testbed, the TWINE project takes an approach similar to our emulation testbed, albeit with more limited support in regard to wireless standards, mobility, etc. [21].

VI. Conclusion

This paper presented an emulation testbed intended for DTN application and protocol experiments. We discussed in detail the main changes that were necessary in order to make possible DTN experiments on the QOMB wireless network emulation testbed, such as multi-interface support, fault injection mechanisms, etc.

Several series of experiments, including some that were not detailed in this paper, demonstrated the practicality of using our testbed for assessing the performance characteristics of DTN2 and IBR-DTN implementations. Our results have shown that DTN2 exhibits poor performance in scenarios with as few as 26 nodes, whereas IBR-DTN behaved as expected in all the tested scenarios. This emphasizes the need to perform repeatable large-scale experiments with DTN applications and protocols.

Our testbed makes possible performance optimization procedures, by allowing to both identify performance bottlenecks through precise controlled experiments, and to test the improved implementation in exactly the same scenarios, so as to confirm that the problems were fixed. Only such procedures guarantee that applying the DTN paradigm to real-life scenarios can be done without affecting network performance.

References

[1] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, M. Schwamborn, *BonnMotion: a mobility scenario generation and analysis tool*, Proc. of the 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools'10), Malaga, Spain, March 15-19, 2010.

[2] R. Beuran, *Introduction to Network Emulation*, Pan Stanford Publishing, 2012.

[3] R. Beuran, L. T. Nguyen, T. Miyachi, J. Nakata, K. Chinen, Y. Tan, Y. Shinoda, *QOMB: A Wireless Network Emulation Testbed*, IEEE Global Communications Conference (GLOBECOM 2009), Honolulu, Hawaii, USA, November 30-December 4, 2009.

[4] R. Beuran, S. Miwa, Y. Shinoda, *Making the Best of Two Worlds: A Framework for Hybrid Experiments*, ACM Intl. Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH 2012), in conjunction with MobiCom 2012, Istanbul, Turkey, August 22-26, 2012, pp. 75-81.

[5] R. Beuran, S. Miwa, Y. Shinoda, *Performance Evaluation of DTN Implementations on a Large-scale Network Emulation Testbed*, ACM Intl. Workshop on Challenged Networks (CHANTS 2012), in conjunction with MobiCom 2012, Istanbul, Turkey, August 22-26, 2012, pp. 39-42.

[6] Delay-Tolerant Network Research Group, *DTN-Bone*, Internet Research Task Force, http://dtnrg.org/wiki/DtnBone.

[7] Delay-Tolerant Network Research Group, *DTNRG home page*, Internet Research Task Force, http://www.dtnrg.org/wiki.

[8] M. Carbone, L.Rizzo, *Dummynet revisited*, ACM SIGCOMM Computer Communications Revue, Vol. 40, No. 2, April 2010.

[9] M. Demmer, E. Brewer, K. Fall, S. Jain, M. Ho, R. Patra, *Implementing Delay Tolerant Networking*, Intel Research Technical Report, IRB-TR-04-020, December 2004.

[10] K. Fall, *A Delay-Tolerant Network Architecture for Challenged Internets*, Intel Research Technical Report, IRB-TR-03-003, February 2003.

[11] E. Koutsogiannis, S. Diamantopoulos, G. Papastergiou, I. Komnios, A. Aggelis, N. Peccia, *Experiences from architecting a DTN Testbed*, Journal of Internet Engineering, Vol. 3, No.1, Dec. 2009, pp. 219-229.

[12] P. Luo, H. Huang, W. Shu, M. Li, M. Wu, *Performance Evaluation of Vehicular DTN Routing under Realistic Mobility Models* Proc. of IEEE Wireless Communications and Networking Conference (WCNC 2008), Las Vegas, Nevada, USA, March 31-April 3, 2008, pp. 2206 - 2211.

[13] M. Meeker, S. Devitt, L. Wu, *Internet Trends*, Morgan Stanley, Report, April 2010.

[14] M. Meeker, L. Wu, *Internet Trends 2011*, Kleiner Perkins Caufield Byers, Report, October 2011.

[15] T. Miyachi, R. Beuran, S. Miwa, Y. Makino, S. Uda, Y. Tan, Y. Shinoda, *Fault Injection on a Large-Scale Network Testbed*, Asian Internet Engineering Conference (AINTEC 2011), Bangkok, Thailand, November 9-11, 2011.

[16] T. Miyachi, K. Chinen, Y. Shinoda, *StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software*, Intl. Conf. on Performance Evaluation Methodologies and Tools (Valuetools 2006), ACM Press, Pisa, Italy, October 2006.

[17] E. Oliver, H. Falaki, *Performance Evaluation and Analysis of Delay Tolerant Networking*, Proc. of ACM/USENIX Conference on Mobile Systems, Applications, and Services (MobiSys 2007), MobiEval Workshop, Puerto Rico, June 2007.

[18] S. Schildt, J. Morgenroth, W.-B. Pottner, L. Wolf, *IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation*, Electronic Comm. of the EASST, Vol. 37, 2011, pp. 1-11.

[19] K. Scott, S. Burleigh, *Bundle Protocol Specification*, RFC 5050, IETF, November 2007.

[20] I. Seskar, K. Nagaraja, S. Nelson, D. Raychaudhuri, *MobilityFirst Future Internet Architecture Project*, ACM AINTEC 2011, Bangkok, Thailand, November 9-11, 2011.

[21] J. Zhou, Z. Ji, R. Bagrodia, *TWINE: A hybrid emulation testbed for wireless networks and applications*, IEEE INFOCOM 2006, Barcelona, Spain, April 23-29, 2006.