Delay-Tolerant Network Protocol Testing and Evaluation

Yong Li, Pan Hui, Depeng Jin, and Sheng Chen

ABSTRACT

Delay-tolerant networks, DTNs, are characterized by lacking end-to-end paths between communication sources and destinations. A variety of routing schemes have been proposed to provide communication services in DTNs, and credible and flexible protocol evaluation tools are in demand in order to test these DTN routing schemes. By examining the evolution of DTN protocol testing and evaluation, this article discusses the trend toward large-scale mobility trace supported emulation, and we propose TUNIE, a large-scale emulation testbed for DTN protocol evaluation based on network virtualization. Unlike the existing simulation tools and real-life testbeds, which either cannot provide a realistic DTN environment setup or are too costly and time-consuming, our proposed TUNIE architecture is capable of simulating reliable DTN environments and obtaining an accurate system performance evaluation. By system prototype and implementation, we demonstrate TUNIE as a flexible platform for evaluating DTN protocol performance.

INTRODUCTION

Delay-tolerant networks (DTNs) [1, 2] have attracted lots of attention in the past 10 years, and many related interesting applications have been experimented and tested, including mobile social networks based on human mobility, sensor networks for wildlife tracking and habitat monitoring, vehicular ad hoc networks for road safety and commercial applications, and deep-space interplanetary networks. In a DTN, most of the time there are no end-to-end paths from communication sources to destinations due to node mobility, wireless propagation effects, sparse node density, and other adverse factors. For this kind of network, traditional ad hoc routing protocols, which rely on end-to-end paths, fail to work [1]. Therefore, a new routing mechanism, called store-carry-and-forward [3], was proposed to provide communication. In order to improve message delivery probability, a variety of routing schemes have been proposed, such as two-hop relaying, spray and wait, and MaxProp [3], which aim to reduce the overhead of epidemic routing. Furthermore, some of these routing schemes claim to obtain optimal system performance, and typically they attempt to achieve short message delivery delay with relatively low transmission cost. However, there is a trade-off between message delivery delay and delivery cost. Generally speaking, shorter delivery delay is obtained at the expense of higher cost, and vice versa. Therefore, it is critically important to accurately evaluate these routing schemes in order to show their advantages and drawbacks objectively.

Recently, theoretical analysis frameworks, such as Markov models [4] and ordinary differential equation (ODE) models [5], are being used to evaluate the performance of DTN protocols. However, these models are far too simplified to be capable of faithfully representing highly complicated DTNs, and the ability of these models to evaluate DTN protocols is severely limited. Therefore, more realistic simulation and experiment-based evaluation tools are needed [3, 6]. Current testing tools can be classified into two types: software-based simulation and testbed-based experimentation. Software-based simulation can be carried out using general network simulation software like NS-2 and OPNET, or specialized DTN simulation tools like ONE [7] and OMNeT++ [8]. Recently emerging real-life DTN testbeds include UMass DieselNet [9] and ORBIT [10], which are built to carry out experiments for evaluating DTN related algorithms and protocols. These testbeds offer realistic DTN environments. However, setting up an experiment in such a testbed involves vast investment in terms of money and time.

In this article, we first review the evolution of the protocol testing and evaluation for DTN and discuss the trend toward large-scale mobility trace supported emulation by surveying the emerging approaches to DTN protocol testing. In order to overcome the shortcomings of both the existing simulation tools and experimental testbeds, we propose TUNIE, a large-scale emulation testbed for DTN protocol evaluation based on network virtualization, which offers the following highly desired features. First, TUNIE enables the implementation of realistic environments for credible evaluation of DTNs by controlling the data transmission through regulated wired and wireless links. Second, it provides deep programmability of networking functions to customize system-level parameters as well as

Yong Li and Depeng Jin are with Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University.

Pan Hui is with Hong Kong University of Science and Technology, Telekom Innovation Laboratories, and Aalto University.

Sheng Chen is with the University of Southampton and King Abdulaziz University.

This work is supported by National Basic Research Program of China (973 Program) (No. 2013CB329105), National Nature Science Foundation of China (No. 61301080, No. 91338102, 61321061, and No. 61171065), National High Technology Research and Development Program (No. 2013AA013501 and No. 2013AA013505), and China's Next Generation Internet (No. CNGI-12-03-007).

0163-6804/15/$25.00 © 2015 IEEE

Categories	Main features	Representative platforms
General simulation software	Software simulated networks, simple mobility models	NS-2, NS-3, OPNET
DTN specific platform	Discrete event simulator, realistic mobility trace integration	OMNeT++[8], The ONE[7]
Experimental testbed	Realistic wireless environment, realistic hardware and system	UMass DieselNet[9], ORBIT[10]

Table 1. Comparison of approaches and tools in DTN protocol testing and evaluation: from simulation to experiment.

The key factors that determine DTN performance are the routing and forwarding algorithms used, and how well their design assumptions match the actual mobility patterns. Many routing schemes have been proposed. Simulation and subsequently experiment testing play an important role in evaluating these DTN protocols.

abundant mobility environments to enable the experimenters to repeat their different evaluations. Third, TUNIE is remotely accessible and sharable by the research community, which substantially reduces the capital costs and human effort required to perform experiments. We implement a prototype of TUNIE to demonstrate that it offers a flexible platform for simulating realistic DTN environments and evaluating DTN protocol performance.

The rest of this article is organized as follows. After reviewing the current testing tools and platforms, we provide the design goals of TUNIE and describe the experiment workflow of TUNIE. We then describe the deployment of TUNIE and conduct some preliminary experiments in our implemented TUNIE to show its flexibility as a DTN performance evaluation platform. We conclude the article in the final section.

From Simulation to Experiment

The performance of a DTN may vary significantly, depending on how the mobile nodes move, how densely the nodes are distributed, and how far apart the sender and receiver are. The key factors that determine DTN performance are the routing and forwarding algorithms used, and how well their design assumptions match the actual mobility patterns. Many routing schemes have been proposed. Simulation and, subsequently, experiment testing play an important role in evaluating these DTN protocols. Table 1 categorizes the testing tools into simulation and experiment-based classes, and summarizes the main features of different schemes.

Simulation Testing

Simulation testing for DTN protocol evaluation began with the use of general network simulation software, such as NS-2, NS-3 and OPNET. These simulation tools are designed for general networks, which also include wireless and mobile networks. Therefore, they have been used for DTN protocol evaluation, particularly in the early days. As mobility patterns are important for characterizing DTNs, mobility generators based on simple models are available for NS-2 and NS-3 as part of their toolsets or as specific extensions.

OMNeT++ [8] is a public source simulation platform that has primarily been used for simulating communication networks, and [8] proposes mechanisms for simulating DTN in the OMNeT++ discrete event simulator. These mechanisms allow open systems of wireless mobile nodes to be simulated, where mobility or contact traces are used to drive the simulation. In this approach, the mobility generations, which are separate from the core OMNeT++ protocol simulations, facilitate importing synthetic or real data from external mobility generators, real mobility tracking data, or real contact traces.

While NS-2 and OPNET can offer sound generic open simulation platforms for packet-based communications, and OMNeT++ is embedded with the specific settings to simulate node mobility, generic support for DTN simulation in these platforms is fairly limited. To alleviate this drawback, the ONE simulator [7] contributes an environment for DTN protocol evaluation with embedded internal and external mobility models, enabling different DTN routing schemes and interactive inspection. ONE is an agent-based discrete event simulation engine. At each simulation step, the engine updates a number of modules that implement the main simulation functions, which include the modeling of node movements, inter-node contacts, routing, and message handling. Result collection and analysis are done through visualization, reports, and post-processing tools.

Experimental Testing

Using the above-mentioned simulation tools, however, it is difficult to realize realistic DTN environments in terms of both physical properties and wireless network phenomena. This is because the mobility, channel, and radio characteristics, power consumption, and many other features of wireless mobile networks interact in very complex relationships, and these software simulators simply cannot faithfully reproduce these highly complex and interdependent characteristics. To address these challenges, ORBIT [10] and UMass DieselNet [9] have been designed and built for realistic mobile networking experimentation.

The ORBIT radio grid testbed was developed for scalable and reproducible evaluation of next-generation wireless network protocols by providing a flexible, open-access, multi-user experimental facility. The ORBIT testbed consists of an indoor radio grid emulator for controlled experimentation and an outdoor field trial network for end-user evaluations in real-world settings. This testbed currently consists of

TUNIE stands for Tsinghua University Network Innovation Environment. Our design focuses on providing a reputable and controllable emulation testbed for accurate DTN protocol evaluation using the technology of network virtualization.

400 wireless nodes having 802.11a/b/g wireless cards laid out in a 20 × 20 grid. In this way, DTN protocols can be assessed by the experimenter via an Internet portal, which provides a variety of services to assist the user with setting up network topology, programming radio nodes, executing experimental code, and collecting measurements.

DieselNet consists of computer-equipped buses, battery-powered nomadic nodes, organic WiFi APs, and a municipal WiFi mesh network serving the area surrounding the University of Massachusetts, Amherst campus. More specifically, UMass DieselNet consists of 40 buses, each carrying a small-form desktop computer with 40 GB of storage and a GPS device. Each bus operates a 802.11b radio that scans for other buses 10 times a second and an 802.11b access point that accepts incoming connections. It is a realistic vehicular DTN testbed, and protocols to be evaluated can be implemented on DieselNet as the first step toward real-world deployment. Moreover, testing on DieselNet allows the experimenter to study the effects of certain critical events, such as delays caused by computation, wireless channel interference, and operating system delays, which could not be perfectly modeled in a software simulator.

Trends

Because software-based simulation has difficulties in realizing realistic wireless link and node mobility properties, it is hard for a software simulator to emulate a complicated DTN environment. Recently, emerging real-life DTN testbeds, like DieselNet and ORBIT, offer realistic DTN environments, and they are built to carry out experiments for evaluating DTN related algorithms and protocols. However, setting up an experiment in such a testbed involves huge costs; moreover, the wider research community may not have access to these testbeds. Furthermore, it is very difficult to set up a truly large-scale experiment in these testbeds. It is highly desired that DTN testing platforms are supported by real-world large-scale mobility traces and are flexible to realize. On the other hand, the recently emerging testbed MoViT [11] shows that emulation is an effective and useful approach to evaluate mobile networks. Based on this trend and the above motivations, we propose a large-scale DTN testbed, TUNIE, which exploits virtualization technology and OpenFlow [12] to implement a realistic and reliable DTN environment, while providing remote accessing and sharing for the research community to emulate customized system-level parameters. Thus, TUNIE offers a realistic testing environment and is convenient to use.

TUNIE System Design

TUNIE stands for Tsinghua University Network Innovation Environment. Our design focuses on providing a reputable and controllable emulation testbed for accurate DTN protocol evaluation using the technology of network virtualization. In this section, we first describe the system goals, and then provide the details of the system in terms of controller, node and link design.

Design Goals

The primary design of TUNIE is to enable controllable and reliable performance evaluation for DTN protocols. Specifically, TUNIE is designed to achieve the following goals.

High Credibility — To avoid inaccurate evaluations of DTN protocols, which may happen when using simulation tools like NS-2 and ONE, TUNIE should provide an accurate assessment approach that reflects the realistic DTN environment. This environment should include realistic node mobility scenarios, wireless link layer properties, and system-level parameters in network layer implementations. The best choice would be to use a realistic wireless testbed. However, the huge cost in terms of time and money involved makes it unrealistic to implement such a wireless testbed. In TUNIE, we rely on a virtualized testbed to implement realistic DTN environments by controlling the data transmission through regulated wireless links, which exhibit intermittent connectivity with time-varying interference, bit rates, error rates, and transmission delays. At the same time, these realistic experiment environments can easily be configured by remotely accessing the central control center.

System-Level Parameter Realization — New DTN algorithms, protocols, and architectures often require specific customizations of system-level parameters. Therefore, the experiment platform must provide deep programmability of networking functions to implement and emulate system-level parameters, such as operation system (OS) information, network stack settings, distributed protocol states, and interactions on different modules. Consequently, TUNIE should offer sufficient customization to enable the implementation of these system-related parameter details in terms of network stack and transmission links in the deployment of new algorithms and protocols. In the implementation of TUNIE, we use virtual machines supporting a customized OS to run full network stack implementation with new routing protocols and applications, which process the network packets with system-level parameters of processing delay and buffer size, and use a centralized control center to control the link behavior according to different mobility models, and passes through a wild wireless interface as required by the experimenter, which results in real-world link parameters of packet loss rate, transmission delay, and so on.

Remotely Accessible and Repeatable Experiments — In the DTN protocol evaluation, we need to validate the performance of the new algorithms using different mobility scenarios. Therefore, a testbed emulator should provide abundant mobility environments to enable the experimenters to repeat their different evaluations and experiments, in which similar environments can be repeated to obtain similar results. Moreover, it should be remotely accessible to provide the DTN research community with a convenient and practical platform to use. TUNIE offers the experimenter a wide range of choices

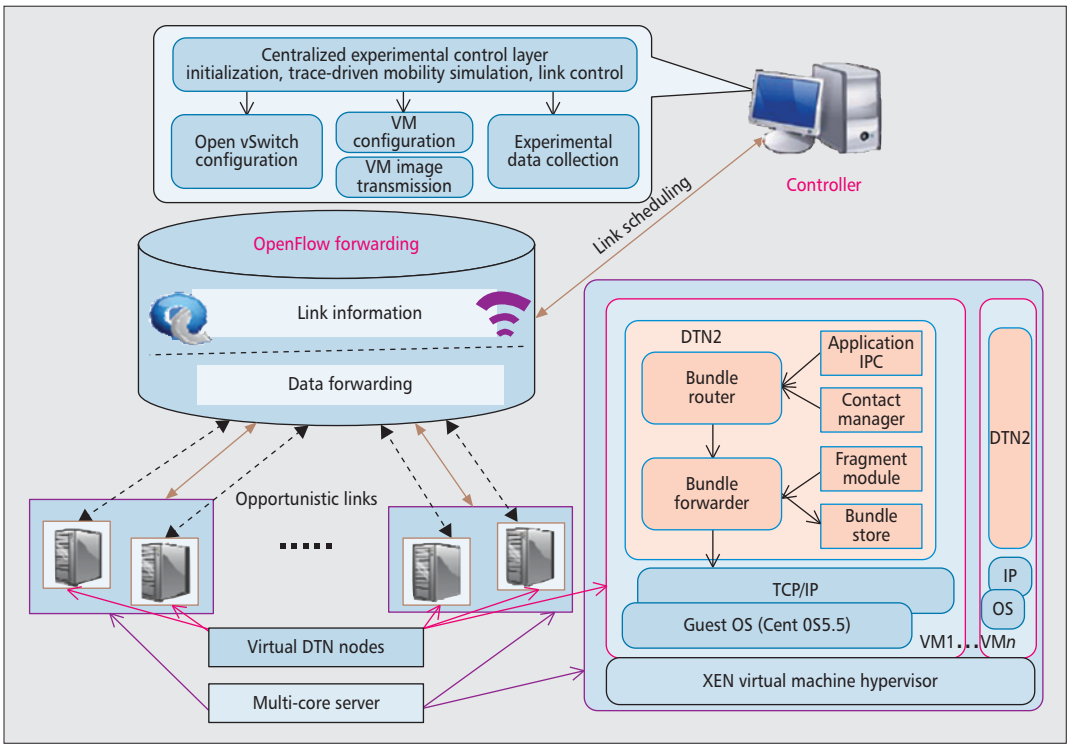

Figure 1. TUNIE architecture: network virtualization based DTN emulation testbed overview.

To enable large-scale experiments and sharable access by many users, TUNIE uses the operation system based network virtualization technology to allow many logical nodes to operate on the same shared physical infrastructure for efficient utilization of the available infrastructure among different experiments.

in terms of mobile environments, and gives the experimenter a high level of control over protocols and software used on the network nodes. In particular, it allows the experimenter to implement a customized system by remote operation and to deal with the issues occurring during the emulation robustly.

Scalability and Shareable Access — Properly evaluating a DTN protocol requires testing its scalability by changing the system size. With the existing testbeds, it is hard to support a large number of nodes, say thousands. On the other hand, as an evaluation platform, a testbed emulator should be a public facility to be shared among many researchers. To enable large-scale experiments and sharable access by many users, TUNIE uses operation system based network virtualization technology to allow many logical nodes to operate on the same shared physical infrastructure for efficient utilization of the available infrastructure among different experiments. The virtual nodes behave like realistic nodes with real-life node mobility and system parameters of network layer implementations, and a unified management system controls the network resource for effective node allocation and usage by all experiments. Due to TUNIE's remote access and sharing capability, the capital costs and human effort required to perform an experiment are substantially reduced, which makes our testbed emulator more economical to run.

System Overview

In order to support large-scale experiments and node customization for credible DTN protocol evaluation, we use the OS and network virtualization technologies to set up virtual DTN emulation platforms. In every DTN node of the TUNIE architecture, an integral of protocol stacks is deployed, which includes the DTN bundle router and bundle forwarder modules, based on the OS visualization. Therefore, the experimenter can easily set up typical routing and forwarding protocols in a DTN node, such as epidemic routing, two-hop forwarding, and spray and wait, or other customized routing and forwarding protocols. To simulate realistic communication links in the DTN paradigm, which relies on intermittent opportunistic connection to transmit packets, we use OpenFlow [12] to control the link events, such as link up and down, according to different inputs of the mobility scenarios. The bandwidths of the transmission links are controlled by the centralized controller, and they vary with time and space. Moreover, we further use the link virtualization technology to control each real-time transmission on the opportunistic links. In this way, we make a virtual wireless interface behave like a real wireless link, including the realistic behaviors of time-varying interference, bit rate, error rate, and transmission delay. This is in contrast to many existing DTN simulators, which either neglect the details of wireless link characteristics and simply assume that any two nodes can communicate with each other when they are in the transmission range of each other, or use the simple time-varying link transmission model to simulate the wireless transmission behaviors as in ONE [7].

The TUNIE architecture is shown in Fig. 1, where two important features can be observed. First, we use OS virtualization to emulate DTN nodes. Specifically, we use a XEN virtual machine hypervisor to run a series of virtual machines as DTN nodes. Each DTN node contains a CentOS supported realistic network stack that includes the upper layer of TCP/IP and core

Data transmissions in TUNIE undergo real-word packet loss rate, transmission delay, and throughput fluctuation. Thus, these important system-level parameters are built in naturally to help obtain accurate and correct performance results for the DTN protocol evaluations.

layer of DTN, which allow users to program each node to customize their own designed algorithms and protocols. Second, OpenFlow is used to control the links of all DTN nodes in order to ensure that they behave like the required real opportunistic links. Specifically, a controller running in a PC transforms the mobility settings given by the experimenter into the link up and down events, and sends these events to the OpenFlow switches, which connect all the virtual nodes through wired and wireless links. The OpenFlow switches control the connectivity between all node pairs to ensure that the transmissions occur in the required opportunistic way, and the wild wireless transmissions ensure they behave as they should in a realistic wireless network. Next, we detail the important components of TUNIE, including node architecture, link structure, and controller.

Node Design

The DTN node in TUNIE is a software-based solution that provides the virtual environment for running a DTN protocol stack and specific applications for the required protocol evaluation settings. As shown in Fig. 1, we use XEN virtualization technology [13] in our system, where multiple concurrent virtual machines running the Linux OS CentOS coexist in a physical machine, and a DTN node is implemented as a virtual machine. In the virtual machine, the core component is the DTN protocol stack, which is running on the TCP/IP protocol stack. In our deployment, we use the open source implementation of DTN bundle protocol, DTN2 [14]. In this DTN stack, Bundle Router and Bundle Forwarder are the two most important components to implement the bundle protocol. Specifically, Bundle Router makes the routing decision based on the application requirements and contact information generated by the contact manager. In the virtual machine, most general router algorithms, including epidemic routing, two-hop relaying, and spray and wait [3], are available to provide the routing functionality. Through the user configuration interface, experimenters can choose their favorite routing components or implement a new one. The physical machine host provides the software interface between the processes in different virtual machines. For a Bundle Forwarder, we use the Click [15] engine to configure the router decision from the Bundle Router into the Click data plane. It then forwards the data into the network interface through the bundle store according to the decision of the fragmentation module. All the packet processing, including the TCP/IP layer packet disassembly and congestion control, will induce delays in packet forwarding. At the same time, the DTN stack with a buffer for message storing, and the buffer size, which is an important system parameter for determining the system performance, are open for user configuration. TUNIE is able to emulate a real DTN system with all these system-level features of the network stack, which are impossible or hard to capture by simulation tools and analytical methods.

The host uses an OpenFlow vSwitch (OVS) [16] to connect virtual nodes, to be discussed later. The OVS can control the bandwidth of each virtual node to share the physical link capacity. After receiving the link bandwidth sent by the controller according to the transmission scheduling, the host can dynamically change the bandwidth. This control mechanism is enabled by the XEN network interface virtualization scheme, where the host can view and control the virtual interface in each virtual node.

Link Design

The virtual link is a key component of TUNIE. We use the link virtualization technology to enable the emulation of wireless opportunistic links. We first set up a virtual interface, which is a tap device, for each virtual node. We then use the software bridge of the OVS in the XEN virtualization environment to connect all the tap devices with the WiFi physical interface in the host node, which itself is connected to the OpenFlow access point (AP). The OVS and OpenFlow AP together let any two virtual nodes set up a link at any time, and can also block any connection at any time. The link control is achieved by the flow management in OpenFlow, which is further controlled by the centralized controller. The link view of TUNIE is illustrated in Fig. 2, where it can be seen that the OpenFlow AP connects the physical multi-core servers through the WiFi interfaces, and this in turn allows virtual nodes to connect with each other through the wireless links. With the OpenFlow link controlling function, all the links among node pairs are controlled by the system; therefore, it is easy for experimenters to simulate different mobility scenarios. On the other hand, the controller controls all the transmissions over virtual links either across multi-servers or insider one physical server, illustrated by virtual link a and b, respectively, in Fig. 2, as they pass the wild WiFi interfaces. Consequently, wireless characteristics, such as radiation patterns of the antennas, are taken into account in the emulation testbed. In this way, data transmissions in TUNIE undergo real-word packet loss rate, transmission delay, and throughput fluctuation. Thus, these important system-level parameters are built in naturally to help obtain accurate and correct performance results for the DTN protocol evaluations.

Controller Design

The controller is a centralized network link control component in the OpenFlow network, which is running on the network OS and has an overview of the network. The controller can inject specific flow rules into the OpenFlow switch according to the behaviors of the network, which depend on the routing algorithm, load balance, and so on. Our TUNIE controller is built on OpenFlow NOX, and is implemented in Java and Python languages. Specifically, we develop a centralized experiment control layer in NOX. The input parameters of this control layer include the number of nodes, mobility model, and emulated applications. Based on the given input, the control layer provides the functions of VM Image transmission and configuration to set up virtual DTN nodes, open vSwitch configuration to control the links, and experimental controlling and data collection, as shown in Fig. 1. To control the experiment, for instance, it first

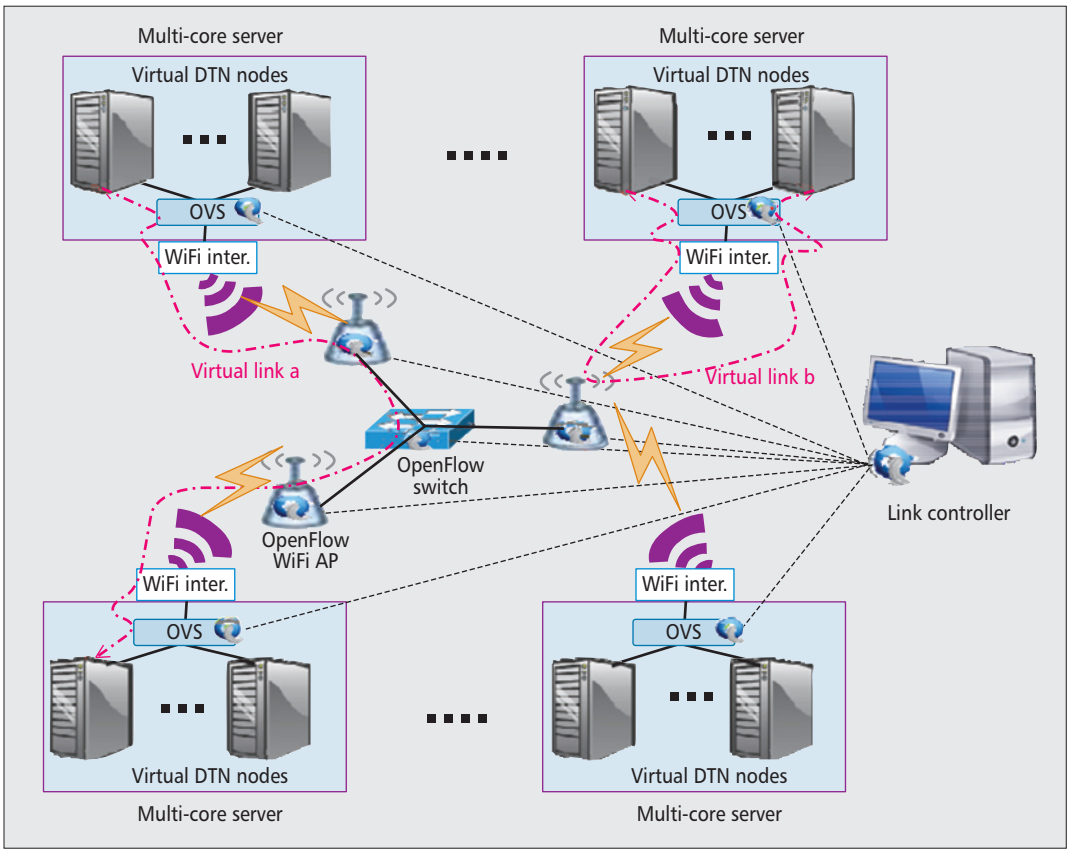

Figure 2. Link design and virtual links in TUNIE.

computes the opportunistic links among all nodes based on the given mobility model to obtain the specific link up and down events. Second, it calculates the data transmission rate of each opportunistic link according to the positions of the communicating nodes. After obtaining the link events and transmission rates, it computes the flow rules at different times, and then sends them to the OpenFlow switch at the appropriate times, as well as deleting certain rules after their link times are done. With this mechanism, we can use the OpenFlow switch to control the links of different nodes to emulate an opportunistic DTN.

TUNIE Experiment Working Flow

The experiment working flow of TUNIE is depicted in Fig. 3. When setting up an experiment in TUNIE, the user needs to configure the controller first by setting the overall network parameters, including the number of nodes, mobility model, and so on. TUNIE will generate the virtual nodes by communicating with the required multi-core servers. The user then needs to log in to the virtual nodes to configure and prototype the algorithms and protocols. The TUNIE controller will translate the experiment settings into link control events and, through the OpenFlow flow rules, controls the links among virtual nodes to ensure that they behave as the required opportunistic links. We now illustrate the experiment working flow by introducing our mobility scenarios, and routing and application settings implemented in TUNIE.

Mobility Scenarios

Mobility is the most important consideration for DTN performance evaluation. Usually, different performance evaluations of a protocol require different mobility models. For example, to evaluate the scalability properties, we may adopt a random mobility model, with which the scale of the network is easy to change. On the other hand, to evaluate the transmission efficiency, we need real traces covering different mobility scenarios in order to get accurate results. To satisfy these different requirements, in TUNIE we integrate different mobility scenarios in the controller, including random mobility models, map-based mobility, and real-world human and vehicular mobility traces, as indicated in Fig. 3.

In terms of random mobility, we cover a broad set of random mobility models by including random waypoint, random walk, and random direction walk. For map-based mobility, we use a map model in the ONE simulator [7]. For realistic human mobility, we integrate four human mobility traces, *Infocom05*, *Infocom06*, *Reality*, and *Cambridge*, in the system. Among these four human mobility traces, Reality was collected from the MIT Reality Mining Project, and the other three were gathered by the Haggle Project. We also integrate two vehicular mobility traces, *Shanghai* and *Beijing*. The Shanghai trace was collected by involving 2019 operational taxis in Shanghai over the whole month of February 2007 without any interruptions. The Beijing

Related to the routing protocols, users can choose epidemic routing, two-hop relaying, and spray and wait, which are available in the system for configuring the DTN nodes. Users can also implement new routing and resource allocation schemes to evaluate their own protocols.

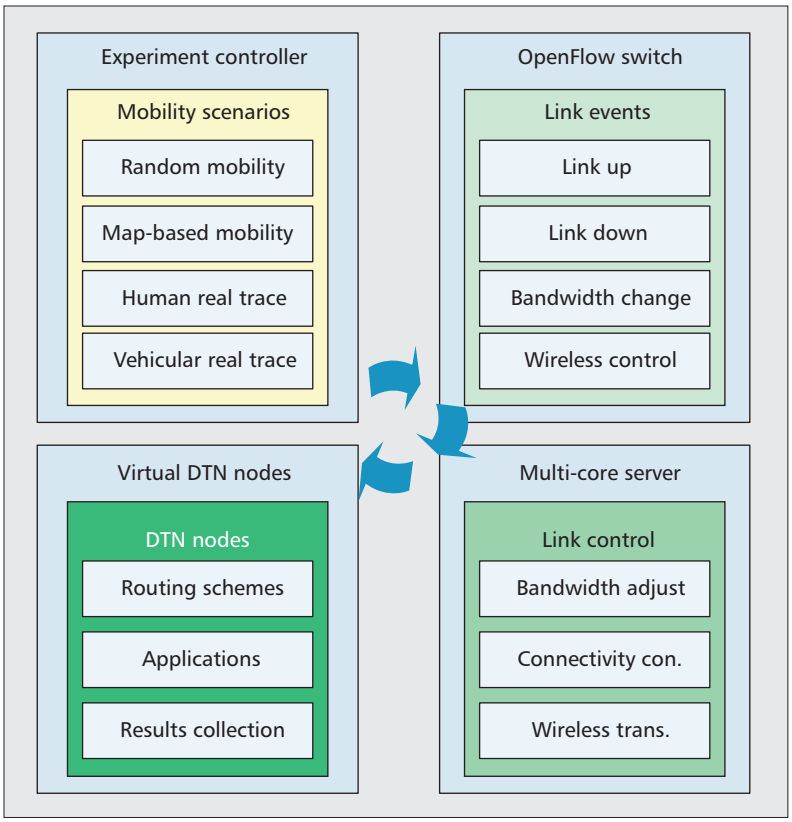

Figure 3. System functions and experiment workflow of TUNIE.

trace includes the mobility traces of 27,000 participating Beijing taxis collected during the entire month of May in 2010, which is the largest vehicular data trace available.

Routing and Applications

Related to the routing protocols, users can choose epidemic routing, two-hop relaying, and spray and wait, which are available in the system for configuring the DTN nodes. Users can also implement new routing and resource allocation schemes to evaluate their own protocols. All these functions are openly accessible in the DTN Bundle module of the DTN stack. In terms of applications, similar to the ONE simulator, TUNIE provides two ways to generate application packets in the virtual nodes: packet generators and external traffic event files. The packet generator in TUNIE creates packets for the selected source and destination with the given packet size and deadline, which are set by the user. A separate tool to generate packet event files is also included in TUNIE. In this way, users can generate messages very conveniently. For both ways of generating application packets, the application is implemented in the module of the Application IPC, which communicates with the router (i.e., injects bundles into the Bundle Router) via an inter-process communication channel in the experiment control layer.

Implementation and Deployment

In implementing the TUNIE design on hardware devices, we use 40 high-performance servers with Intel Xeon X5550 2.6 GB four core CPU, 16 GB memory, and WiFi interface as the hosts to provide the virtual DTN nodes. For the OpenFlow switches, we use 8 WiFi APs of a Broadcom chip and 3 Pronto OpenFlow-enabled switches as the OpenFlow devices, where the WiFi APs provide the wireless connections of the servers through their WiFi interfaces. The system-level parameters and characteristics are summarized in Table 2, which provides detailed information, in terms of nodes, links, and DTN protocol stack, on our implemented testbed emulator. This virtualized programmable DTN emulation testbed contains the resources of software virtualization component and OpenFlow component. With our unified user configuration interface in the controller and host, experimenters have access to all the resources in the platform, and can build their own experiments to test the designed DTN algorithms and protocols. Moreover, the same implemented code of these evaluated targets can directly run on a real-life DTN system, and no translation is required from the testbed emulator to the real DTN environment.

Based on the virtualized environment, we install the virtual DTN nodes with the DTN2 protocol stack on the TCP/IP network stack, and we also install a network connecting with all the physical multi-core servers to support the platform management system. We design the system with the GUI interface by web service for users to apply the DTN node resources, and to use them by configuring and customizing the network. All these operations and configurations over the virtual environment are carried out by web-based remote access, and experimenters are able to build their own experiments concurrently in their individual and separated virtual networks. Although building TUNIE involves the cost of the required hardware servers and switches, it is significantly less than deploying a similar large-scale dedicated hardware testbed. Moreover, such a dedicated testbed can only run a single experiment, but our TUNIE testbed emulator enables multiple different experiments to run concurrently, and its hardware resources can be remotely accessed, controlled, and shared by many researchers. Additionally, unlike a dedicated DTN testbed, the invested hardware resources are not restricted to DTN experiments, and TUNIE can conveniently be used for other network experiments and application. Clearly, our TUNIE testbed emulator provides an economical, repeatable, and reliable platform for DTN protocol testing and evaluation, and it will be open to the wider DTN research community.

We carried out some basic experiments to investigate the capability of the system. The well-known ODE model is widely used to model the message propagation in DTN network [5]. Here, we verify this model in our testbed emulator to investigate the efficiency and accuracy of TUNIE as a DTN performance evaluation platform. We evaluated the accuracy of the ODE model by comparing the theoretical results obtained based on the model given in [5] with the experimental results, which were obtained by simulating the message dissemination under the epidemic routing with the random walk mobility model in TUNIE. The system settings and performance metrics obtained from the experiment on

	Category	Parameters
Node virtualization	Physical server	Intel X5550 2.6G 4 core CPU, 16 GB memory, 800 GB hard disc, and Cisco 350 WiFi adaptor.
	Virtual node	1 virtual CPU, 256 MB memory, 4 GB hard disc, virtual tap network interface, CentOS.
	Virtualization efficiency	32 virtual nodes/physical server, full virtualization, CPU utilization efficiency > 80 %.
Link virtualization	Wired port and link	Tap rate = 10,000 ± 100kb/s, Open vSwitch port rate control: max = 20 Mb/s, min = 0.5 Mb/s.
	Wireless links	Frequency: 2.4–2.4897GHz; Data Rates: 1, 2, 5.5, and 11 Mb/s; transmit power: 0-20dBm.
DTN protocol stack	Routing protocols	Static, flooding, two-hop relaying, spray and wait, delay-tolerant link state routing.
	Transmission protocols	TCP, UDP, Ethernet, BlueTooth, serial.
	Storage size setting	Bundle storage: 2 MB–10 GB; message storage: 16 KB–2 MB.

Table 2. System-level parameters and characteristics of our deployed TUNIE testbed emulator.

TUNIE in terms of one-hop transmission and network-level performance are summarized in Table 3.

We simulated the network with different network sizes of N from 100 to 1000, where 20 percent of the nodes in the network were randomly chosen to be infected, and the system was simulated 100 times. From the average deviation between the theoretical and experimental results of the average infected node ratio in Table 3, we inferred that the number of message-infected nodes computed from the ODE model agreed with the average values obtained by the experiment. On the other hand, the deviations, which were influenced by TUNIE's system-level parameters of network stack and wireless links, indicated that the deviations happened in realistic DTN environments with network stack interactions and wild wireless transmissions. These aspects could not be obtained by the analytical method, which only captured the average system performance. These results clearly demonstrate the credibility and accuracy of TUNIE as an emulation testbed to evaluate DTN performance.

Category	Metrics	Values
One-hop performance	Wireless transmission rate	5.5 Mb/s
	Average wireless link throughput	2.4 Mb/s
	Throughput standard deviation	1.3 Mb/s
	Link packet loss rate	9.5%
Network performance	The number of nodes (N)	100∼1000
	System simulated time (T)	5×10^3 s
	Average message delivery ratio	91.2%
	Average message transmission delay	1.7×10^3 s
	Average deviation of infected ratio	99% ± 8%

Table 3. System-level settings and performance of the experiment of ODE model validation in TUNIE.

Conclusions

In this article, we have reviewed the evolution of DTN protocol testing and evaluation. Based on a survey of the existing simulation tools and experimental testbeds, particularly their advantages and drawbacks, we have designed and implemented TUNIE, a network virtualization based DTN testbed emulator. TUNIE integrates XEN virtualization and OpenFlow technologies. Specifically, XEN enables setting up large-scale DTN networks, and OpenFlow is used to emulate the DTN opportunistic links. TUNIE opens up a wide range of choices to users, in terms of mobility scenarios, routing protocols, and applications, and also allows users to design their own experimental environments conveniently. Our initial implementation validated that TUNIE is an efficient platform to evaluate DTN performance, which needs to further integrate other abundant models about the network properties (i.e., link models from MoViT[11] and device characteristics (i.e., energy consumption) to simulate some specific DTN environments. TUNIE's web-based interface for remotely accessing, controlling, and sharing ensures that TUNIE will be open to the wider DTN research community.

References

[1] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," *Proc. ACM SIGCOMM 2003 Conf. Applications, Technologies, Architectures, and Protocols for Computer Commun.*, Karlsruhe, Germany, Aug. 25–29, 2003, pp. 27–34.

[2] K. Fall and S. Farrell, "DTN: An Architectural Retrospective," *IEEE JSAC*, vol. 26, no. 5, June 2008, pp. 828–36.

[3] Z. Zhang, "Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay Tolerant Networks: Overview and Challenges," *IEEE Commun. Surveys & Tutorials*, vol. 8, no. 1, 2006, pp. 24–37.

[4] Y. Li *et al.*, "Evaluating the Impact of Social Selfishness on the Epidemic Routing in Delay Tolerant Networks," *IEEE Commun. Lett.*, vol. 14, no. 11, Nov. 2010, pp. 1026–28.
[5] X. Zhang *et al.*, "Performance Modeling of Epidemic Routing," *Computer Networks*, vol. 51, no. 10, Oct. 2007, pp. 2867–91.
[6] S. Jain, K. Fall, and R. Patra, "Routing in a Delay Tolerant Network," *Proc. ACM SIGCOMM 2004 Conf. Applications, Technologies, Architectures, and Protocols for Computer Commun.*, Kyoto, Japan, Aug. 30–Sept. 3, 2004, pp. 145–58.
[7] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," *Proc. 2nd Int'l. Conf. Simulation Tools and Techniques for Commun., Networks and Sys.*, Rome, Italy, Mar. 2–6, 2009, pp. 50–55.
[8] O. R. Helgason and K. V. Jónsson, "Opportunistic Networking in OMNeT++," *Proc. ACM 1st Int. Conf. Simulation Tools and Techniques for Commun., Networks and Sys.*, Marseille, France, Mar. 3–7, 2008, pp. 76–82.
[9] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN Routing as a Resource Allocation Problem," *Proc. ACM SIGCOMM 2007 Conf. Applications, Technologies, Architectures, and Protocols for Computer Commun.*, Kyoto, Japan, Aug. 27–31, 2007, pp. 373–84.
[10] W. Ivancic *et al.*, "Experience with Delay-Tolerant Networking from ORBIT," *Proc. IEEE 4th Conf. Advanced Satellite Mobile Sys.*, Bologna, Italy, Aug. 26–28, 2008, pp. 173–78.
[11] E. Giordano *et al.*, "MoViT: the Mobile Network Virtualized Testbed," *Proc. ACM VANET '12*, Lake District, U.K., June 25, 2012, pp. 3–12.
[12] N. McKeown *et al.*, "Openflow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Commun. Rev.*, vol. 38, no. 2, April. 2008, pp. 69–74.
[13] P. Barham et al., "Xen and the Art of Virtualization," *ACM SIGOPS Op. Sys. Rev.*, vol. 37, no. 5, Dec. 2003, pp. 164–77.
[14] S. Burleigh, "Interplanetary Overlay Network: An Implementation of the DTN Bundle Protocol," *Proc. IEEE Consumer Commun. and Networking Conf.*, Las Vegas, NV, Jan. 11–13, 2007, pp. 222–26.
[15] E. Kohler et al., "The Click Modular Router," *ACM Trans. Computer Sys.*, vol. 18, no. 3, Aug. 2000, pp. 263–97.
[16] B. Pfaff et al., "Extending Networking into the Virtualization Layer," *Proc. 8th ACM Workshop on Hot Topics in Networks*, New York, NY, Oct. 22–23, 2009, pp. 1–6.

BIOGRAPHIES

YONG LI [M'09]received his B.S. degree in electronics and information engineering from Huazhong University of Science and Technology, Wuhan, China, in 2007 and his Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China, in 2012. During July to August 2012 and 2013, he was a visiting research associate with Telekom Innovation Laboratories and Hong Kong University of Science and Technology, respectively. During December 2013 to March 2014, he was a visiting scientist with the University of Miami, Florida. He is currently a faculty member of the Department of Electronic Engineering, Tsinghua University. His research interests are in the areas of networking and communications, including mobile opportunistic networks, device-to-device communication, software-defined networks, network virtualization, and future Internet.

PAN HUI received his Ph.D degree from the Computer Laboratory, University of Cambridge, and earned his M.Phil. and B.Eng. from the Department of Electrical and Electronic Engineering, University of Hong Kong. He is currently a faculty member of the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology, where he directs the System and Media Lab. He also serves as a Distinguished Scientist of Telekom Innovation Laboratories (T-labs) Germany and an adjunct professor of social computing and networking at Aalto University, Finland. Before returning to Hong Kong, he spent several years in T-labs and Intel Research Cambridge. He has published more than 100 research papers, and has several granted and pending European patents. He has founded and chaired several IEEE/ACM conferences/workshops, and served on the Technical Program Committees of numerous international conferences and workshops including IEEE INFOCOM, SECON, MASS, GLOBECOM, WCNC, and ITC.

DEPENG JIN received his B.S. and Ph.D. degrees from Tsinghua University in 1995 and 1999, respectively, both in electronics engineering. He is an associate professor at Tsinghua University and vice chair of Department of Electronic Engineering. He was awarded the National Scientific and Technological Innovation Prize (Second Class) in 2002. His research fields include telecommunications, high-speed networks, ASIC design, and future Internet architecture.

SHENG CHEN [M'90, SM'97, F'08] obtained his B.Eng. degree from the East China Petroleum Institute, Dongying, China, in January 1982, and his Ph.D. degree from City University, London, United Kingdom, in September 1986, both in control engineering. In 2005, he was awarded a D.Sc. from the University of Southampton, United Kingdom. From 1986 to 1999, he held research and academic appointments at the Universities of Sheffield, Edinburgh, and Portsmouth, all in the United Kingdom. Since 1999, he has been with the Department of Electronics and Computer Science, University of Southampton, where he currently holds the post of professor in intelligent systems and signal processing. He is a distinguished adjunct professor at King Abdulaziz University, Jeddah, Saudi Arabia. He is a Chartered Engineer (CEng) and a Fellow of IET (FIET). His recent research interests include adaptive signal processing, wireless communications, modeling and identification of nonlinear systems, neural network and machine learning, intelligent control system design, evolutionary computation methods, and optimization. He has published over 470 research papers. He is an ISI highly cited researcher in the engineering category (March 2004).