Connectivity Emulation Testbed for IoT Devices and Networks

Nadir Javed(✉) and Bilhanan Silverajan

Tampere University of Technology, Tampere, Finland
{nadir.javed,bilhanan.silverajan}@tut.fi

Abstract. This paper describes our ongoing effort in creating a distributed highly scalable and resilient platform that can model network interactions among a very large number of devices, in terms of their wireless and wired network characteristics as well as multiradio hardware capabilities. Such an emulation platform was realized as a service overlay network atop the PlanetLab distributed testbed. Our initial results suggest that the approach undertaken is highly feasible to model both device heterogeneity ranging from simple sensors to more powerful devices, as well as wireless network characteristics to customize link reliability, channel throughput as well as bandwidth availability.

Keywords: IoT · PlanetLab · Testbed · Connectivity

1 Introduction

The Internet of Things (IoT) is expected to be constituted of billions of interconnected nodes and an equally significant number of networks [1]. These nodes comprise powerful devices as well as complexity and resource limited nodes such as sensors and actuators. In addition to high-speed fixed and wireless networks, the IoT is expected to comprise lossy, unreliable and limited bandwidth networks too. Such a diversity of connected nodes and networks inevitably impacts the types of service interactions as well as network communications, in both client-server, as well as peer-to-peer configurations. Nodes such as smart phones possess hardware allowing multiple radio technologies to co-exist, enabling multi-radio communication with other nodes using links of varying bandwidth and latency. Gateway nodes also allow packets from one kind of radio technology and access network to traverse another. Wireless sensor nodes introduce multi-hop relays into the network. Obviously this implies that measuring traffic flows among disparate types of networks, and traffic characteristics of pairwise node-based interactions are not trivial. The management of these networks and nodes, as well as lookups and discovery, are challenging problems to solve, considering the deployment scale.

In this paper, a scalable device and network emulation testbed for IoT is described, that allows such investigations to occur, from the device to the network, to subsequently execute services and monitor application level behavior.

© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2014
V.C.M. Leung et al. (Eds.): TridentCom 2014, LNICST 137, pp. 146–155, 2014.
DOI: 10.1007/978-3-319-13326-3_15

This testbed, based on device-level interface characteristics and network conditions, resulted in a prototype architecture deployed atop PlanetLab [2]. PlanetLab guarantees neither constant network connectivity nor machine uptimes. Our prototype leverages both the availability of multiple hosts on demand, as well as link unreliability as positive aspects: Instantiation of emulated nodes does not impact overall system resources, while the intrinsic unreliability of host uptimes as well as connectivity can be typical of resource constrained nodes, which either go into sleep state or turn off their uplinks in an effort to conserve energy. The main objectives of our work are:

Network heterogeneity. Nodes in our testbed should feature emulation of several types of network interfaces and properties. Network connections feature diverse link characteristics, as well as link quality, packet loss and latency.

Flexibility. The testbed should serve as a foundation for wide research in deploying new types of services and applications, as well as the introduction of new application-level protocols. Such services, applications and protocols can be connection-oriented, connectionless, client-server or peer-to-peer based.

Scalability. The testbed should offer the ability to emulate and instantiate devices in the order of thousands to tens of thousands. Instantiation and management of instantiated emulated nodes should be accomplished using intuitive mechanisms that do not impact the execution nor the performance of the physical hosts atop which the emulated nodes run.

Remote Node Management. The testbed should allow remote configuration and management with a web-based front-end. Managing large numbers of nodes should be performed by allowing nodes to be tagged, for easy retrieval afterwards.

The rest of this paper is structured as follows: Section 2 presents related work. Sections 3 and 4 discuss the architecture, design as well as the implementation of important components in our testbed. Testing and verification is discussed in Section 5 while Section 6 concludes the paper.

2 Related Work

In published literature, a number of projects undertake active testbed research and deployment.

The MagNets project [3] aimed at deploying a next-generation wireless access network testbed infrastructure in the city of Berlin, where heterogeneous devices possessed by university students are allowed free access to an operator supported network. The Pan-European Laboratory (PanLab) concept [4] introduces a resource federation framework allowing multi-domain testbeds that provide heterogeneous crosslayer infrastructures for broad testing and experimentation. The SmartSantander [5] project aims to create a city-wide test facility for the experimentation of architectures, key enabling technologies, services and applications for the Internet of Things. It is conceived to provide a platform for large scale experimentation under real-life conditions. A unified testbed platform was developed to emulate LTE over Wired Ethernet, that can be used to examine the key aspects of an LTE system in realtime, including real time uplink and downlink

scheduling, QoS parameters, and Android end-user applications [6]. The Distributed Network Emulator (DNEmu) [7] investigates how realistic network experiment can be performed involving globally distributed physical nodes under heterogeneous environments where a requirement of experimentation control between the real world network and emulated/simulated networks is introduced.

3 Design

We envision an emulation testbed with various device instances as shown in Figure 1a. The PlanetLab network is abstracted as a cloud, while squares represent PlanetLab nodes atop which various internetworked device instances and their available communication links are modeled. The network capabilities of such device instances are modeled focusing on the link reliability, bandwidth limitation and possible time delay.

As Figure 1.b shows, the testbed architecture comprises a central management server controlling and managing available PlanetLab hosts, setting up device instances on these hosts and emulating their network interfaces. An interface to the PlanetLab Central (PLC) server is used to fetch detailed node information such as node locations, addresses and uptime status. PLC provides an RPC-based API for this purpose [8].

A database is needed to maintain and record the state of the testbed. It used by the management server for storing and retrieving data essential for setting up a runtime environment, such as information for configuration of device instances as well as network links and characteristics. Such information is also retrieved by the webserver to be presented to the user for management and use of the emulated devices.

User-defined tags are supported by the platform to identify PlanetLab nodes and device instances, either individually or as groups. Tags associated with nodes and device instances are stored in the database as well.

This server also provides a web-based user-interface through which the platform can be deployed and managed. Figure 2 depicts browser windows, showing

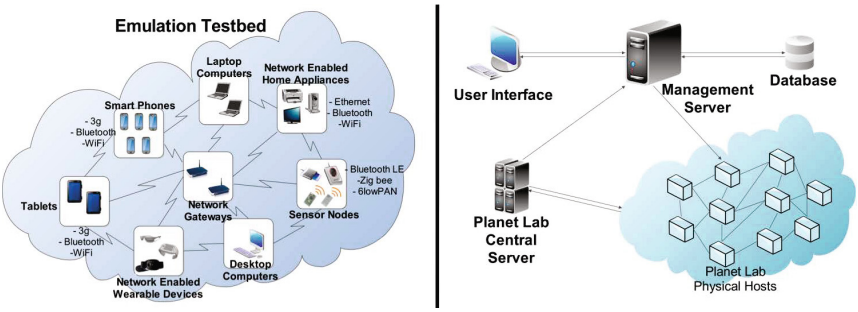

Fig. 1. a) Concept of device emulation on PlanetLab hosts b) Architecture and major components in the emulation platform

details of emulated devices such as the interfaces available for an emulated device instance, user-defined descriptions, device tags, number of instances and IDs.

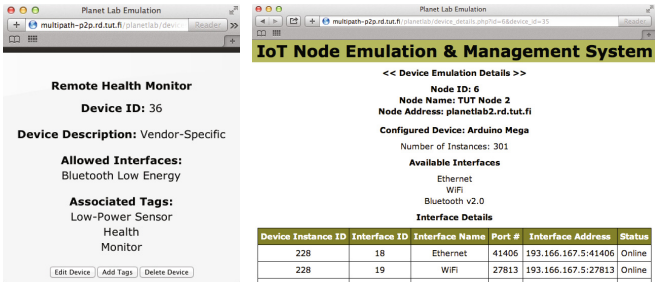

Fig. 2. Browser views of emulated devices

In this example, the user has defined device instances to emulate a remote health monitoring sensor and an Arduino device capable of communication over Bluetooth, Ethernet and WiFi based interfaces. The device ID is automatically generated and supplied by the system. The server also supplies other views aggregating all devices defined by a user, as well as physical PlanetLab nodes to be added into an existing testbed. The addition of various types of tags to be associated with PlanetLab nodes and devices to be emulated can be performed via this interface as well.

4 Implementation

Figure 3 presents a detailed view on how the emulation testbed has been implemented. The management server forms the core of the system which controls all the other components involved. The main system processes have been developed using PHP since the system uses a web based interface for user interaction. The user interface was implemented with a combination of HTML and JavaScript. To use the PLC API for fetching node information, a Python script was implemented. The database engine is based on MySQL. The server and the database use AJAX-based communication.

In order for the management server to set up the emulation testbed, it needs to be able to connect to the PlanetLab nodes defined by the user, and configure device instances. Therefore in addition to these components, SSH-based client functionality was also provided to the management server, with which it forms secure connections to PlanetLab nodes through which the required configuration commands are transmitted and the results are obtained. The SSH communication is established using public and private keys instead of passwords. The public key is uploaded to the PlanetLab Central server from where it gets propagated to all the hosts that are being used. The private key is stored on the management server and is used for authentication with the remote node.

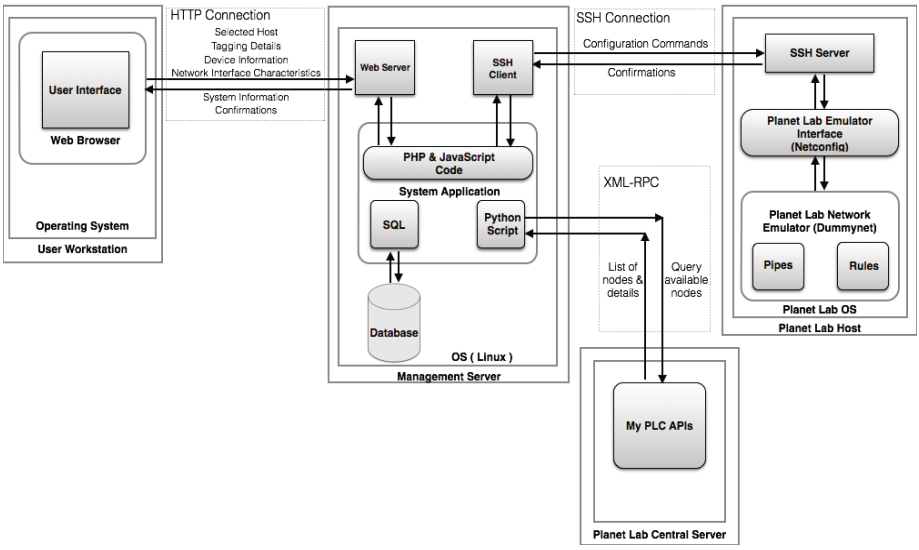

Fig. 3. Testbed implementation and existing connection types between them

PlanetLab provides a command-line tool for node-based bandwidth management. The tool is based on Dummynet, a powerful and flexible tool for testing network protocols and topologies [9]. Each physical machine runs a customized version of Dummynet that cannot be modified or replaced by end-users. Instead, a userspace command called *netconfig* is provided at each node for controlling bandwidth, network latency and packet loss ratio for incoming or outgoing connections, based on one or more known ports or addresses. In our testbed, we utilize *netconfig* to emulate both uplink and downlink of different network interfaces which are distinguished by using different port numbers. All incoming and outgoing traffic is monitored, and once a packet is detected having the same source or destination port number, rules are applied that have been specified for the traffic on that port number. For example, a sample *netconfig* configuration command invoked by an end-user for network emulation on a PlanetLab node could be:

```
host-> netconfig config SERVICE 6361 IN bw 2Mbit/s delay 2ms plr 0.2 OUT bw 1MBit/s delay 1ms plr 0.1
```

This configures the emulated link to intercept all the traffic flowing on port number 6361 and will force the incoming packets to a bandwidth of 2Mbit/s, cause a delay of 2 milliseconds and drop 20 % of the packets since the packet loss ratio is set to 0.2. Similarly all the outgoing packets will be forced to a bandwidth of 1Mbit/s, a delay of 1 millisecond and 10% packets will be dropped.

In order to start emulating the devices, the user first needs to add some nodes to the testbed, define the required network interfaces and their characteristics,

define new devices in the system and associate the existing network interfaces to the device. After the required details are present, the user selects the nodes on which the devices are to be emulated. The system application at this point queries the database for node details and the list of available devices that can be emulated on the selected nodes. This information is then presented to the user who selects the device type and the number of instances that needs to be initialized on each node. The first part of the process is represented in Figure 4 as a message sequence chart showing the communication exchange between the different entities involved.

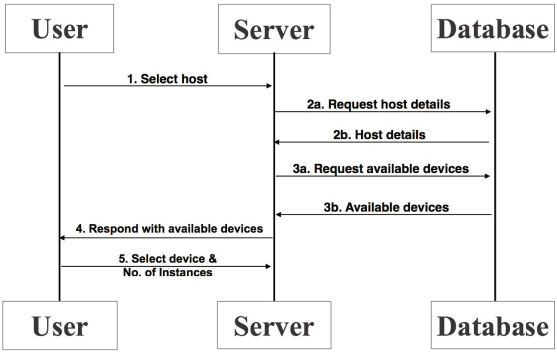

Fig. 4. First part of the message sequence for device emulation process

Once the server receives the user's choice of device and the number of instances that needs to be emulated, it queries the database for device details. The system application then queries the database for details of each associated interface. For each interface, a random port number is generated that is not being already used on the selected host for some other emulated link. Based on this port number and the interface details, a configuration command is generated by the system for execution on the node to emulate the requested network interface. This is the second part of the process and details can be seen in Figure 5.

After the configuration command is ready for all the required network interfaces of the device that is to be emulated, the system establishes an SSH connection with the selected PlanetLab node. Once the connection is established it executes in a loop all the configuration commands for each interface for each of the device instance that needs to be emulated. The system receives the result for each command that is executed and updates the records in database accordingly. Upon completion, the user is notified of the configuration results and is presented with the essential information required to utilize the newly instantiated device. This final part is represented in Figure 6.

After *netconfig* is executed on the PlanetLab node it creates the appropriate queues for handling the traffic, known as pipes, and also creates certain rules, which are then used for governing the flow of traffic through these pipes.

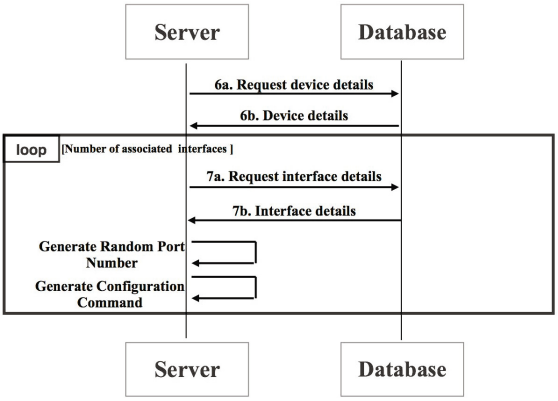

Fig. 5. Second part of the message sequence for device emulation process

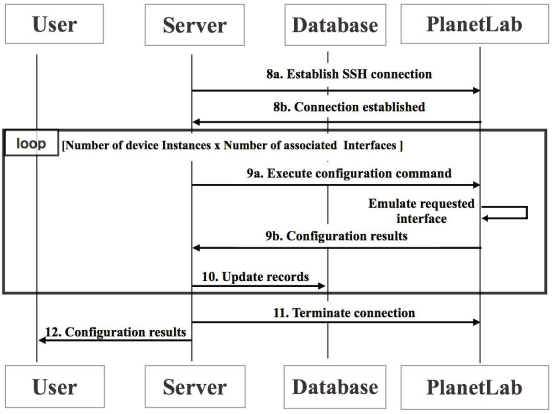

Fig. 6. Final part of the message sequence for device emulation process

Dummynet as an emulator makes use of these queues for enforcing the custom bandwidth, delays and packet drops. Every emulated network link has one or more pipes associated with it, and once the network packet matches the specified rule, it is put inside this pipe and the outward or inward flow is then controlled so that it conforms to the requested bandwidth. Similarly there could be some time constraints applied to the traffic flowing through the pipe for the latency effect and some packets could be randomly dropped from the queue depending on the packet loss ratio the user has configured.

5 Testing and Verification

The design and implementation of the emulation testbed were tested under live conditions on the PlanetLab environment. Unit testing was performed on the

management server as well as the database, before widespread device instantiation was tested over approximately 50 Planetlab nodes, although it is relatively trivial to increase the number of testbed nodes to several hundreds or thousands. On each node, we successfully tested instantiation of at least 300 emulated devices as depicted in the earlier Fig. 2.

In order to verify that our testbed is configuring network interfaces properly, we tested the configured links across different emulated devices using a network measurement utility called *Iperf* [10]. *Iperf* works as a service in our device instances, supporting both client and server mode. It allowed us to create TCP and UDP streams between two hosts and provided throughput measurements for the underlying network.

To provide a simple example for this paper, we defined a test device in the testbed having a network interface with an uplink and downlink bandwidth of 5 Mbit/s and we deployed this device on a PlanetLab host. *Iperf* was then installed on this host and the bandwidth was measured for the traffic flowing on the port on which the interface has been configured.

Figure 7 shows the bandwidth measurement for the incoming traffic and Figure 8 shows the measurement for outgoing traffic. As it can be seen from these figures, the bandwidth observed on the configured interface conforms to the bandwidth that was configured.

```
[dcetut_Multipath_P2P@planetlab2 ~]$ iperf -s -p 5345
------------------------------------------------------------
Server listening on TCP port 5345
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 193.166.167.5 port 5345 connected with 193.166.167.4 port 40390
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.6 sec  6.12 MBytes  4.83 Mbits/sec
```

Fig. 7. Incoming traffic bitrate verification

```
[dcetut_Multipath_P2P@planetlab2 ~]$ iperf -c planetlab1.rd.tut.fi -p 5345
------------------------------------------------------------
Client connecting to planetlab1.rd.tut.fi, TCP port 5345
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 193.166.167.5 port 53143 connected with 193.166.167.4 port 5345
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.3 sec  6.12 MBytes  5.00 Mbits/sec
```

Fig. 8. Outgoing traffic bitrate verification

In order to verify that latency and packet loss factors are also functional on the interface, it was modified first to have a delay of 50ms along with the bandwidth of 5Mbit/s. The results obtained are shown in Figure 9, clearly showing a drop in the bandwidth due to the time delay on the interface.

Secondly the same interface was configured now to have a packet loss ratio of 0.1 i.e. 10 % of the packets on the interface would be dropped randomly. The results from this configuration are presented in Figure 10, also showing a drop in the observed bandwidth.

```
[dcetut_Multipath_P2P@planetlab2 ~]$ iperf -s -p 5345
------------------------------------------------------------
Server listening on TCP port 5345
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 193.166.167.5 port 5345 connected with 193.166.167.4 port 43349
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.6 sec  5.62 MBytes  4.45 Mbits/sec
```

Fig. 9. Time delay affecting bandwidth

```
[dcetut_Multipath_P2P@planetlab2 ~]$ iperf -s -p 5345
------------------------------------------------------------
Server listening on TCP port 5345
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 193.166.167.5 port 5345 connected with 193.166.167.4 port 58947
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.9 sec  1.62 MBytes  1.25 Mbits/sec
```

Fig. 10. Packet loss ratio

6 Conclusions and Future Work

The main outcome of this work has been the development of a distributed platform that allows us to emulate multiple devices on PlanetLab nodes that possess multiple links of varying characteristics to simulate fixed, wireless and virtual interfaces found in mobile devices and resource challenged nodes. While we achieved the target set out in our prototype testbed, we expect greater usefulness can be achieved by modeling other characteristics. These include various processor architectures, execution speeds as well as storage requirements. This remains a challenge as physical PlanetLab machines are highly homogeneous in terms of hardware as well as operating systems, typically running on x86-based workstations. However we remain optimistic that in future, research projects would arise to take on such a challenge for emulating the hardware characteristics of devices. This would undoubtedly affect startup and configuration times as well, as a PlanetLab node would have no idea of the type of device it is supposed to instantiate until the command is issued by the management server. In such a scenario, it can be envisioned that an additional component would be necessary in order to transfer binary device images to end-hosts for successful emulation.

The current architecture is aimed towards providing a single realm of control, i.e. the management of the emulated devices is centralized towards a single server while information about running nodes and their interfaces is stored in a single database. This is highly suitable for scenarios whereby management is controlled by a single organization. Such scenarios include a smart grid operator, nation-wide traffic management systems as well as smart city based management solutions. Commands for emulation are issued over the SSH protocol as blocking operations. As future work, we intend to investigate protocol driven approaches towards the instantiation and subsequent management of the emulated nodes. This would imply adding a management interface to each instantiated node over which well defined request and response messages would be sent to set or retrieve various types of information regarding the emulated nodes. Access control as well as transport layer security need to be well considered using this strategy.

While bandwidth and link characteristics were successfully controlled over TCP-based connection-oriented interactions, an operating system software bug on physical PlanetLab machines unfortunately prevented us from achieving similar results with UDP-based datagrams. At the time of writing, we are still in the process of troubleshooting the issue together with the PlanetLab administration. However, our results suggest that the approach undertaken is highly feasible to model both device heterogeneity ranging from simple sensors to more powerful devices, as well as wireless network characteristics to customize link reliability, channel throughput as well as bandwidth availability.

References

1. Ericsson, More than 50 Billion Connected Devices, White Paper (2011). http://www.ericsson.com/res/docs/whitepapers/wp50billions.pdf
2. PlanetLab International Testbed. http://www.planet-lab.org
3. Karrer, R.P., et al.: Magnets-experiences from deploying a joint research-operational next-generation wireless access network testbed. In: Testbeds and Research Infrastructure for the Development of Networks and Communities, TridentCom 2007, IEEE (2007)
4. Sebastian, W., et al.: Pan-European testbed and experimental facility federation-architecture refinement and implementation. International Journal of Communication Networks and Distributed Systems **5**(1/2), 67–87 (2010)
5. Luis, S., et al.: SmartSantander: The meeting point between Future Internet research and experimentation and the smart cities. In: Future Network and Mobile Summit (FutureNetw), IEEE (2011)
6. Chertov, R., Kim, J., Chen, J.: LTE Emulation over Wired Ethernet. In: Korakis, T., Zink, M., Ott, M. (eds.) TridentCom 2012. LNICST, vol. 44, pp. 18–32. Springer, Heidelberg (2012)
7. Tazaki, H., Asaeda, H.: DNEmu: Design and Implementation of Distributed Network Emulation for Smooth Experimentation Control. In: Korakis, T., Zink, M., Ott, M. (eds.) TridentCom 2012. LNICST, vol. 44, pp. 162–177. Springer, Heidelberg (2012)
8. PlanetLab, PlanetLab Central API Documentation. https://www.planet-lab.eu/db/doc/PLCAPI.php
9. Carbone, M., Rizzo, L.: Dummynet Revisited. ACM SIGCOMM Computer Communication Review **40**(2), 12–20 (2010)
10. Iperf project. http://iperf.sourceforge.net/