Testbed for Measuring Protocols and Real-Time Applications

Jose Saldana, José Ruiz-Mas, Eduardo Viruete, Jenifer Murillo, Julián Fernández-Navajas and José I. Aznar
Communication Technologies Group and the Aragon Institute of Engineering Research
Dpt. IEC Ada Byron Building, CPS Univ. Zaragoza
50018 Zaragoza
{jsaldana, jruiz, eviruete, jenifer.murillo, navajas, jiaznar}@unizar.es

Abstract

This paper presents a Xen virtualization-based testbed for wired and wireless networks. Although big test infrastructures have been developed by research groups around the world, smaller testbeds can also be interesting, especially for the first stages of deployment of new systems and protocols. It is a hybrid testbed, as it includes simulation and emulation tools. But there exists a previous offline simulation stage, in order to avoid computer load, which could limit the maximum number of hosts to be included into the scenario. Two uses of the testbed are presented, one for wireless networks which deploys some handover measures with MIPv4, and the first tests of a distributed Call Admission Control (CAC) for an IP Telephony system. The uses show that measurements obtained with our testbed are similar to the ones published by other groups using real machines.

1. Introduction

Although in its beginning the Internet was a research infrastructure, nowadays it has become an operational environment, and it is not mainly used as an experimental platform. Many universities and R&D departments that work on protocols and distributed applications need to verify their behavior in realistic conditions. Sometimes, tests are very difficult to carry out because of the number of machines that are required.

There are various possibilities to solve this problem. One of them is the use of simulation tools (e.g. Opnet, ns-2), which allow the making of controlled and repeatable measurements with low costs. They have two drawbacks: the first one is computational load, especially when the network scenario consists of a large number of machines. The second one is their lack of accuracy, because they have specific implementations of protocols, not allowing the use of whatever applications and Operating Systems (OS).

Another option is the use of real hardware, including the protocol stack of the OS. This can be a good solution, but also a large number of testbeds and emulators [1] have been deployed. Some of them are hybrid, combining the advantages of simulation, emulation and real equipment tests.

In the present work we explain the deployment of a testbed that emulates a network by means of virtualization, allowing us to implement a set of virtual nodes in one physical machine. Nodes participating in the communication will be virtual machines adequately connected.

This paper also presents two different uses of our testbed for both wired and wireless networks. They will be useful to validate the testbed, comparing our results with the ones obtained by other groups using real hardware. These comparatives will show the utility of the testbed, especially in the first steps of the development of new systems and distributed applications. Our tool includes a first offline simulation and scenario generation stage, in order to avoid scalability problems caused by the need of real-time simulations.

In the next section we will talk about related work. In section 3 the architecture and components of the testbed are presented. The next section explains two uses of the testbed. This paper ends with the conclusions section.

2. Related Work

In recent years, many large scale network experiment facilities have been started around the globe [2], [3]. These environments are usually shared by research groups in different countries. Some of them integrate emulators to imitate the behavior of system elements, e.g. the movements

of the machines, or the quick changes of the radio channel. One of these big platforms is PlanetLab [4], which allows deploying and evaluating new protocols and services. Each node runs a virtual machine monitor that isolates services and applications from one another.

In an emulation-based experiment, the system we want to measure is represented by some surrogate systems, and by other systems used as real [1]. Some parts of the test have a bigger level of abstraction than others. Some of them are simulated and some of them are real. As network emulation combines real elements with abstracted ones, it must run in real time.

We have notice of many hybrid test platforms. They simulate a part of the system and emulate other parts. EMWin [5] has an emulated MAC layer, imitating the radio link in a wired testbed. NCTU [6] is able to simulate protocols of wired and wireless IP Networks. The traffic generation nodes are emulated. The traffic is captured by the simulator, and it calculates the influence of the traffic on the wireless part.

Some emulators like vBET [7] have used User Mode Linux (UML), which is a virtualization solution. The virtual nodes are connected via a simulated network, which is below the driver of the network card.

The main advantage of virtualization is that virtual machines use actual implementations of protocols and OS. Thus, the used traffic is exactly the same as in real applications. And it is very easy to establish an isolated environment, so the same traffic patterns can be repeated a lot of times, in the same or different network conditions, providing repeatability. Finally, synchronization can be easily obtained, as all the virtual machines are in the same physical one.

The main disadvantages are that network behavior has to be emulated, because virtual bridges have no bandwidth limits, and the need of monitoring the processor load of the main machine. If the processor load is very high, delays can be produced by that cause, and not by the network.

Xen is a paravirtualization solution, which requires the guest OS to be specifically compiled to run in the virtual machine. Some works have made a comparative between virtualization platforms [8], and they have showed the good behavior of Xen in terms of overhead, linearity and isolation between virtual machines.

3. System Architecture

The testbed uses different kinds of emulation in each level of the protocol stack. First, each node is emulated via a virtual machine. One of our aims is to provide a network scenario that can be run in a commodity PC or in a small LAN.

Two different tools are used to emulate link level: the Linux tool Traffic Control (*tc*) and *Mackill*. *tc* is used as a way to emulate the different bandwidths of network links. It has some parameters as *limit/latency*, *rate* and *burst*. Fig. 1 includes a scheme of its behavior.

Mackill is used to emulate visibility of the nodes. It is a part of APE (Ad hoc Protocol Evaluation) testbed [9]. It consists of a kernel module that adds a MAC filter to the protocol stack, and drops packets coming from certain addresses (Fig. 1). During test time, packets proceeding from non-visible nodes are discarded. *Mackill* reads the information of node's visibility from a previously generated file in order to emulate mobility. As it is not a part of the standard Linux, *Mackill* requires a kernel recompilation. Traffic shaping is applied when packets are sent, while traffic dropping is applied to packets once they have arrived to the virtual network card.

To emulate the IP level behavior, we have used the *NetEm* tool [10]. Internally, it is implemented with two nested packet queues. When packets are en-queued, they are time stamped with a send time, and put into the holding queue. Concurrently, a timer moves packets from the first queue to the nested one. This allows to introduce controlled delays, packet losses, duplication and reordering with different distributions and statistics.

We have created two networks: one for control and another for tests. Control network allows the access to nodes, in order to obtain the desired traffic measures, avoiding the interference with test network.

The system is used in three stages (Fig. 2). First of all, an offline scenario generation phase has to be deployed. A simulation tool has to be used in order to obtain the time moments when some events will occur. The final goal of this phase is a set of files which include the parameters that will be used during the emulations, e.g. the positions of each node in discrete time moments, the moments of call arrival, etc

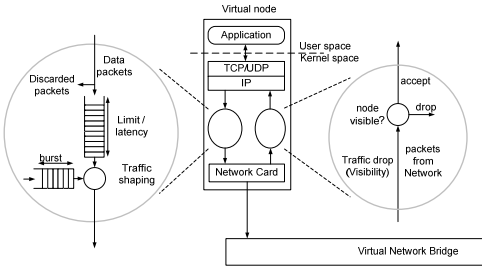

Figure 1. Network model

Figure 2. Testbed usage stages scheme

The testbed should be able to emulate a series of parameters of real networks. The scenario configuration sub-stage is in charge of setting the correct parameters of each node and network element: kind of networks, number of nodes, bandwidth, packet size, and protocols.

In the second stage, the nodes and the different levels are emulated. The scenario is built and traffic is generated, using different statistical distributions of inter-departure time and packet size. We have obtained log files with the sending and receiving times of each packet. These files will be used in the next stage.

As we are trying to avoid calculations during the development of the test, departure and arrival time of each packet have to be analyzed after the end of the test. By this way we will be able to measure objective Quality of Service (QoS) parameters as One-way delay (OWD), Round-trip time (RTT), jitter, packet loss, etc., that we can use to calculate MOS (Mean Opinion Score) or other subjective parameters related to QoE.

We have used a machine with the Operating System CentOS 5. The version of the Linux core is 2.6.18-8.1.15. It has a Core 2 Duo at 2.40 GHz processor, 2MB of Cache level 2, and 4GB RAM. Virtual machines also run CentOS 5. The version of Xen is 3.03-25.0.4.

It is necessary to monitor the machine's processor during the emulations to assure that it has not a heavy processing charge. In that case, delay times could be caused by processor delays instead of network delays. We have used *top* and *mpstat*, which is a part of the *sysstat* package.

4. Testbed usage examples

4.1. Wireless mobility. Handover latencies

In the first usage example we use a network simulator to generate the random movements of the hosts by a scenario, and traffic is sent between virtual machines while movements are emulated. MIPv4 mobility protocol has been used, and some measurements of handover delays for multimedia flows have been obtained.

We have modified *AnSim* [11] for the implementation of the simulation sub-stage. It is an application that allows the user to generate the random movements of the nodes. The user can tune some parameters: attenuation level, mobility model, scenario file, number of nodes, etc. It is also possible to select some hotspots of the Random Waypoint Model (RWM).

We have added to *AnSim* new functionalities, such as fixed nodes, and the possibility to specify which node will act as a router for the others. We can define different kinds of links to connect nodes, each one with its speed and peculiar parameters, thus emulating mobility in a scenario with a network infrastructure.

With the obtained off-line simulation of the nodes' movements, *Mackill* filters at MAC level the packets that come from a node which has no visibility with the node that is executing. In our testbed, the input of the application is the movement file generated by the simulator of nodes' movements. Recently we have improved the loss model, in order to include not only free space path loss but ground reflected wave as well.

We have built a scenario (Fig. 3) able to emulate the pass of a node from a network to another while a multimedia session is on. This scenario should allow a wide set of measurements, so we will be able to compare different situations.

MIPv4 introduces mobility agents to provide service to nodes that are not in its original network. There is a *Home Agent* (HA) that pays attention to the traffic of the home nodes that are out of the network. The *Foreign Agent* (FA) handles the traffic of nodes from other networks

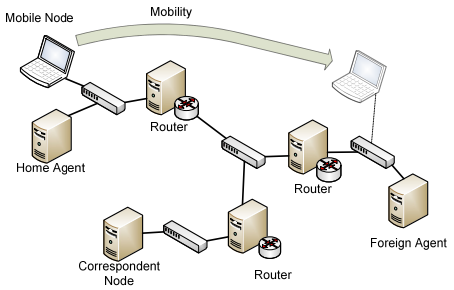

Figure 3. Mobility scenario

that are currently visiting its network. The *Mobile Node* (MN) is outside its original network; and finally, the *Correspondent Node* (CN) is communicating with the MN.

In this stage, each simulated node has been translated to a virtual machine and the calculated movements of the nodes are emulated with the correspondent tool for dropping packets that go between two nodes which have no visibility.

We have used Dynamics-Hut implementation of MIPv4. It is able to work in *MN* and *FA decapsulation* modes, and has the possibility of reverse or triangle tunneling.

Let us see a graph obtained during a handover with a packet cadence of 12 ms. The information bandwidth is 28 kbps. We have selected this packet cadence because we need accuracy to measure times between 80 and 100 ms. In Fig. 4 we can observe the OWD during a handover. In this case, 7 packets have been lost. Thus, we can estimate handover time in 84 ms. OWD increases after the handover due to tunneling.

We have also done some handover time measurements, based on packet loss, to compare *FA* and *MN decapsulation*. Sending UDP packets of 42 bytes every 10 ms, to emulate a VoIP session, we have obtained in the first case 9.2 lost packets average, and 9.4 packets with *MN decapsulation*. We have measured 10 handovers for each case. We conclude that there are not great differences between the two operation modes. These times fit with [12] in a similar case. In that study the measured handover time of MIPv4 was 104.5 ms.

An interesting comparison is also the difference between handovers that occur when the MN goes from its Home Network to the Foreign Network (direct handover), and the inverse ones. Logically, we can expect handovers to be faster in the second case, as less signalling is required when the MN returns to its home network. We have sent the same traffic that in the previous case. In the inverse case we have lost 1.9 packets. As can be seen, inverse handovers are faster.

4.2. Distributed IP Telephony system

IP Telephony services represent an interesting solution for enterprises since they not only imply savings, but also availability and security features. VoIP is a real-time service in which delay and packet loss directly impair calls' quality, and users demand a QoS similar to the one guaranteed for PSTN. Currently we are working on a Call Admission Control (CAC) for a SIP-based IP Telephony system, which implementation and first tests are being developed in the virtualization testbed. In this case, the testbed will not emulate wireless networks, so tools for simulating mobility and visibility are not necessary.

The IP Telephony system corresponds to an enterprise with several central offices placed in different countries. We assume that it does not have control over the Internet, nor over network parameters and the infrastructure, so an end-to-end approach has to be used. New incoming call acceptance paradigm consists of, while accepting the call, the remaining ongoing calls are not affected in terms of QoS. The PBX is configured within the centralized data centre. Furthermore, Internet is used for telephone traffic delivery among offices, instead of dedicated lines. Fig. 5 shows the different elements in the scenario.

There is a local agent in each office, which is in charge of the CAC. When it receives an INVITE from the PBX, it accepts or rejects it, depending on the number of established calls in that moment. There is a maximum number of

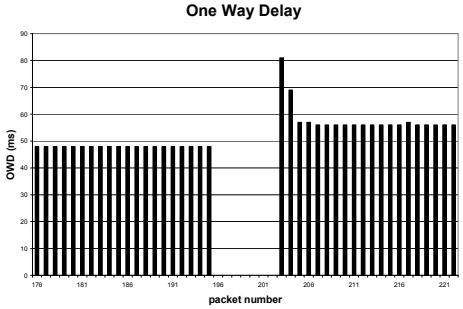

Figure 4. One Way Delay during a handover

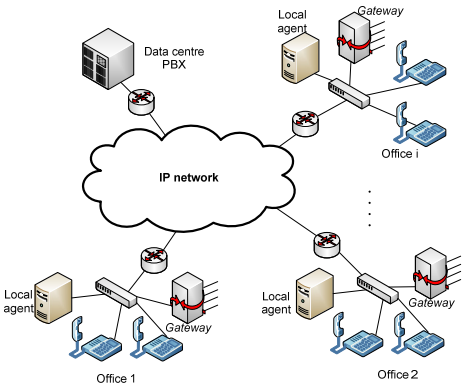

Figure 5. IP Telephony system

simultaneous calls, which is the main parameter of the CAC. For calls with destination in PSTN, a redirection can be used if the local gateway is busy. Although SIP messages go to the PBX, RTP traffic is directly sent from one phone to another.

Matlab has been used to obtain the moments of phone calls and their durations during a busy hour. Different traffic distributions are being tested in order to emulate the telephone traffic between different branch offices of an enterprise.

The software tools chosen for the developed scenario should be free, and require low computational load, due to the fact that they run within a virtualized environment. For the system's PBX we have selected *Asterisk* 1.6. OpenSIPS 1.4. has also been used, as it is a SIP proxy which includes *redirect* option and can use external databases in order to implement CAC decisions. Finally, PJSUA 1.0. has been used for the soft phones and gateways. *Expect* Linux tool is used in order to manage it, establishing and hanging the calls in the adequate moment, depending on the times generated with *Matlab*. They can be executed by the testbed with different network behaviors, e.g. with and without local agent, different traffic distributions, different number of users, etc.

Office routers have a connection of 1 Mbps. The *tc* buffer discards packets that spend more than 110 ms in it. *NetEm* adds 40 ms in central offices and 20 ms in data centre. Each RTP call has a bandwidth of 24 kbps at IP level.

Background traffic has the next size distribution [13]: 50% of packets are of 40 bytes, 10% of 576 bytes and 40% of 1500 bytes, all of them at IP level. This traffic has been used in order to saturate the access link of each central office. We have used UDP instead of TCP, in order to avoid the effect of TCP flow control. Thus, the background traffic is always the same, making the system work in the worst case.

Log files are analyzed after the execution in order to obtain graphs of QoS parameters. Fig. 6 (a) and 6 (b) show the behavior of the system in terms of OWD and packet loss. These curves are the upper bound for the values in case the maximum number of calls of the CAC is set to each value. They are represented as a function of background traffic, so each curve has a different moment in which total traffic gets above the bandwidth limit. The existence of a queue that discards packets spending more than 110 ms implies an upper bound for OWD.

Fig. 6 (c) shows the MOS [14]. It can be seen that in the case of 20 calls, the system only achieves a value of 3.5 if background traffic is smaller than 700 kbps. With 15 calls this limit is 900 kbps, and in other cases this happens when the offered traffic is above the bandwidth limit.

5. Conclusions

This paper describes a testbed that uses Xen in order to include a set of virtual machines in a physical one. It is used in three stages: it begins with the scenario generation. After that, simulated nodes are translated to virtual ones, and traffic is sent and captured in the emulation phase. Thus, it is a hybrid testbed as it uses simulation and emulation. Finally, captures are analyzed offline in order to obtain measures of QoS parameters.

Two uses of the testbed have been presented: the first one simulates a mobility scenario, and obtains handover latencies with MIPv4. The second use shows how the testbed can be useful for the deployment of a distributed system, such as a CAC for IP Telephony. It has also been shown that the testbed has flexibility to deploy different tests, with both wired and wireless networks.

Acknowledgments

This work has been partially financed by Cheque Tecnológico 2009/2010 Project, of Aragon I+D Agency, of the Government of Aragon, and Cátedra Telefónica Project, of University of Zaragoza.

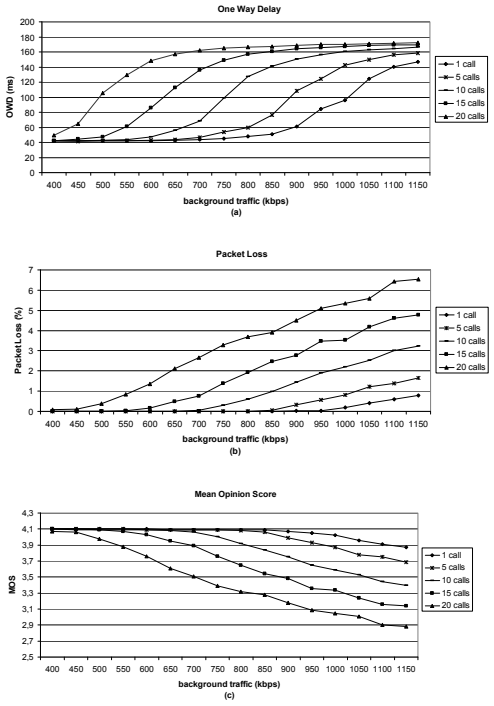

Figure 6. One Way Delay, Packet Loss and MOS of RTP traffic

References

[1] E. Göktürk, "A stance on emulation and testbeds", in Proc. 21st European Conference on Modelling and Simulation ECMS 2007.

[2] J. S. Turner, "A proposed architecture for the GENI backbone platform", In Proc. Architecture for Networking and Communications Systems, 2006.

[3] A. Gavras, A. Karila, S. Fdida, M. May, M. Potts, "Future internet research and experimentation: the FIRE initiative", ACM SIGCOMM Computer Communication Review, v.37 n.3, 2007.

[4] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, M. Wawrzoniak, "Operating System Support for Planetary-Scale Network Services", in Proc. of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04), San Francisco, CA, 2004.

[5] P. Zheng, L. M. Ni, "EMWin: emulating a mobile wireless network using a wired network", In Proceedings of the 5th ACM international Workshop on Wireless Mobile Multimedia, Atlanta, 2002.

[6] S. Y. Wang, "Using the innovative NCTUns 3.0 network simulator and emulator to facilitate network researches", Testbeds and Research Infrastructures for the Development of Networks and Communities, TRIDENTCOM 2006. 2nd International Conference on, pp.4-184, Barcelona, 2006.

[7] X. Jiang, D. Xu, "vbet: a vm-based emulation testbed", in Proc. of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research MoMeTools'03, ACM Press, pp. 95–104, New York, 2003.

[8] B. Quetier, V. Neri, F. Cappello, "Selecting A Virtualization System For Grid/P2P Large Scale Emulation", in Proc. of the Workshop on Experimental Grid testbeds for the assessment of large-scale applications and tools EXPGRID'06, Paris, 2006.

[9] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordströ, C. Tschudin, "A Large-scale Testbed for Reproducible Ad hoc Protocol Evaluations", in Proc. IEEE Wireless Communications and Networking Conference WCNC'02, 2002.

[10] S. Hemminger, "Network Emulation with NetEm". In Proceedings of Linux Conference AU, Canberra, 2005.

[11] H. Hellbrück, S. Fischer, "Towards analysis and simulation of ad-hoc networks", in ICWN02: Proceedings of the International Conference on Wireless Networks, pages 69–75, Las Vegas, 2002.

[12] A. Cabellos-Aparicio, H. Julian-Bertomeu, J. Núñez-Martínez, L. Jakab, R. Serral-Gracià, J. Domingo-Pascual, "Measurement-Based Comparison of IPv4/IPv6 Mobility Protocols on a WLAN Scenario", in Proceedings of Networks UK HET-NET Ilkley, UK, 2005.

[13] Cooperative Association for Internet Data Analysis "NASA Ames Internet Exchange Packet Length Distributions".

[14] "The E-Model", http://www.itu.int/ITU-T/studygroups/com12/emodelv1/calcul.php