Virtualization Technologies for DTN Testbeds

Carlo Caini[1], Rosario Firrincieli[1], Daniele Lacamera[2], and Marco Livini[1]

[1] DEIS/ARCES, University of Bologna, Bologna, Italy
[2] Sadel S.p.A., Bologna, Italy
{carlo.caini,rosario.firrincieli}@unibo.it, root@danielinux.net, marco.livini@studio.unibo.it

Abstract. At present, Internet is based on the availability of a continuous path from the source to the sink node and on limited delays. These assumptions do not hold in "challenged networks", which comprise a wide variety of different environments, from sensor networks to space communications (including satellite systems). These networks are the preferred target of Delay/Disruption Tolerant Networking (DTN), an innovative networking architecture able to cope with long delays, channel disruptions and limited or intermittent connectivity. Given the increasing interest in DTN, there is urgent need for suitable tools for DTN performance evaluation. In general, there are two approaches to performance evaluation in networking: simulation and real testbeds. In this paper, after an in-depth discussion of advantages and disadvantages of both, a third way based on a virtualization is proposed and tested for DTN environments, for which it seems particularly suitable. To validate this assumption, a virtual counterpart of a real testbed is set-up using Virtual Testbed Toolkit (VTT) components. A series of tests is then performed by considering DTN transmission on a heterogeneous network including a GEO satellite link. The close match between real and virtual testbed results confirms the validity of the virtual approach for accurate performance evaluations in DTN environments.

1 Introduction

A key assumption of present Internet is the availability of a continuous path from the source to the sink node. This, however, is not always true in the "challenged networks", which comprise a wide variety of different environments, like sensor networks and space communications. These networks are the preferred target of Delay/Disruption Tolerant Networking (DTN) [1], [2], which provides more robust network architecture in cases of long delays, channel disruptions and limited or intermittent connectivity.

Among the many possible challenged environments, heterogeneous networks that include satellite links should be also considered. This is because of the long propagation delay (Round Trip Time – RTT – is about 600 ms in a GEO system), the possible presence of relatively high PER (Packet Error Rate) values of due to adverse propagation conditions on the satellite link and, when considering mobile satellite terminals, the presence of disruptions caused by tunnels and buildings.

In dealing with performance evaluation of new protocols or architectures, two approaches are possible. The first is based on simulation, the second on the use of real testbeds. Each has advantages and disadvantages, as it will be shown below. In brief, network simulators, such as ns-2 [3] and many others, by relying on a single workstation, are easy to set up and maintain, but depend on the accuracy of protocol implementations and/or modeling. Moreover, they generally require code duplication, as specific version of new protocols must be developed and maintained for the simulator environment. This can lead to measurement inconsistency, caused by bugs or other code discrepancies. Testbeds avoid this problem by making use of real protocol and architecture implementations, thus providing a test environment much closer to reality. On the negative side, testbeds imply much greater costs and higher maintenance. For PC-based testbeds, costs increase roughly linearly with the number of nodes, while in simulation the cost is independent of network layout complexity.

In this paper a third way, based on virtualization, is proposed and tested for DTN environments, to which it seems particularly suited. Virtualization expands the concept of emulation, so an entire testbed can be emulated with even one off-the-shelf fast PC. The main goal is to combine the accuracy and code commonality of real testbeds with the benefits of network simulators: lower costs, higher flexibility and reduced maintenance. With DTN architectures, an additional advantage is that the most important DTN bundle layer implementations (DTN2 and ION) are developed for Linux and there is still little support of DTN in network simulators. The paper will show that virtualization technologies are a good match for DTN environments.

2 Simulation vs. Real Testbeds

2.1 Simulation

Network simulators provide opportunities for fast and efficient experimentation. Two main categories exist: simulators that target to a very specific protocol or network type, and simulators that address a wide range of protocols. Several simulation packages, like Cnet [4], Opnet [5], Qualnet [6] and Insane [7], come into the latter category. However, at present the most commonly adopted wide scope tool is ns-2, a discrete event simulator developed and maintained for networking research by LBNL, University of California, Berkeley campus [3]. Thanks to its wide diffusion and public domain design, the network research community can easily share results and additional modules.

2.2 Linux Testbeds

GNU/Linux is the preferred operating system when setting up a testbed for the experimental evaluation of new protocols and architectures, for a number of reasons. Linux is free, so it can be installed with no expense other than the hardware. It is fully customizable in all its components; thanks to its GNU General Public License [9] the user is allowed to freely access and modify the source code of the Linux kernel and of all system tools. Moreover, there are a number of additional networking applications freely available on the Internet.

An essential feature of Linux is that the standard code implementing network functions is research oriented, besides also being perfectly suitable for the ordinary user. For example, Linux already offers several TCP variants in addition to the standard NewReno algorithm, such as TCP Hybla [10] without packet spacing [11] and Hoe's initial slow start threshold estimate [12]), HighSpeed TCP, TCP Westwood+, TCP CUBIC (the Linux default) and many others. Other important features, such as the SACK option and the Forward RTO (F-RTO) [13], are also available. All these TCP enhancements can be activated by means of their own individual control switches.

The GNU libraries offer a useful interface to read TCP variables during a connection through the *socket()* system call, which is the *TCP_INFO* socket option. If this option is set when creating a socket, the user application can access TCP-related values during a connection, by directly communicating with the Linux TCP layer. Another commonly used evaluation tool is the "libpcap" library and all the applications built on it, like tcpdump [14]. These latter tools make network interfaces work in "promiscuous" mode, capturing every single packet that passes through and collecting statistics on TCP flows. Despite their advantages, in particular the fact that these tools are already available, in our experience, more powerful instruments are needed for a in-depth analysis of TCP.. To this end, we added some debugging routines to the standard kernel code, using the MultiTCP [15] patch.

2.3 Advantages and Disadvantages of the First Two Alternatives

In this section, we compare simulation and testbeds with regard to a number of typical experimentation issues.

Test management

There is no doubt that it is usually much easier and faster to perform complex test campaigns by simulation than a testbed, because the latter requires a much more complex hardware and software. However, we have shown that by adopting centralized control tools, as in TATPA (Testbed on Advanced Transport Protocols and Architecture) [16], this disadvantage can be removed. It should of course be stressed that realizing such a complex management interface is in itself quite a demanding and expensive task, while for simulation it comes for free.

Another management aspect to be considered is the possibility of granting concurrent access to resources. In simulation it is usually possible to carry out several runs in parallel, and a number of users can use the simulator at the same time, although, given its relatively limited cost, it is more likely that they use of different PCs. By contrast, in the case of testbeds, where only one run can be launched at a time, the use of hardware resources must be strictly regulated, to avoid any potential conflict. But even in this case, this problem can be resolved by an appropriate interface, to manage an efficient access scheduling.

Reproducibility

Reproducibility is one of the main principles of the scientific method. Experimental results produced by a group of researchers are reproducible if they can be confirmed by other independent researchers in the same operational conditions. Simulators generally guarantee reproducibility if run over the same architecture and all the random sources are correctly seeded. Regarding testbeds, only partial reproducibility can be achieved, because of the lack of synchronicity between testbed components, and unavoidable hardware and software discrepancies between different testbeds. However, the assessment of average performance parameters, such as goodput, is not impaired, provided that tests are carried out in "clean" environments, i.e. without external "interference", which is a prerequisite to assure adequate reproducibility of results. Analyses performed on "real Internet", so frequent in the literature, should eventually complement, and not substitute for, analyses in clean environments.

Hardware Requirements
Hardware (and related costs) is definitely a point in favor of simulation. While a simulation requires a single workstation, an emulation testbed generally requires several workstations and/or additional dedicated hardware like hubs, switches and routers. The TATPA testbed, for example, consists of up to ten PCs with one or more network cards (depending whether they act as end-points or routers) and connected by cross-cables and switches.

Maintenance
Maintenance includes fixing bugs, adding new features and updating software and hardware components. Ns-2 updates are frequent, and it is necessary to check how new versions impact on non standard modules, and to introduce when necessary the required modifications. Testbeds have the same problem whenever a new version of Linux kernel is released. Finally, as a testbed always consists of several units and maintenance increases with the number of components, although some savings of scale can be possible.

Extensibility and Inter-Working
Extensibility quantifies the ability to easily extend a system, with new features or other modifications; Inter-working is the capacity of different systems to work together, sharing information and data. Both arguments for the testbed approach, thanks to the limited number of standard network interface. As an example, to add new modules (even based on proprietary software or hardware platforms) in an Ethernet based testbed, it is enough that they adopt the same NIC (Network Interface Card) standard. With simulation, new modules can inter-work only if expressly developed for a given simulator software. This weak point of simulation can be mitigated by wide diffusion of a given the simulator (as for ns-2), which facilitates sharing of new modules.

Layer Implementation
Layer implementation indicates the capacity of representing the entire ISO/OSI stack as closely as possible. This important point mainly depends on the simulation and emulation tool adopted. In general there must be a trade-off between complexity and close representation. For example, a satellite connection can be simply modeled by setting a large delay and possibly a significant packet error rate (PER) [17]. More complex models can include a choice among different probability density functions for random losses (e.g. two state models for link availability), or the implementation of link functions (e.g. MAC procedures) and so on [18]. With complex models, the emulator has the additional constraint that all procedures must be performed in real time.

Accuracy and code commonality with existing systems
Last but not least, it should be noted that testbeds use the same protocol and architecture as real systems. There is therefore a higher level of accuracy than simulation can offer. Moreover, using the same code of existing systems speeds up the development of new versions and avoids parallel maintenance of two versions of the same code (one for simulation, one for real operating systems), which is both quite demanding and inevitably prone to error (additional bugs in code conversion). An interesting attempt to reduce this problem, limited however to TCP congestion control, is the "Linux TCP implementation for ns-2" developed by Caltech [19], which reuses Linux code for TCP congestion control algorithms on ns-2.

Finally, let us point out here that possible dual use (research and deployment) of code developed for real operating systems was certainly contemplated by the developers of the DTN2 bundle layer reference implementation for Linux. It is worth citing the DTN2 description given on the DTNRG website: "The goal of this implementation is to clearly embody the components of the DTN architecture, while also providing a robust and flexible software framework for experimentation, extension, and real-world deployment".

3 The Virtual Approach

Emulation is usually employed in testbeds to reproduce radio channel characteristics in real time (e.g. satellite, WLAN, cellular networks). Virtualization expands this concept to all the testbed components, so an entire testbed can be emulated with one modern PC. The main goal is to combine the advantages of real testbeds, such as extensibility, inter-working, accuracy and protocol commonality with real systems, with the benefits of network simulators, such as lower costs, higher flexibility, reduced maintenance and result reproducibility. The virtual testbed toolkit presented in this paper, based on free software, goes further and tries to preserve the real-time constraint, an essential requirement for performance evaluation of time-sensitive protocol on wireless networks.

Although the advantages of the virtualization approach are many and significant, there are also limits to its applicability. They are mainly related to the complexity of the testbed layout (number of virtual machines necessary to implement nodes) and the transmission rates supported (the higher, the more computationally demanding). These limits are closely related to the computational resources of the host machine and the efficiency of the virtualization hardware and software supports. Incidentally, to cope with demanding environments the use of multiple host machines can be considered as well, thus leading to the concept of a *distributed* virtual testbed, by contrast to an *integrated* virtual testbed, where there is just one host machine.

As regards the DTN application field, it is worth noting that in DTN environments the bandwidths are usually limited (e.g. no high-speed networks are considered), which makes it feasible to implement even relatively complex layouts, by preserving the real-time constraint. Moreover, in some cases, like security or routing experiments, real-time constraint is not essential, thus further expanding the number of nodes supported. It seems therefore that DTN environments and virtualization technologies are a perfect match. Experimental results presented below, obtained using integrated virtual testbed, confirm the validity of this assumption.

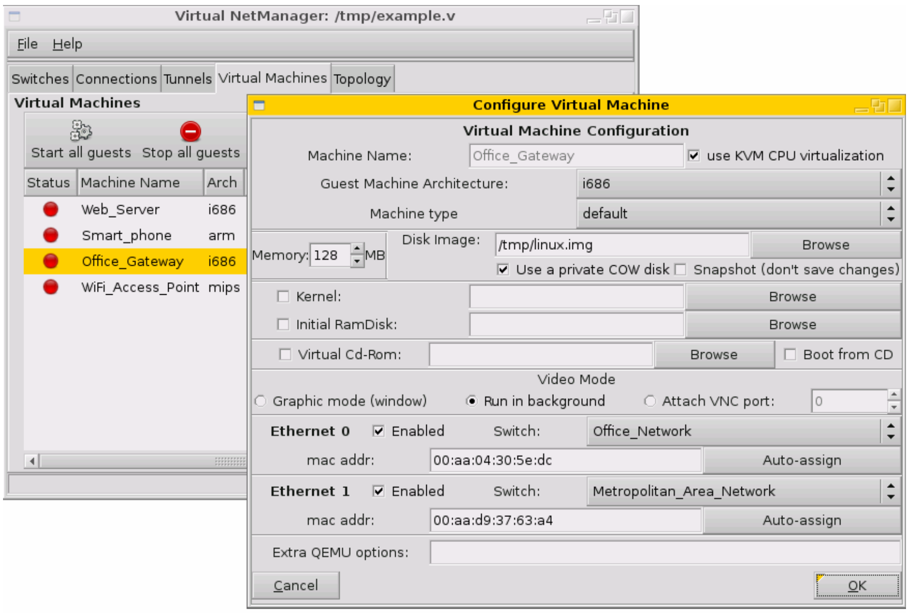
3.1 The Virtual Testbed Toolkit (VTT)

The idea of exploiting network virtualization to concentrate many testbed components on one PC has already appeared in the literature. To provide some examples, we cite [20], where the implementation of a virtual router based on Xen technology is presented, and [21], which describes a hybrid testbed, composed of real mesh nodes and a virtualized environment. Virtualization is further exploited in [22] and [23], where VMs are used to build testbeds dedicated to the evaluation of security issues. In [24] the aim is to create an "inbox" protocol development environment.

However, the Virtual Testbed Toolkit (VTT) used in this paper is innovative with respect to other proposals, in that it aims to preserve time accuracy, in order to offer reliable results even when dealing with time-sensitive performance tests (e.g. goodput evaluations). This is a challenge due to the limited processing resources of the real host machine. For this reason, the virtualization approach, when applied to a DTN environment, is validated by comparing virtual testbed results with those achieved by the real TATPA testbed, based on a cluster of Linux PCs.

This validation is performed by considering time-sensitive DTN performance (e.g. goodput) on a heterogeneous network scenario including satellite links. Other tests, with non time-sensitive performance, could also be carried out, but would be much less demanding.

VTT implements network nodes as independent Kernel-based Virtual Machines (KVMs), while other network devices and connections are emulated through the Virtual Distributed Ethernet (VDE) project tools [25]. A powerful graphic interface, the "Virtual NetManager", allows the user to design the desired network layout and to configure and control the network components, as shown in Fig. 1. The toolkit is



Status	Machine Name	Arch
●	Web_Server	i686
●	Smart_phone	arm
●	Office_Gateway	i686
●	WiFi_Access_Point	mips

Machine Name: Office_Gateway ✓ use KVM CPU virtualization
Guest Machine Architecture: i686
Machine type: default
Memory: 128 MB
Disk Image: /tmp/linux.img
✓ Use a private COW disk ☐ Snapshot (don't save changes)
☐ Kernel:
☐ Initial RamDisk:
☐ Virtual Cd-Rom: ☐ Boot from CD
Graphic mode (window) [✓ Run in background] Attach VNC port: 0
Ethernet 0 ✓ Enabled Switch: Office_Network
mac addr: 00:aa:04:30:5e:dc
Ethernet 1 ✓ Enabled Switch: Metropolitan_Area_Network
mac addr: 00:aa:d9:37:63:a4
Extra QEMU options:

Fig. 1. Virtual NetManager graphical interface; virtual machine configuration

distributed under the GPL2 free license. All information about downloading and installation can be found in [26]. It should be pointed out, however, that the software is under development and some work is still necessary to improve the reliability of the most recent components.

The authors have already dealt with related issues in [27], where a virtual counterpart of the TATPA testbed, namely VITT (Virtual Integrated TCP Testbed) was proposed. However, the present work differs in many respects: first, the focus here is on DTN and not on TCP alone; second, here there is a toolkit that allows great layout configurability, and not just a fixed layout testbed; third, the Virtual NetManager interface has been introduced; fourth, VTT benefits for all the VDE enhancements introduced in the meantime.

4 Numerical Results

DTN bundle layer performance evaluations obtained using the VTT tools are compared here with those obtained by the TATPA testbed. The aim is to validate the reliability of virtualization when applied to DTN environments, rather than to present new performance results. In other words, the novelty of the numerical results presented here lies *in the way* they have been obtained (i.e. through VTT tools), and not in their absolute values. In fact, the environment considered is actually the same as that in [28], where TATPA results on DTN were first presented. The hardware platform used for running VTT tools is an “Intel® Core™2 Quad Q6600” with 2.4 MHz clock, support of Intel® Virtual Technology and 4 GB of RAM.

Tests refer to a satellite environment characterized by the continuous availability of the end-to-end path. Time-sensitive performance is evaluated by means of the DTNperf_2 tool [29]. The DTNperf_2 “transmission window” W is set to one, which means that bundles are sent one by one, (a new bundle is sent only at reception of previous bundle “delivered” status report). A bundle size of 1 Mbyte is assumed, to limit the bundle layer overhead; as we are aiming at reliable transmissions, the bundle protocol “custody transfer” option is enabled and TCP is adopted as convergence layer.

Tests are carried out using the layout represented in Fig. 2, both for both the virtual and TATPA testbeds. DTN features are enabled in three nodes: the satellite sender and receiver, and the intermediate node R2 (satellite gateway). Note that, by contrast, wired sender and receiver are not DTN nodes, as they are used here just to generate background TCP traffic when necessary. The satellite end-to-end DTN “connection” consists of two hops; the first from the satellite sender to R2, the second from R2 to the satellite receiver (see Fig. 2 and Fig. 3). In the former (RTT = 25 ms, PER = 0) NewReno is used, and in the latter (variable RTT, PER = 0 or 1 %) Hybla, a TCP variant optimized for satellite links [10]. NewReno is also used in background wired TCP connections (RTT = 25 ms, PER = 0). Tests last 180 s and are repeated three times in order to average DTN goodput and calculate confidence intervals. Three evaluation scenarios are considered below: congestion only, link losses only and congestion and link losses.

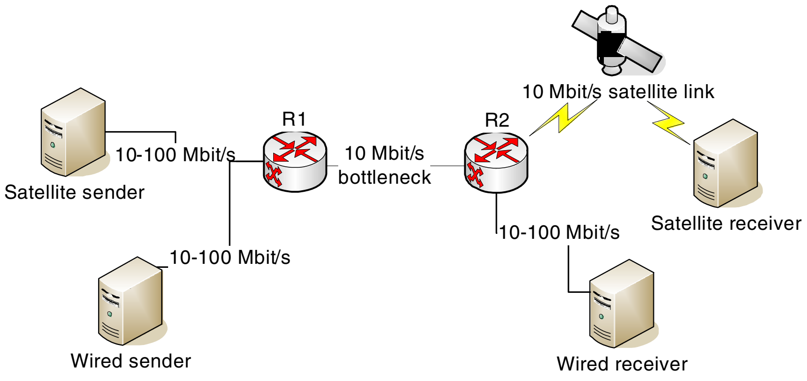


Fig. 2. Layout used in both virtual and TATPA testbeds

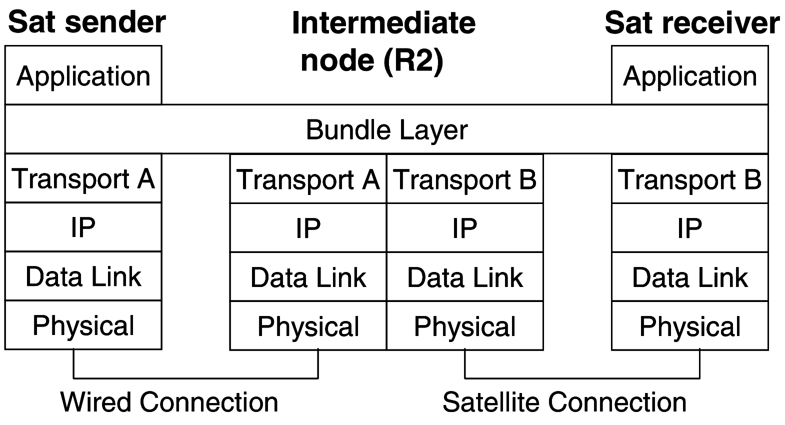


Fig. 3. DTN protocol stack used in tests

4.1 Performance Comparison in the Presence of Congestion

The first evaluation scenario is a DTN transfer between satellite end nodes (with variable RTT on the satellite hop) and 5 wired TCP connections in background.

In Fig. 4, TATPA and VTT results are compared, referring to the goodput achieved by the satellite connection for three RTTs. It shows how VTT and TATPA results are very similar, for all RTTs. This confirms the accuracy of the virtual approach, with all the aforementioned advantages deriving from the use of a single PC.

As regards goodput, this is almost independent of RTT. This is due to the DTN architecture which, by splitting the satellite connection into two parts, allows a fair subdivision of the bottleneck, because the 5 TCP background connections and the satellite sender-R2 connection have the same RTT. On the satellite link, the improvements due to Hybla are limited, because there are neither congestion events nor link losses.

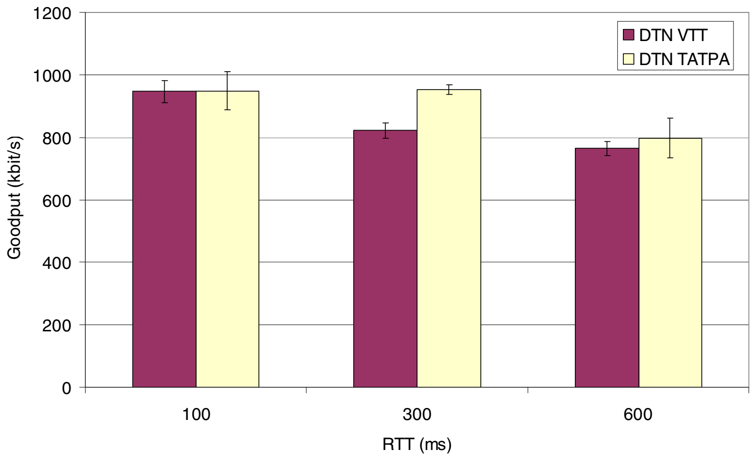


Fig. 4. Performance comparison between VTT and TATPA. Presence of congestion on the R1-R2 bottleneck: goodput of a satellite connection vs. its RTT; 5 background wired connection (RTT=25 ms) active.

4.2 Performance Comparison in the Presence of Link Losses

Unlike in the previous case, here there are no competing background connections on the first (wired) DTN hop. However, end-to-end performance is impaired by a high PER on the second DTN hop (the satellite one), which becomes the bottleneck here. Results, given in Fig. 5, show a good match between TATPA and VTT outcomes, in line with the previous case.

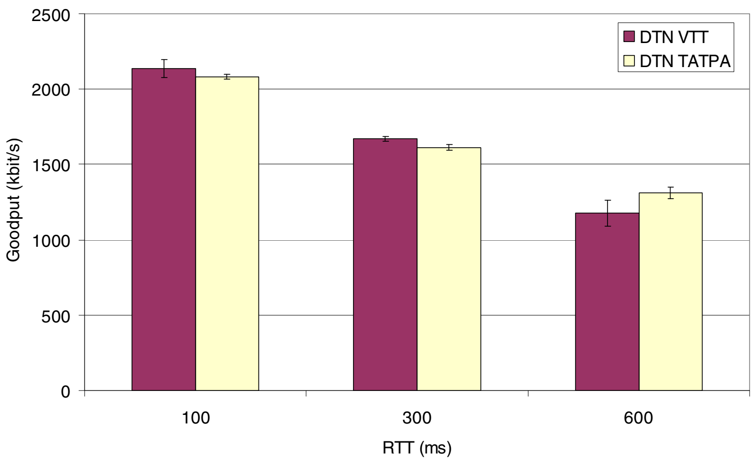


Fig. 5. Performance comparison between VTT and TATPA. Presence of losses (PER = 1%) on the satellite link: goodput of a satellite connection vs. its RTT; no background traffic.

Here performance very much depends on the ability of the TCP protocol used on the satellite hop to cope with random losses (i.e. losses not due to congestion). The protocol used here is Hybla, which actually provides much better resilience against random losses than other TCP variants, in particular for long RTTs. Interested readers are referred to [30] for an exhaustive comparison between Hybla and other TCP variants.

4.3 Performance Comparison in the Presence of Congestion and Link Losses

In the third scenario we have the simultaneous presence of both congestion on the first DTN hop and a high PER on the satellite one. Results, given in Fig. 6, refer as above to the end-to-end goodput of the DTN transfer between the two "satellite" end nodes. As in the previous scenarios, VTT and TATPA results closely match, demonstrating how, even in this complex scenario, the virtual approach can provide the same level as accuracy of a real testbed.

As regards performance, the limited impairment with RTT can be ascribed to both DTN architecture and Hybla. On the first DTN hop, where background traffic is present, the "TCP splitting" performed by the DTN architecture allows a fair sharing of bandwidth resources among all the connections on the R1-R2 bottleneck. On the second DTN hop, where a high PER affects the satellite link, Hybla resilience to random losses limits their negative impact on performance. Note that, as a result, the additional performance degradation due to PER on the satellite hop, with respect to the congestion only case (Fig. 4), is limited. Better performance could have been obtained by making use of a larger DTNperf_2 window, but this would have been outside the scope of the present tests.

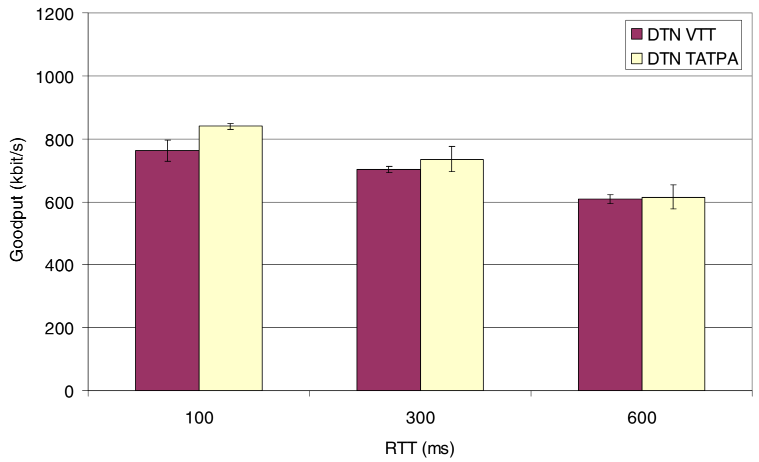


Fig. 6. Performance comparison between VTT and TATPA. Presence of both congestion on the R1-R2 bottleneck and losses (PER = 1%) on the satellite link: goodput of a satellite connection vs. its RTT; 5 background wired connection (RTT=25 ms) active.

5 Conclusions

In this paper, the use of virtualization technologies for DTN bundle protocol architecture is proposed and discussed. The aim is to set-up an entire DTN testbed in just one PC, thus combining the accuracy of real testbeds with the benefits of network simulators in terms of cost, test reproducibility, and maintenance. The results obtained by means of the Virtual Testbed Toolkit used to set up an integrated virtual counterpart of the real TATPA testbed show a good match in all benchmark tests considered. This confirms the validity of the virtualization technologies for performance evaluations in DTN environments.

References

1. Cerf, V., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., Weiss, H.: Delay-Tolerant Networking Architecture. IETF RFC 4838 (April 2007)
2. Fall, K., Farrell, S.: DTN: an architectural retrospective. IEEE Journal on Selected Areas in Commun 26(5), 828–836 (2008)
3. Network Simulator ns-2 University of California, Berkeley, `http://www.isi.edu/nsnam/ns/`
4. Cnet network simulator, `http://www.csse.uwa.edu.au/cnet/`
5. Opnet network simulator, `http://www.opnet.com/`
6. Qualnet network simulator, `http://www.scalable-networks.com`
7. Mah, B.A.: Insane Users Manual, The Tenet Group Computer Science Division. Univ. California, Berkeley (1996)
8. NCTUns, `http://nsl10.csie.nctu.edu.tw/`
9. GNU General Public License, `http://www.gnu.org/gpl`
10. Caini, C., Firrincieli, R.: TCP Hybla: a TCP Enhancement for Heterogeneous Networks. Wiley Int. J. Satellite Commun. Netw. 22, 547–566 (2004)
11. Caini, C., Firrincieli, R.: Packet spreading techniques to avoid bursty traffic in long RTT TCP connections. In: Proc. IEEE Vehicular Technology Conference VTC 2004-Spring, Milano (IT), May 2004, vol. 5, pp. 2906–2910 (2004)
12. Hoe, J.C.: Improving the Start-up Behavior of a Congestion Control Scheme for TCP. In: Proc. ACM SIGCOMM 1996, pp. 270–280 (1996)
13. Sarolahti, P., Kojo, M., Raatikainen, K.: F-RTO: an enhanced recovery algorithm for TCP retransmission timeouts. ACM SIGCOMM Computer Communication Review 33(2), 51–63 (2003)
14. TCP Dump, `http://www.tcpdump.org`
15. Caini, C., Firrincieli, R., Lacamera, D.: A Linux Based Multi TCP Implementation for Experimental Evaluation of TCP Enhancements. In: Proc. SPECTS 2005, Philadelphia, July 2005, pp. 875–883 (2005)
16. Caini, C., Firrincieli, R., Lacamera, D., Tamagnini, S., Tiraferri, D.: The TATPA. testbed. In: Proc. of IEEE/Create-Net Tridentcom 2007, Orlando, USA (2007)
17. Parker, S., Schmechel, C.: Some Testing Tools for TCP Implementors. IETF RFC 2398 (August 1998)
18. Marchese, M.: TCP Modifications over Satellite Channels: Study and Performance Evaluation. International Journal of Satellite Communications, Special Issue on IP 19(1), 93–110 (2001)

19. Wei, D.X., Cao, P.: NS-2 TCP-Linux: an NS-2 TCP implementation with congestion control algorithms from Linux. In: Proc. of ValueTool 2006 – Workshop of NS-2, October 2006, pp. 1–9 (2006)
20. Egi, N., Greenhalgh, A., Handley, M., Hoerdt, M., Mathy, L., Schooley, T.: Evaluating Xen for Router Virtualization. In: Proc. of IEEE ICCCN 2007, Honolulu, Hawaii USA, August 2007, pp. 1256–1261 (2007)
21. Zimmermann, A., Gunes, M., Wenig, M., Meis, U., Ritzerfeld, J.: How to Study Wireless Mesh Networks: A hybrid Testbed Approach. In: Proc. of AINA 2007, Niagara Falls, Canada, May 2007, pp. 853–860 (2007)
22. Volynkin, A., Skormin, V.: Large-scale Reconfigurable Virtual Testbed for Information Security Experiments. In: Proc. of IEEE/Create-NetTridentcom 2007, Orlando, USA (2007)
23. Duchamp, D., DeAngelis, G.: A Hypervisor Based Security Testbed. In: Proc. of DETER 2007, Boston, USA (August 2007)
24. Huang, X.W., Sharma, R., Keshav, S.: The ENTRAPID Protocol Development Environment. In: Proc. of IEEE INFOCOMM 1999, pp. 1107–1115 (1999)
25. Davoli, R.: VDE: Virtual Distributed Ethernet. In: Proc. of IEEE/Create-Net Tridentcom 2005, Trento, Italy, May 2005, pp. 213–220 (2005)
26. wiki V2, http://wiki.virtualsquare.org/
27. Caini, C., Davoli, R., Firrincieli, R., Lacamera, D.: Virtual Integrated TCP Testbed (VITT). In: Proc. Create-Net Tridentcom, Innsbruck, Austria, March 2008, pp. 1–6 (2008)
28. Caini, C., Cornice, P., Firrincieli, R., Lacamera, D.: A DTN Approach to Satellite Communications. IEEE Journal on Selected Areas in Communications, special issue on Delay and Disruption Tolerant Wireless Communication 26(5), 820–827 (2008)
29. Caini, C., Cornice, P., Firrincieli, R., Livini, M.: DTNperf_2: a Performance Evaluation tool for Delay/Disruption Tolerant Networking. In: Proc. E-DTN, St. Petersburg, Russia, September 2009, pp. 1–6 (2009)
30. Caini, C., Firrincieli, R., Lacamera, D.: Comparative Performance Evaluation of TCP variants on Satellite Environments. In: Proc. IEEE ICC, Dresden, Germany, June 2009, pp. 1–5 (2009)