Developing, Deploying and Evaluating Protocols with ManetLab

François Vessaz[1], Benoît Garbinato[1], Arielle Moro[1], Adrian Holzer[2]

[1] Université de Lausanne, Lausanne, Switzerland
{francois.vessaz,benoit.garbinato,arielle.moro}@unil.ch
[2] École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
adrian.holzer@epfl.ch

Abstract. Evaluating the performance of MANET-specific communication protocols is essential to build robust mobile ad hoc applications. Unfortunately, most existing evaluation results are either based on simulations – which makes it difficult to draw conclusions beyond confined lab settings – or they are based on custom testbed results – which makes it difficult to reproduce them. In order to overcome this challenge, we introduce ManetLab, a modular and configurable software framework for creating and running testbeds to evaluate MANET-specific protocols. With ManetLab, one can easily configure and automate reproducible protocol executions on standard computer hardware, and thus provides both the *accuracy* of testbed-based evaluations and the *reproducibility* of simulation-based evaluations. After presenting ManetLab's extensible architecture, based on the notion of modular protocol stack, we show how it helps evaluate the performance of different broadcast protocols in real MANETs and how its results compare with simulation-based results.

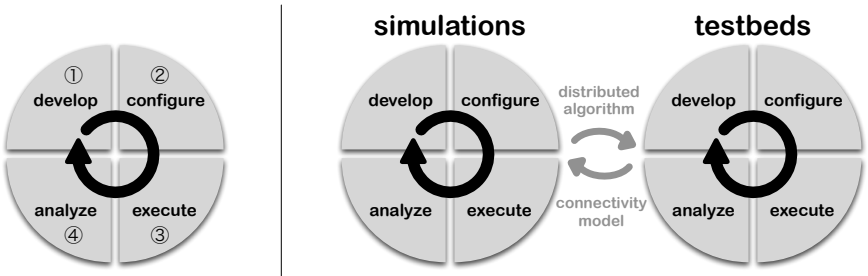
1 Introduction

With the tidal wave created by the arrival of smart devices and tablets, the prospects of seeing MANET-based apps pop up in the distributed systems landscape has become more promising than ever. To encourage the emergence of such apps, system developers must provide solid communication building blocks for application developers, such as multi-hop broadcast, multicast, unicast, and other dissemination and routing protocols. Along that line, a large amount of research effort have been spent investigating mobile ad hoc routing protocols over the past decade. Central to this effort are the specialized tools that allow researcher to *develop* and *evaluate* their protocols.

1.1 Protocol development and evaluation

The development of an effective and efficient protocol, be it wired, wireless infrastructure-based or ad hoc, is an iterative process consisting of four steps, as illustrated in Figure 1a. For a start, one has to devise the protocol in the form of a distributed algorithm, ideally proving it formally and ultimately implementing

it in some programming language (Step 1). Then, one has to configure some test environment in which the protocol will be executed (Step 2) and run the actual tests (Step 3). Finally, one has to analyze the collected data (Step 4), which might then lead to fine-tune the protocol and trigger a new iteration.



(a) general iterative process

(b) combining simulations and testbeds

Fig. 1: Development of communication protocols

Existing tools for evaluating protocol performance can be categorised as either *simulators* or *testeds*. When using testbeds, the iterative process sketched in Figure 1a can be very time consuming, especially if one wants to evaluate performances in various distributed environments, which is clearly a must. This is particularly true for Steps 2 and 3, since they imply the deployment of the protocol code and of the test configuration to various distributed nodes, the launching of the testbed execution and the gathering of the results obtained at each individual node. For this reason, dedicated tools aimed at facilitating the creation and execution of testbeds have been proposed by the research community for specific families of distributed environments. This is for example the case of PlanetLab[3] for large-scale distributed systems [10].

Performance evaluation tools for MANETs. When it comes to evaluate the performance of MANET-specific protocols however, until now researchers have had essentially the choice between the reproducibility offered by simulators and the accuracy offered by testbeds. On the positive side, simulators are widely used by the research community and their source code is generally accessible online, which makes simulation-based evaluations fairly reproducible. Unfortunately, as they rely on the modeling of complex physical and logical parameters, it is difficult to draw general conclusions about the behavior of such protocols in real settings [27]. Section 5 further discusses evaluation tools for MANETs.

Testbeds on the contrary rely on real mobile ad hoc networking and therefore tend to offer a high-level of accuracy. For this very reason however, they also tend to impose a high development and deployment barrier [1]. In addition, most testbeds are not directly available to other researchers and often require

[3] `http://www.planet-lab.org`

specialized or incompletely specified hardware. For these reasons, the level of reproducibility of the resulting performance evaluation is generally quite low.

1.2 Contribution and roadmap

The ManetLab framework precisely aims at filling this gap, by supporting both accurate and reproducible performance evaluations of MANET-specific protocols, in a similar way PlanetLab does it for large-scale distributed systems.

In Section 2, we discuss the need for a tool such as ManetLab and its key requirements to achieve high accuracy and reproducibility. In Section 3, we present ManetLab in details, by showing how it helps develop protocol layers and assemble them into a full protocol stack, which can then be deployed on remote nodes. We also introduce ManetLab's graphical tool, which helps configure performance evaluations, launch them and gather the corresponding results. In Section 4, we then compare the results of various performance evaluations obtained using ManetLab with those obtained with two simulations tools. While simulation-based performance evaluations are fairly accurate for simple MANET environments, they diverge significantly from the results obtained in reality for more complex environments, typically involving physical obstacles between nodes. Interestingly, when injecting the topological constraints observed with ManetLab back into the two considered simulators, simulation-based evaluation tend to augment their level of accuracy. Finally, we discuss existing testbed for MANETs in Section 5, and ongoing work on improving ManetLab in Section 6.

2 Achieving accuracy and reproducibility

To be useful, performance evaluations of MANET-specific protocols should obviously achieve a high degree of both accuracy and reproducibility. As a consequence, tools supporting such evaluations should mimic real-life environments as accurately as possible and should allow researchers to easily reproduce experiments with the purpose of comparing evaluation results.

2.1 Combining simulations and testbeds

Although this paper focuses on the need for a robust testbed tool, we advocate the combination of simulations and testbeds, as illustrated in Figure 1b. While simulations can indeed be considered an acceptable first approximation, communication protocols tend to perform in more unpredictable way when actually deployed in a real MANET than in an infrastructure-based network, be it wired or wireless. For this reason, we believe that ultimately accuracy can only be achieved with testbed approaches, at least at that stage of the evaluation.

In the early phases of a protocol development, simulations can be very useful to validate the protocol in simple environment settings, e.g., in the absence of walls or obstacles, using a basic wireless propagation model. Once this first validation is done, the protocol should then be evaluated in a real mobile ad hoc

network, using testbeds. As shown in Figure 1b, the results of testbed-oriented evaluations can then be injected back into simulations, typically in the form of a more accurate model of the wireless connectivity among network nodes. This is precisely the approach we follow in Section 4.

2.2 Creating accurate and reproducible testbeds

While various simulation tools exists, some of which have become de facto standards, the situation is very different when it comes to testbeds for mobile ad hoc networks. Moreover, coming up with universal and rigid testbeds for MANETs might not even be a desirable goal, given the great variability of actual deployment settings. Rather, we believe that there exists a need for a framework that facilitates the development, deployment and evaluation of MANET-specific protocols, by making it easy to create accurate and reproducible testbeds. That is, *accuracy* and *reproducibility* should be the two key requirements for such a testbed framework.

Accuracy. Devising a tool that offers an accurate evaluation of the behavior of a protocol running in a MANET can be very challenging. The central issue stems from the fact that MANETs tend to exhibit very erratic behaviors in terms of connectivity and of reliability, depending on their physical environment and on how nodes are moving. In addition, the various layers that stack up, in particular TCP/IP, tend to distort the actual performance evaluation of communication protocols in MANETs. For this reason, an adequate testbed framework should offer flexibility in protocol layering, all the way down to the lowest-level layers, typically by making it easy to compose protocol stacks from elemental layers.

Reproducibility. To evaluate the accuracy of results provided by an evaluation tool, other researchers must be able to reproduce the testbeds described in the literature, and scrutinize the evaluation tool itself. Thus, it is important that any evaluation tool, in particular a testbed framework, be easily accessible for the research community. A testbed framework should in addition be configurable, in order to easily switch from one deployment setting to another, and it should offer support for automatically launching evaluations and gathering results.

3 Introducing ManetLab

ManetLab is a framework supporting the creation and execution of accurate and reproducible testbeds for MANETs-specific protocols, using mainstream hardware.[4] On each computer where ManetLab is installed, the wireless network interface is used to connect the MANET, while the wired network interface is used as control network to provide feedback about the protocol performance. More specifically, as illustrated in Figure 2, each computer running ManetLab is hosting an *agent* connected to the MANET. In addition, one of the computers

[4] ManetLab runs on Apple's computers with Mac OS X 10.7 or higher.

hosts the *controller*, which acts as a conductor orchestrating the protocol execution.[5] That is, the controller uploads the protocol stack to each agent, triggers the execution of the protocol and collects feedback from all agents about the protocol execution, via the wired network interface. In the following, we discuss how each step pictured in Figure 1 is performed with ManetLab.

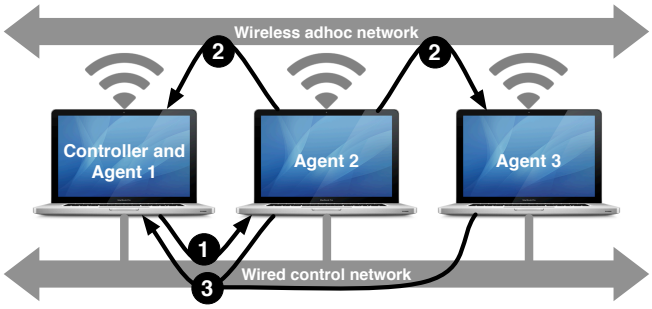


Fig. 2: ManetLab — Using an ad hoc network and a control network

3.1 Development

In order to test an ad hoc protocol, one has to first implement it. ManetLab proposes an API[6] to help MANET-specific protocol developers in this task. The protocol must be implemented in Objective-C and designed as layers, inheriting from the MLStackLayer class, in a stack (MLStack) provided by the API. The layer above the stack represents the application, whereas the layer below the stack is the antenna. At any given time, a ManetLab node is executing at most one protocol stack. Communication between nodes and between layers inside a stack is achieved via message passing (MLMessage). Figure 3 illustrates a stack containing two layers, i.e., a fragmentation layer and a gossip layer.

As pointed out in [27], researchers rarely make the effort to provide the source code of their MANET-specific protocols to the community, which makes it very difficult to seriously compare different protocols pursuing the same goal. Moreover, even when the source code is provided, the absence of a standardized tool to compose and deploy protocols leads to low reproducibility of most research results. For this reason, ManetLab proposes a plugin architecture that makes it easy to package all the layers (subclasses of MLStackLayer) and message types (subclasses of MLMessage) composing a MANET-specific protocol stack. As a result, stacks created using ManetLab can be shared and reused by other researchers. In addition, ManetLab is an open source project, which further promotes the scrutiny and reproducibility of testbeds relying on it.

[5] The example depicted in Figure 2 is further discussed in Section 3.3.

[6] The API is distributed with the ManetLab software and available at `http://doplab.unil.ch/manetlab`

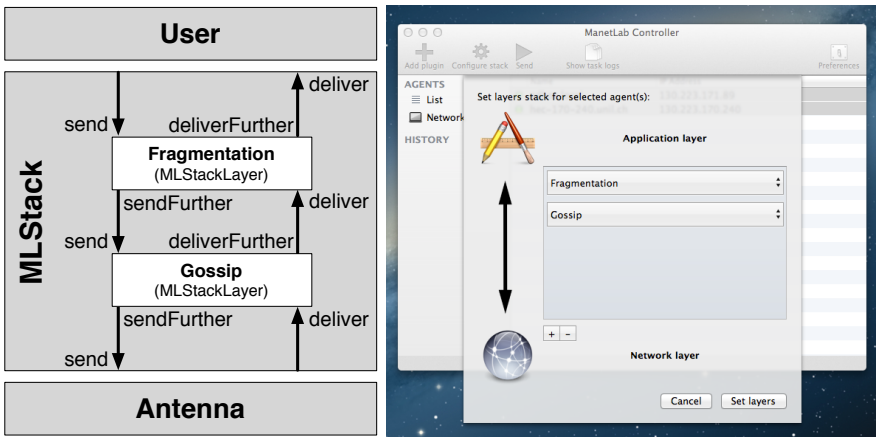


Fig. 3: A stack with two layers (left) and its corresponding GUI (right).

3.2 Configuration

The configuration step greatly differs in a simulator-based approach and in a testbed-based approach. However, in both cases, configuration entails *installation* of the tool (simulator or testbed) and its *parameterization*, and then *deployment* of the protocol code in the tool.

Installation. Installing a simulator is usually non-trivial because it implies to download the source code from the Internet and then to build the simulator from that code. As for testbeds, they are rarely made publicly available and when they are, they tend to be even more difficult to install and to use, since they often require specialized hardware. In contrast, ManetLab requires just a few clicks to be installed on a standard desktop or laptop computer.[7] Yet, if one wants to access its source code, it is also made available via GitHub.[8]

Parameterization. For both simulators and testbeds, parameterization implies to load the protocol stack into the tool, usually in some binary form. Apart from this obvious step, parameterization is where simulators and testbeds differ the most. As testbeds rely on real MANET implementations, one has only a few parameters to set, e.g., the wireless channel used to communicate; this is typically the case with ManetLab. In addition, as already suggested in Section 3.1, ManetLab makes it easy to dynamically load plugins containing protocol layers into the tool, to then graphically compose a protocol stack from these layers and deploy it of each node of the MANET (Figure 3). When it comes to simulators however, many more parameters have to be set, such as the mobility model, the connectivity model, the node distribution model, the interference model, etc. In terms of *accuracy*, this step is critical because unrealistic values may result in misleading or even erroneous performance evaluations.

[7] ManetLab executable is available from `http://doplab.unil.ch/manetlab`.
[8] ManetLab source code is available from `http://github.com/doplab/ManetLab`.

Deployment. When using testbeds, one of the major obstacles to reproducibility often lies in the need to deploy and maintain specialized hardware, typically in the form of prototype devices. In order to solve this problem, at least partially, ManetLab is implemented on the OS X platform, which is widespread in research institutions today. In addition, Apple's hardware is known to be very standardized and traceable, e.g., using a tool like Mactraker,[9] which is clearly an advantage in term of *reproducibility*. Moreover, building ManetLab on top of OS X, which shares the same code basis as iOS when it comes to low-level services, opens the opportunity to port ManetLab to iOS devices in the future.

3.3 Execution

With ManetLab, protocols communicate using the IEEE 802.11 wireless ad hoc network (IBSS mode), so the accuracy of performance evaluations is naturally ensured. On the controller, the graphical user interface shown in Figure 4 is used to prepare and launch the testbed execution.

Execution example. Arrows pictured in Figure 2 illustrate an execution with ManetLab, where the controller simply requests one agent to broadcast a message. First, the controller asks Agent 2 to broadcast a message (Arrow 1). As a result, Agent 2 does indeed broadcast a message on the wireless ad hoc network (Arrow 2). Finally, all agents have received the message and provide feedback to the controller, using the wired and reliable control network (Arrow 3).

Offline control mode. Since the controller communicates with agents via a wired control network, ManetLab does not allow to test protocol with mobility in its first version. To overcome this limitation, we are currently implementing an *offline control mode*, which uses the wireless network for both control messages and protocol messages. The idea is to have each agent log its evaluation results locally during the testbed execution, so that it can send them to the controller after the execution.

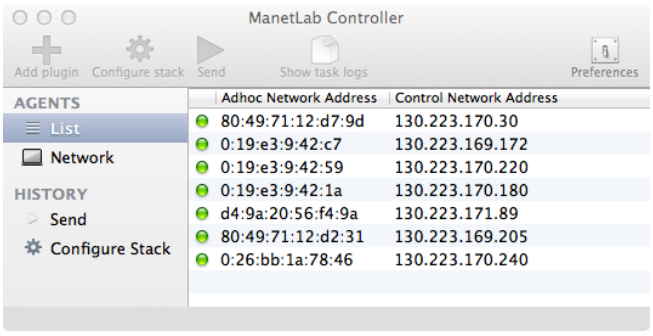


Fig. 4: ManetLab Controller — Graphical user interface

[9] `http://mactracker.ca`

3.4 Analysis

Most simulators and testbeds do not provide specific analysis tools. Rather, they allow protocol developers to produce log files or populate databases, which can then be fed into some analysis tool, such as a graphical network animator like NetAnim for example. This is also the case of the ManetLab testbed framework: its API offers a log methods that allows developers to produce whatever trace they need for their performance evaluation.

4 ManetLab testbeds *versus* simulations

To compare performance evaluations obtained from ManetLab with those obtained from simulations, we study the behavior of various broadcast protocols in ManetLab and in two simulators. These two simulators are ***NS-3*** [16], the latest simulator from the *NS* family, and ***Sinalgo***,[10] a simple Java-based simulator we used to evaluate several of our own MANET-specific protocols [12,13,14]. For our comparison, we rely on two simple measures: the ***delivery ratio***, defined as the *number of nodes who received a message over the total number of nodes*, and the ***forward ratio***, defined as the *number of nodes who send or retransmit the message over the total number of nodes*. Each measure is the average of 1'000 distinct executions.

4.1 Network settings

In order to avoid a potential distortion or overhead caused by TCP/IP (in particular its routing scheme), all protocols are directly using the MAC layer when it comes to broadcast a message in the MANET. Along that line, we parameterize the MAC layer of each tool in a similar way, as discussed hereafter.

ManetLab. Sincle ManetLab is a real MANET implementation, there are only a very few settings we can change. All other settings are constraints deriving from the operating system and the hardware on which ManetLab is running. Basically, ManetLab creates an IEEE 802.11a ad hoc network with a theoretical data rate of maximum 6 MBits/s for broadcast.

NS-3. We use the *WifiNetDevice* from NS-3 with the *YansWifiChannel* and the *YansWifiPhy* models. We set all the settings we can to similar values of what ManetLab uses on real computers. That is, we configure NS-3 to use an IEEE 802.11a physical layer model, with a data rate of maximum 5.5 Mbits/s and a MTU of 1'500 bytes.

Sinalgo. Being a higher-level simulator than NS-3, Sinalgo does not rely on an implementation of the IEEE 802.11 standard. So we configure Sinalgo to send messages smaller than 1'500 bytes, with a *Unit Disk Graph* connectivity model and a *Signal to Interference plus Noise Ratio* interference model.

[10] `http://www.disco.ethz.ch/projects/sinalgo/`

4.2 Protocols, environments and communication patterns

Because we aim at providing a solid first comparison, we consider a number of *broadcast protocols*, *physical environments* and *communication patterns*. They are presented in details hereafter.

Broadcast protocols. We consider three probabilistic broadcast protocols, namely ***Simple Flooding***, ***Gossip***, and ***Counter-Based Scheme*** (CBS), which are well-known to the research community. With Simple Flooding [15], each node systematically retransmits a message the first time it receives it, so the delivery ratio is always equal to the forward ratio. With Gossip [30], each node retransmits a message with a probability p the first time it receives it. For our comparisons, we set p to 0.7, 0.5, and 0.2. Finally, with CBS [30], a node waits for some random delay between 0 and w_{max} before retransmitting a message, only if it received it only once. That is, if a node receives a message more than once, it does not retransmit it. For CBS, we set w_{max} to 0.1 and 0.5 seconds.

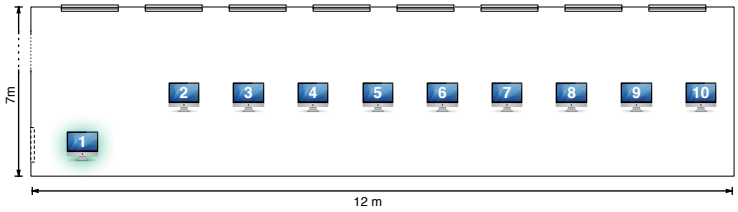


Fig. 5: Sketch of the open space environment.

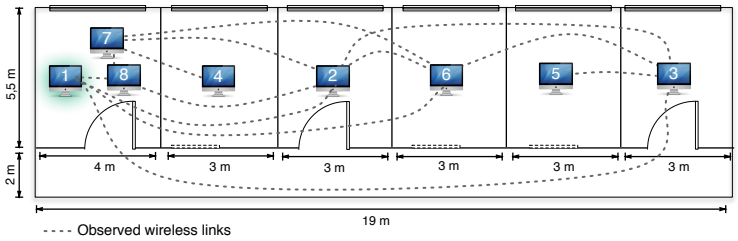


Fig. 6: Sketch of the private offices environment.

Physical environments. We consider two environments: an ***open space*** and ***private offices***. In the open space, ten computers are placed in one open space as illustrated in Figure 5; this is typically the case in a classroom. With private offices, eight computers are placed in adjacent offices as depicted in Figure 6. In each physical environment, Computer 1 acts as the initial broadcaster.

Communication patterns. We consider two communication patterns: a ***one-shot*** message, which corresponds to a low network load, and the ***streaming*** of 1'000 messages, which correspond to a high network load. With the *one-shot*

pattern, Computer 1 broadcasts a single 1'400-bytes message. Those 1'400 bytes are encapsulated in just one network frame consisting of 1'485 bytes, including headers. With the *streaming* pattern, Computer 1 sends 1.4 Mbytes, which are fragmented into 1'000 network frames of 1'485 bytes each.

4.3 Results in the open space environment

Figure 7 shows the delivery and forward ratios of the one-shot communication pattern. Since all nodes are connected (fully connected graph) and there are almost no interferences, the delivery ratios are strictly equal to 1.0 for all protocols in NS-3 and Sinalgo, and above 0.99 for ManetLab. For this reason, the forward ratios tend to converge towards their theoretical values, i.e., 1 for flooding, p for gossip and much smaller values for CBS. Overall, we can say that in this scenario (one-shot in an open space), simulations are quite accurate since they faithfully mimic the results obtained by ManetLab in a real MANET.

In the second scenario (streaming in an open space), interferences start to disturb the behavior of the protocols and affect both the delivery ratio and the forward ratio, as shown in Figure 8. The more messages are transmitted, e.g., for flooding or for gossip with $p = 0.7$, the more the delivery ratio decreases. Interestingly, the delivery ratios of ManetLab are over 0.7, whereas the delivery ratios of NS-3 and Sinalgo are under 0.6. That is, the interferences models used in the two simulators are discarding too many frames, which indicates that their accuracy is diminishing. As for the forward ratios, they tend to be only slightly lower with the simulators than with ManetLab.

4.4 Results in the private offices environment

In the private offices environment, the real MANET experienced by ManetLab is no longer a fully connected graph, due to various physical obstacles (mainly walls but also furniture, possibly people, etc.). It is thus not surprising that the delivery ratio of a one-shot communication pattern in ManetLab tends to drop compared to the open space, as shown in Figure 9. NS-3 and Sinalgo, on the contrary, continue to view the MANET as a fully connected graph, so their delivery ratios remain strictly equal to 1. This clearly indicates that their level of accuracy is dropping. As for the forward ratios, NS-3 and Sinalgo have similar results to those of ManetLab except for CBS.

Streaming in private offices is by far the worst scenario when it comes to the delivery ratio, as shown in Figure 10. Both physical obstacles and interferences are significantly decreasing the performances of all protocols. Again, the accuracy of NS-3 and Sinalgo is compromised, as they only roughly approximate the delivery ratio observed with ManetLab. Furthermore, since fewer nodes receive the messages being broadcast, the forward ratios with NS-3 and Sinalgo are also dropping and thus diverge from those observed with ManetLab.

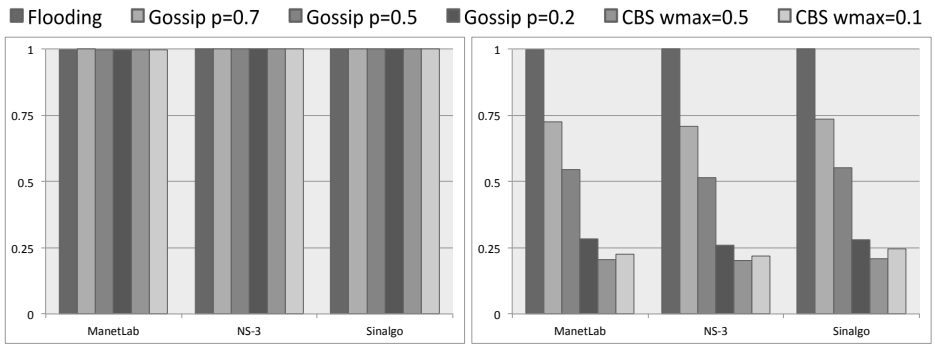


Fig. 7: One-shot in an open space – delivery ratio (left) and forward ratio (right).

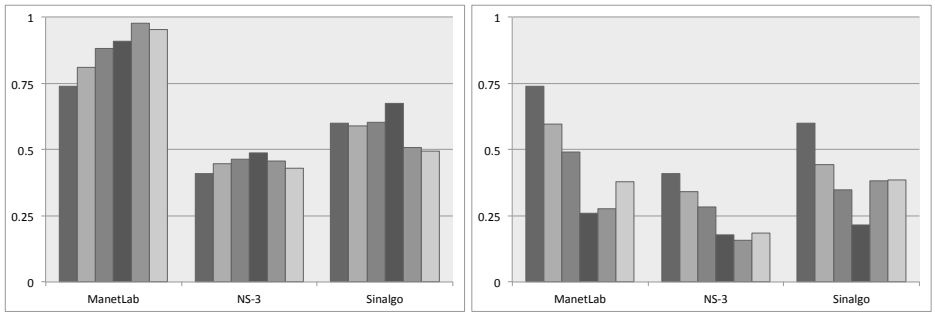


Fig. 8: Streaming in an open space – delivery ratio (left) and forward ratio (right).

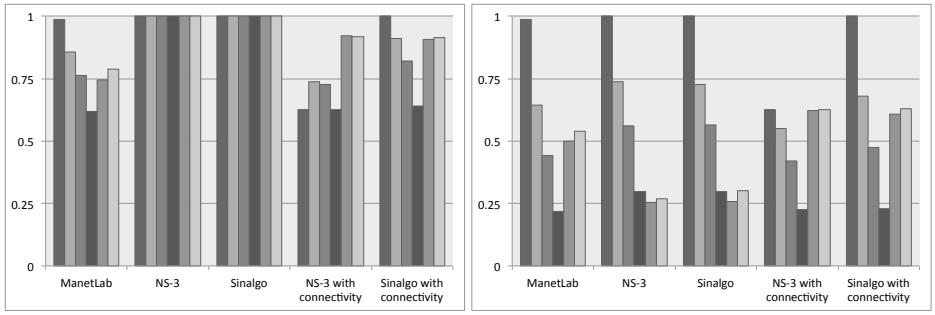


Fig. 9: One-shot in private offices – delivery ratio (left) and forward ratio (right).

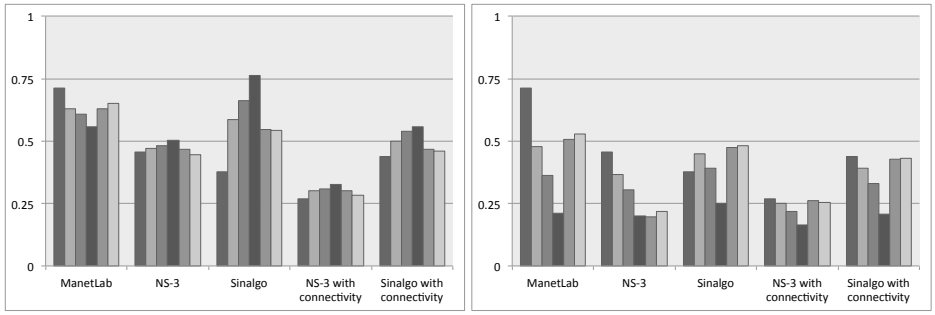


Fig. 10: Streaming in private offices – delivery ratio (left) and forward ratio(right).

4.5 Injecting the observed connectivity into simulations

It seems reasonable to assume that the drop in accuracy we observe for both NS-3 and Sinalgo, when considering private offices, is largely due to their erroneous modeling of the MANET connectivity. In order to confirm this assumption, we inject the connectivity graph experienced by ManetLab (see Figure 6) into NS-3 and Sinalgo, and we re-run our performance evaluations.

As shown in Figure 9, after the injection the accuracy of both NS-3 and Sinalgo is improved for the one-shot communication pattern. Interestingly, and somewhat surprisingly, the results for the simple flooding protocol are more accurate with Sinalgo than with NS-3. With the streaming communication patterns however, injecting the connectivity graph is not sufficient to improve the accuracy of NS-3 and Sinalgo, as shown in Figure 10. It seems that the effect of interferences, combined with a lower connectivity, leads both simulators to produce results that are significantly lower than what happens in reality.

5 Related work

To evaluate the behavior of their protocols, researchers should rely on simulators and testbeds that aim at providing *accurate* and *reproducible* performance evaluations. Hereafter, using these two dimensions, we review a wide range of evaluation tools for MANETs found in the literature [27,29,25,17,26] and we compare them with ManetLab. For accuracy, we focus on their communication support, as this is a critical element when it comes to evaluate performance in a MANET. For reproducibility, we assess the availability of the tools.

5.1 Communication support

Simulators on the one hand do not provide a real implementation of a wireless communication layer. For this reason, assessing the accuracy of their communication support, i.e., of their modelling of wireless communications, is very difficult and can only be achieved by comparing their results with those of a real MANET (as we did in Section 4). Such tools include NS-2 [9], NS-3 [16], GloMoSim [34] and its commercial version Qual Net[11], OPNET [8], OMNet++ [33], and others such as Sinalgo or JIST / SWANS [3,4].

Testbeds on the other hand tend to be more accurate because they rely on real wireless communication links. Such tools include Castadiva [18], MASSIVE [28], MobiEmu [35], mLab [22], Carnegie Mellon University Wireless Emulator [23], ORBIT [31], Seawind [24] and WHYNET [36] or JEmu [19], PoEM [20], and of course ManetLab. Some testbeds however tend to oversimplify topological constraints, e.g., by simply piling up a stack of wireless devices. In addition, while one-hop communication is provided by all testbeds, multi-hop communication is only found in tools such as WHYNET, RoofNet [7], ManetLab and Airplug-emu [6]. Other testbeds also provide multi-hop communication, but they shorten

[11] `http://www.scalable-networks.com/content/products/qualnet`

their wifi range. Such tools include ORBIT, TrueMobile [21] and MiNT [11]. Other tools provide a logical multi hop communication, such as mLab, MobiEmu and Castadiva.

5.2 Tool availability

To assess the availability of each tool, we evaluate if it is available *online*, if its *source code* is disclosed and available for download and installation, if detailed *documentation* is provided, and if *specialized hardware* is required in order to run testbeds relying on that tool.

Online availability. While many of the surveyed tools are available online for download, just as ManetLab, some other tools are only described in scientific papers, with no further details provided online. This is for instance the case of TrueMobile, PoEM, MASSIVE, JEmu, and the tool described by Barolli et al. [2]. This makes it very hard for other researchers to get a hold of these tools and reproduce experiments.

Source code availability and documentation. In order to evaluate the accuracy of a performance evaluation tool, providing the disclosed source code is another important aspect. Most reviewed simulators, except GloMoSim and OPNet, provide a downloadable version of their code. Among testbeds however, source code becomes much more scarce: only Castadiva, MobiEmu, mLab, Airplug-emu and MIT Roofnet provide access to their code, some of them without much documentation. ManetLab on the other hand provides both its source code and an easy-to-install binary file, with documentation and examples online.

Specialized hardware. While most simulators can be easily deployed on almost any computer, many testbeds require specialized hardware. This requirement makes it harder for other researchers to install the testbed and execute existing protocols. Moreover, some testbeds are deployed in specific lab settings and allow remote users to connect, such as CMUTrueMobile (based on the Emulab testbed [32]), which offers access to its testbed built on custom robots, or ORBIT which offers a testbed of 400 fixed WiFi devices placed in a grid formation on the ceiling of a single room. Other tools are devised to use special hardware, such as MiNT-m that uses Roomba vacuum cleaner robots as underlying hardware in order to support mobility and custom hardware on which to run protocols. Airplug-emu is another such example and is designed to emulate vehicular network and runs on laptops connected to specific GPS and radio receivers. Remote solutions have the advantage of side-stepping the tool deployment stage, and often allowing node mobility, but they impose restrictions on the execution scenarios. Other tools along with ManetLab can be deployed on standard equipment, which makes it easier for others to deploy and evaluate them. These tools include: Castadiva, MobileEmu, mLab and Airplug-emu.

6 Conclusion

Even though performance evaluation is central when it comes to designing robust MANET-specific communication protocols, we believe this problem has not been addressed in a satisfactory manner so far. Either protocols were evaluated through simulations and the results might not be valid in a real MANET environment, or they were evaluated in a customized testbed, which makes it hard to reproduce experiments. In this paper we presented ManetLab as a solution to this conundrum: ManetLab aims at offering the best of both worlds, i.e., accurate and reproducible results. In future work, we plan to extend ManetLab to iOS devices and to add an offline control mode.

Acknowledgement. This research is partially funded by the Swiss National Science Foundation under project numbers 138092 and 140762.

References

1. M. Al-Bado, C. Sengul, and R. Merz. What details are needed for wireless simulations? - a study of a site-specific indoor wireless model. In *INFOCOM'12*, 289–297.
2. L. Barolli, M. Ikeda, F. Xhafa, and A. Durresi. A testbed for manets: Implementation, experiences and learned lessons. In *IEEE Sys. Journ.*, 4(2):243–252, 2010.
3. R. Barr, Z. Haas, and R. van Renesse. Jist: An efficient approach to simulation using virtual machines. In *Software Practice & Experience* 35(6):539–576, 2005.
4. R. Barr, Z. Haas, and R. van Renesse. Scalable wireless ad hoc network simulation. In *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad hoc Wireless, and Peer-to-Peer Networks, ch. 19*, 2005.
5. Y. Benchaïb and C. Chaudet. Virmanel: a mobile multihop network virtualization tool. In *WiNTECH'12*, 67–74.
6. A. Buisset, B. Ducourthial, F. El Ali and S. Khalfallah. Vehicular networks emulation. In *ICCCN'10*, 1–7.
7. B. A. Chambers. The grid roofnet: A rooftop adhoc wireless network. In *M.S. Thesis, MIT, Cambridge, Massachusetts, June 2002.*
8. X. Chang. Network simulations with opnet. In *Wintersim'99*, 307–314.
9. Q. Chen, F. Schmidt-Eisenlohr, D. Jiang, M. Torrent-Moreno, L. Delgrossi, and H. Hartenstein. Overhaul of ieee 802.11 modeling and simulation in ns-2. In *MSWiM'07*, 159–168.
10. B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. In *ACM CCR*, 33(3), 2003.
11. P. De, A. Raniwala, S. Sharma, and T. Chiueh. Mint: A miniaturized network testbed for mobile wireless research. In *INFOCOM'05*, 2731–2742.
12. B. Garbinato, A. Holzer and F. Vessaz. Six-Shot Broadcast: A Context-Aware Algorithm for Efficient Message Diffusion in MANETs. In *DOA'08*, 625–638.
13. B. Garbinato, A. Holzer and F. Vessaz. Context-aware broadcasting approaches in mobile ad hoc networks. In *Computer Networks*, 1210 – 1228, 2010.
14. B. Garbinato, A. Holzer and F. Vessaz. Six-Shot Multicast: A Location-Aware Strategy for Efficient Message Routing in MANETs. In *NCA'10*, 1–9.

15. W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MOBICOM'99*, 174–185.
16. T. R. Henderson, M. Lacage, and G. F. Riley. Network simulations with the ns-3 simulator. In *SIGCOMM'08*.
17. J. Hortelano, J.-C. Cano, C. T. Calafate, and P. Manzoni. Testing applications in manet environments through emulation. *EURASIP J. Wirel. Commun. Netw.*, 2009:47:1–47:9, 2009.
18. J. Hortelano, M.acher, J.-C. Cano, C. T. Calafate, and P. Manzoni. Castadiva: A Test-Bed Architecture for Mobile AD HOC Networks. In *PIMRC'07*, 1–5.
19. H. T. J. Flynn and D. O'Mahony. Jemu: A real time emulation system for mobile ad hoc networks. In *Symp. on Tel. Sys. Res.*, 2001.
20. W. Jiang and C. Zhang. A portable real-time emulator for testing multi-radio manets. In *IPDPS'06*, 169–169.
21. D. Johnson, T. Stack, R. Fish, D. Flickinger, R. Ricci, and J. Lepreau. Truemobile: A mobile robotic wireless and sensor network testbed?, flux technical note ftn-2005-02. In *INFOCOM'06*.
22. A. Karygiannis and E. Antonakakis. mlab: A mobile ad hoc network test bed. In *SecPerU'05*.
23. K. Borries, W. Xiaohui, G. Judd, P. Steenkiste and D. Stancil. Experience with a wireless network testbed based on signal propagation emulation. In *EW'10*.
24. M. Kojo, A. Gurtov, J. Manner, P. Sarolahti, T. Alanko, and K. Raatikainen. Seawind: a wireless network emulator. In *MMB'01*.
25. M. Kropff, T. Krop, M. Hollick, P. Mogre, and R. Steinmetz. A survey on real world and emulation testbeds for mobile ad hoc networks. In *RIDENTCOM'06*.
26. E. Kulla, M. Ikeda, L. Barolli, F. Xhafa, and J. Iwashige. A survey on manet testbeds and mobility models. In *CSA'12*, 651–657.
27. S. Kurkowski, T. Camp, and M. Colagrosso. MANET simulation studies: the incredibles. In *Mob. Comput. Commun. Rev.*, 9(4):50–61, 2005.
28. M. Matthes, H. Biehl, M. Lauer and O. Drobnik. Massive: An emulation environment for mobile ad-hoc networks. In *WONS'05*, 54–59.
29. M. M. Qabajeh, A. A. Hashim, O. O. Khalifa, L. K. Qabajeh and J. I. Daoud. Performance evaluation in manets environment. In *Australian J. of Basic and Appl. Sciences, 6(1):143-148*, 2012.
30. S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *MobiCom'99*, 151–162.
31. D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu and M. Singh. Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. In *WCNC'05*, 1664–1669.
32. T. Stack, R. Fish, D. M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau. Mobile emulab: A robotic wireless and sensor network testbed. In *INFOCOM'06*, 1–12.
33. A. Varga. The omnet++ discrete event simulation system. In *ESM'01*.
34. X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: A library for parallel simulation of large-scale wireless networks. In *PADS'98*, 154–161.
35. Y. Zhang and W. Li. An integrated environment for testing mobile ad-hoc networks. In *MobiHoc'02*.
36. J. Zhou, Z. Ji, M. Varshney, Z. Xu, Y. Yang, M. Marina, and R. Bagrodia. Whynet: a hybrid testbed for large-scale, heterogeneous and adaptive wireless networks. In *WiNTECH '06*, 111–112.