Report on the testbed results of the BattleMesh V10

Alessandro GGAGNI
Leonardo MACCARI
Gioacchino MAZZUCCO
Claudio PISA
Bastian ??
Johan ??
??

Abstract

The Wireless Battle of the Mesh is an yearly event that brings together people from across the world to test and compare the performance of different routing protocols for ad-hoc and mesh networks, like Babel, B.A.T.M.A.N., BMX6, OLSR, 802.11s. Every year the community gathers and set-up a testbed on which the protocols are run, developed, debugged and tested, and some performance measures are extracted. While the initial spirit of the event was to set-up a competition between the protocols (as the name suggests), with time it changed into a moment of exchange of experience, collective development of innovations in the field of mesh networks and wireless open source networking software. This document reports on the results of the tenth edition of the Battle of The Mesh event.

11 giugno 2017

1 BattleMesh v10

The Battle of the Mesh (WBM), as the official website says:

> It is a tournament with a social character. If you are a mesh networking enthusiast, community networking activist, or have an interest in mesh networks you might want to check this out!
>
> The goal of the WirelessBattleMesh events is to set-up hands-on testbed for each available mesh routing protocol with a standard test procedure for the different mesh networks. During the different WBM events, similar hardware and software configuration will be used based on the OpenWRT BoardSupportPackage and packages for each protocol implementation. The WBM events are also a great opportunity to develop testing tools for PHY/MAC radio layers (drivers, scripts and PHY analyzers).

WBM is now at its tenth edition, and is organized by a motivated and large group of people (approximately 80-100 participants in the whole week in the last editions).

2 The Testbed

This year, the testbed was realized with 17 TP-Link WDR4300 routers, equipped with two wireless interfaces (operating in the 2.4 and 5.0 GHz bands), and 5 Ubiquiti Unifi AC Pro. The latter were used to perform local testing and support to the tests, but did not participate to the routing, since they are equipped with a different chipset. The nodes were configured to set-up two independent networks, a “management” network, running on the 2.4GHz, and a “testing” network, running on the 5 GHz. The management network was configured with the IEEE 802.11s protocol, and was used only to access manage the nodes and perform tasks. The testing network was made with interfaces configured in ad-hoc mode and, for each test, was using a specific routing protocol.

fig. 1 reports the topology of the network as exported by one of the network node, when running the OLSRv1 protocol. This topology was used to understand and roughly guess the property of the network. The underlying image is the map of the XX museum that hosted the event .

put museum name here

Routers numbered from 1 to 17 are the TP-Link, while router 31 is an Ubiquiti router that is used in this case to extract the network topology. In the real tests this router does not participate to the network functioning. The transparency of the links represents the badness of the link quality as reported by the OLSRv1 protocol, using the ETX metric [**?**]. A solid link means ETX = 1 (good link), while a transparent link means ETX ¿ 1 (the highest, the worse).

Note that the topology in fig. 1 does not necessarily represent the topology used by other routing protocols, but gives an approximate idea of what links are directly connected. Based on this topology, exported in the netJSON format[1], we calculated the weighted shortest path between any couple of node, using the networkX python

[1] A generic format for exporting a network topology, see `http://netjson.org`

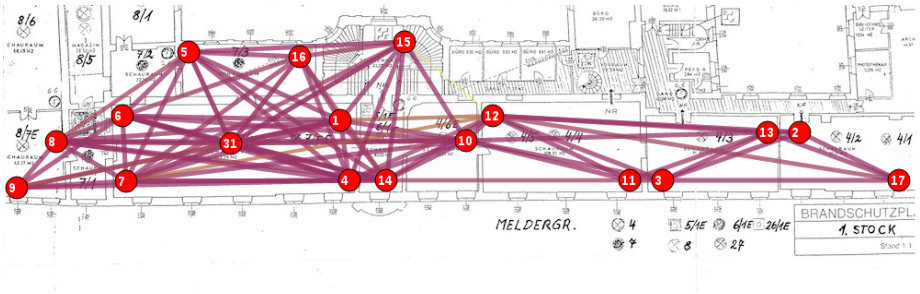

Fig. 1: Network topology, as exported from the OLSRv1 protocol

Name	*Desciption*	*Link*
BABEL		
BATMAN Advanced v4		
BATMAN Advanced v5		
BMX7		
BMX7TUN		
OLSRv1		
OLSRv2		
OLSRv2_MPR		

graph library. We repeated the process twice, with two different transmission power level for the testing network. With the transmission power set to 17dBm the network was considered too dense, and thus of little interest for the routing function. Therefore the power was lowered to 10dBm, and produced the distribution of path costs reported in fig. 2. Each entry is computed as follows, given the network graph and a couple of nodes A, B, the shortest path between A and B is computed via networkX and then the sum of the ETX values for the path is done[2]. We selected 4 nodes to perform the tests, node 8, 9, 17, and 2, that are at the extreme ends of the topology. fig. 2 reports the corresponding shortest path costs in the ranking.

put link to the code here

2.1 The Protocols and the Experiments

Several protocols were tested during the WBM, not all the protocols (or their variants) have been tested on all the configuration. **??** reports the list of the protocols, a brief description, and the link to the source code.

Four out of six days that make the BOM were devoted to set-up the testbed, and only the last two were dedicate to the testing itself. The procedure of set-up and testing is error prone due to a vast set of reasons that range from the need to use a recent OpenWRT/LEDE version, the potential incompatibility with the last version of the protocols, their configuration, and last, but not least, the eventual bugs that are found while testing and need to be fixed. This is a very important part of the BOM, possibly the

[2] For the code used for this task, see `http://github.com/`

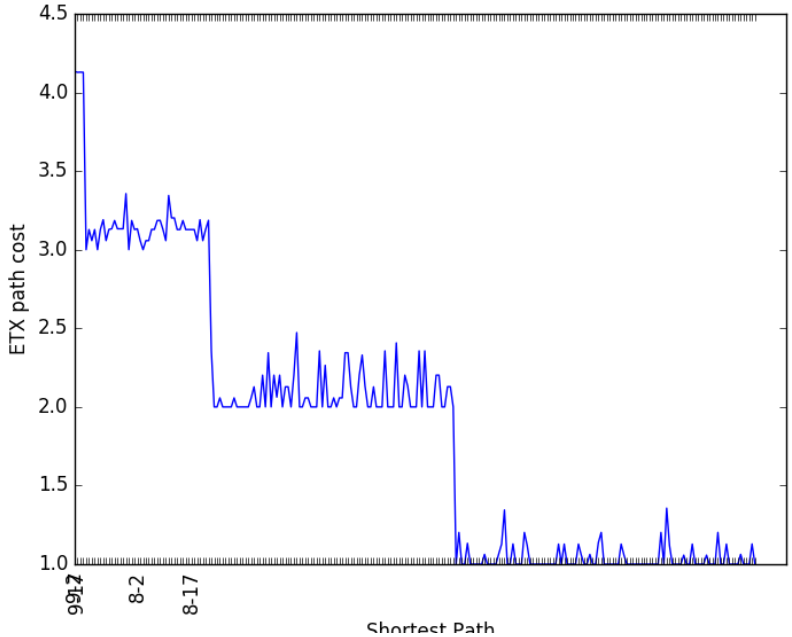

Fig. 2: The ranked list of the ETX weight of all the shortest paths in the network.

most important under the technical point of view (the social side of the BOM as equally important). During the tests the developers of the protocols (that are generally present at the event) test new features, compare different strategies and inevitably stumble upon unknown bugs in their protocols. This process is vital for them to improve their software and stabilize their code, and is arguably even more important than the results of the tests themselves.

Three set of experiments were planned, and two were fully performed:

- Ping test: a session of 100 pings from node 8 to 17, 8 to 2, 9 to 17, and 9 to 2 were performed to measure the loss and the delay distribution. This test was repeated only once.
- iperf test: a batch of 10 iperf sessions were run (10s each) between node 8 and 17 and from node 9 to node 2. This experiment was repeated with two more variants, one in which node 14 and 10 were running another iperf with a 5Mbps limit, and a second one in which another 5Mbps session was running from node 4 to node 15. These added sessions were not recorded but where used to increase the level of congestion in the central part of the network.
- pingall test: in this experiment, a random subset of all the possible couples was taken and an iperf session was performed. This test should have been repeated for all the protocols, with an without Airtime fairness enabled, but was not possible to complete.

3 The Results

3.1 Ping Tests

Figure 3 reports the percentage of lost packets in the ping tests. It shows that OLSRv1 and BMX are the ones that choose paths that are more conservative, so they deliver all or almost all their pings. OLSRv2, BATMAN4 and, to a lower extent BABEL are the ones that, instead, tend to choose paths that are more lossy. BATMAN5 largely underperforms compared to the others. This is an example of a typical situation in the BOM. BATMAN5 does not have a stable release, and the developers brought to the BOM a testing version, to initially study its performance. During the tests, these outlier performances made it possible to spot the presence of previously unknown software bugs, which become visible only when tested on networks larger than a certain size. It was not possible to patch BATMAN5 before the end of the BOM and thus, from now on the results of BATMAN5 are omitted.

Figure 4 reports data about the distribution of the RTT measured with ping. What clearly emerges is that BMX is consistently using paths with larger delays, and that BABEL on three cases over 4 has a very large range of values. OLSRv2 performs worse than OLSRv1 in the majority of the cases, with delays comparable to BATMAN. Note that OLSRv2, BATMAN, and BABEL are advantaged in this comparison, since they have non-zero loss. It is reasonable to assume that the packets that could not be delivered, if they could reach the destination would probably have a large RTT.

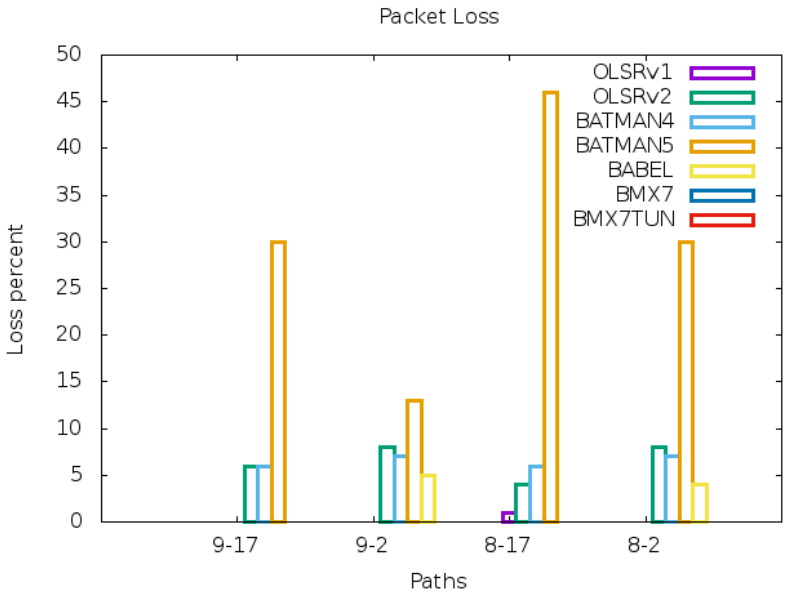

Fig. 3: The percentage of lost packets per protocol

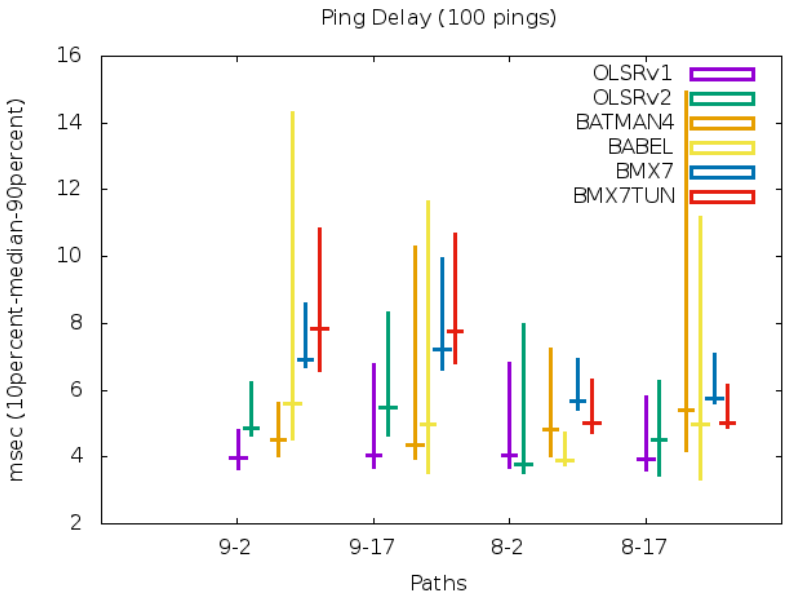

Fig. 4: The values of the 10th percentile, 90th percentile, and median value of the RTT for all the pings

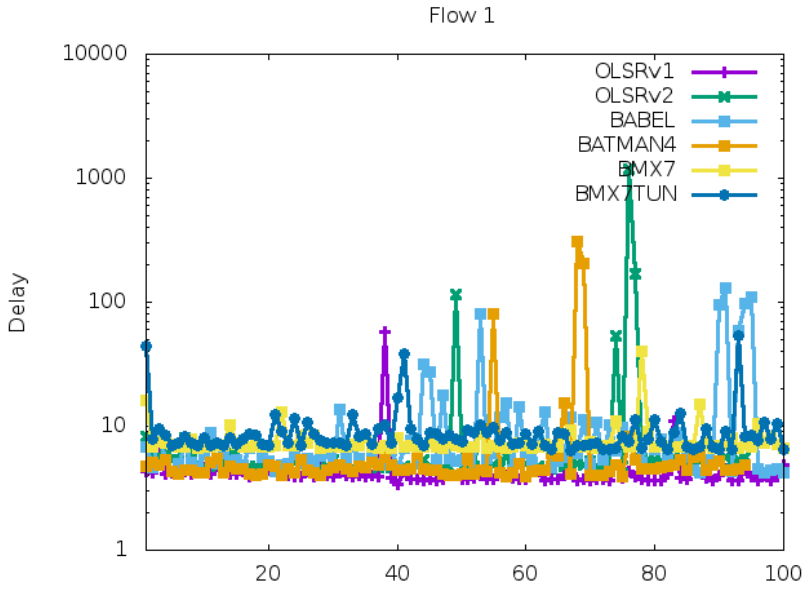

Fig. 5: The values of the RTT per each ping, 9 to 17

Finally, figs. 5 to 8 report the whole set of RTT measured for each run. Note that the tests were done in sequence, so the network conditions may have varied from one test to the next one, note also the log scale on y axis. It is clear that there are two phenomenons, one is the difference from one protocol to another that impacts the median value, the other is the presence of many outliers that influence the whiskers of fig. 4.

3.2 iperf Test

BMX7 without traffic, enjoys better TCP performance due to the zero loss and and more stable delay (see image of delay). when load increases we don't know.

4 Suggestion for the Future Battlemeshers

- assign microtasks to people in the room. scream the outloud, document them on othe web (like an etherpad) and use something to catch attention (a trumpet!) when necessary. This way some tasks can be delegated to random people, especially the creation of commands to parse results, or to create graphs.
- use cable to reach the nodes (or powerline)
- do small tests, so you understand what happens

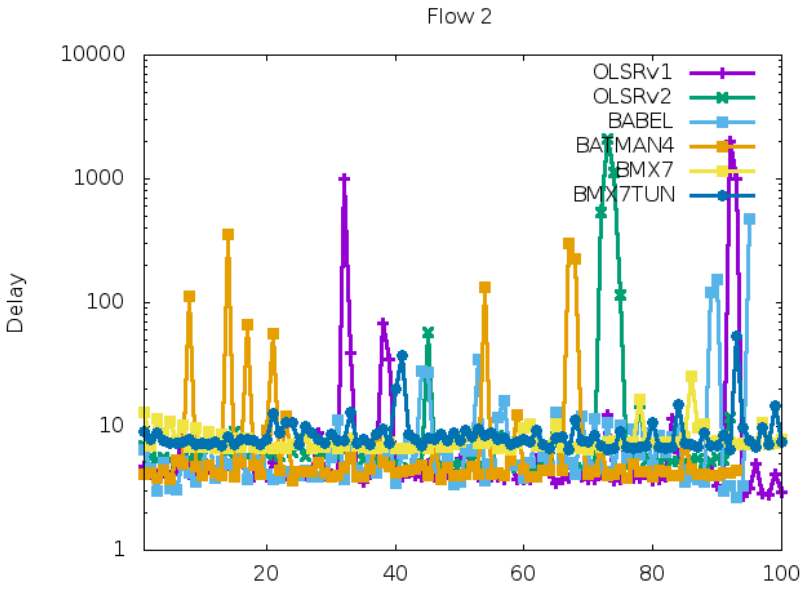

Fig. 6: The values of the RTT per each ping, 9 to 2

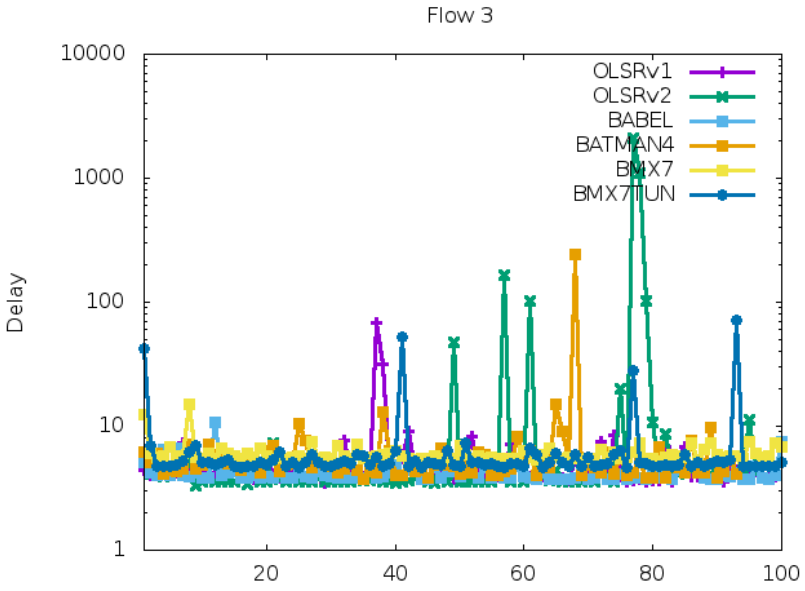

Fig. 7: The values of the RTT per each ping, 8 to 17

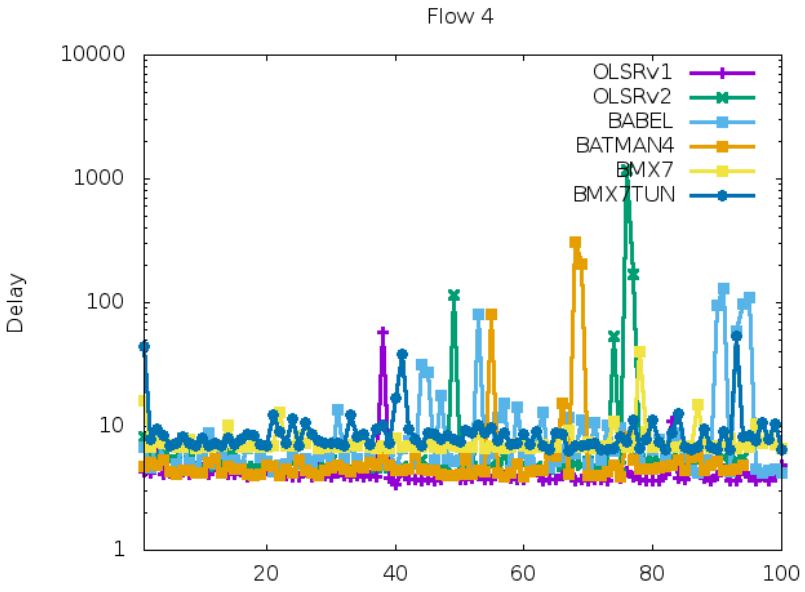

Fig. 8: The values of the RTT per each ping, 8 to 2

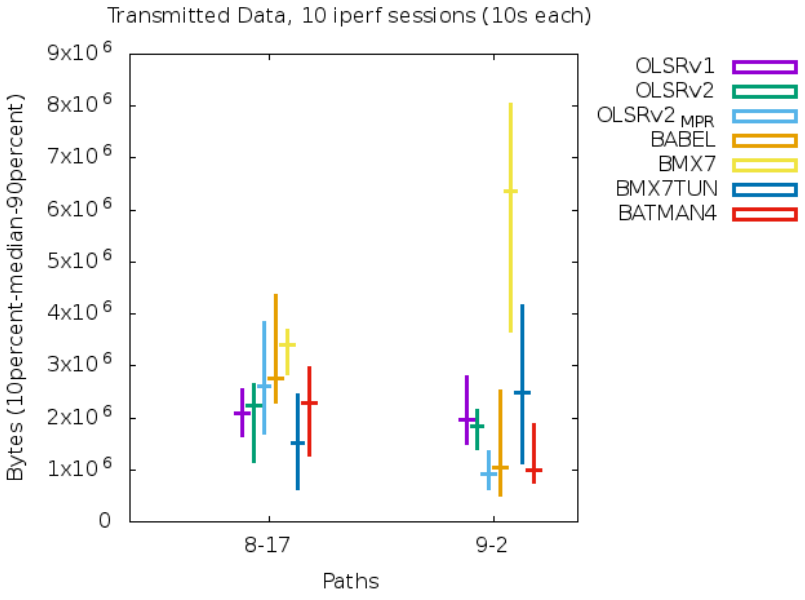

Fig. 9

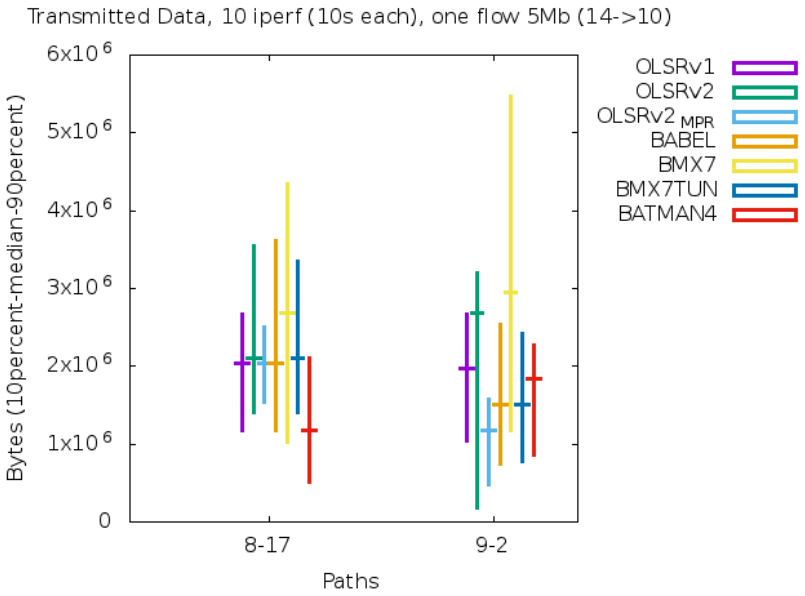
Transmitted Data, 10 iperf (10s each), one flow 5Mb (14->10)
Bytes (10percent-median-90percent)
6x10^6
5x10^6
4x10^6
3x10^6
2x10^6
1x10^6
0
8-17
9-2
Paths
OLSRv1
OLSRv2
OLSRv2 MPR
BABEL
BMX7
BMX7TUN
BATMAN4

Fig. 10

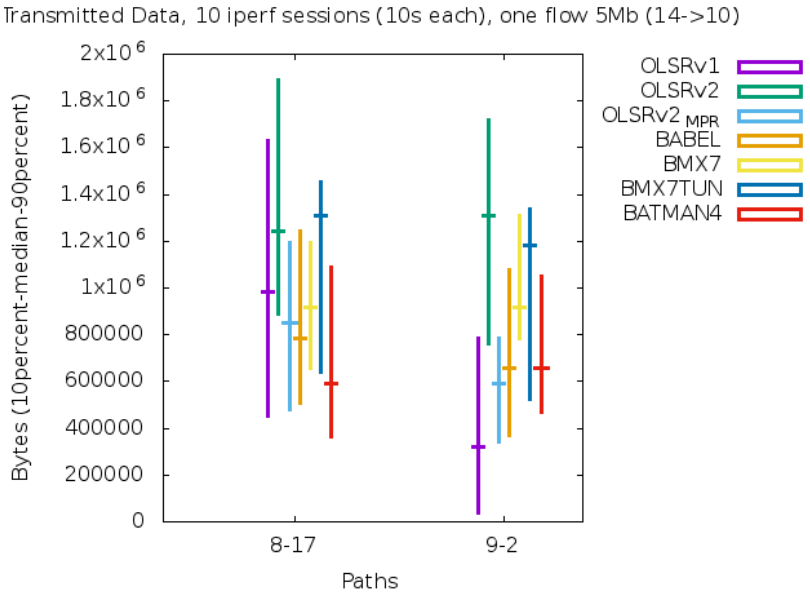
Transmitted Data, 10 iperf sessions (10s each), one flow 5Mb (14->10)
Bytes (10percent-median-90percent)
2x10^6
1.8x10^6
1.6x10^6
1.4x10^6
1.2x10^6
1x10^6
800000
600000
400000
200000
0
8-17
9-2
Paths
OLSRv1
OLSRv2
OLSRv2 MPR
BABEL
BMX7
BMX7TUN
BATMAN4

Fig. 11

- fork the testbed. use a small testbed for the devs to make their patches, otherwise when something does not work they will lock the testbed for too long.
- bring some hardware to perform night-time batched long tests, like a raspberry

5 Thanks

thank the organizers here

Riferimenti bibliografici