Measurement Results from Wireless Battle Mesh Version 7

Type: Measurement Analysis (work in progress)
Creation date: June 15, 2014

Event:
Sublab. Leipzig, Germany
12th to 18th of May 2014
http://battlemesh.org/BattleMeshV7

Contents

1 Introduction

WBM...

2 Data and System Repositories

```
http://wibed.confine-project.eu

https://github.com/battlemesh/wibed (buildroot)
https://github.com/battlemesh/wibed-battlemesh-experiment (package)
http://wiki.confine-project.eu/wibed:start
https://github.com/axn/wbm2pdf  (this stuff, branch wbmv7 in future)

Raw measurement data:
http://wibed.confine-project.eu/resultsdir/wbmv7-axn-16_2014-05-16_19-28-43 (stationary scenarios)
http://wibed.confine-project.eu/resultsdir/wbmv7-axn-17_2014-05-16_20-13-20 (broken crossed streams scenario)
http://wibed.confine-project.eu/resultsdir/wbmv7-axn-19_2014-05-16_21-35-33 (mobile scenarios)
```

3 Testbed ad Experiment Descripiton

3.1 Deployment

During the first days of the event a total of 20 wibed nodes have been deployed. 16 wibed-nodes have beend spread over 3 different floors in the main event building. About 10 of these 16 nodes were located in the main event hall (approximately 300 sqare-meters workshop room) with highest node density in a particular corner of this room (deathroom) and the 6 in the below and above floor of the event hall. Three more nodes have been placed in a neighboring building with wireless connectivity. One node was battery powerded for allowing mobile-node scenarios. In fact not all node positions were always exactly known as nodes were sometimes moved to fullfill specific experimentation-scenario requirements. In each building 1 of the wibed-nodes were configured as GW nodes and blocked for experimental usage. The remaining 18 nodes were shared between three different experimentation groups for running tests and different scenarios (each node was used by at most one experimentation group at any time).

All experiments were performed in a single 5Ghz channel. However, due to the presence of around 50 participans with wireless laptops and several other actively used wireless equipment, also the used 5GHz channel was likely affected by non-testbed related interference.

The experiment presented in this work was lead by one of these groups and used the following 16 nodes:

NodeID	Location	exp:axn-16 (stationary)	exp:axn-17 (broken)	exp:axn-19 (mobile)
164a7a	deathroom			
3b3a90	workshopRoom			
3b3d70	????			
3e9db0	deathroom??	9db0->1ab0		9db0->4174
51aac8	halleAnfang			aac8->4174
8a417e	deathroom	417e->4174	417e->1ab0	
c24174	HalleEnde (mobile)		4174->1936	
c2427a	deathroom??			427a->4174
ce3360	EloiTable			

e4b63a	mustiTable			
e60a62	halleMitte			
e60aac	deathroom			
e60ad6	deathroom			
e61936	axelsTable	1936->4174	1936->4174	1936->4174
f41ab0	kloschi (building B)	1ab0->4174	1ab0->417e	1ab0->4174

3.2 Experiment Configuration

The highly dynamic and uncontrollable interference in the measurement environment made it inpossible to ensure equal conditions for sequential experiment executions. Therefore, to ensure equal (fair) environment conditions for all tested protocols, all routing protocols were running and observed in parallel on all nodes, thus all being always exposed to the exactly same environmental conditions. Only netperf-based throughput tests, seeking to measure the achievable performance of the end-to-end routing path established by the individual protocols, were performed sequentially.

The accepted downside of this approach is of course that protocol overhead introduced by one protocol or by protocol-observing tools like ping (causing total overhead in the order of few KB/s, as can be seen from later measurements) slightly affects the maximum achievable end-to-end throughput of other protocols (in the order of seveal MB/s).

To avoid protocol-bootstrapping effects (eg unfinished neighbor-, path- or topology-discovery), all routing protocol deamons were started at least 200 seconds before any measurement.

Followup measurements were executed from a singe selected node by launching a pre-deployed test script [?] which...

Each followup measurement last 200 seconds during which the following additional tools were used to observe protocol performance and overhead:

- ping

3.3 Experiments

The experiment focused on measuring the overhead and performance of 5 different mesh routing protocol implementations in static and mobile scenarios. The five tested protocols were batman-advanced (batadv), bmx6, olsr, olsr2, and babel. Unfortunately, we just discovered after the measurements that the babel protocol daemon was not configured correctly, leading to broken routing decisions for multi-hop path. Therefore all babel-protocol related measurements are discareded in the following discussion.

3.4 Stationary Nodes Measurements

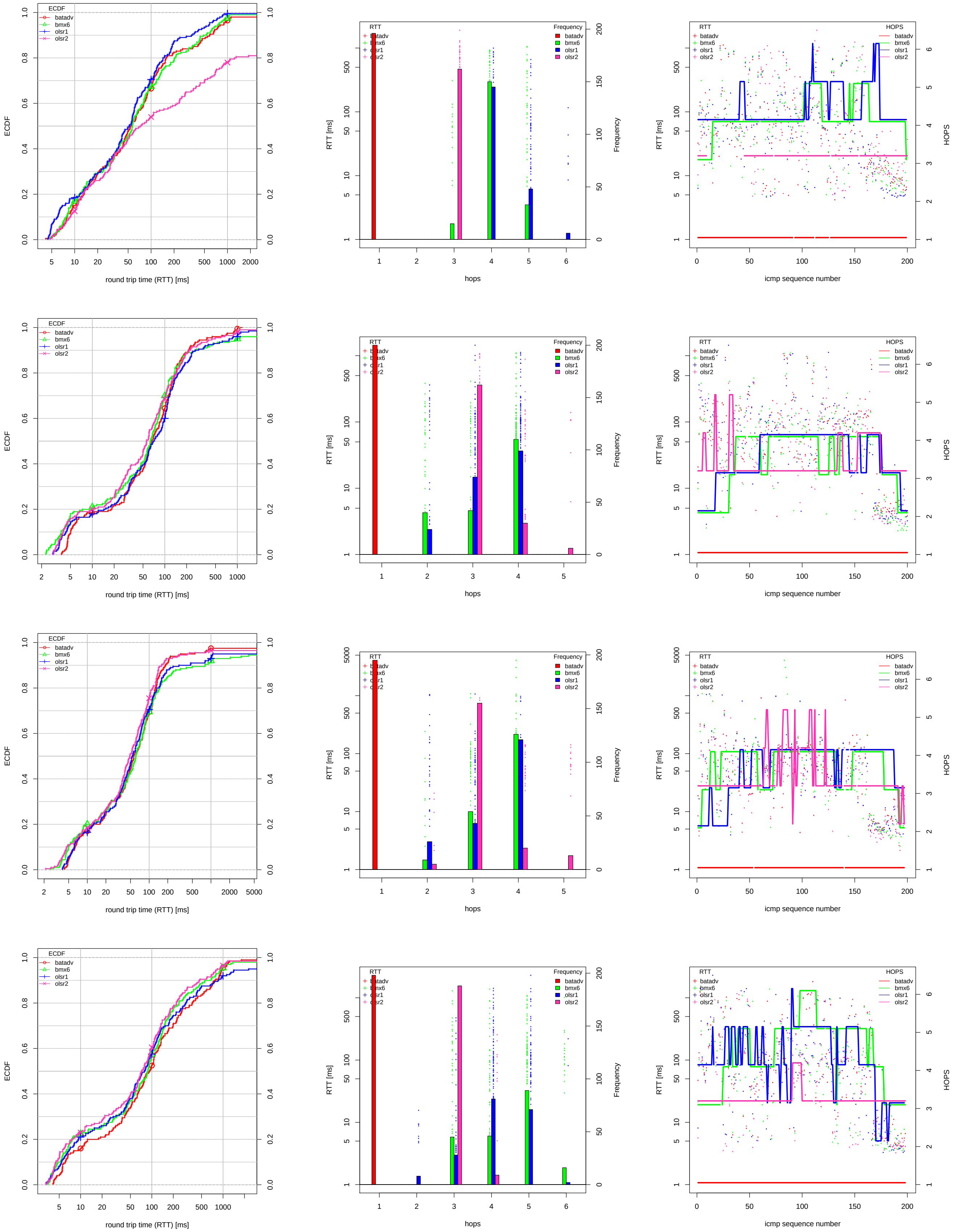

Table 1: End-to-end ping6 performance between two stationary nodes: 9db0-1ab0, 417e-4174, 1936-4174, 1ab0-4174

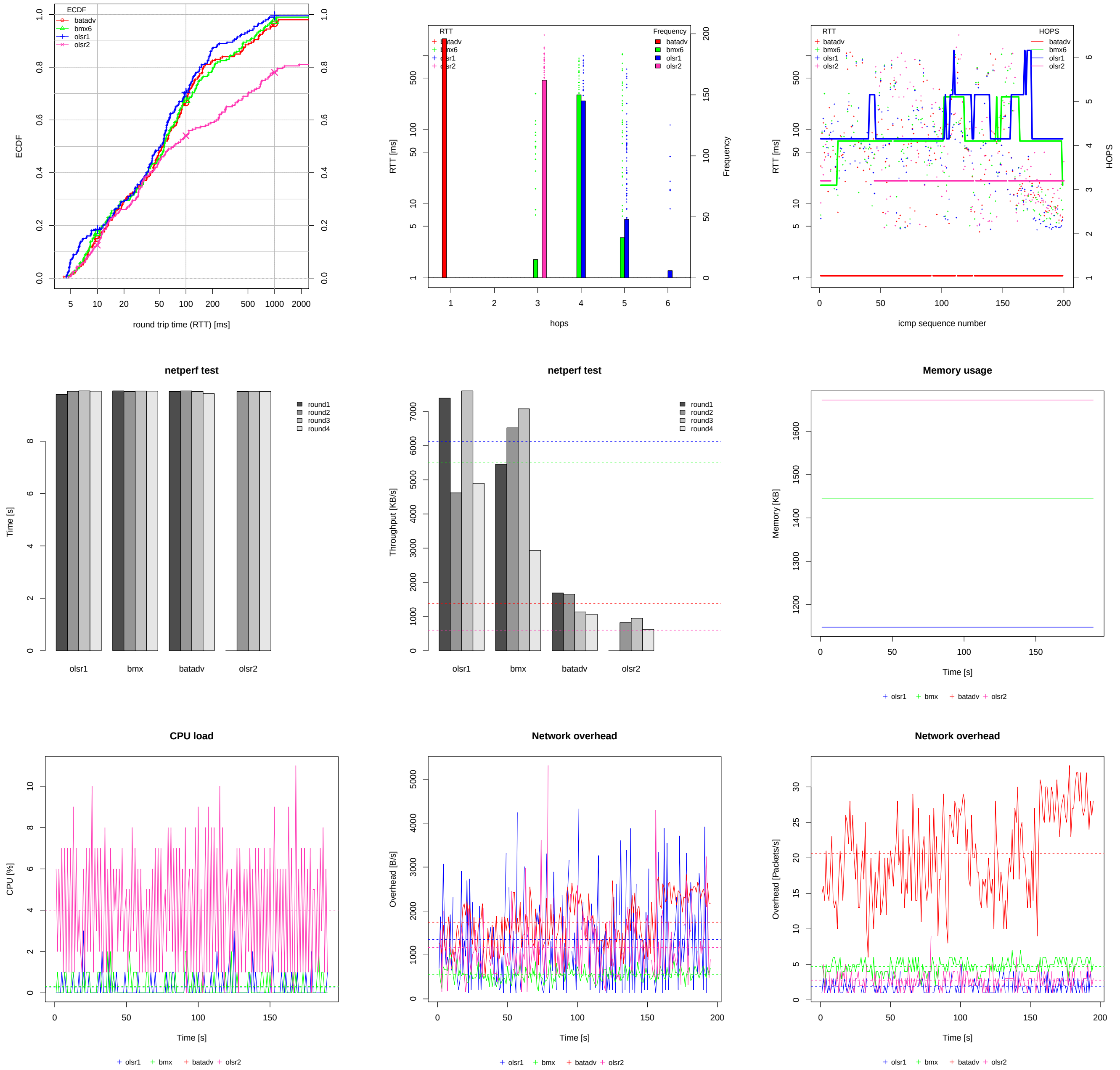

Table 2: Overhead and end-to-end performance between two stationary nodes: 3e9db0 and 1ab0

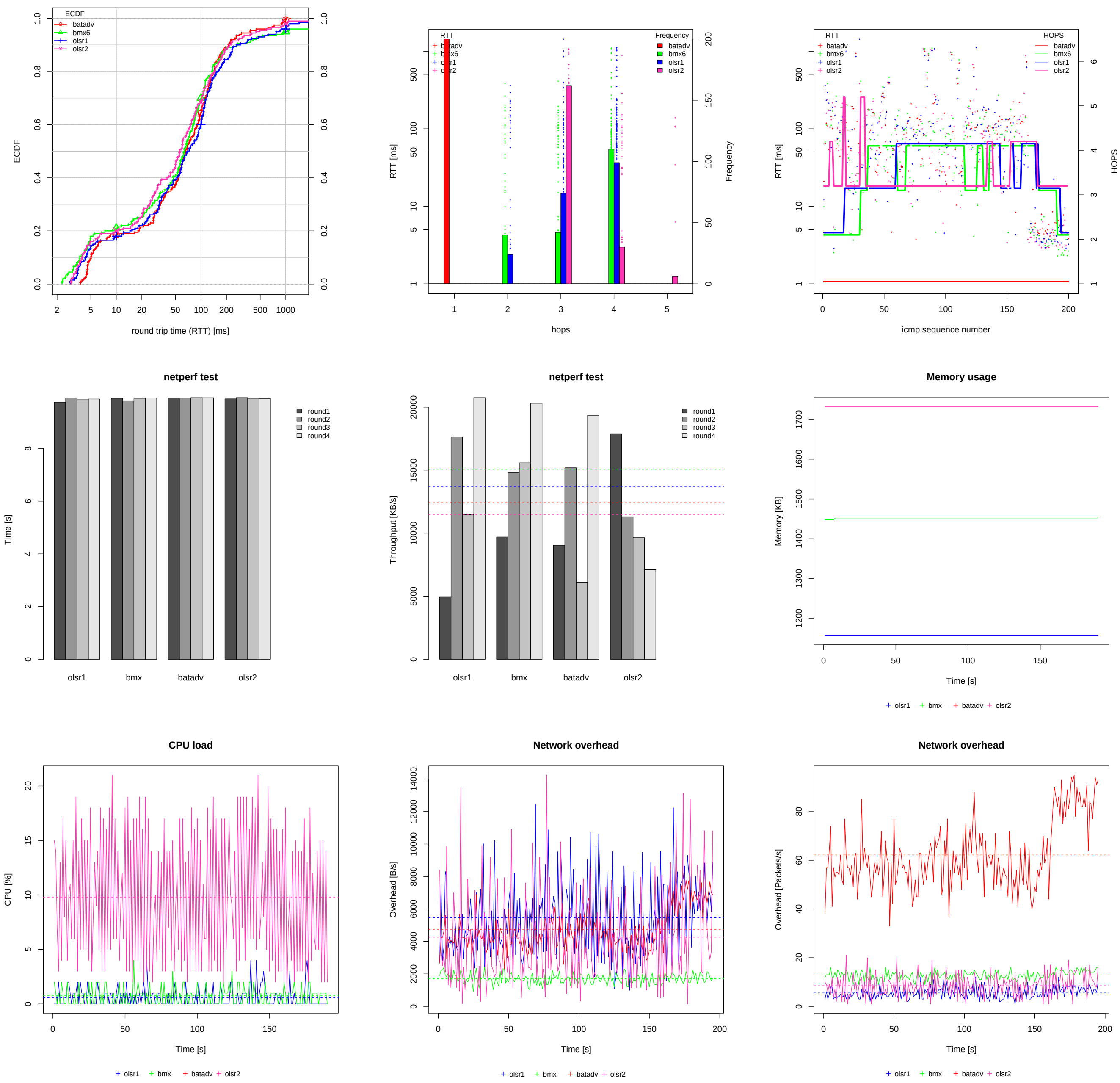

Table 3: Overhead and end-to-end performance between two stationary nodes: 8e417e and c24174

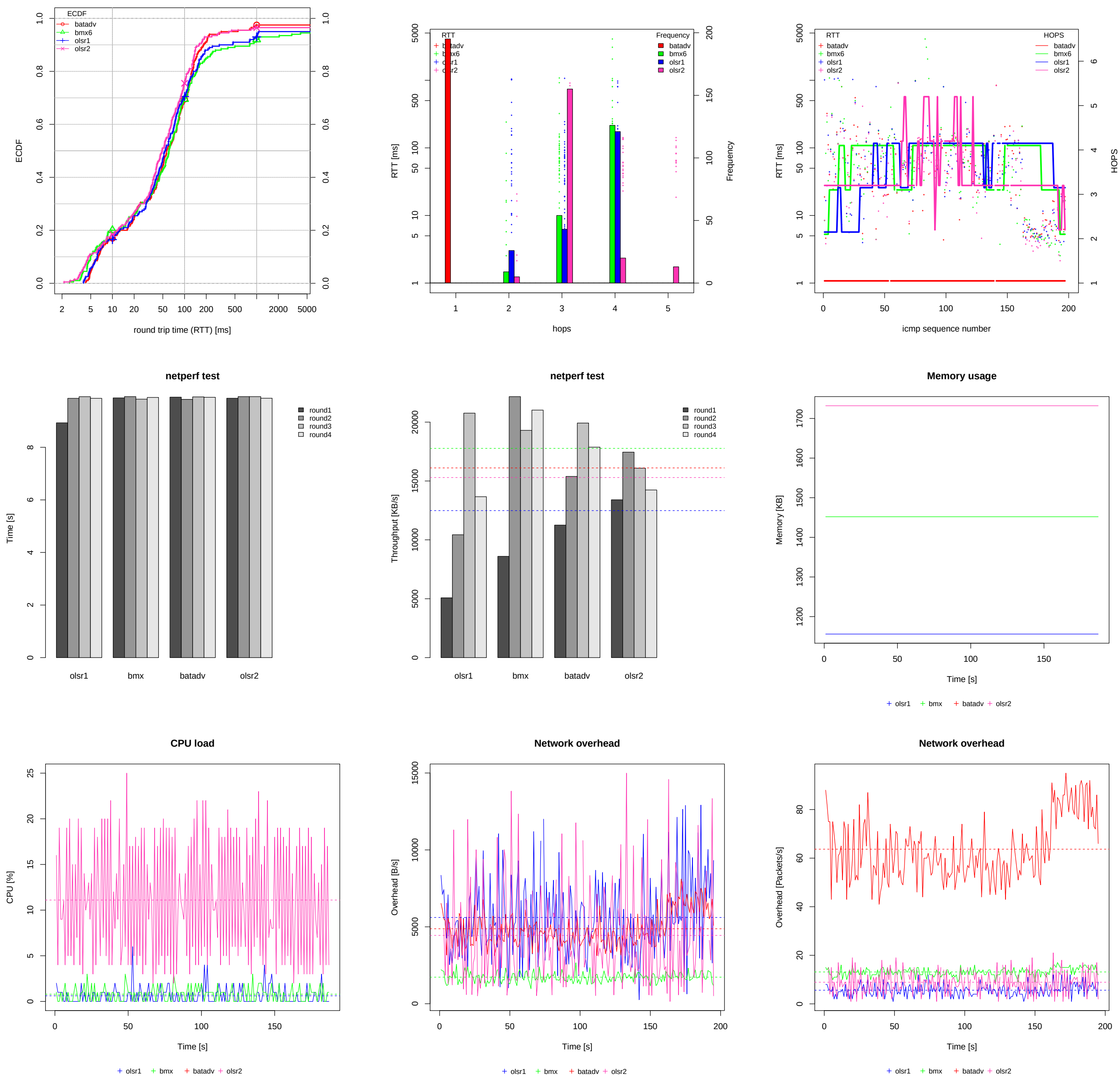

Table 4: Overhead and end-to-end performance between two stationary nodes: e61936 and c24174

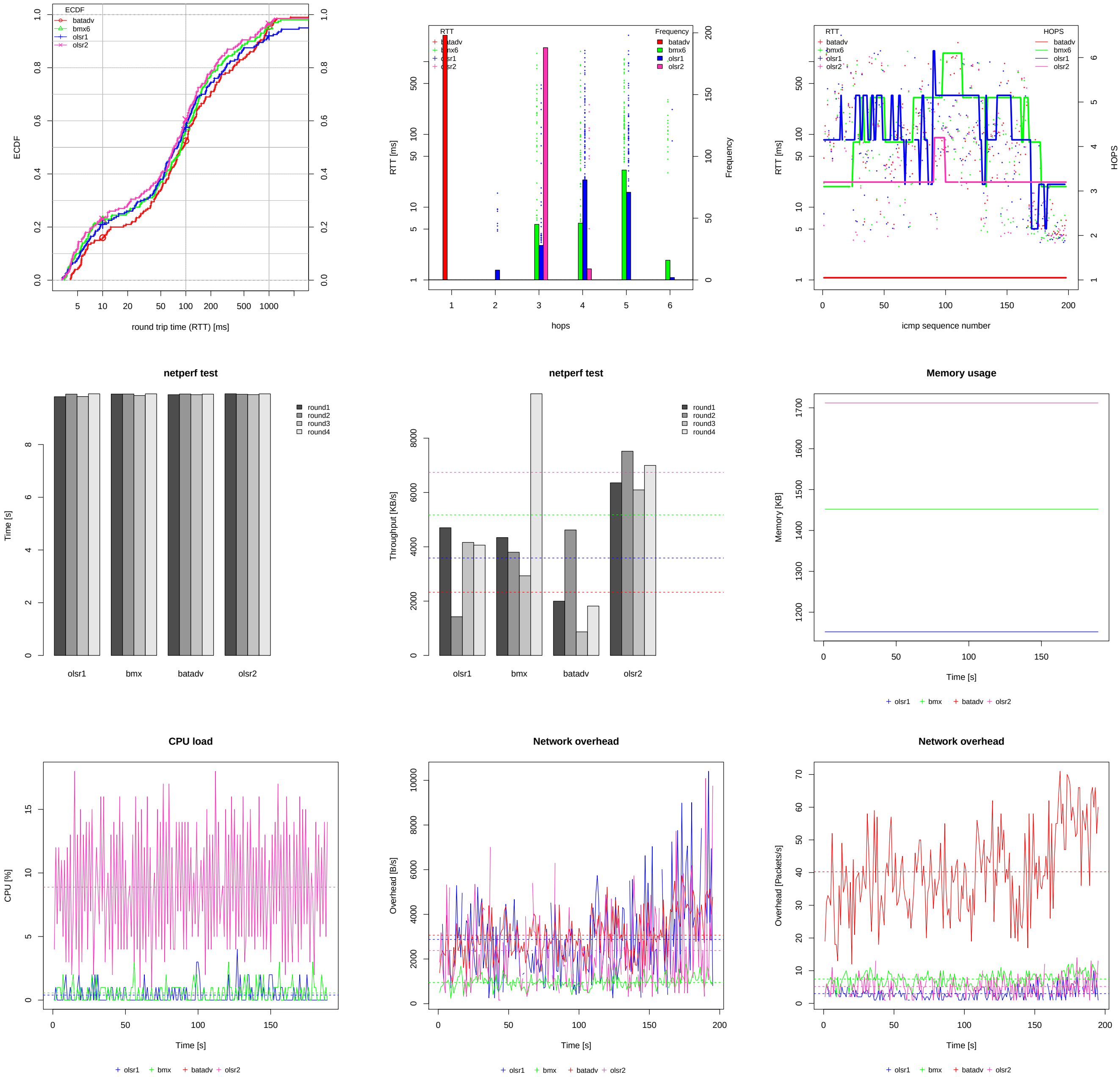

Table 5: Overhead and end-to-end performance between two stationary nodes: f41ab0 and c24174

3.5 Mobile Node Measurements

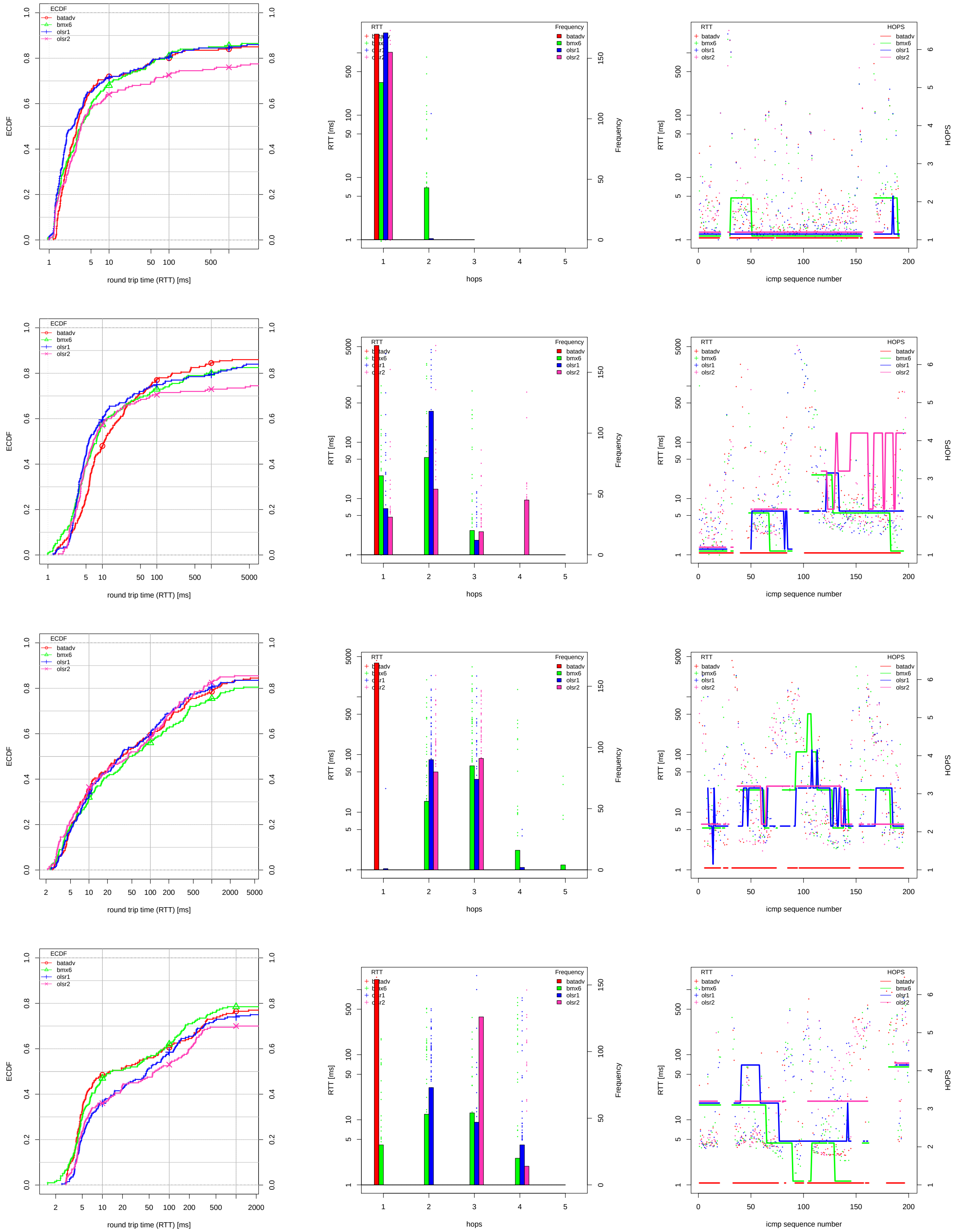

Table 6: End-to-end ping6 performance to mobile node 4174 from aac8, 1936, 1ab0

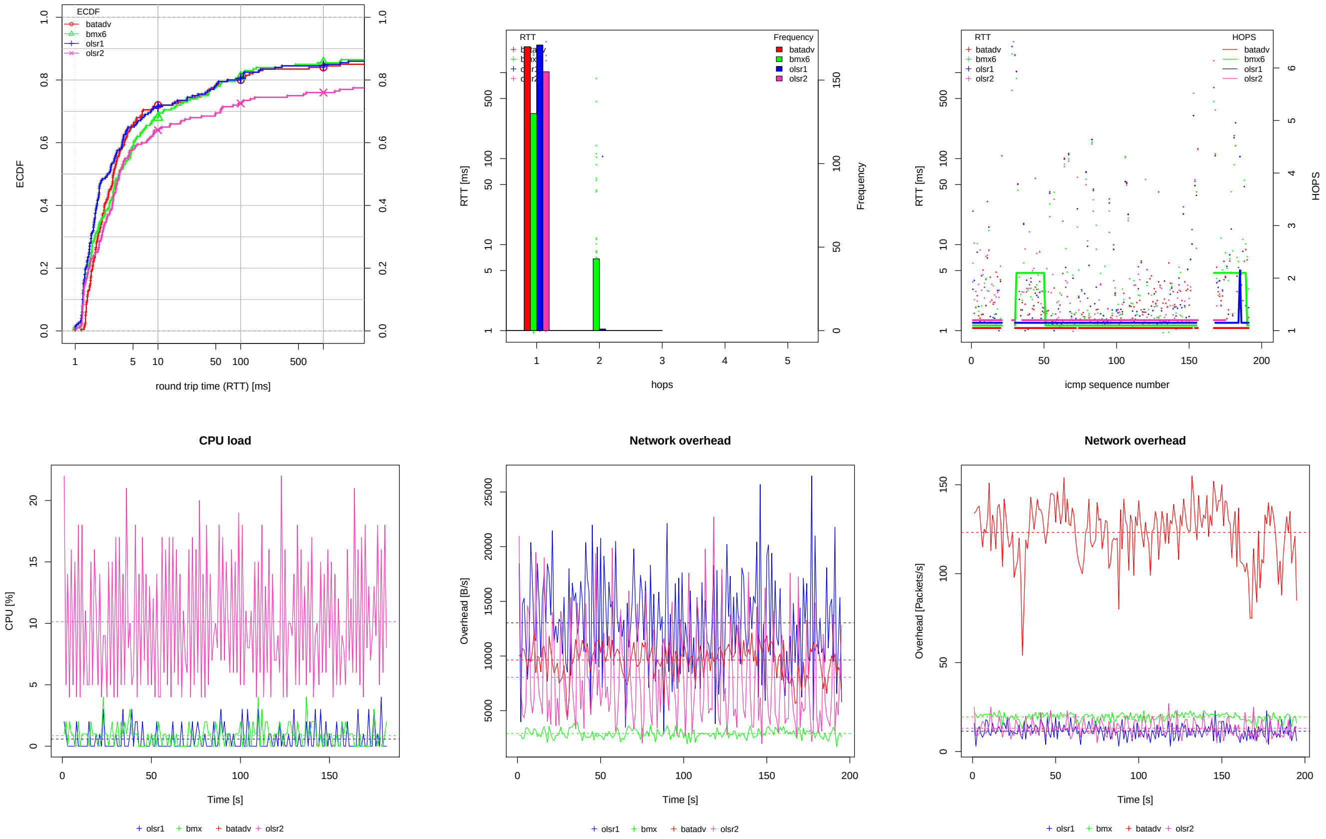

Table 7: Overhead and end-to-end performance to mobile node c24174 from stationary node 51aac8

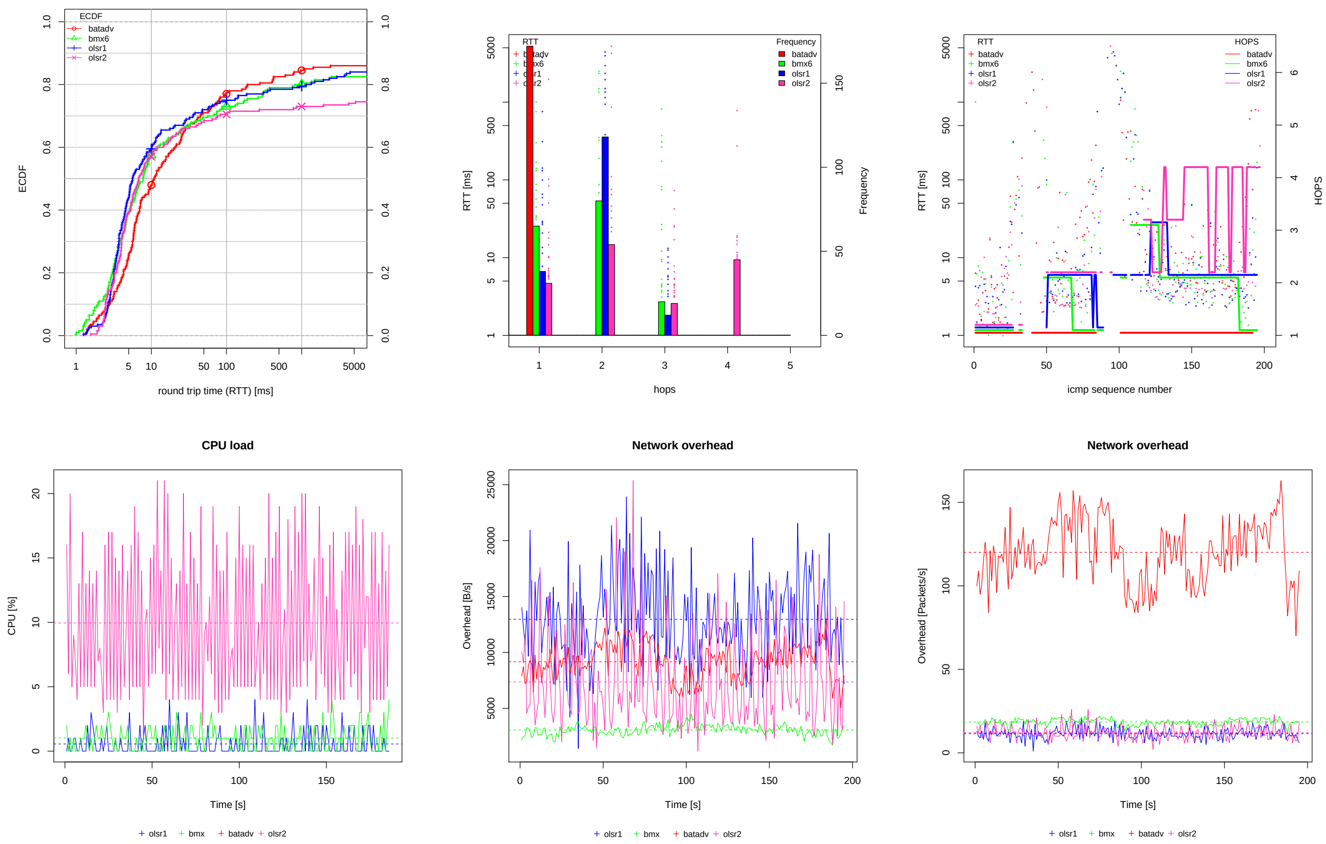

Table 8: Overhead and end-to-end performance to mobile node c24174 from stationary node e61936

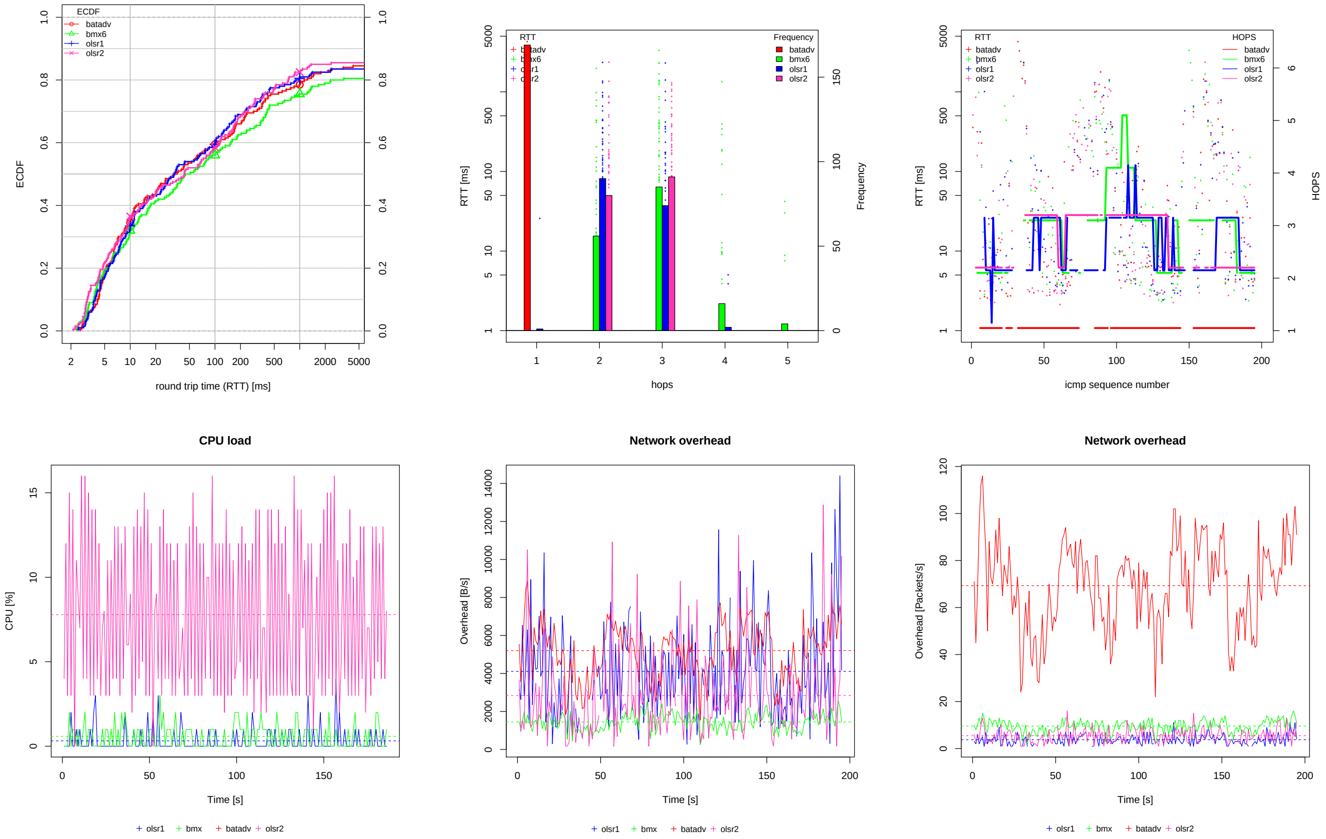

Table 9: Overhead and end-to-end performance to mobile node c24174 from stationary node f41ab0

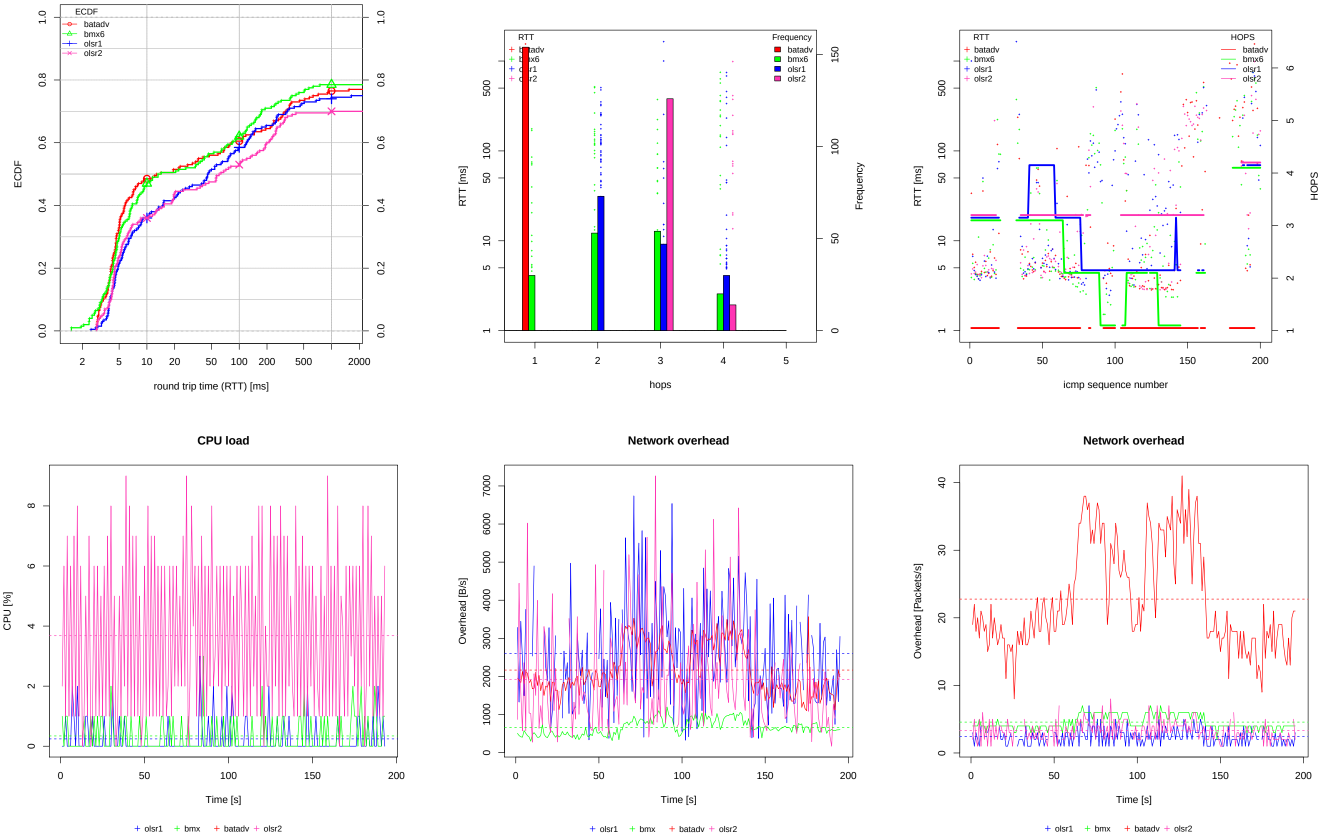

Table 10: Overhead and end-to-end performance to mobile node c24174 from stationary node c2427a

3.6 Mobile Scenarios

4 TCP Throughput Measurements

5 Recommendations for next battlemesh

- Traceroute and mrt often show high packet for intermediate nodes. This is due to a kind of denial-of-service mechanism enabled by default in Linux kernel. WIth this mechanism the kernel simply discards frequent icmp responses (eg due to exceeded TTL values). This behavior can be disabled by lowering the default net.ipv6.icmp.ratelimit=1000 setting, eg via: sysctl -w net.ipv6.icmp.ratelimit=10

6 Appendix