Appendix G
Test Report for 100 Test Runs for the Hazelcast Prototype

Appendix for Master Thesis:
Different Paths to High Availability
by Introducing Redundancy
in a Distributed SCADA System

Morten Andersen, Årskort 19941458
Master of Information Technology, Software Development
Department of Computer Science, University of Aarhus
8200 Århus N, Denmark

June 14, 2011

Contents

1 Comments to the Graphs

This appendix contains the detailed test graphs for all 100 test runs for the Hazelcast based prototype. The unprocessed data that these graphs are created from, are included in the source archive for the thesis in the test-runs/hazelcast-timings folder.

Figure 1 below is a summary of all 100 tests in one graph, showing the *maximum* observed client data age for each of the 100 test runs as the solid line, and the *mean value over the max values* as the dashed line.

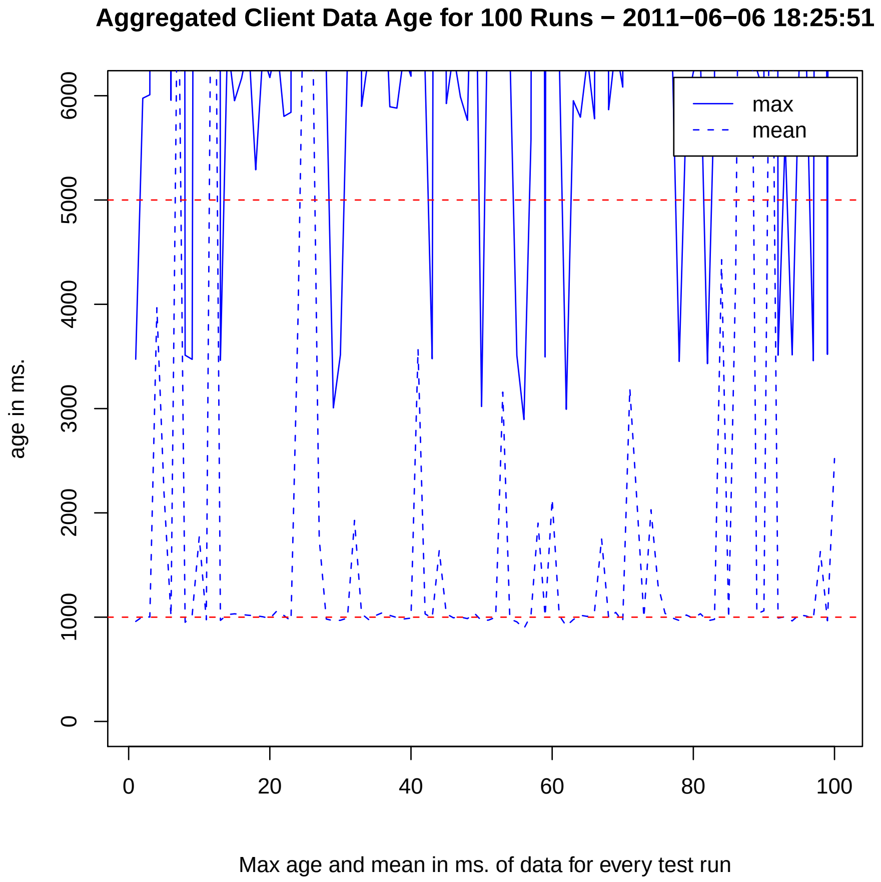


Figure 1: Aggregated max and mean of the max values for all 100 test runs.

Figure 2 shows the same data as figure 1, but is not capped at 6000 ms for the y-axis, showing in full extend how badly the prototype performed at some of the failover situations.

The two figures shows that only 20% of the test runs was successful in that the maximum observed value for the test run was below the 5 seconds deadline (QAS1). For the remaining 80% percent of the tests there were at least one failover situation that failed in fulfilling QAS1.

Based on these aggregated graphs it is relevant to study the graphs for the individual test runs in chapter 3 to search for a pattern in the failing tests.

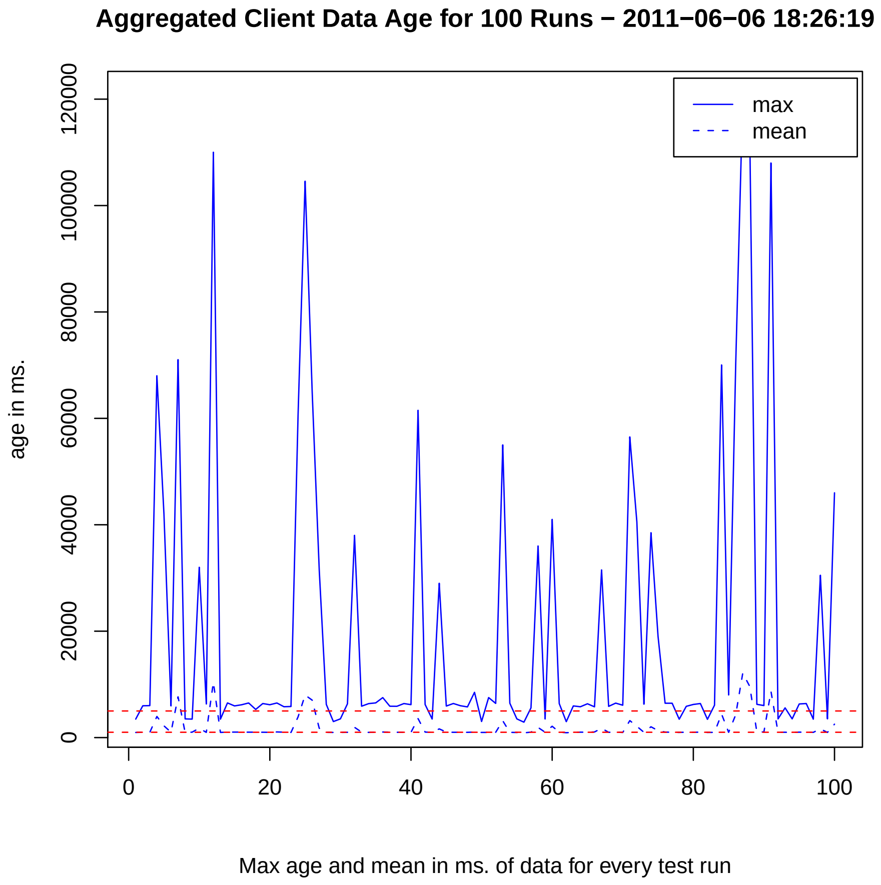


Figure 2: Aggregated max and mean of the max values for all 100 test runs.

2 Network Measurements

The two figures on the following page shows the network measurements of the nodes in the test system before and after the 100 test runs.

It is seen that the network layout as reported by the hop length between the nodes are stable during the test. This verifies that no live migration of running systems happens in the Amazon EC2 data center.

With respect to the network round trip times (rtt) 4 notable outliers are seen in the test after the tests was performed. These are the 4 white spots in figure 4, with rtt crossing 100 ms. This is somewhat surprising and shows that one must expect a certain degree of variation in network latency in the test system. This would have been a problem if the deadlines in the wind farm SCADA system has been in the millisecond range, instead of in the seconds range, and might be a factor in explaining the large deviations in the tests. But as this variation does not seem to have a negative influence on the end-to-end based prototype it is not likely that these network fluctuations can be attributed as the cause.

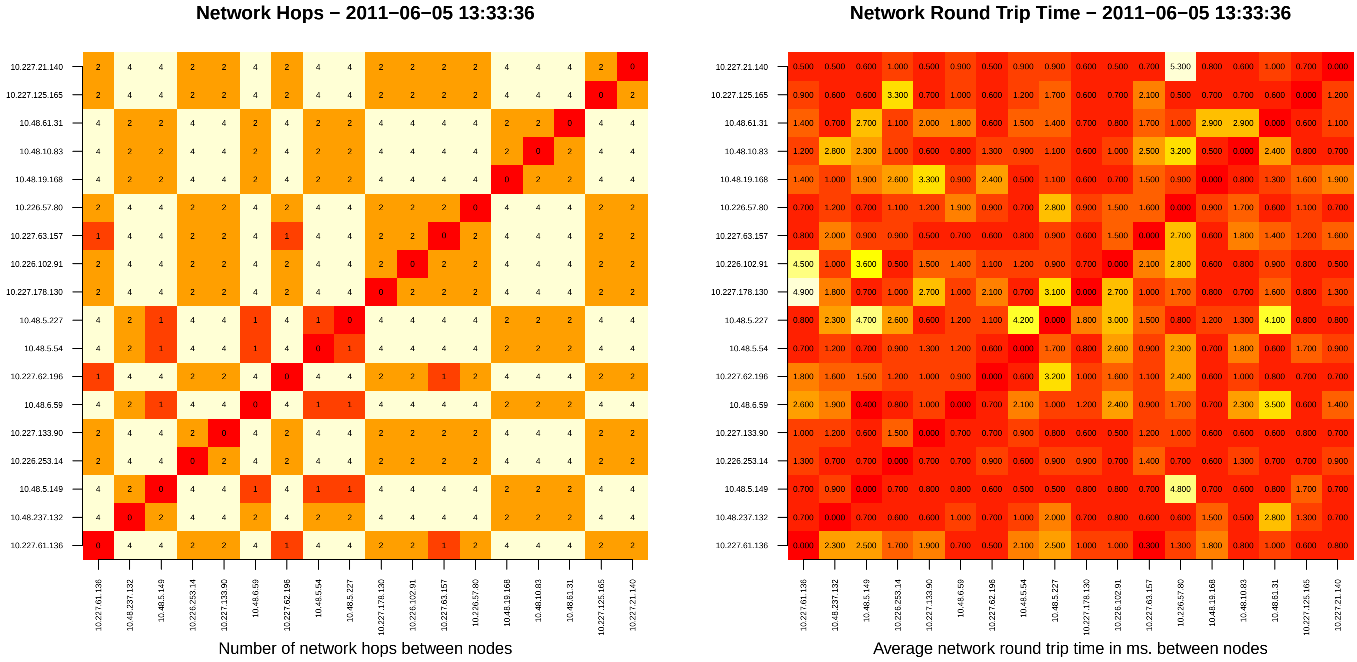


Figure 3: Network measurements *before* the test runs.

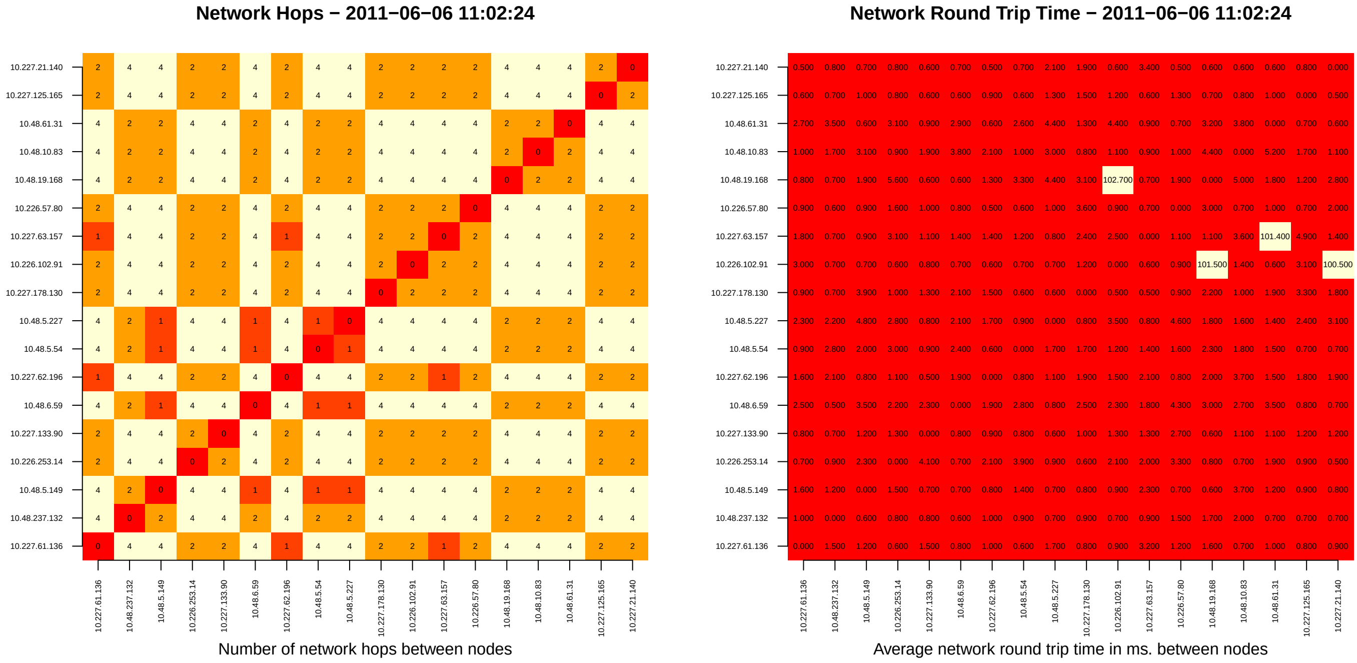


Figure 4: Network measurements *after* the test runs.

3 Measurements for the 100 Prototype Test Runs

The following 100 images each depict the measured timings during *one* prototype test run. It is seen that some of the test runs, such as test number 1, nicely fulfills the QAS deadlines. Then there are tests like test number 2 where there is a single failover situation where the 5 seconds deadline (QAS1) is slightly exceeded.

But then there are also some of the test runs where the prototype almost breaks down, like test number 4, 5 and 7 where no data seems to reach the clients in the first couple of periods. The most extreme of these failures are test number 12, 25, 26, 87, 88 and 91 where the data age reaches almost 110 seconds before falling back to 1 second.

The actual data for one of the problematic periods in test number 12 are depicted below in table 1 for one of the clients. From the min column, showing the minimum age of the 100 sensors that the client subscribes to, it is clear that some sensor values are steadily coming through to the client. Based on the mean column values of 25 seconds and the max values of 100 seconds it can be estimated that if the age for the 100 sensor values falls in just two groups, one group with ages around 1 second, and one group with values around 100 seconds, approximately 25 of the sensor values are not getting updated. This could be interpreted as data from 2 or 3 turbine nodes are not getting through to the client.

```
"min";"mean";"max"
 ...;.....;......
 544;24173;100017
 544;24472;101017
 544;24771;102017
 544;25070;103017
 547;25372;104020
 543;25666;105016
 548;25970;106021
 543;26264;107016
 551;26571;108024
 543;26862;109016
 526;12018;110021
 522;733;977
 525;736;981
```

Table 1: Data from one of the clients from test run number 12 (source archive file: test-runs/hazelcast-timings/ 20110605_160332_timings/46.137.54.3.csv).

But this analysis of what is actually going on is only a hypothesis, that will need to be verified by collecting more fine grained data from the nodes. As the problem only seems to be present at the beginning of the test runs, it could to be attributed to some of the turbine nodes not joining the Hazelcast cluster correctly. A hypothesis that would also need further debugging to verify.

So summarizing there are at least two problems with the prototype:

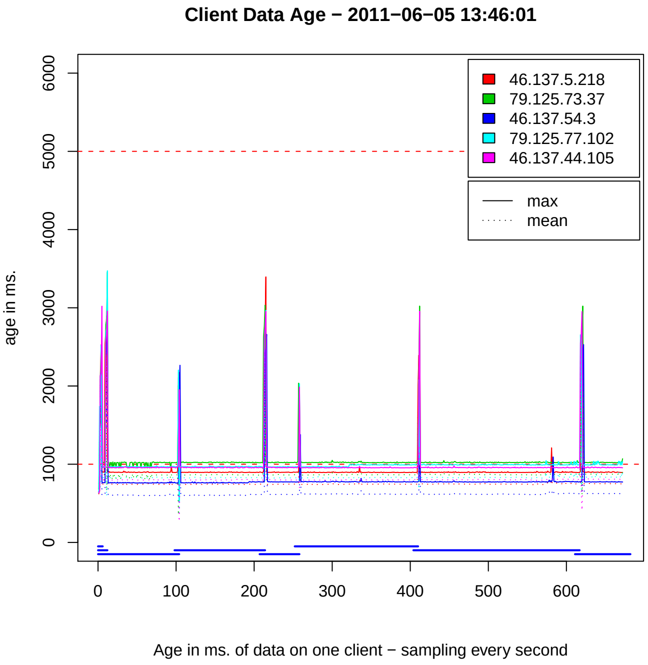
1. Occasionally small exceedings of the 5 second deadline. This problem

might be hard to solve, as it is not expected to be a direct bug in the source code, but more a performance problem with the Hazelcast multicast algorithm.

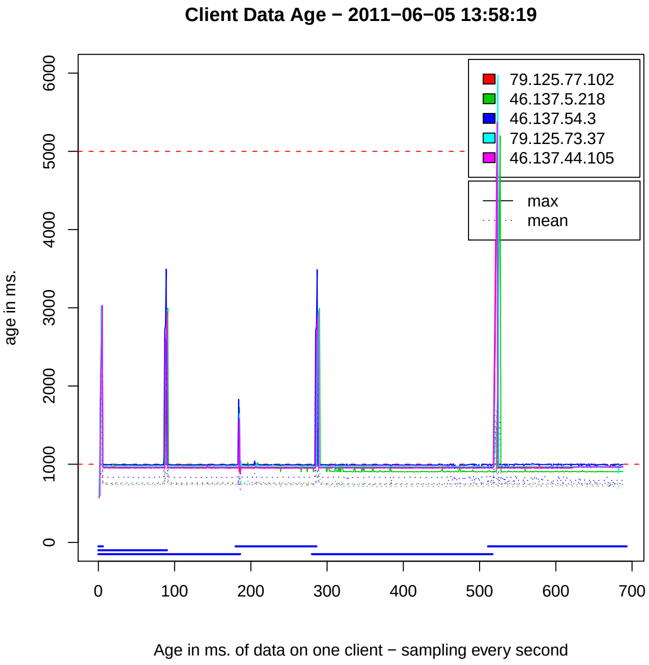
2. Problems at startup for some tests, leading to significantly exceeding the 5 second deadline. This problem is probably attributable to a bug in the source code. With some effort it is assumed that it would be possible to find and fix this.

3.1 Crash After Test Run 17

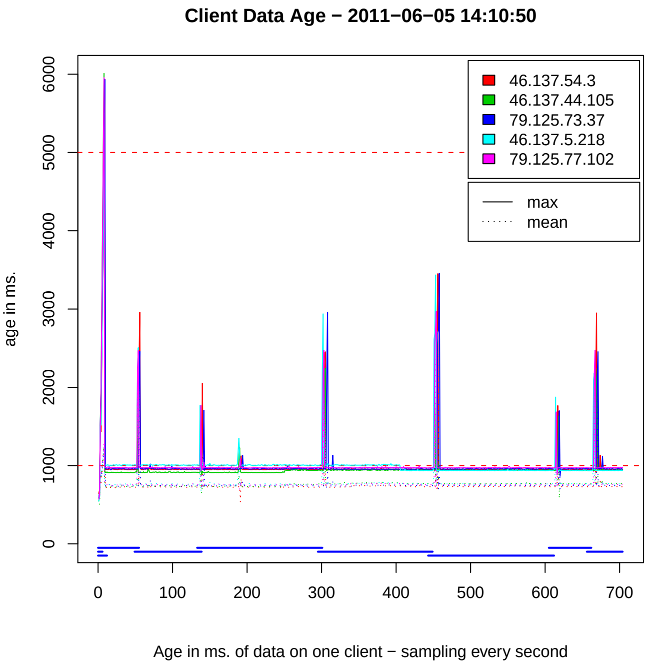
During the test run the management console running on the local workstation crashed after test run number 17. Therefore the test was restarted at that point for the remaining 83 tests. When the test runs was later analysed it was noticed that this restart poses a statistical problem as the seed for the restarted sequence was also reset, wherefore test number 1 and test number 18 is actually a test of the same crash sequence (and so on up until test number 17 and 34). Seen from a straight statistical view the Hazelcast test should be rerun to be completely comparable to the other test run. But from a pragmatic view it is not assumed that this would lead to any change in the conclusion about the lacking performance of the Hazelcast prototype.



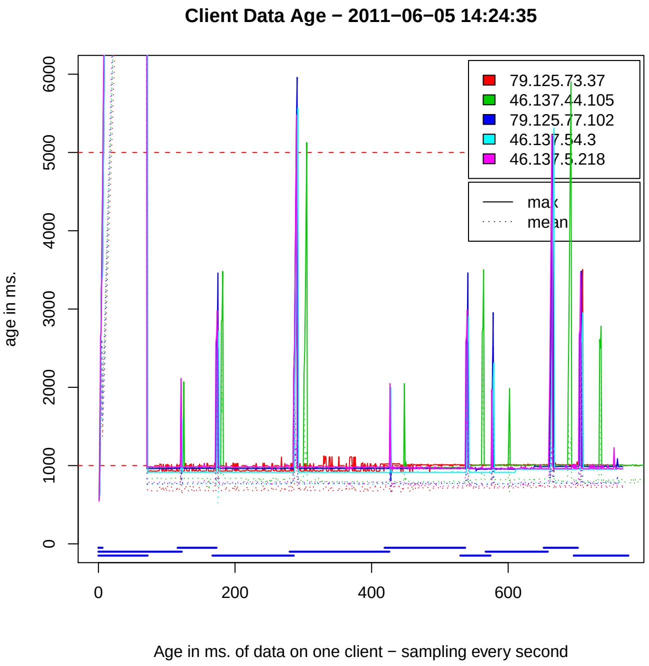
(Test run 1)



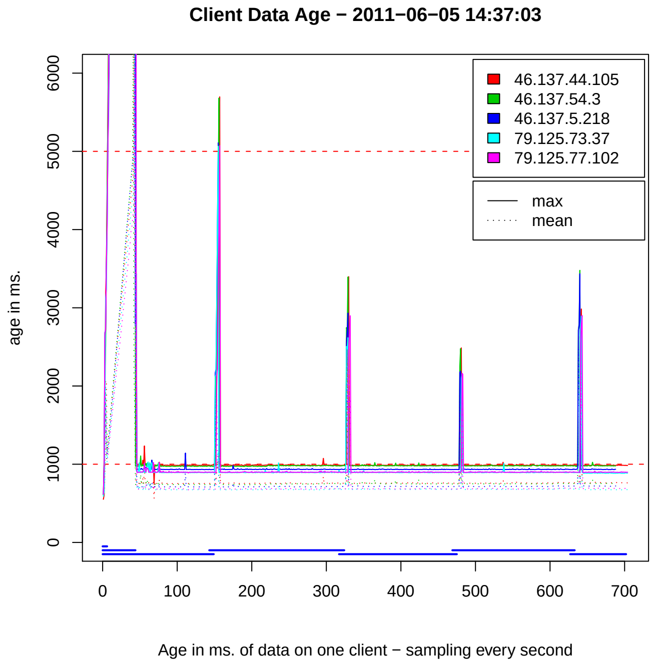
(Test run 2)



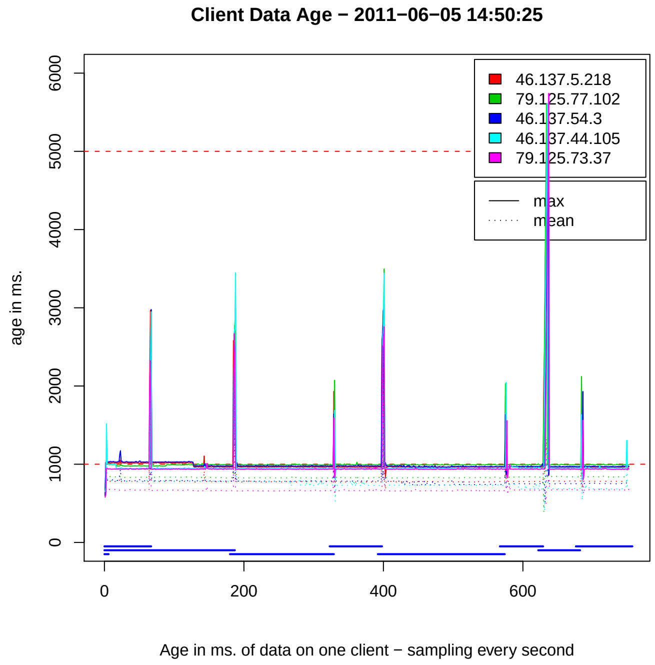
(Test run 3)



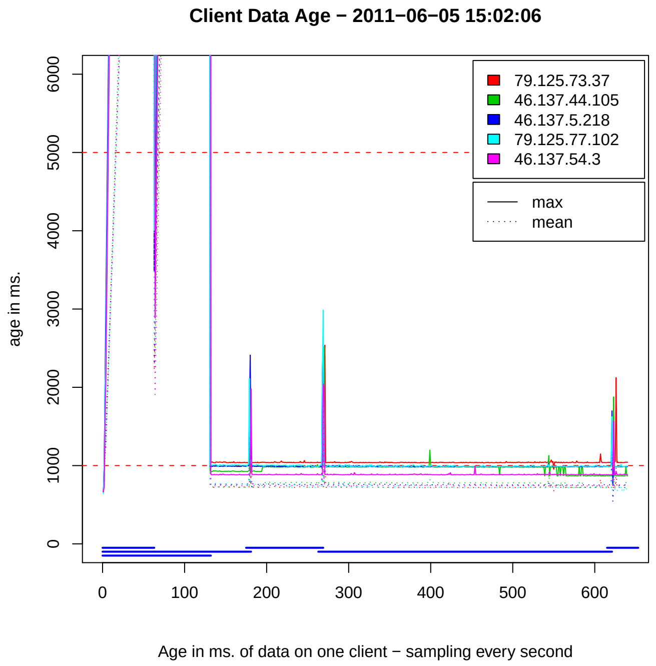
(Test run 4)



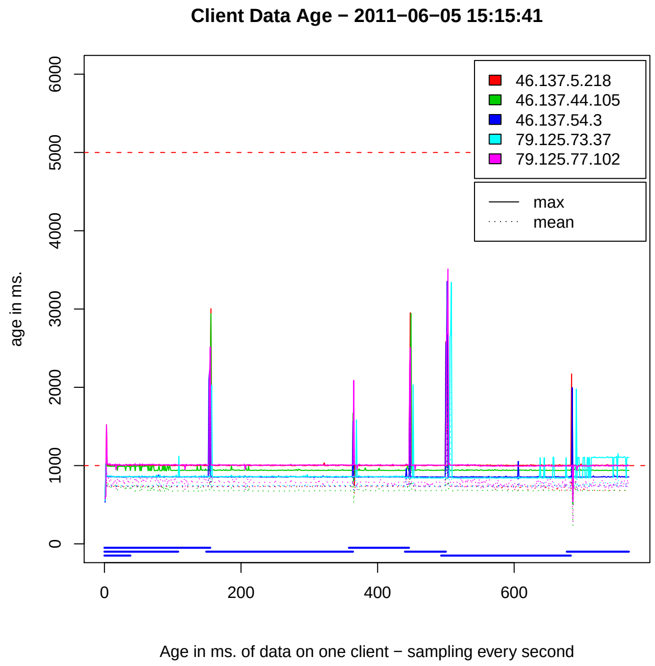
(Test run 5)



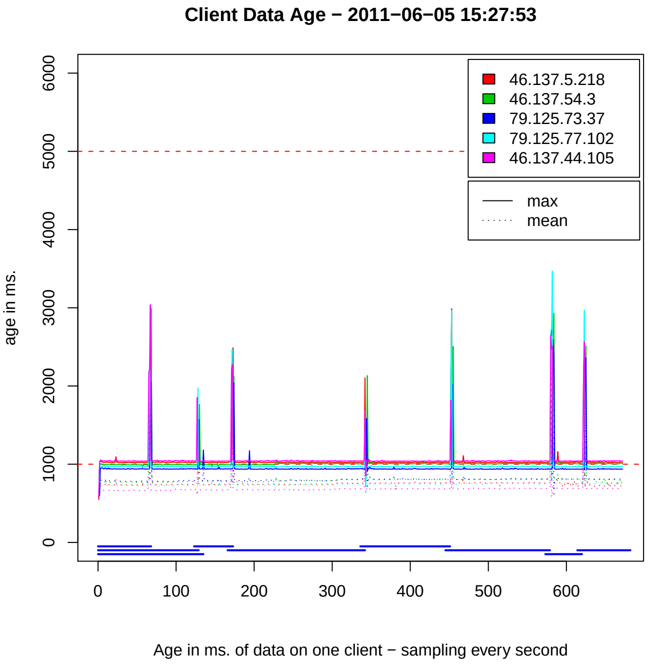
(Test run 6)



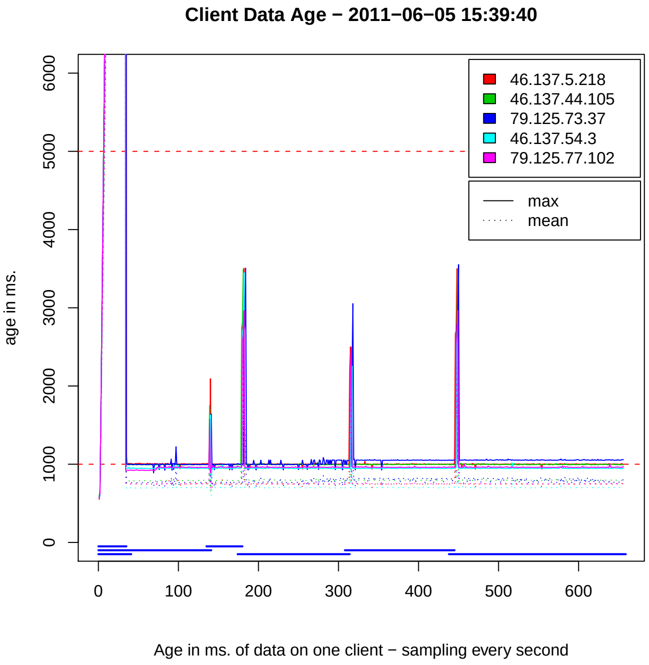
(Test run 7)



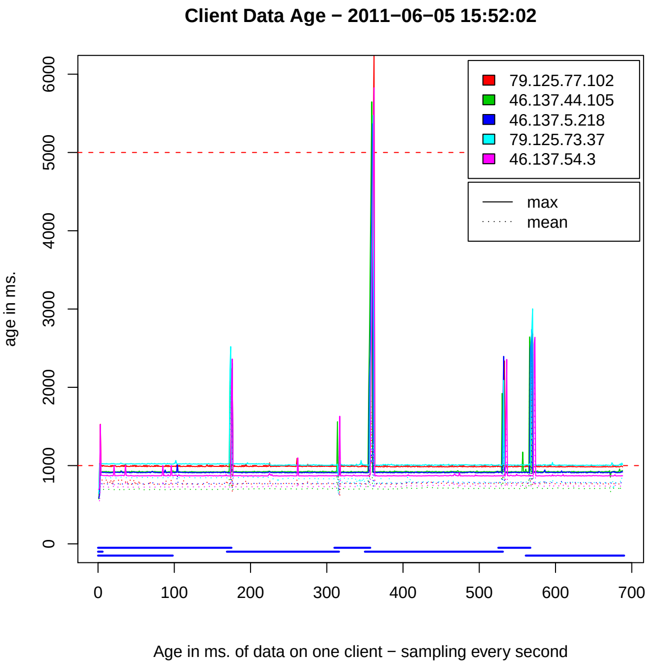
(Test run 8)



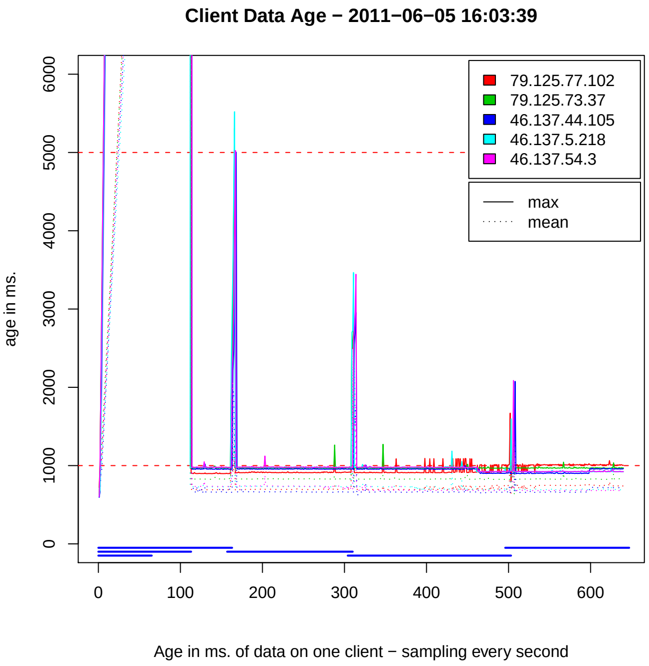
(Test run 9)



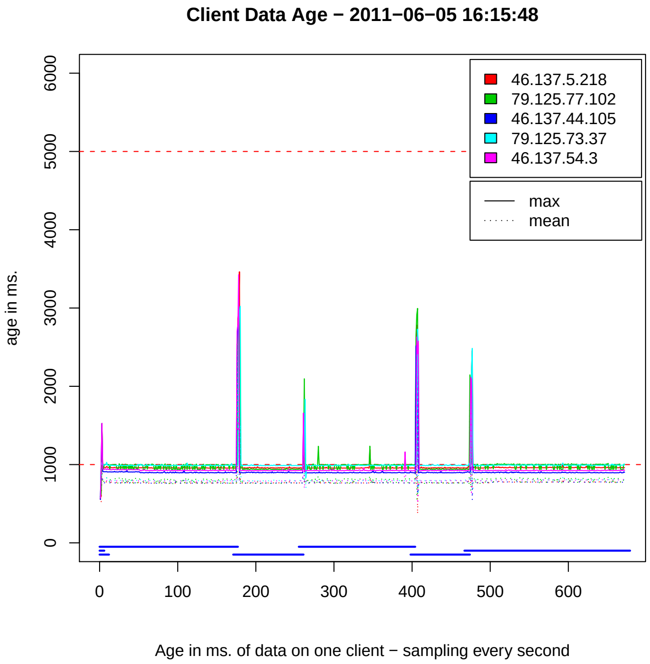
(Test run 10)



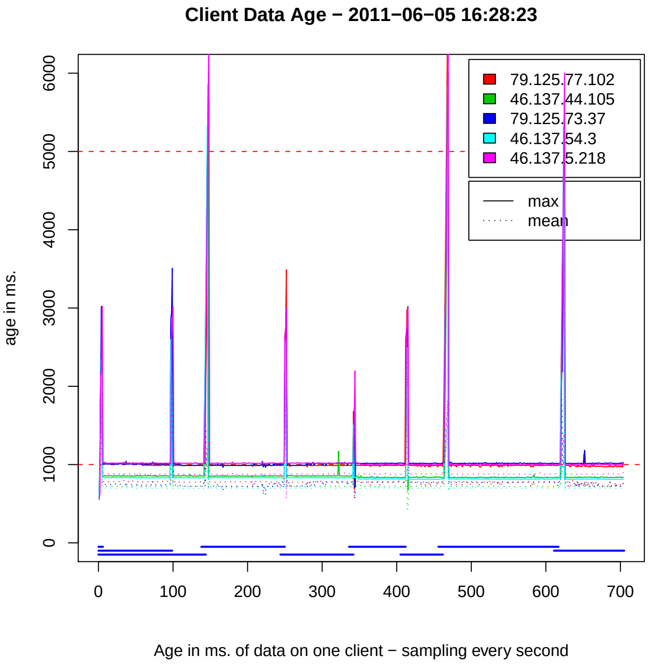
(Test run 11)



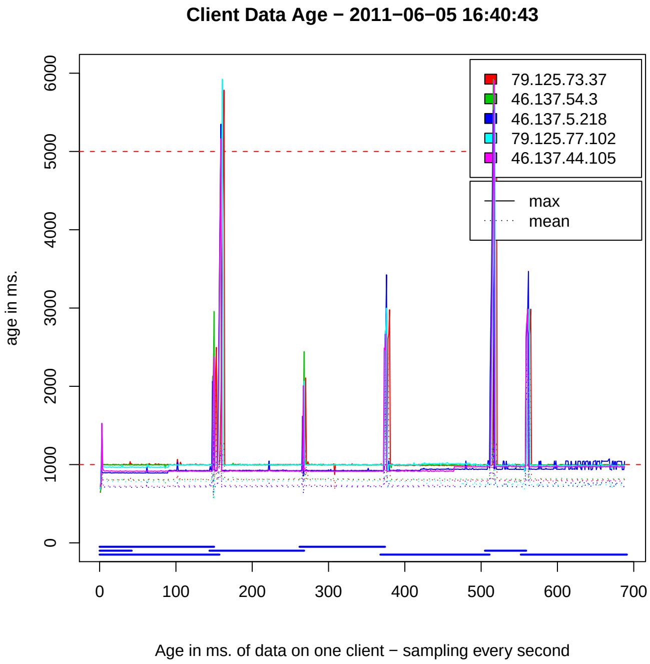
(Test run 12)



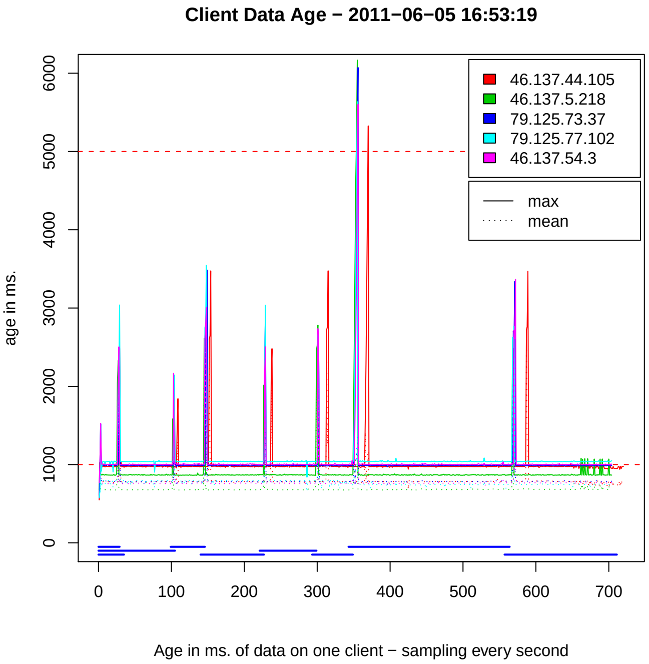
(Test run 13)



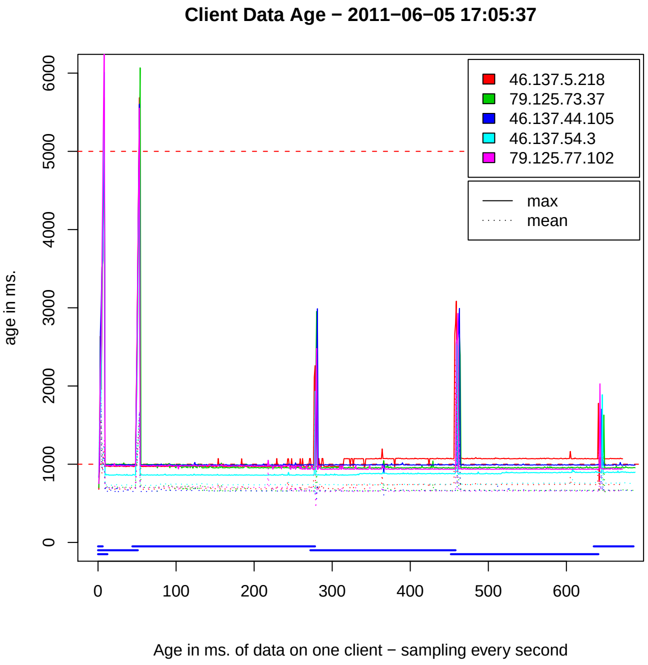
(Test run 14)



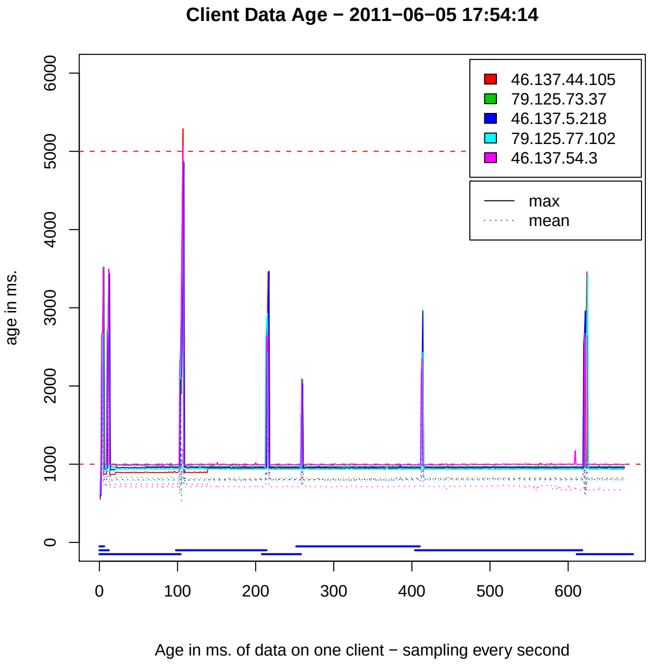
(Test run 15)



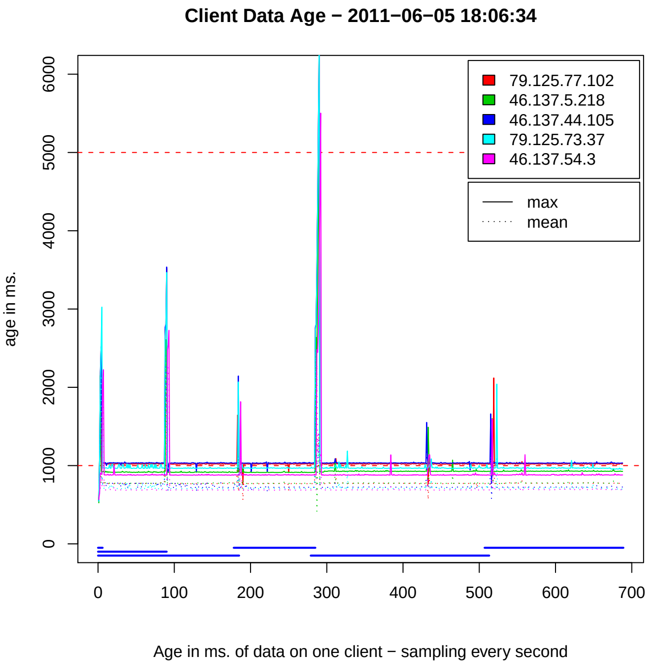
(Test run 16)



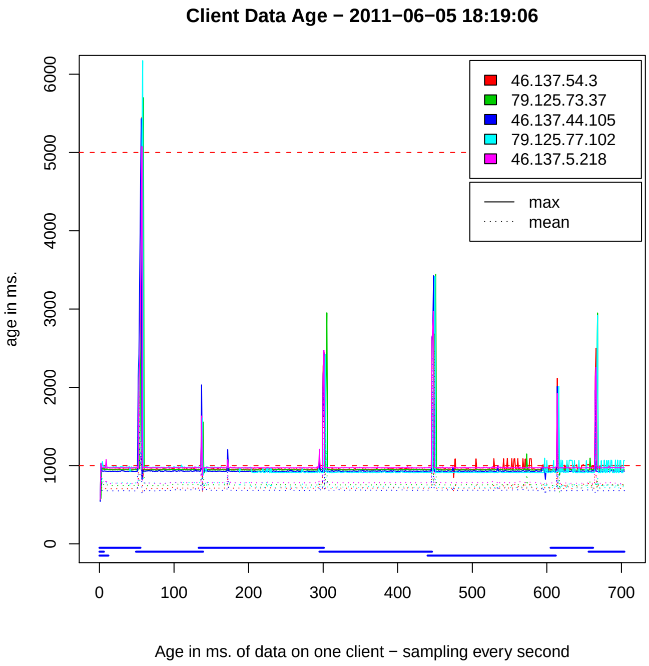
(Test run 17) Last test before crash.



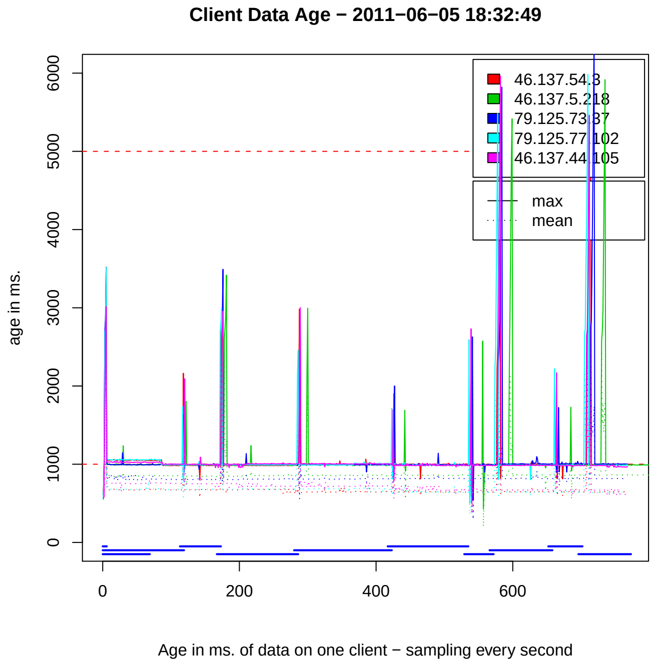
(Test run 18) First test after crash.



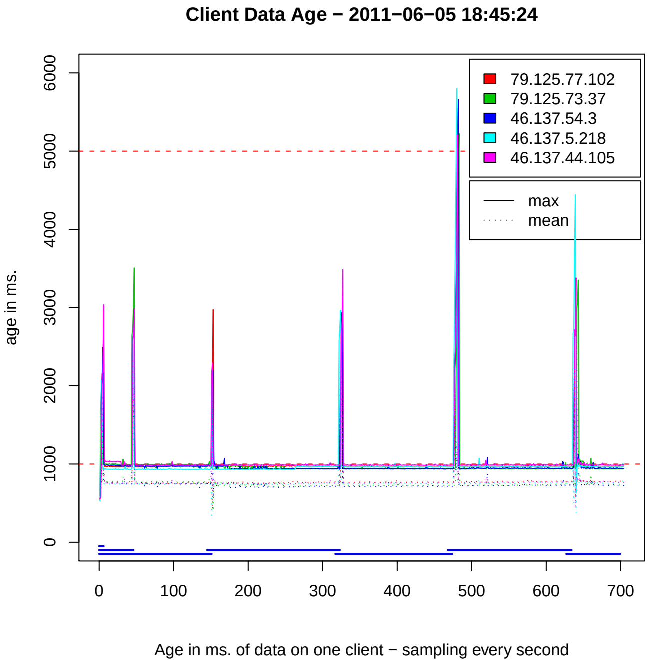
(Test run 19)



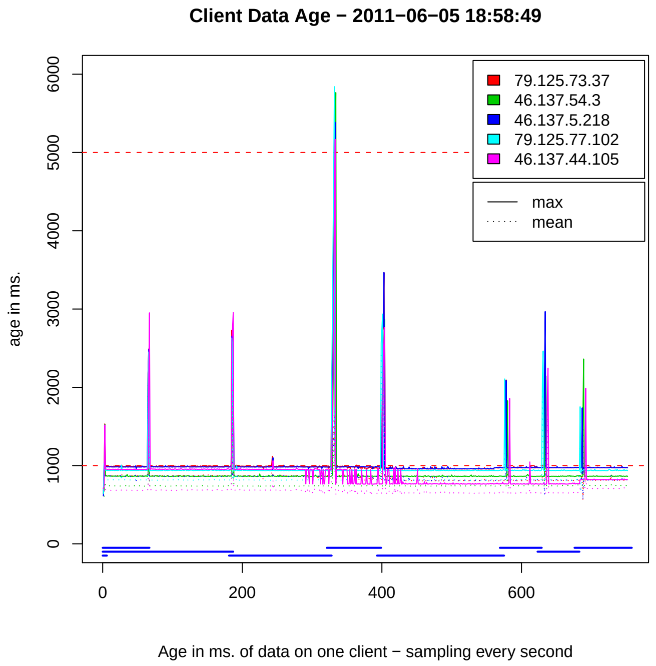
(Test run 20)



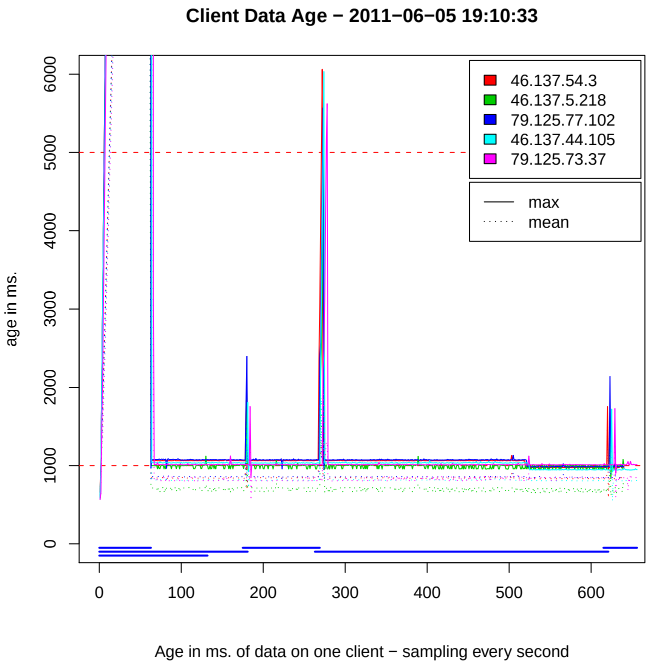
(Test run 21)



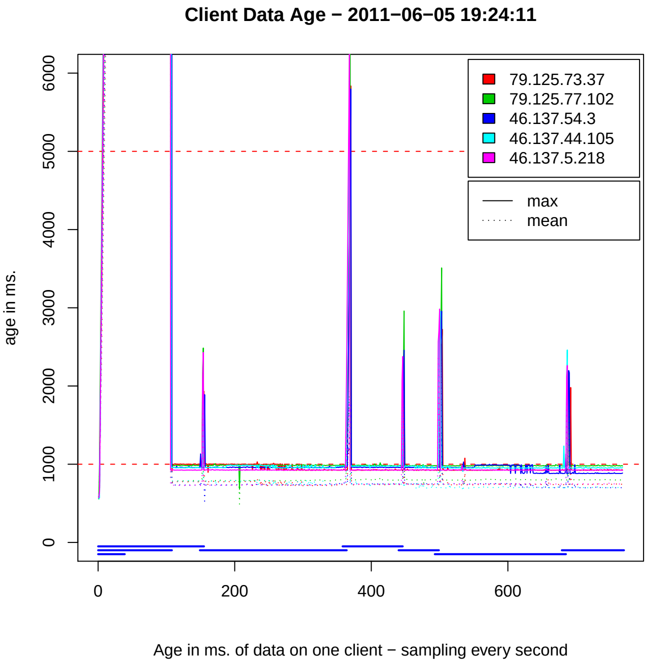
(Test run 22)



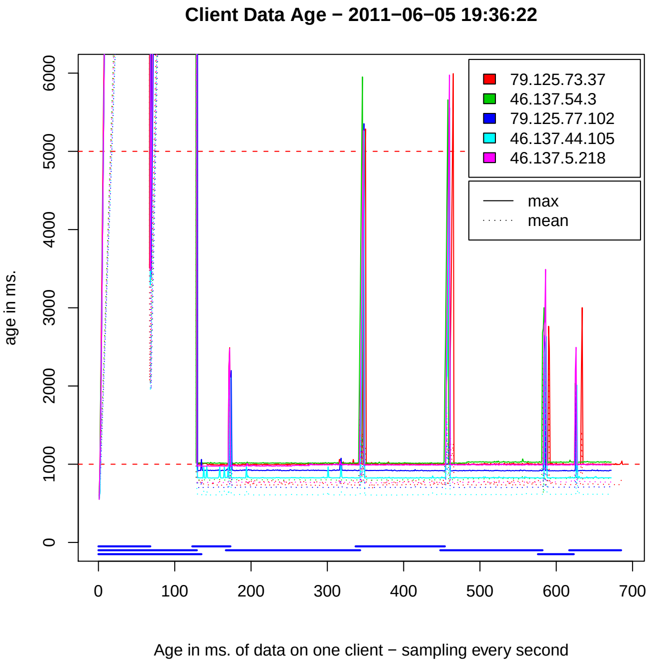
(Test run 23)



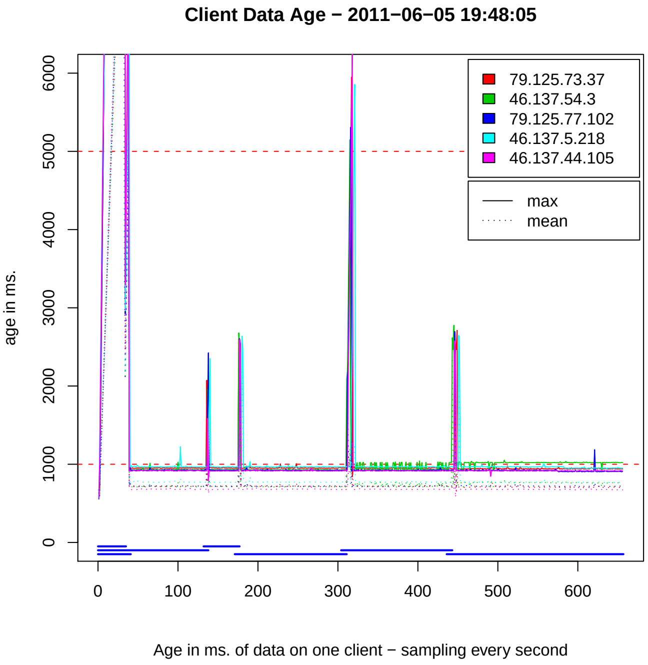
(Test run 24)



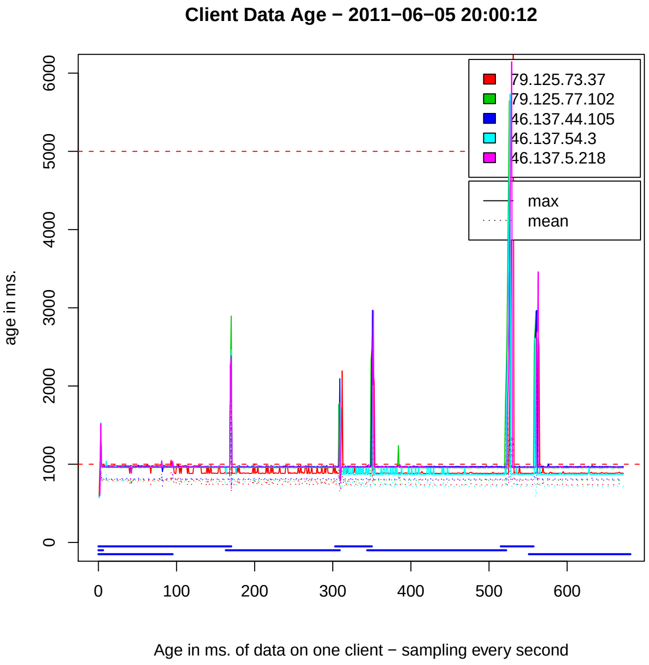
(Test run 25)



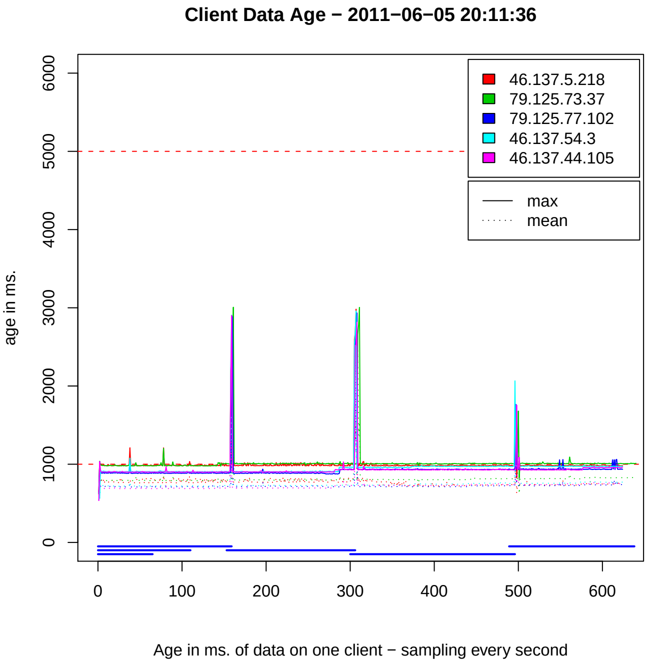
(Test run 26)



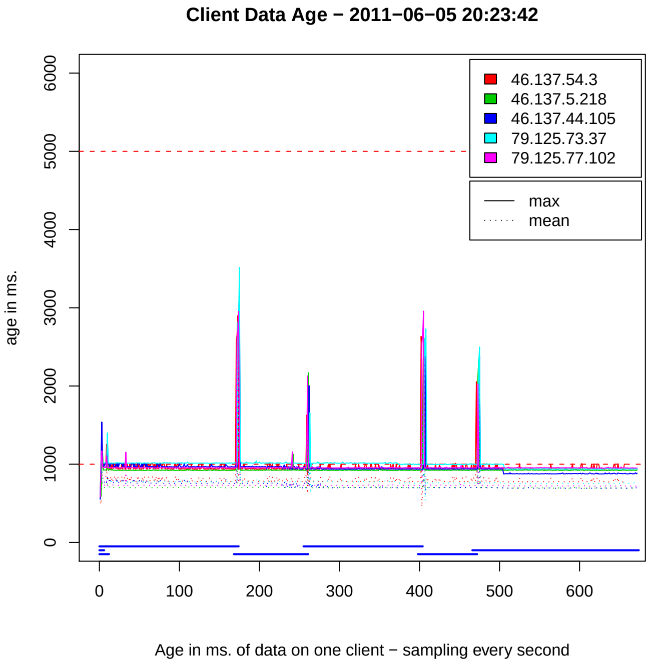
(Test run 27)



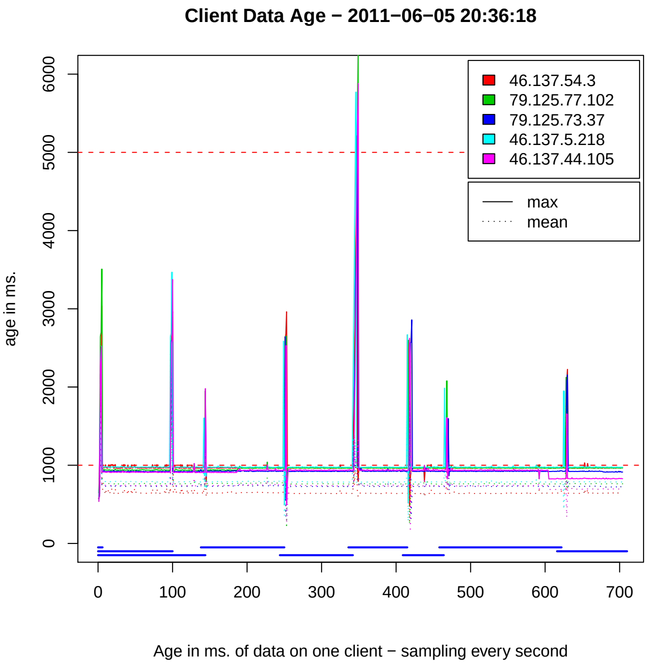
(Test run 28)



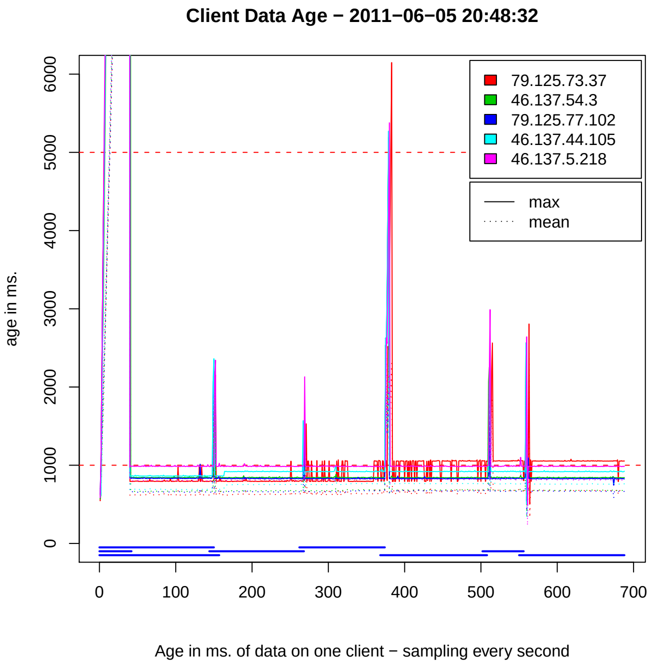
(Test run 29)



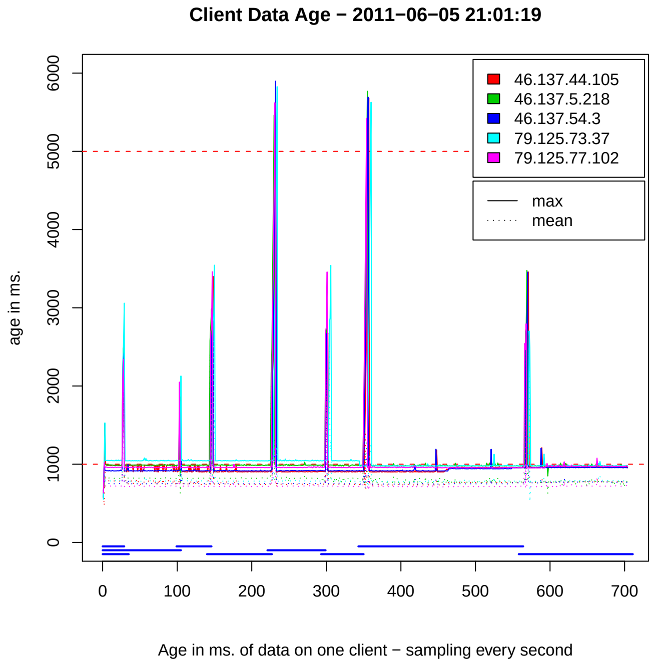
(Test run 30)



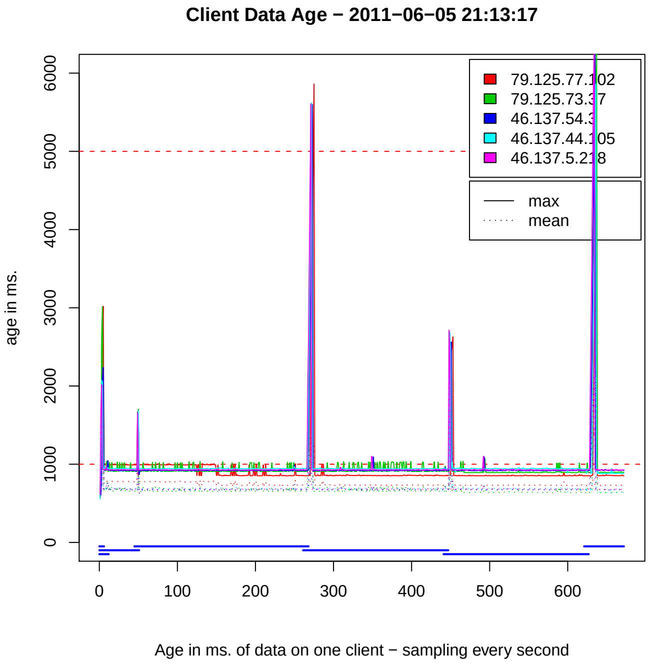
(Test run 31)



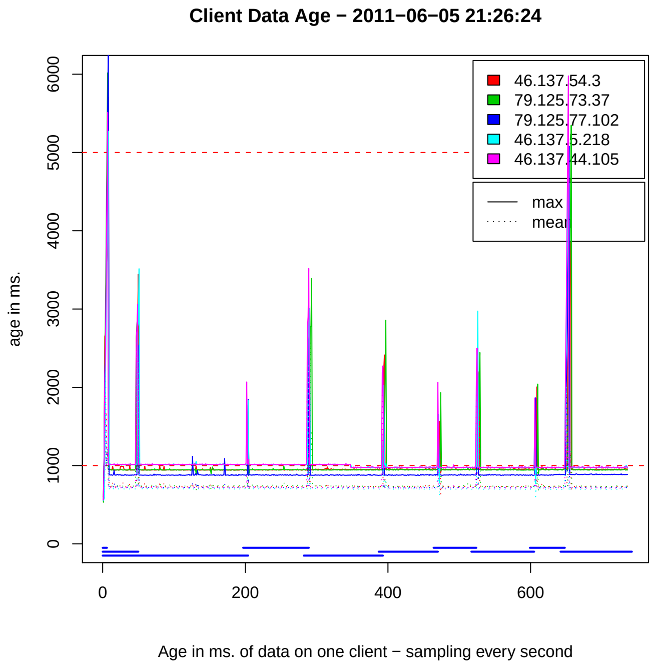
(Test run 32)



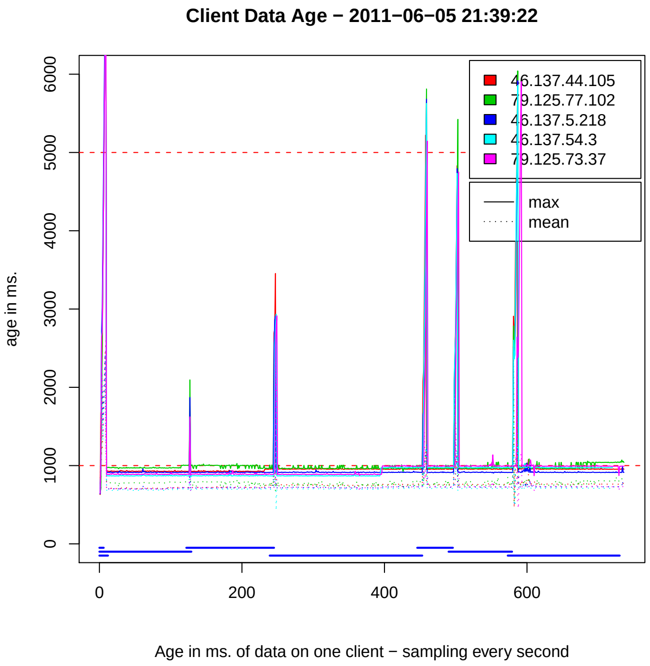
(Test run 33)



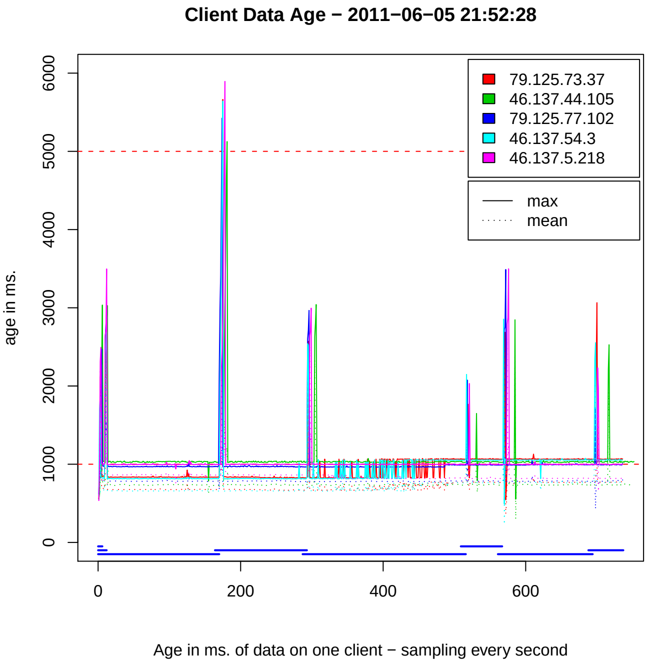
(Test run 34)



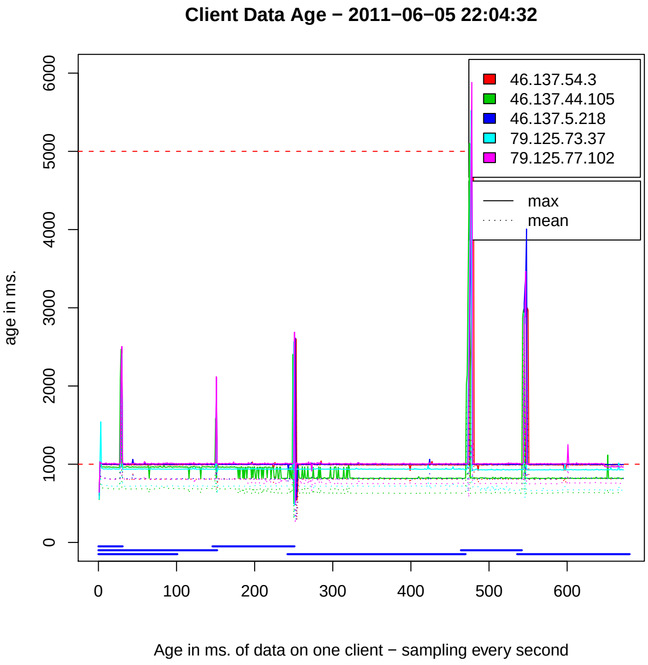
(Test run 35)



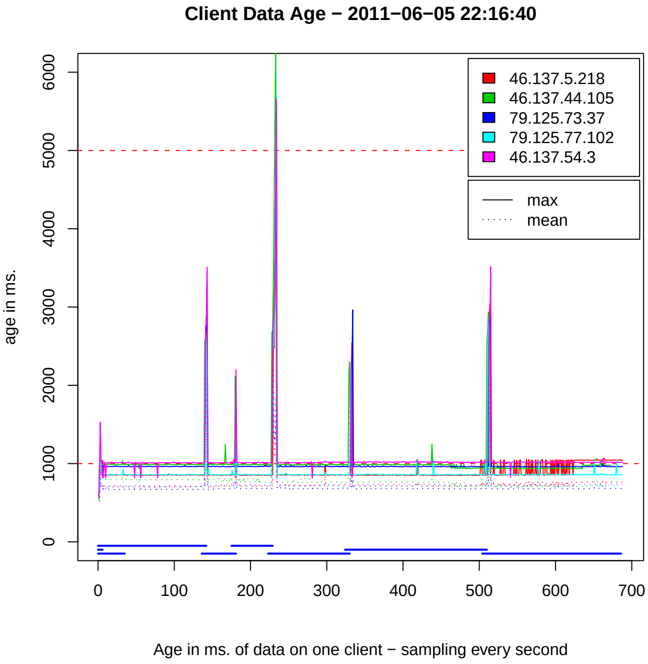
(Test run 36)



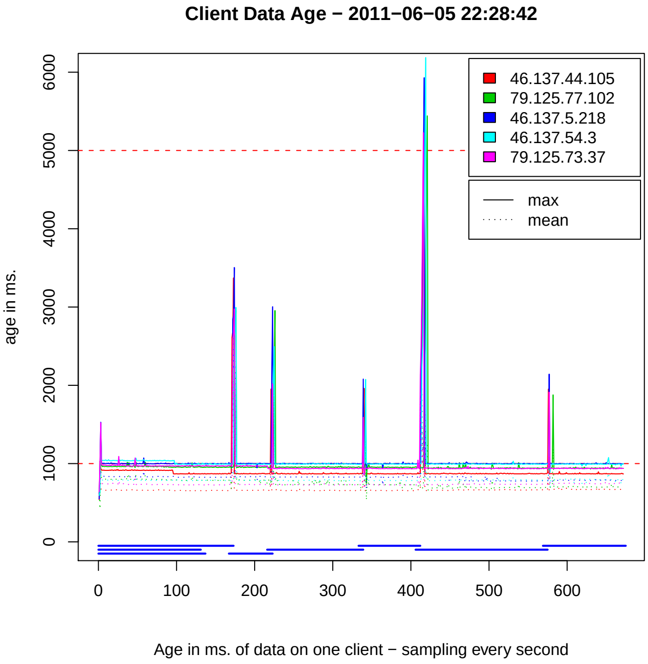
(Test run 37)



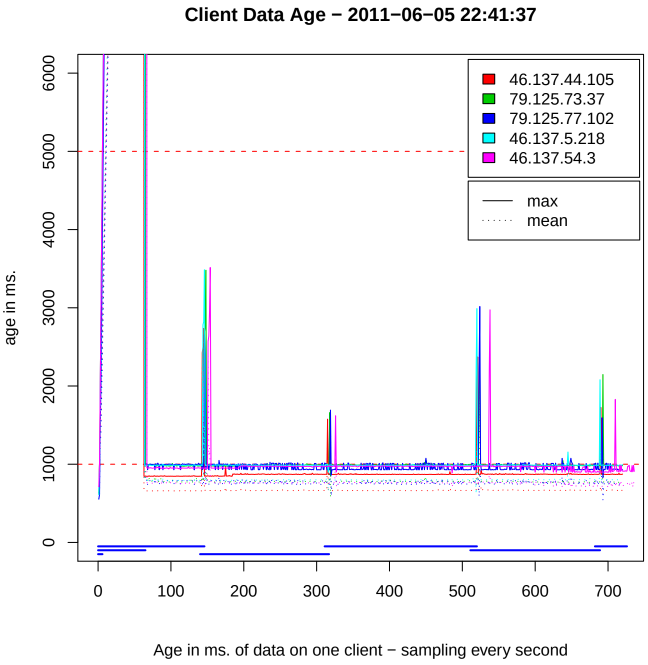
(Test run 38)



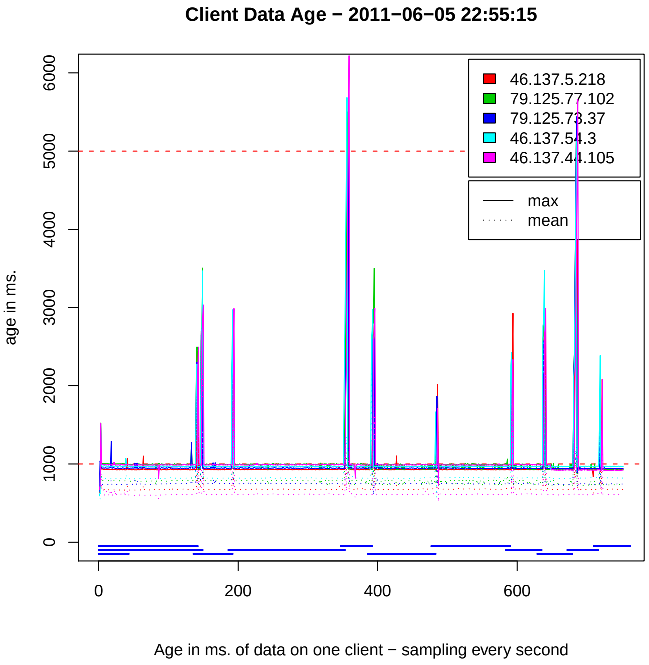
(Test run 39)



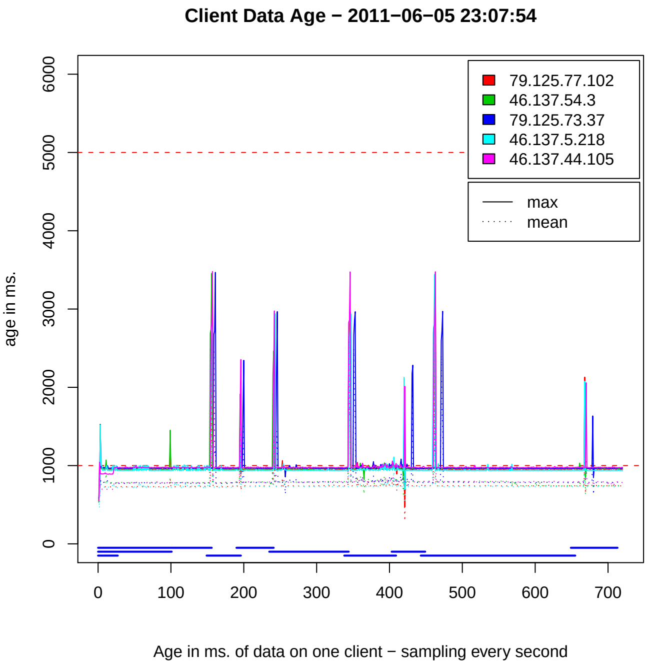
(Test run 40)



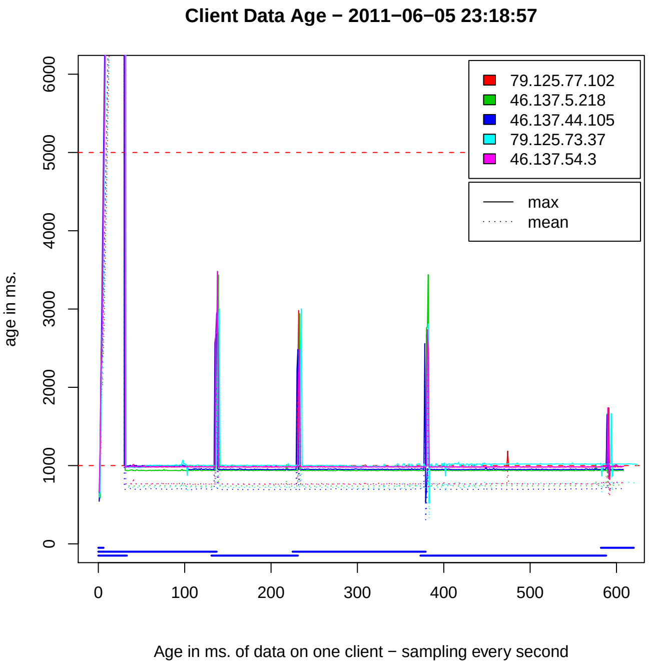
(Test run 41)



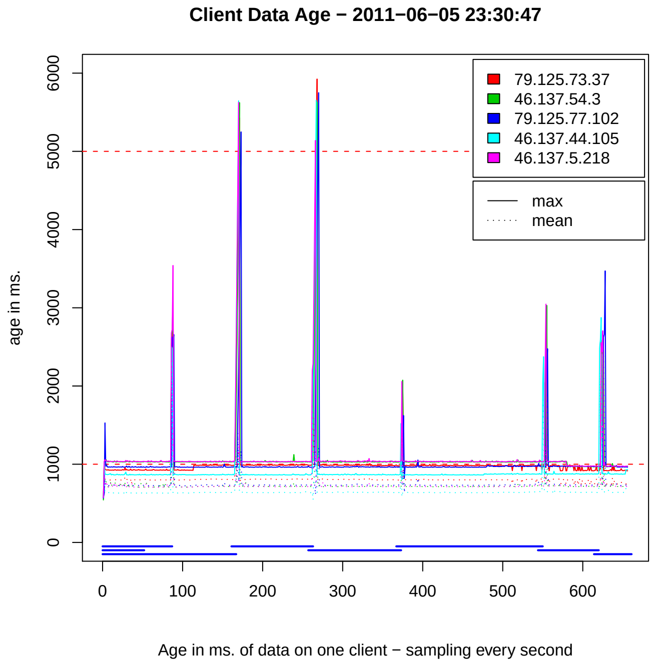
(Test run 42)



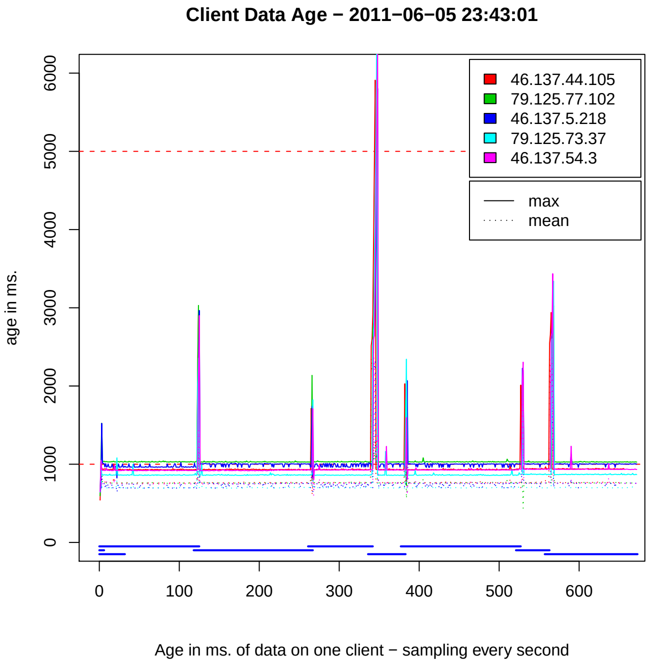
(Test run 43)



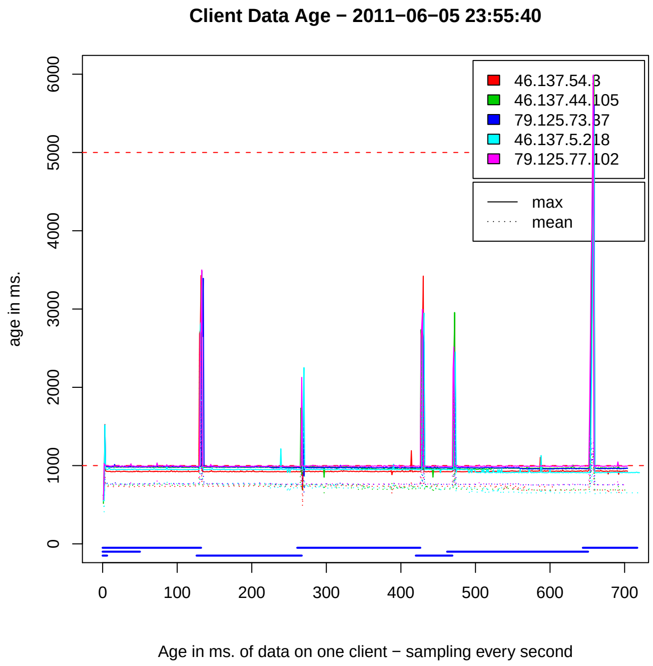
(Test run 44)



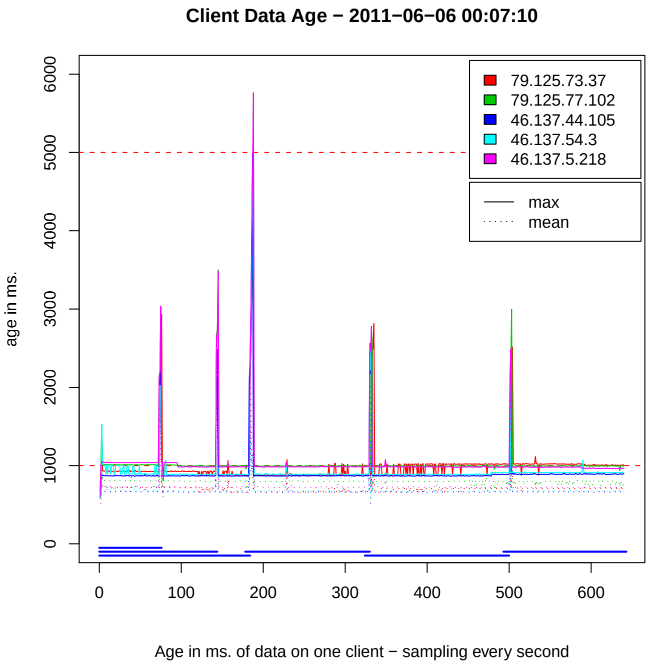
(Test run 45)



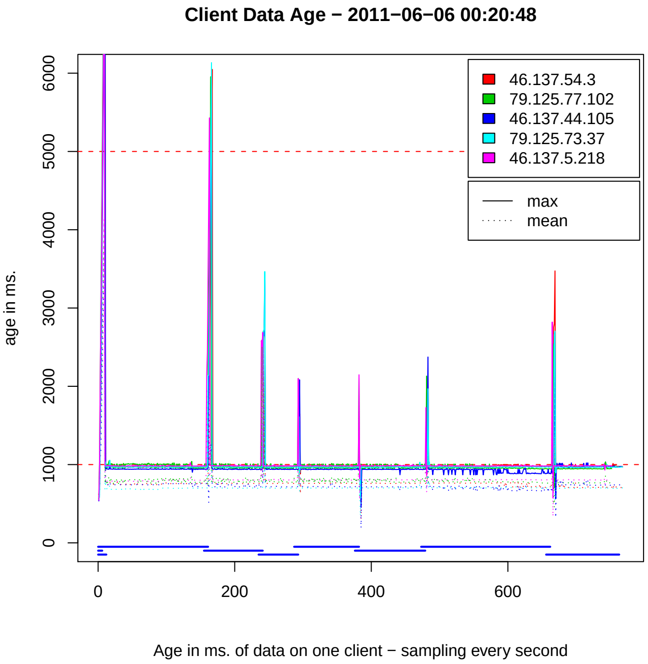
(Test run 46)



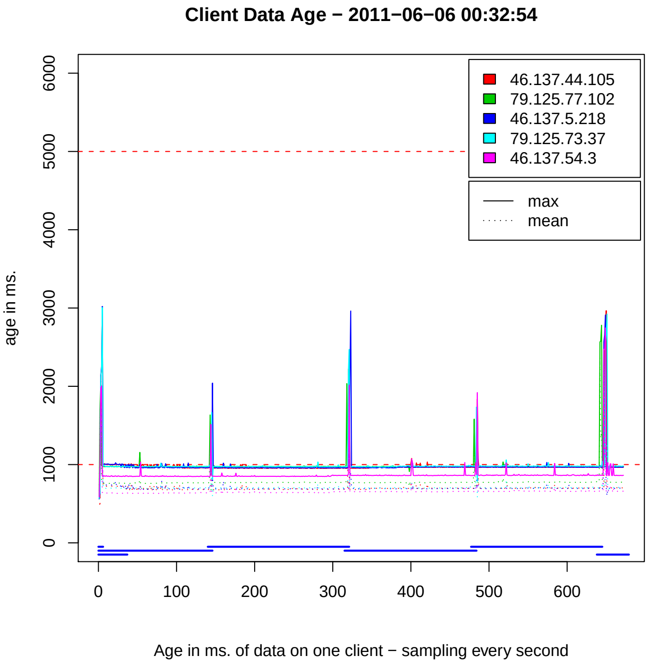
(Test run 47)



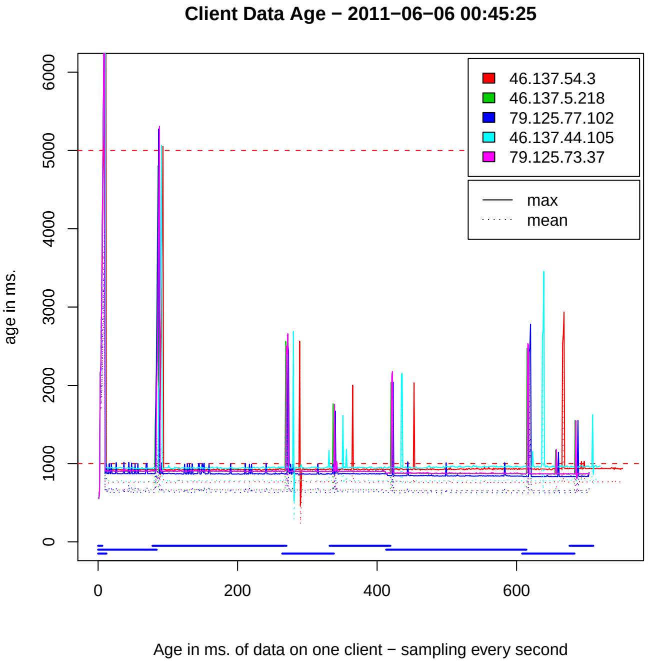
(Test run 48)



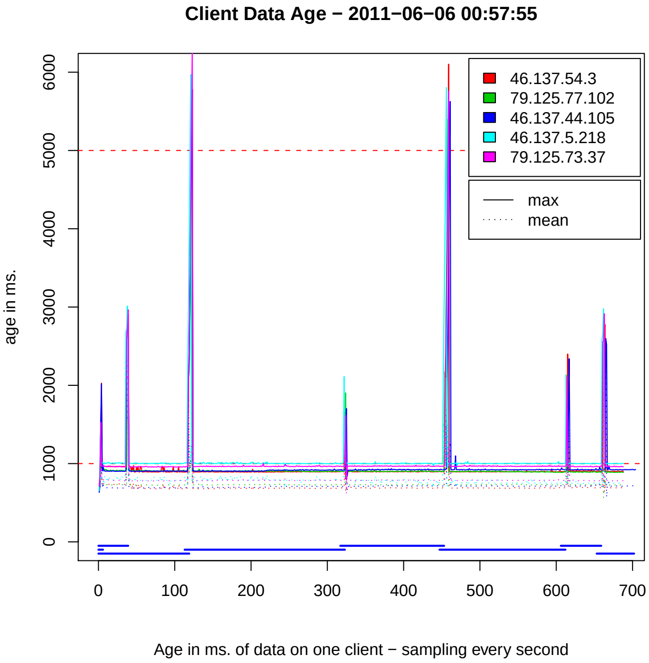
(Test run 49)



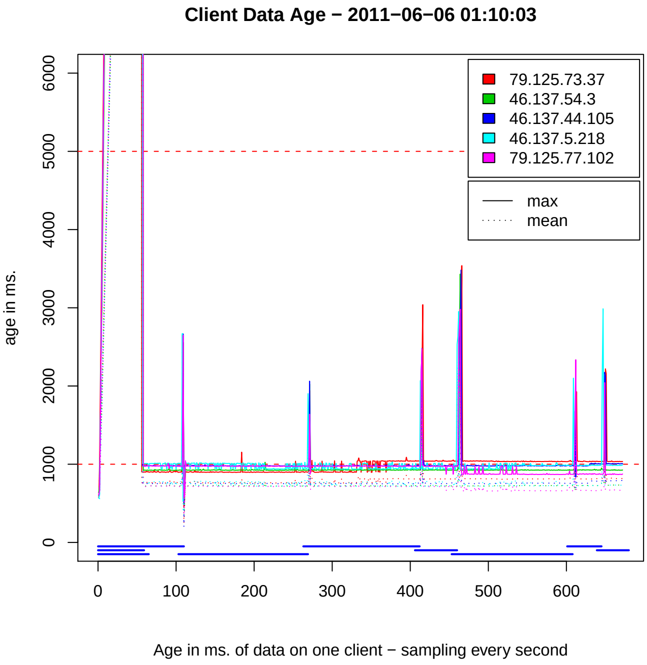
(Test run 50)



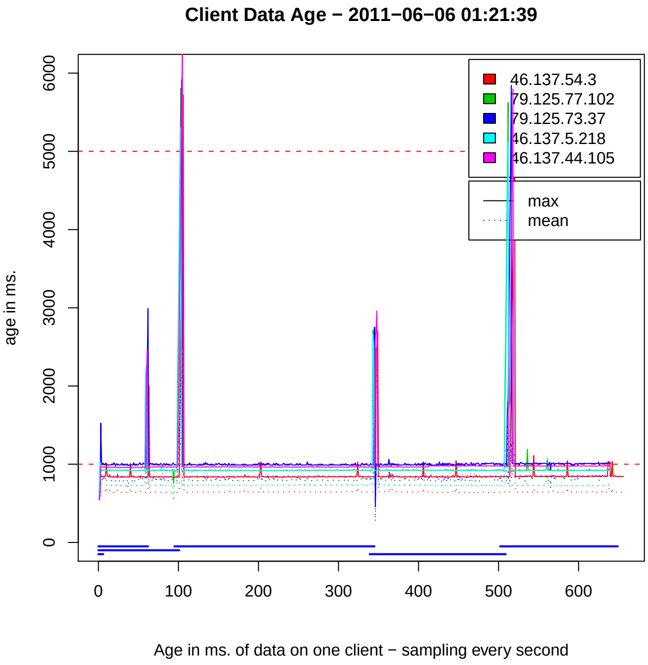
(Test run 51)



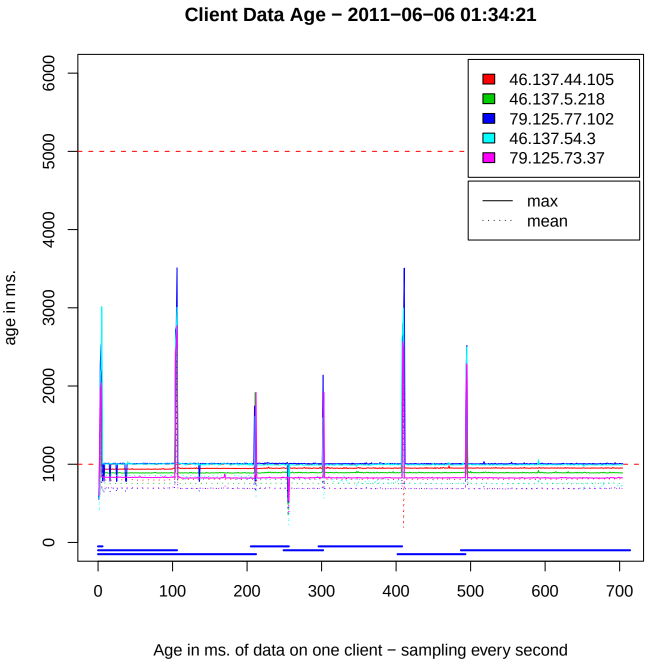
(Test run 52)



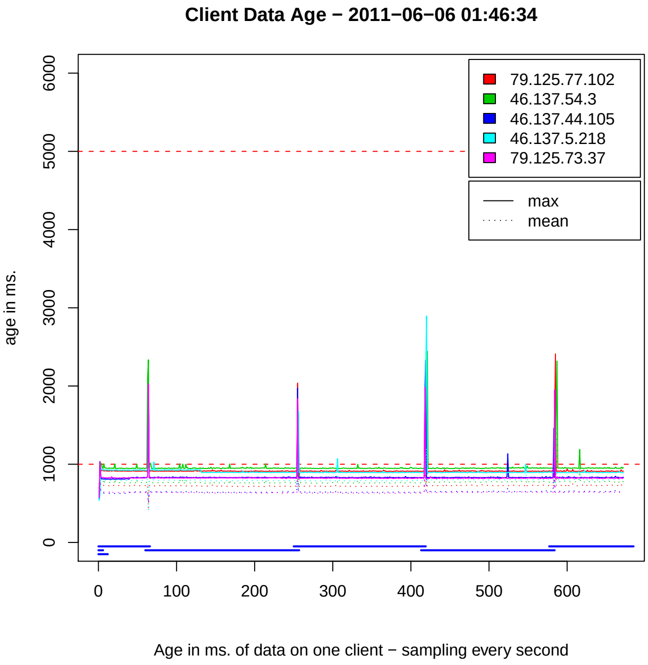
(Test run 53)



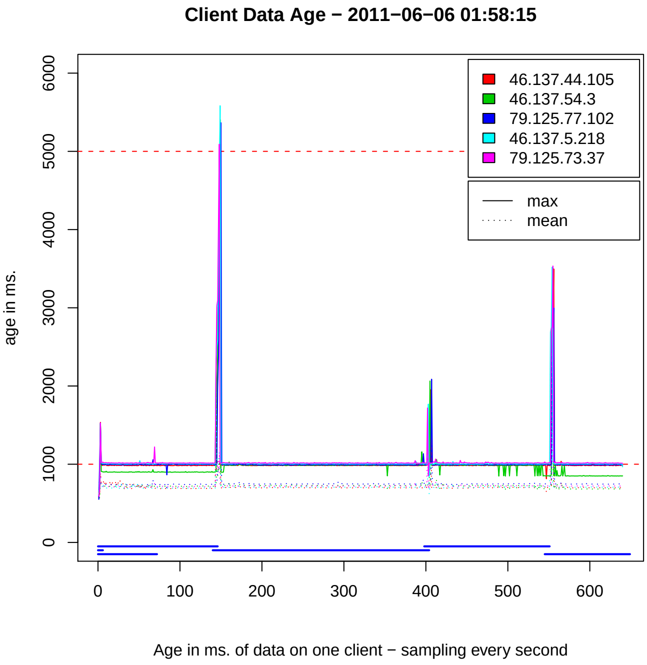
(Test run 54)



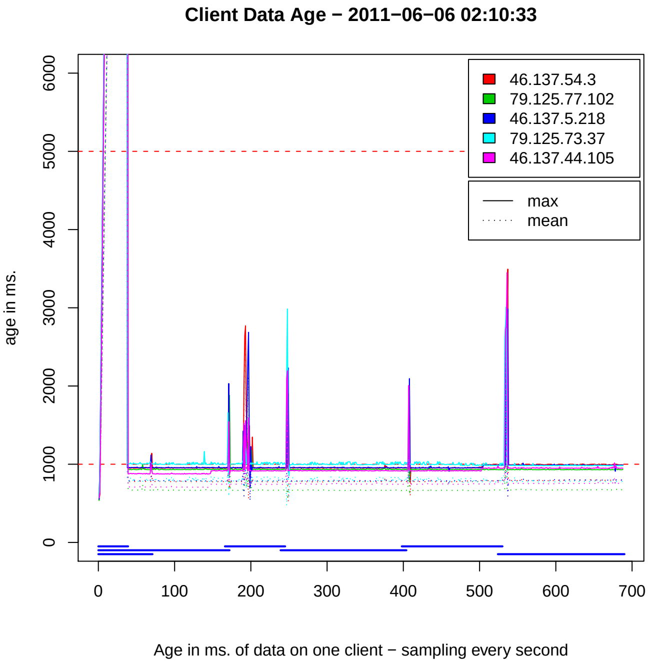
(Test run 55)



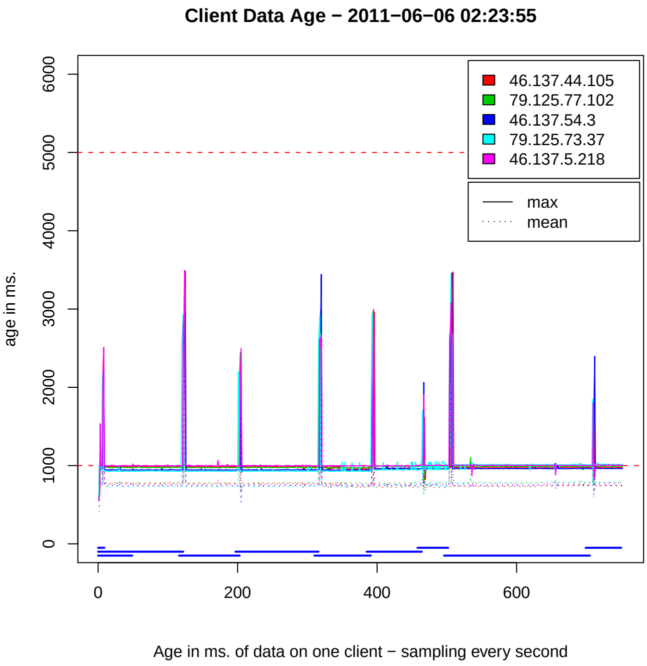
(Test run 56)



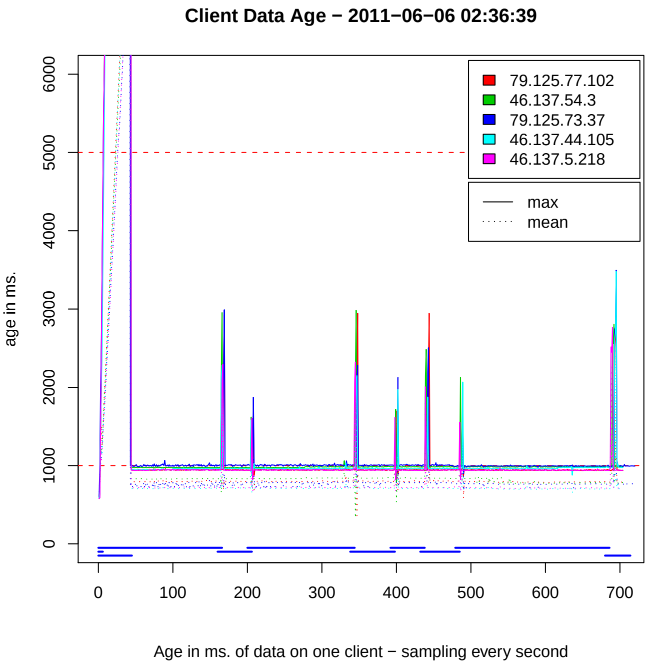
(Test run 57)



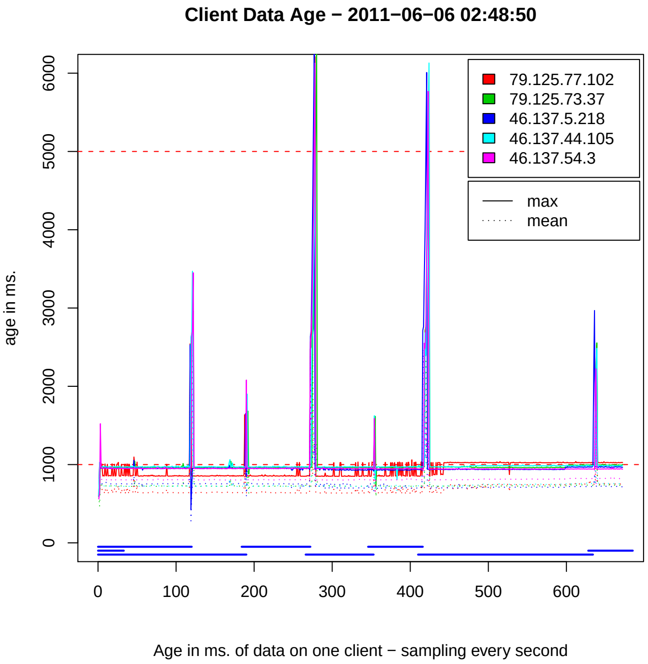
(Test run 58)



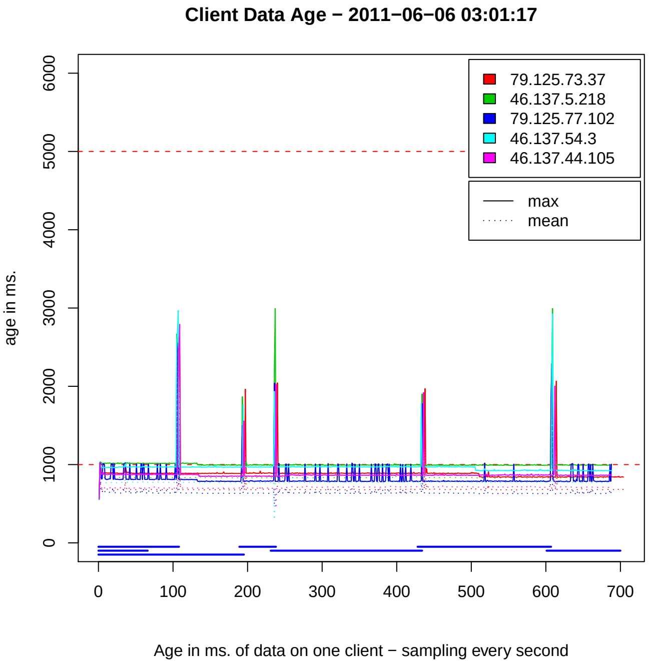
(Test run 59)



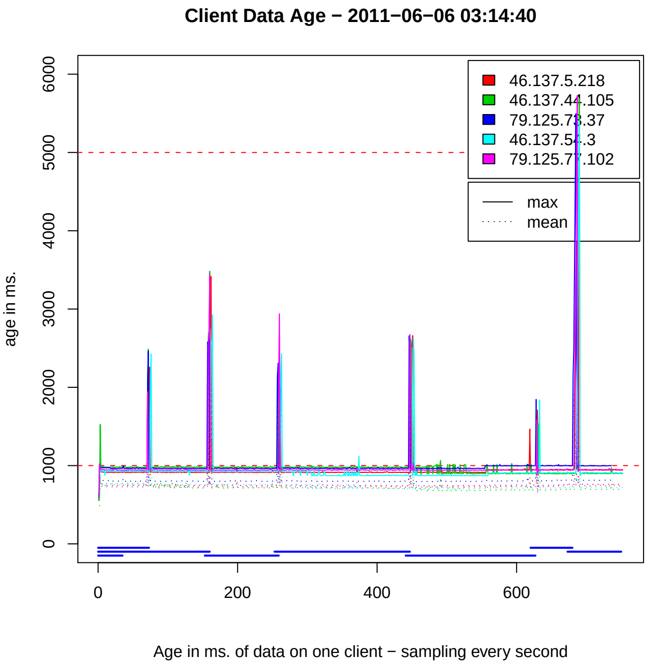
(Test run 60)



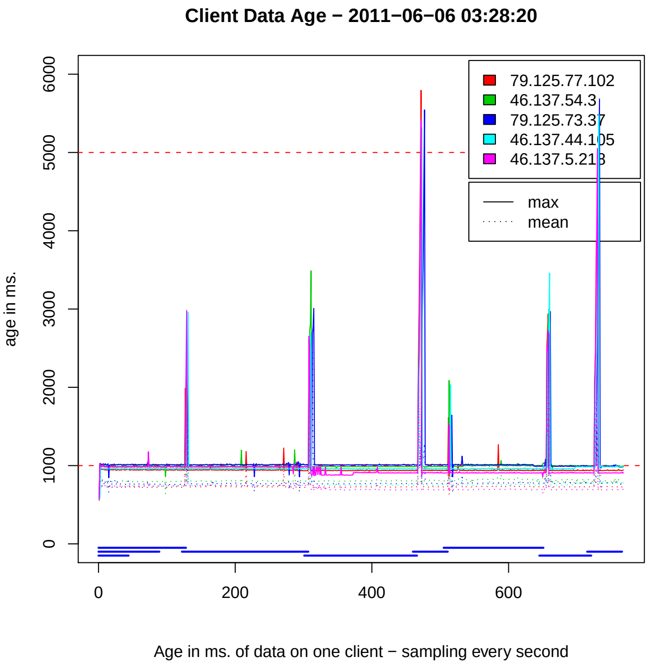
(Test run 61)



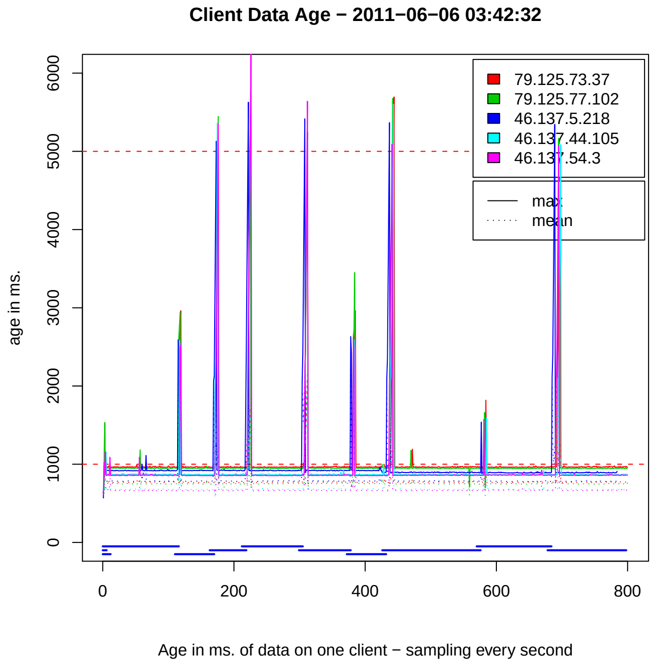
(Test run 62)



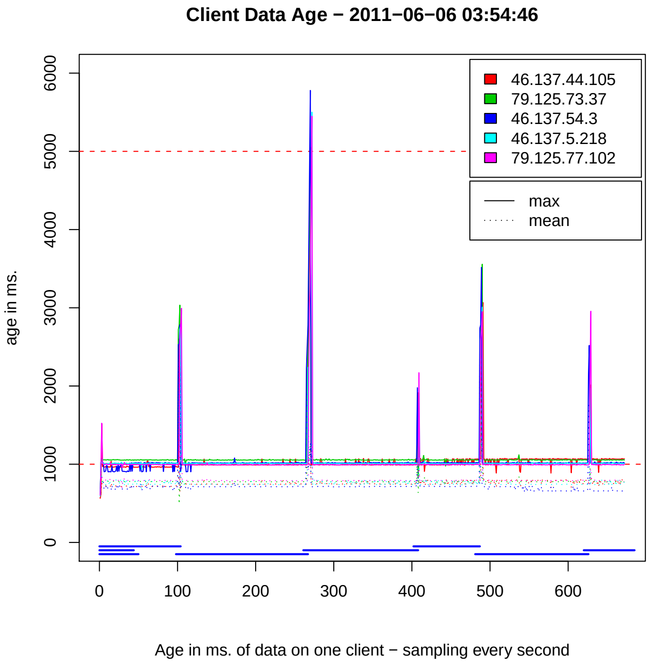
(Test run 63)



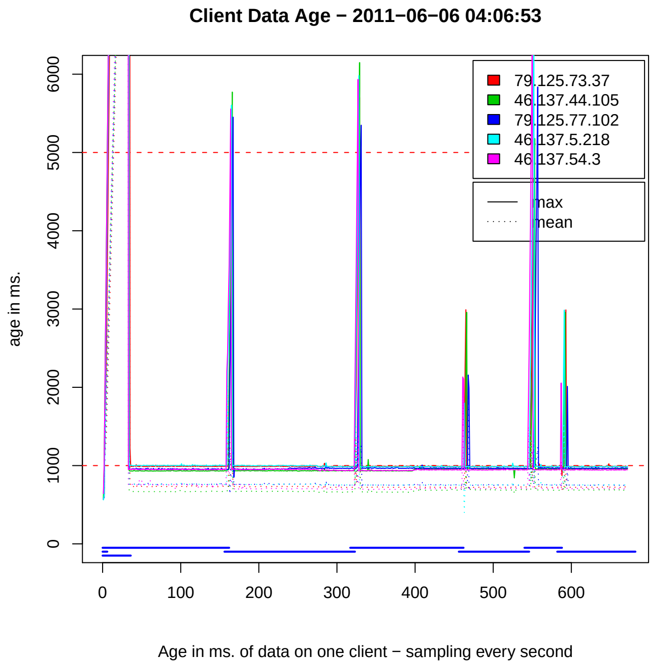
(Test run 64)



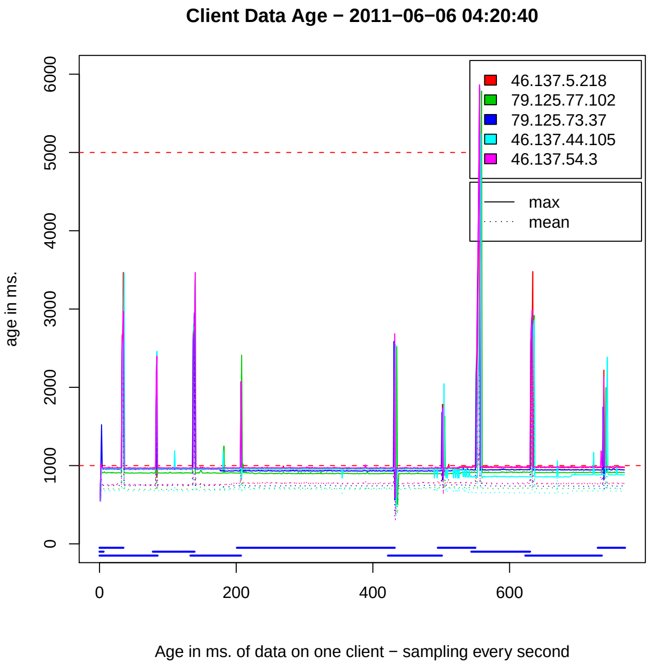
(Test run 65)



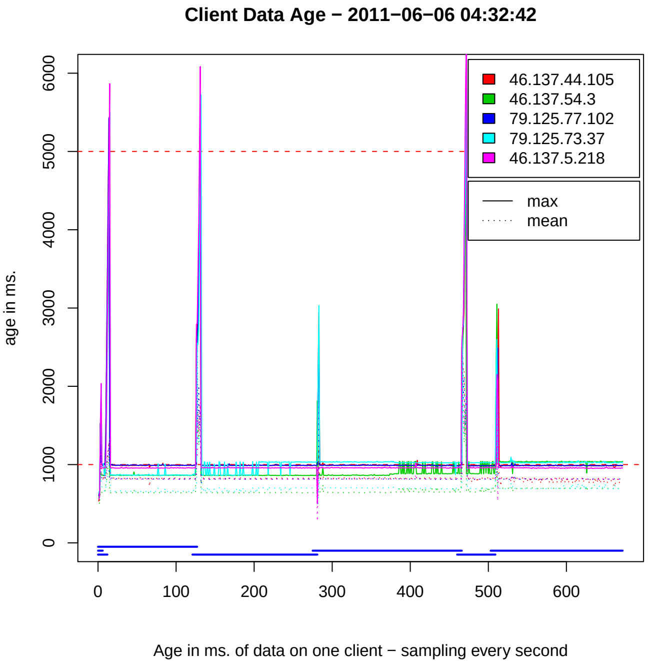
(Test run 66)



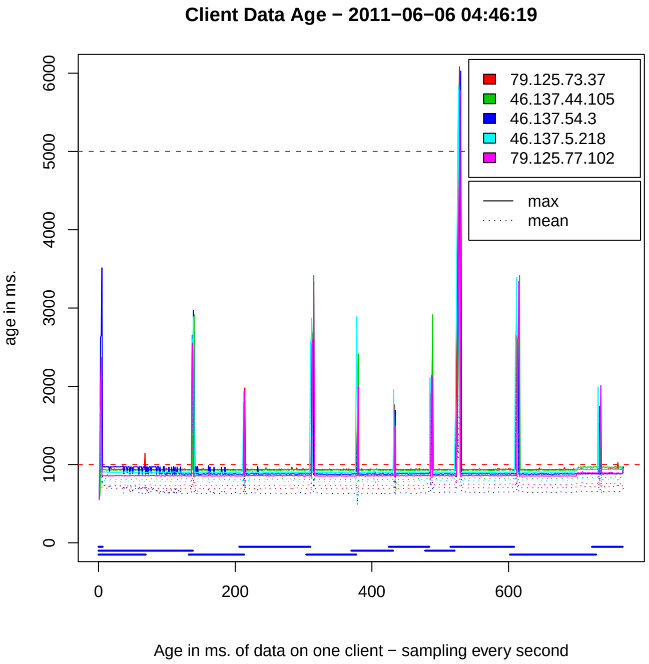
(Test run 67)



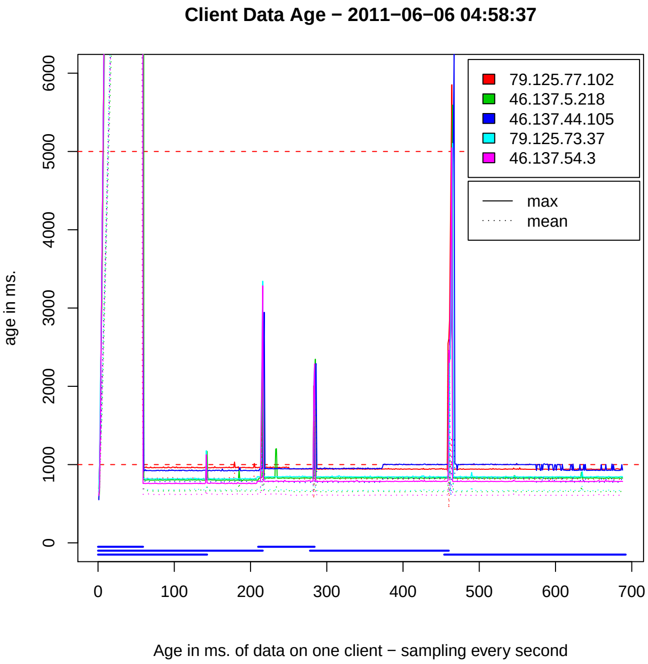
(Test run 68)



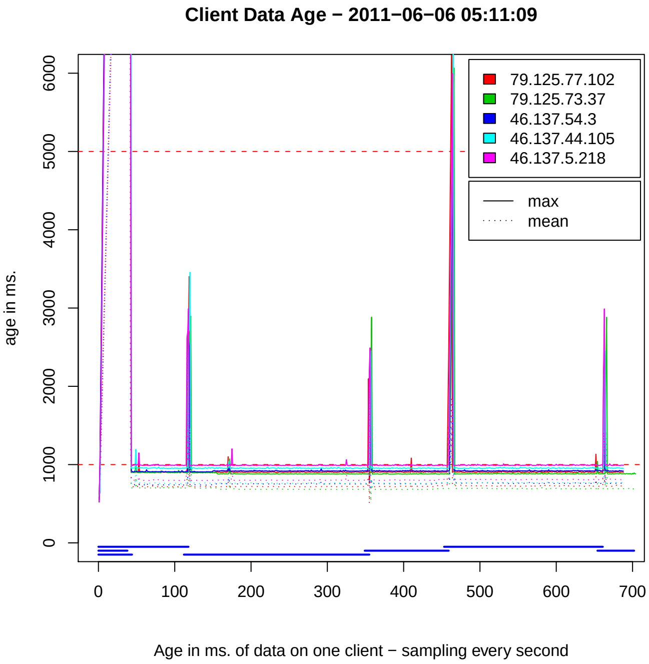
(Test run 69)



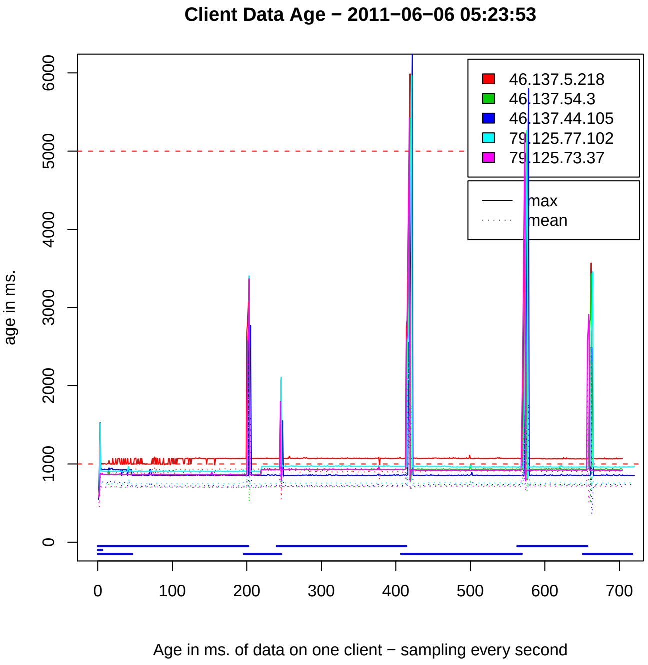
(Test run 70)



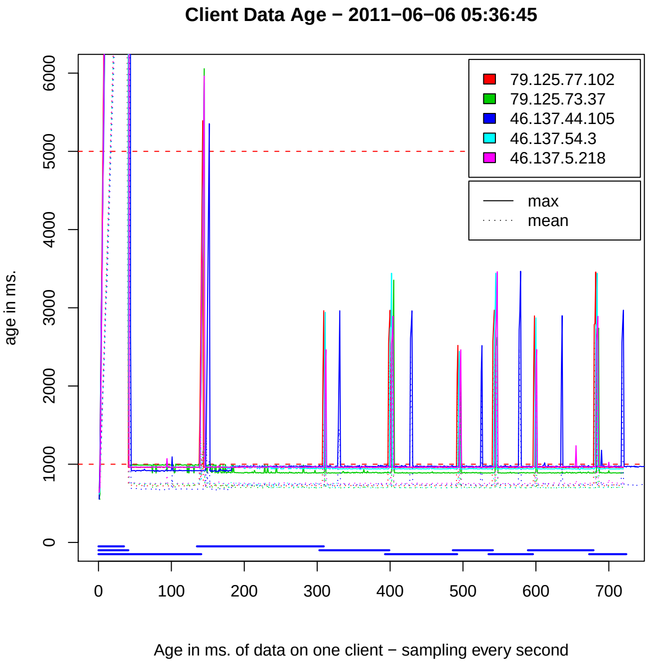
(Test run 71)



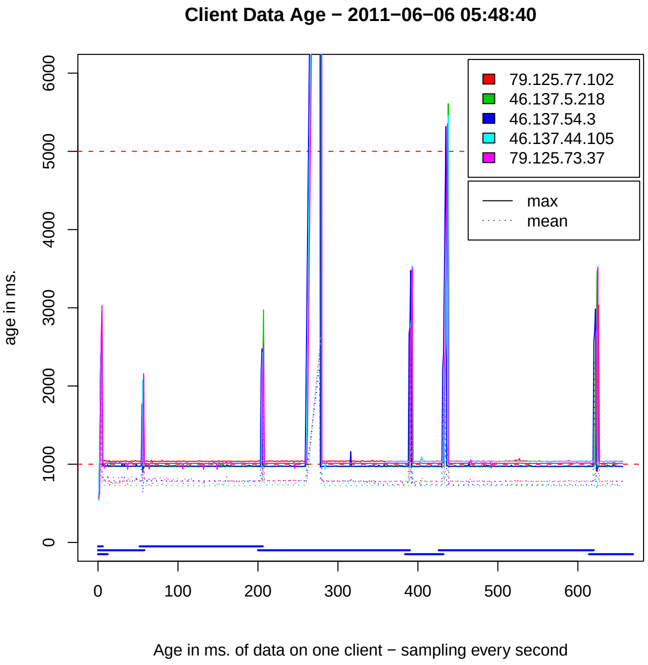
(Test run 72)



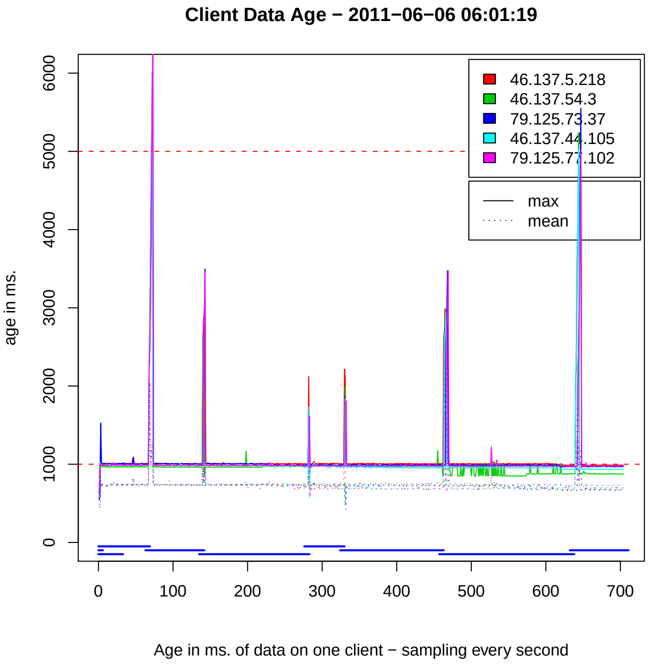
(Test run 73)



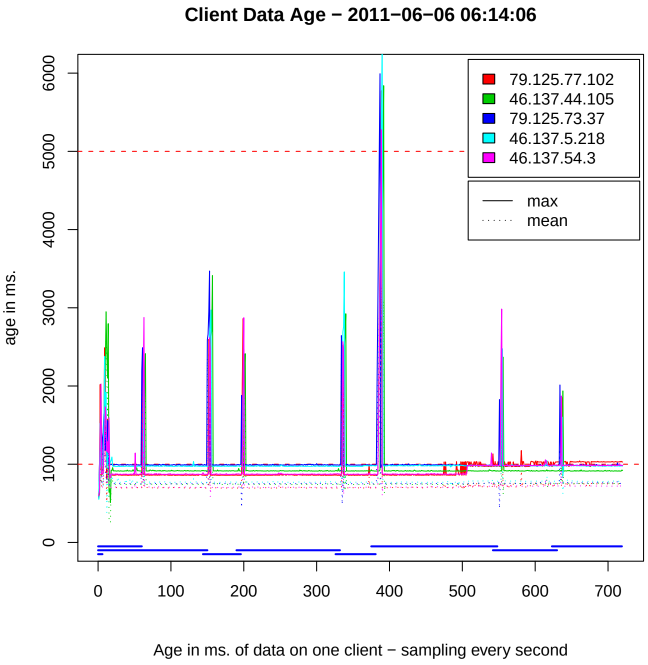
(Test run 74)



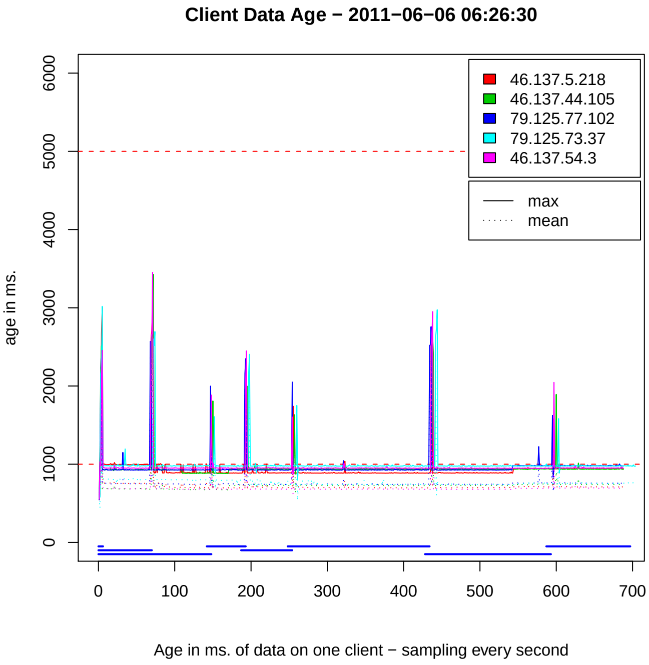
(Test run 75)



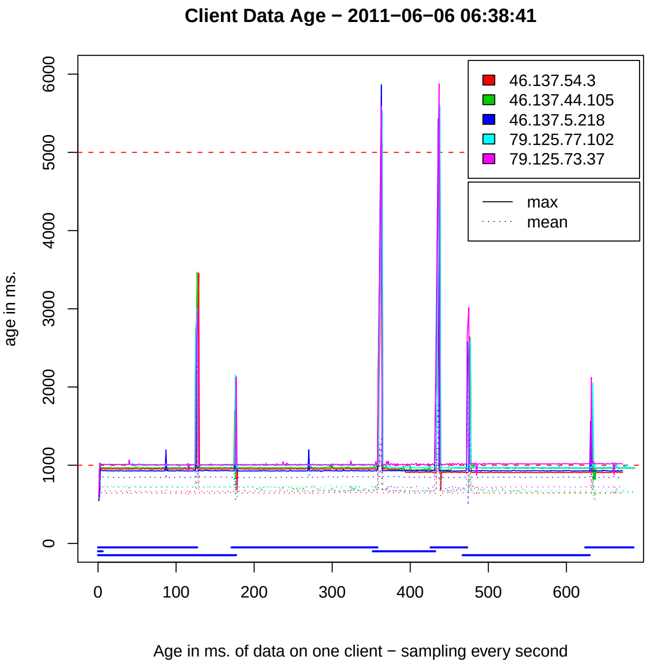
(Test run 76)



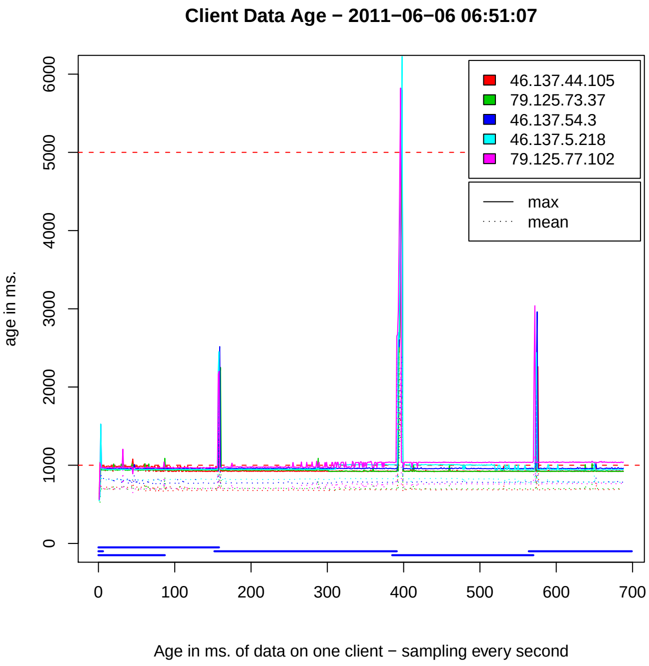
(Test run 77)



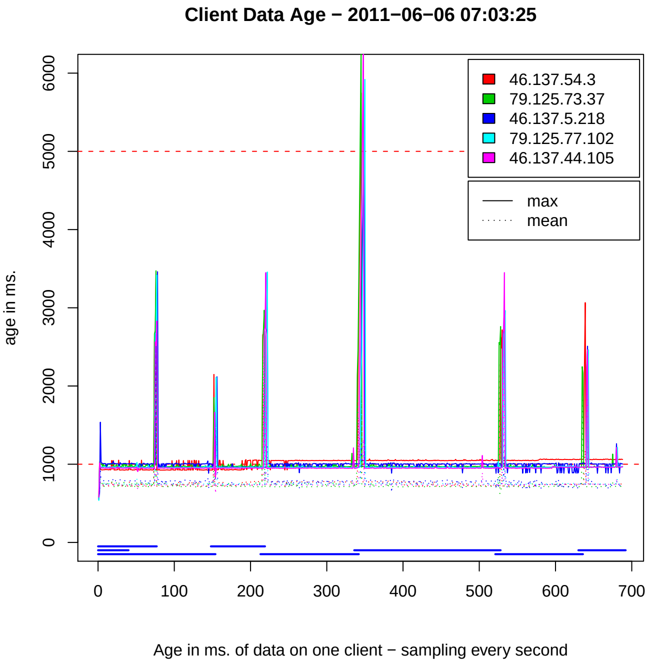
(Test run 78)



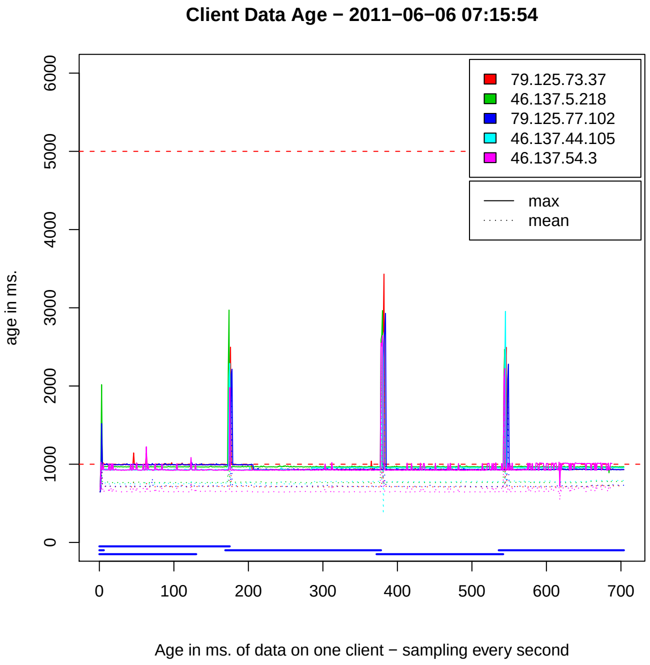
(Test run 79)



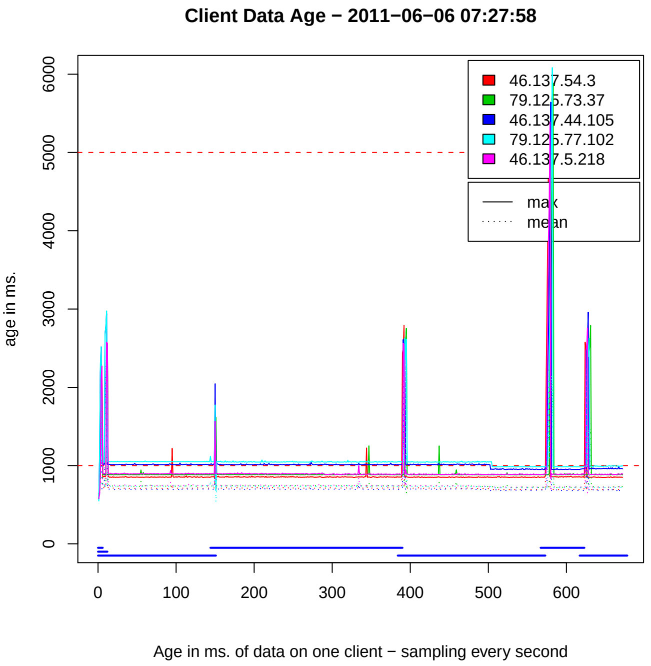
(Test run 80)



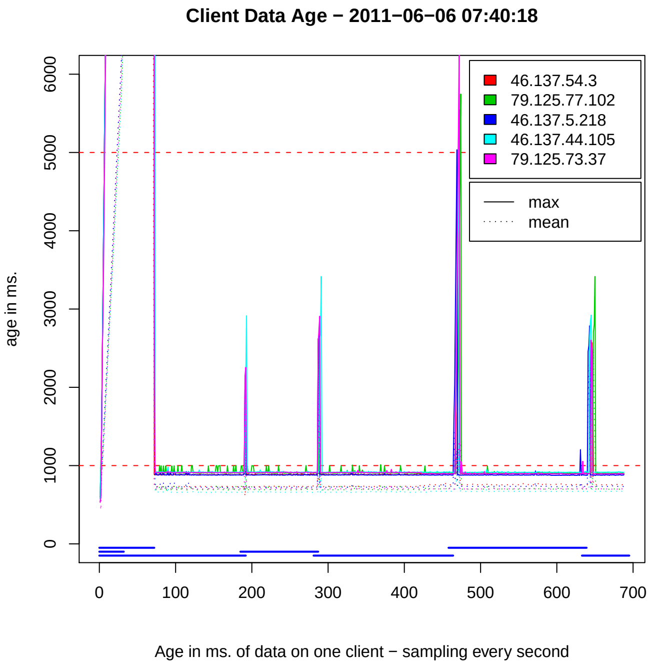
(Test run 81)



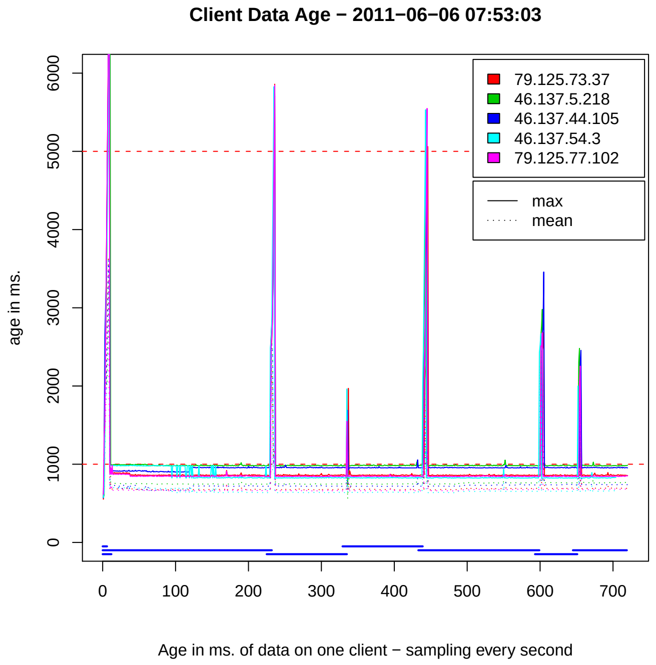
(Test run 82)



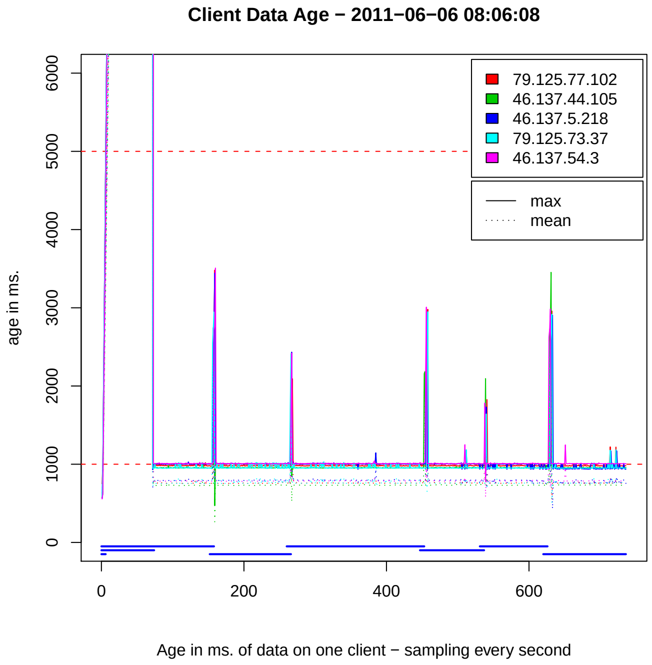
(Test run 83)



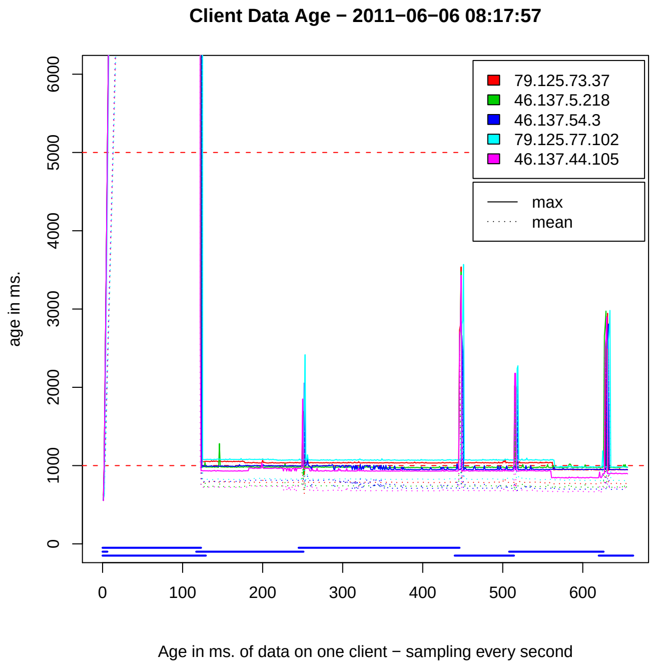
(Test run 84)



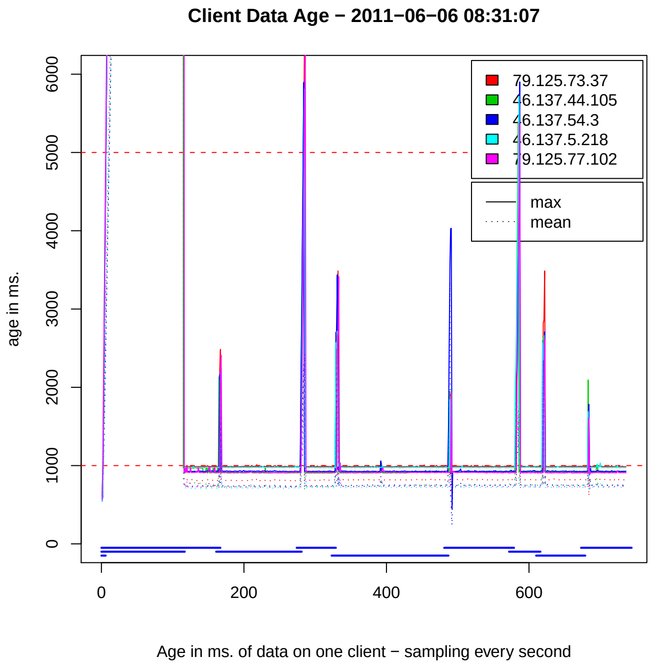
(Test run 85)



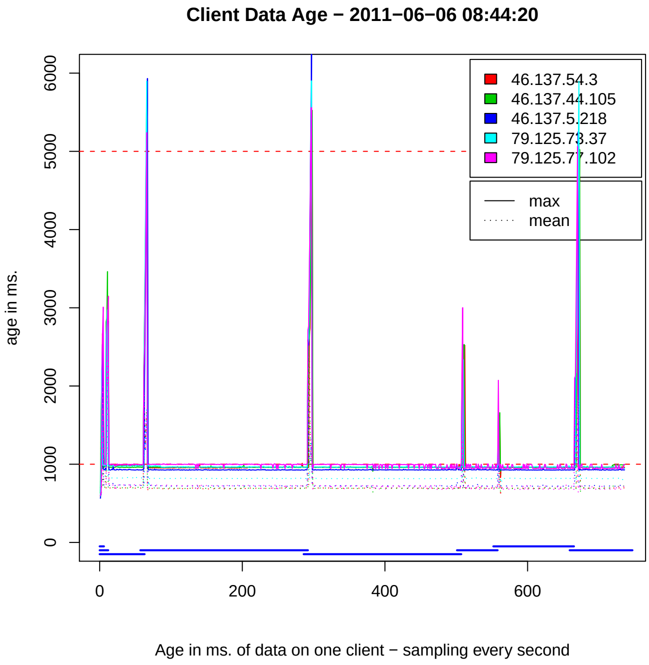
(Test run 86)



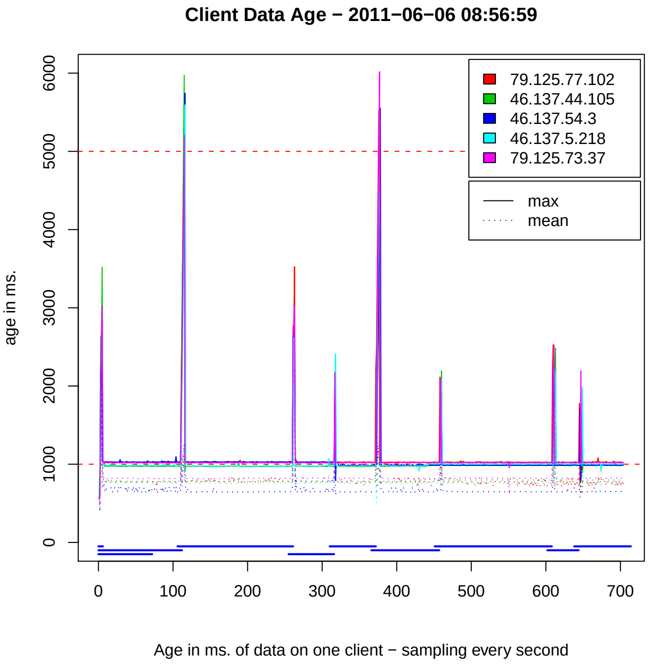
(Test run 87)



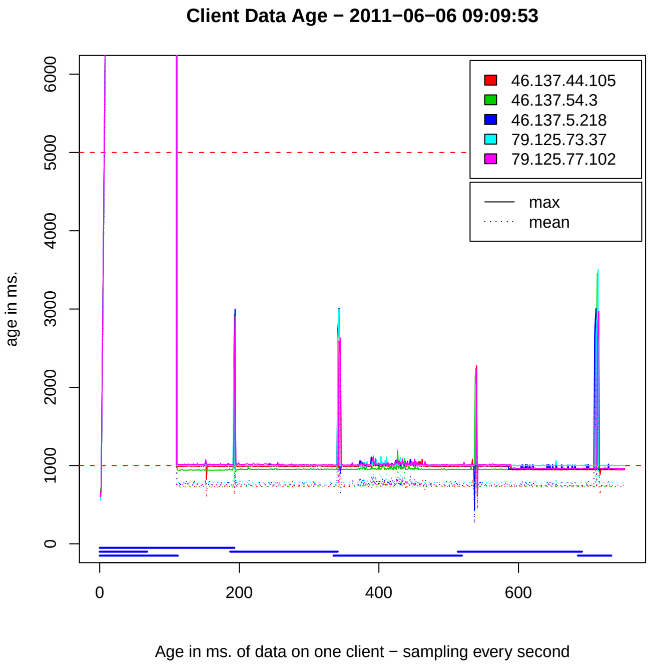
(Test run 88)



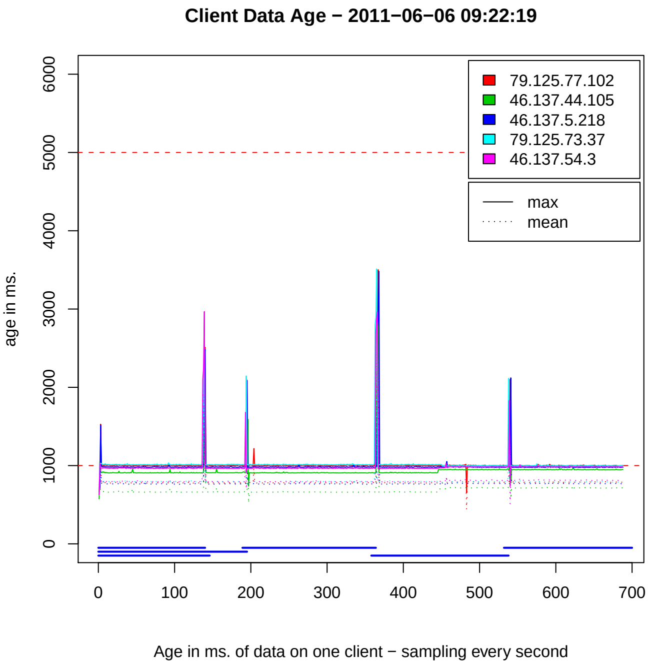
(Test run 89)



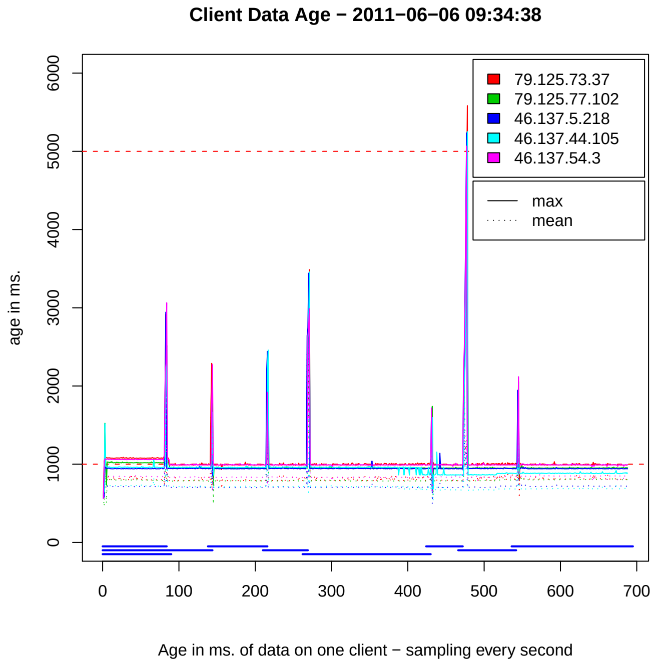
(Test run 90)



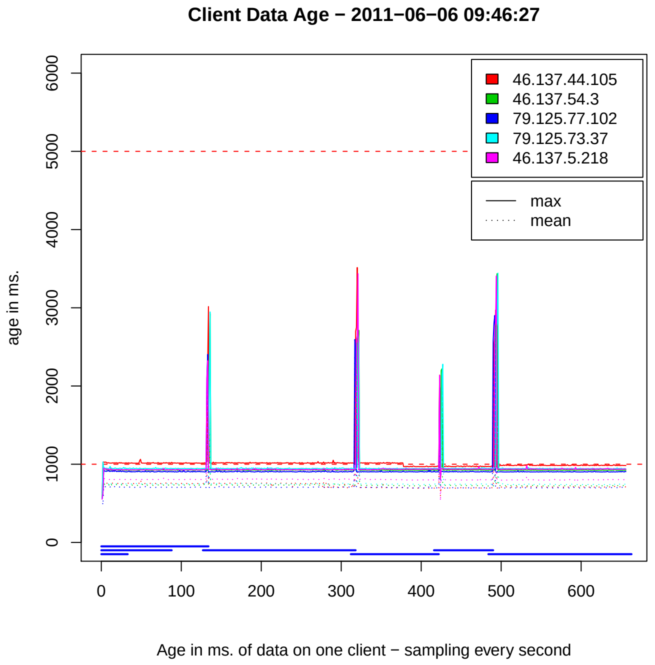
(Test run 91)



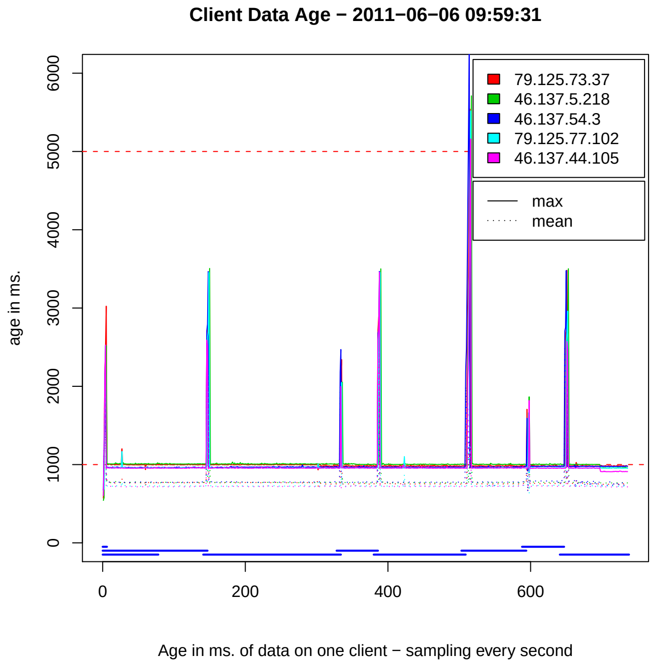
(Test run 92)



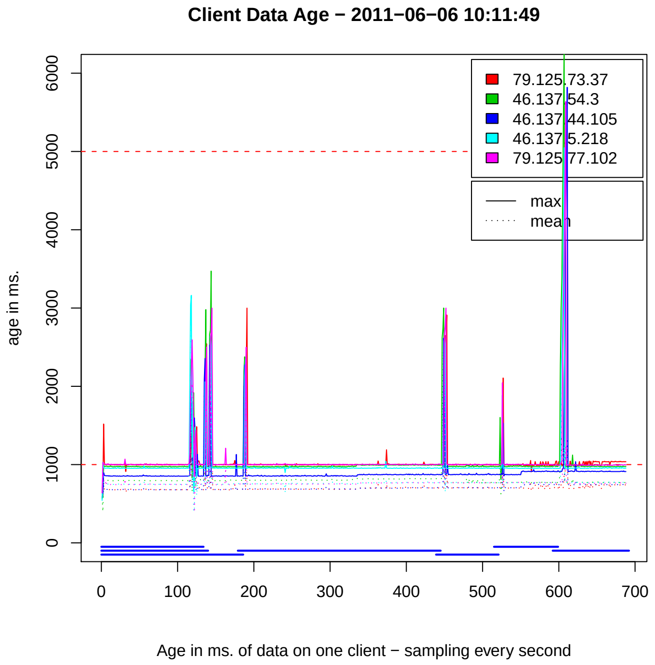
(Test run 93)



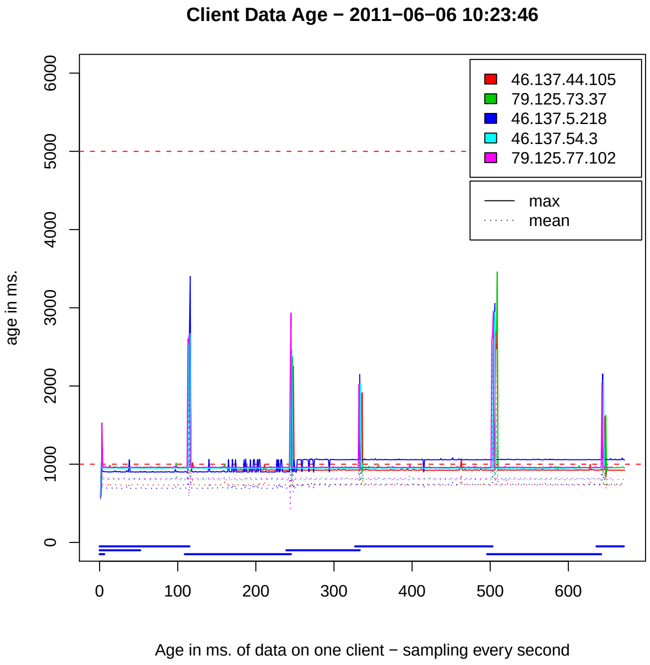
(Test run 94)



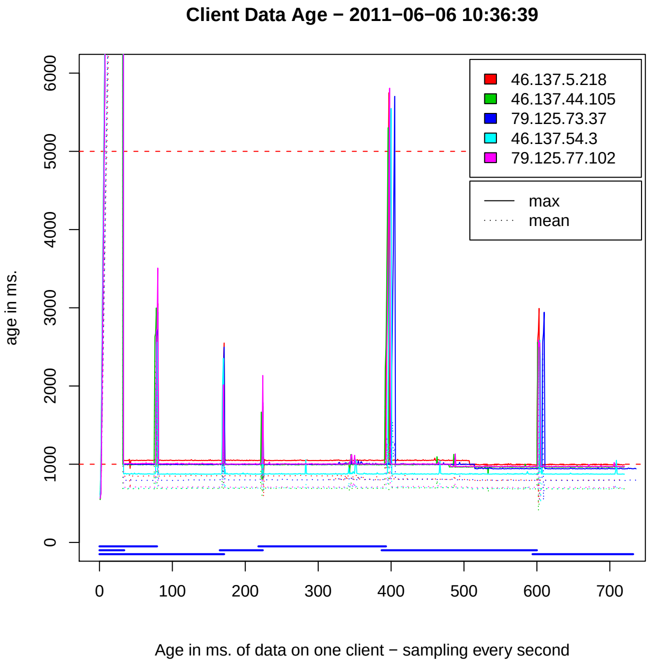
(Test run 95)



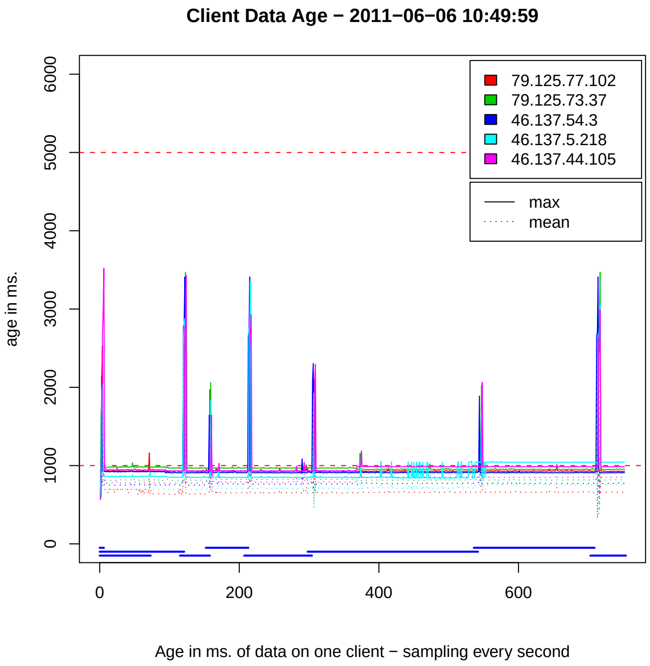
(Test run 96)



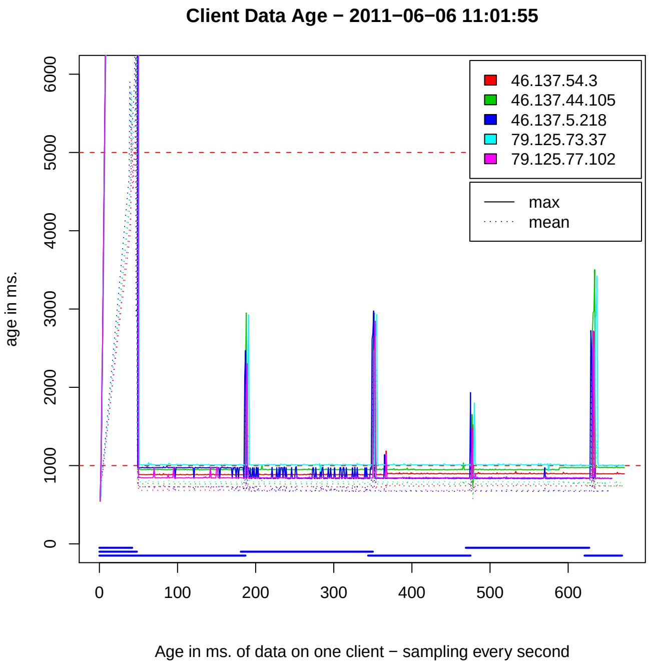
(Test run 97)



(Test run 98)



(Test run 99)



(Test run 100)