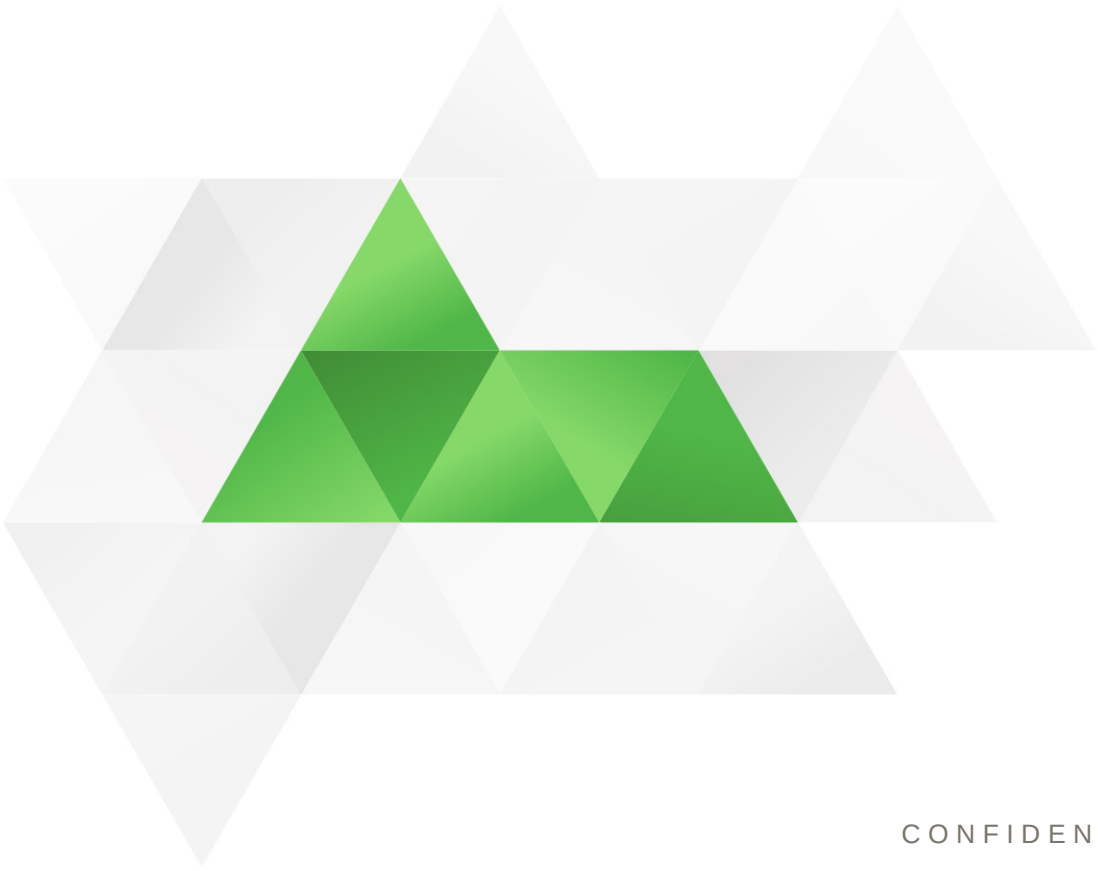
CAPLIN

Liberator 6.1

Benchmarks

December 2013



CONFIDENTIAL

Contents

1 Preface 1
1.1 What this document contains 1
1.2 Who should read this document 1
1.3 Related documents 1
1.4 Feedback 1
1.5 Acknowledgments 1
2 Overview 2
2.1 About the benchmark tests 2
Message latency versus CPU usage 2
Test setup 2
2.2 Headline results 3
2.3 Caplin's benchmark tools 4
3 Test scenarios 6
Low updates 6
Medium updates 7
High updates 7
High updates - batching 7
Very high updates 8
Very high updates - batching 8
4 Test results 9
4.1 Low updates 9
4.2 Medium updates 11
4.3 High updates 13
4.4 High updates - batching 15
4.5 Very high updates 16
4.6 Very high updates - batching 18
4.7 Message sizes 19
5 How Caplin's benchmark tests were conducted 20
5.1 Test method 20
Approach 20
Test setup 20
5.2 Test configurations 21
5.3 Test software 22
Caplin Liberator server 22
Operating system 22
Test DataSource application (Benchsrc) 22

Test RTTP client application (Benchrttp) 22
5.4 Test hardware 23
Main server 23
Main harness 23
Client harnesses 24
5.5 The network 24
6 Frequently asked questions 25
6.1 What burst configuration should we use? 25
6.2 How many threads should we configure? 25
DataSource threads 25
Session threads 26
CPU resource considerations 26
6.3 How do message sizes affect performance? 26
6.4 How does network latency affect performance? 26
6.5 How much bandwidth will our Liberator use? 26
6.6 How many subscriptions can Liberator handle? 27
6.7 How much disk space will our Liberator need? 27
7 Glossary of terms and acronyms 28

1 Preface

1.1 What this document contains

This document details the results of a set of performance benchmark tests carried out on Caplin Liberator 6.1. It is hoped that the information provided in this report will assist customers in production capacity planning when deploying Liberator 6.1.

- Section 2.1 on page 2 gives a summary of the tests performed by Caplin Systems.
- Section 2.2 on page 3 summarizes the main results of the tests relating to typical performance profiles for Caplin Liberator.
- Section 3 on page 6 describes the test scenarios.
- Section 4 on page 9 contains the results, with performance graphs, of the standard tests.
- Section 5 on page 20 gives detailed information on how the benchmark tests were conducted.
- Section 6 on page 25 addresses frequently asked questions about how to configure and tune Liberator and its environment to achieve the required performance.

1.2 Who should read this document

This document is intended for anyone who is evaluating Caplin Liberator's performance characteristics, or who is planning to deploy Caplin Liberator. Typical readers would be:

- Technical Managers
- System Architects
- System Administrators

1.3 Related documents

[1] **Liberator 4.5 Benchmarks**

1.4 Feedback

Customer feedback can only improve the quality of our product documentation, and we would welcome any comments, criticisms or suggestions you may have regarding this document.

Please email your feedback to documentation@caplin.com.

1.5 Acknowledgments

2 Overview

2.1 About the benchmark tests

The benchmark tests detailed in this document are designed to show how Caplin Liberator will perform on the Linux® platform, when deployed as a real-time financial internet hub, streaming data updates to web-based financial trading applications.

The tests cover:

- Low update rates, typical of a low-end information portal
- Medium update rates, typical of a low-volatility trading system, such as credit trading
- High update rates, typical of a high-activity trading system such as FX
- High update rates, using batching
- Very high update rates, representing the most extreme online trading requirements
- Very high update rates, using batching

The main factor affecting the overall performance of Liberator is the power of the machine on which it runs. The tests were conducted on servers representing typical commercially available machines that can be used to host Web servers and server applications. A single Liberator instance was run on one machine, while test harnesses were run on other machines to provide data and client processes.

Message latency versus CPU usage

The key item measured in the tests was the end-to-end message latency against the number of logged in clients, and by implication the number of update messages being sent out to the totality of the connected client base.

Although some of the test results show CPU usage, in practice, end-to-end message latency is more significant as a measure of Liberator performance than CPU usage. Message latency has a direct impact on users and may increase long before CPU usage reaches its maximum. The aim of sizing a system incorporating Caplin Liberator should be to achieve a maximum desired message latency for a given maximum update rate.

Test setup

For detailed information on the test set up used at Caplin Systems, see How Caplin's benchmark tests were conducted on page 20.

It is hoped that the information provided in this report will assist customers in production capacity planning. However, while the tests were designed to emulate real-world traffic and user scenarios, they were conducted using specific hardware running in an isolated environment, and therefore no guarantees can be made that identical results will be achieved in other environments.

2.2 Headline results

Details of the test scenarios, environments, full results and graphs can be seen in later sections, but here are some headline results for the tests run:

- 100,000 users each receive 1 message/second with a latency of under 4 milliseconds.
- 40,000 users each receive 10 messages/second with a latency of only 3 milliseconds.
- 12,000 users each receive 50 messages/second with a latency of only 6 milliseconds.
- 30,000 users each receive 50 messages/second with a latency of under 55 milliseconds.
- 4,000 users each receive 100 messages/second with a latency of only 9 milliseconds.
- 16,000 users each receive 100 messages/second with a latency of 55 milliseconds.

2.3 Caplin's benchmark tools

Benchmarking a streaming server such as Liberator in a realistic manner is a challenge, because of the need to simulate the large numbers of users and high update rates that would be encountered in the real-world business environments where the server is typically deployed. Caplin Systems has produced a suite of tools, the Benchtools that make such benchmarks easier to set up and run.

The Benchtools suite was used to run the benchmark tests described in this document. These tools are also available for Caplin's customers to measure the performance of Caplin Liberator in their own environments.

The following diagram indicates how the Benchtool components integrate with Caplin Platform:

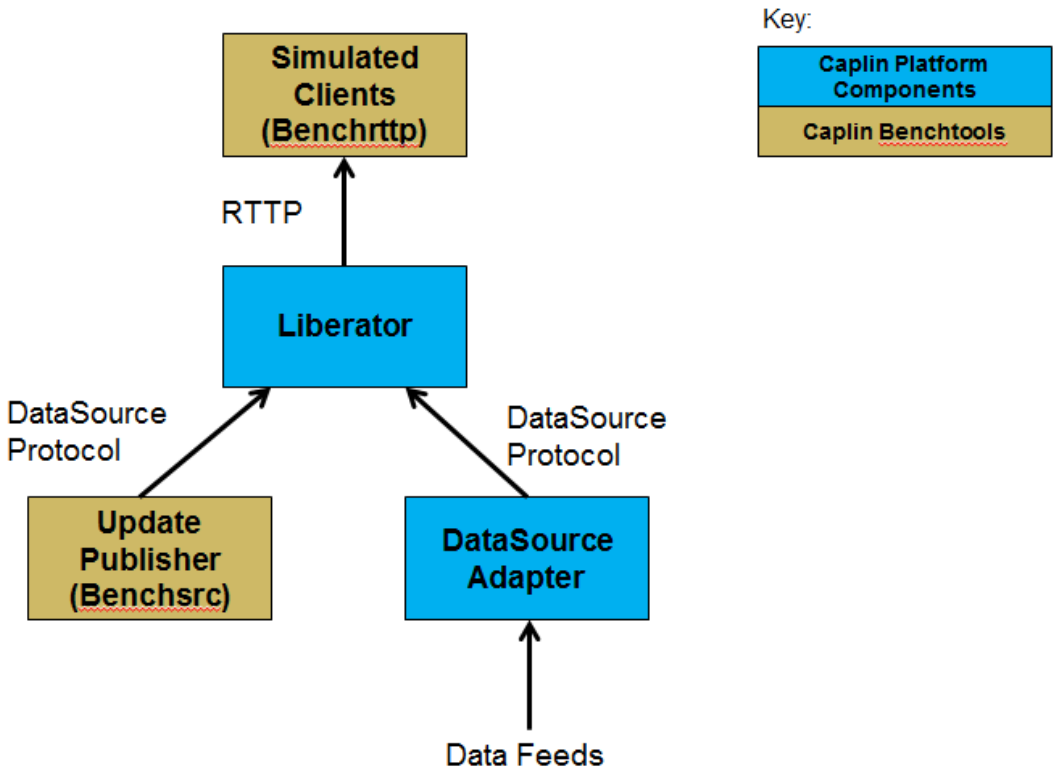


Figure 2–1: Benchtool components in Caplin Platform

The Benchtools package comprises two components:

- **Benchsrc**
  Caplin's update publisher which generates updates of a configurable size and frequency.
- **Benchrttp**
  Caplin's scalable client simulator. This creates real RTTP sessions, but simulates clients being spread across many computers/browsers.

As with StreamLink clients, simulated clients use the RTTP protocol to establish a streaming connection to Liberator's HTTP port. Similar to other DataSources, Benchsrc sends updates to Liberator's DataSource port via the DataSource protocol.

The architecture shown in the diagram allows two test configurations:

1. Benchsrc generates updates.
2. Another publisher generates updates to a feed such as Reuters RMDS.

In both configurations Benchrttp clients consume the updates. All the test in this document use Benchsrc (configuration 1).

3 Test scenarios

The following sections describe the test scenarios used in these benchmarks. These scenarios are designed to simulate different types of activity and data rates, as commonly seen in real time financial applications, and they demonstrate the kind of performance that Liberator can achieve.

The exact numbers can vary significantly between different business scenarios, *and therefore Caplin always advises customers to run benchmarks that reflect their actual requirements and data update rates*. Caplin's benchmark tools make this easier to do; once the test environment is setup using these tools, it is easy to configure and test different scenarios. Benchmarks can either be run against a test backend, such as Benchsrc (which the tests described in this document use), or against real data supplied by the customer.

Each scenario consists of a source of data and a set of clients.

The source of data (benchsrc) is configured to supply a set of subjects, at a known update rate. Each client subscribes to a subset of these subjects, choosing at random. As the number of clients is increased, more of the source subjects are subscribed to, up to the point when all source subjects are subscribed to. In these tests, the point at which all source subjects are subscribed to is quite early in the test.

In all tests the message size is 54 bytes unless otherwise stated. This is a 5 field message, typical of financial applications.

Low updates

This scenario is a good base test, as it is typical of a low-end information portal.

Source subjects	1,000
Update rate per subject	0.5 updates/second
Total source update rate	500 updates/second
Subscriptions per client	2
Update rate per client	1 update/second
Number of DataSources	1

Medium updates

Compared to the Low updates scenario, this scenario has an increased update rate, increased number of subjects, and more subscriptions from each client. This level of subscriptions and data rates is typical of a low-volatility trading system, such as credit trading.

Source subjects	4,000
Update rate per subject	0.5 updates/second
Total source update rate	2,000 updates/second
Subscriptions per client	20
Update rate per client	10 updates/second
Number of DataSources	2

High updates

Compared to the Low and Medium updates scenarios, this scenario has an even higher update rate, number of subjects, and number of subscriptions from each client. This is typical of a high-activity trading system such as FX.

Source subjects	10,000
Update rate per subject	0.5 updates/second
Total source update rate	5,000 updates/second
Subscriptions per client	100
Update rate per client	50 updates/second
Number of DataSources	2

High updates - batching

This scenario also simulates a high-activity trading system. However, in this test, Liberator has been configured to batch messages together at the point they are sent to the client. This will increase the number of clients that Liberator will handle, at the expense of some latency.

Very high updates

This scenario simulates a very high-end single dealer platform, where each client has a large number of fast moving instruments on their screen.

This is the most onerous scenario tested. Compared to the other scenarios, it has a much larger data set, a higher update rate per object, and a higher number of subscriptions per client. It represents the most extreme online trading requirements.

Source subjects	20,000
Update rate per subject	1 update/second
Total source update rate	20,000 updates/second
Subscriptions per client	100
Update rate per client	100 updates/second
Number of DataSources	2

Very high updates - batching

This scenario also simulates a very high-end single dealer platform, where each client has a large number of fast moving instruments on their screen. However, in this test, Liberator has been configured to batch messages together at the point they are sent to the client. This will increase the number of clients that Liberator will handle, at the expense of some latency.

4 Test results

Each of the three scenarios below shows two graphs. The first graph is a plot of the mean latency as the number of clients increases. Each point on the graph represents the mean of all the messages received in a 30-second period. The second graph shows more detail: the blue plot is the latency range (the maximum and minimum latency received in that period), the white line embedded in the latency range shows the mean latency, and the darker line shows the CPU usage of the Liberator throughout the test.

4.1 Low updates

The low updates scenario is really a test of how many users the system can support. The update rate of 1 message/second is low and not very indicative of financial applications. Each client in this scenario receives 54 bytes/second.

The following graph shows Liberator supporting 100,000 clients, each receiving 1 message/second, with mean latency under 4 milliseconds. At 30,000 clients the mean latency is a mere 1 millisecond. At 100,000 clients Liberator is publishing 43.2 Mbit/second.

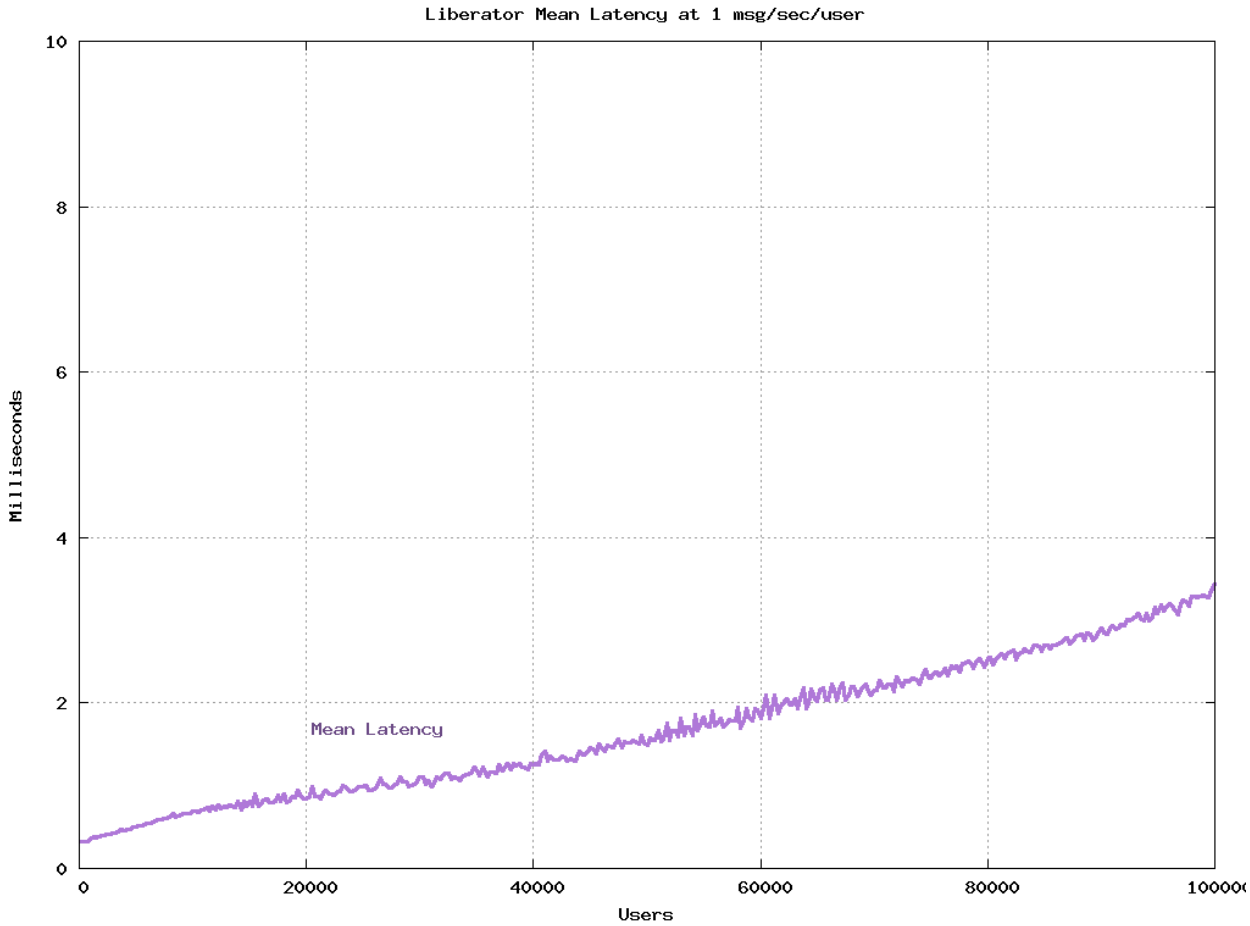


Figure 4–1: Low updates – mean latency

The next graph shows more detail on the latency of messages in this test run. The shaded area shows the range of latencies; the interesting part of this is the top, which represents the maximum latency of any message in that period. The maximum single latency up to 100,000 users is 20-24 milliseconds. For the majority of the test, the maximum latency is about 2-3 times the mean.

The other part of the graph is the CPU usage. At 100,000 clients, the CPU usage of the machine is only 17%. This shows that the machine and Liberator have a lot more to give in this scenario. However, the test was only set up for 100,000 clients, and more client harness machines would be needed to take it much further.

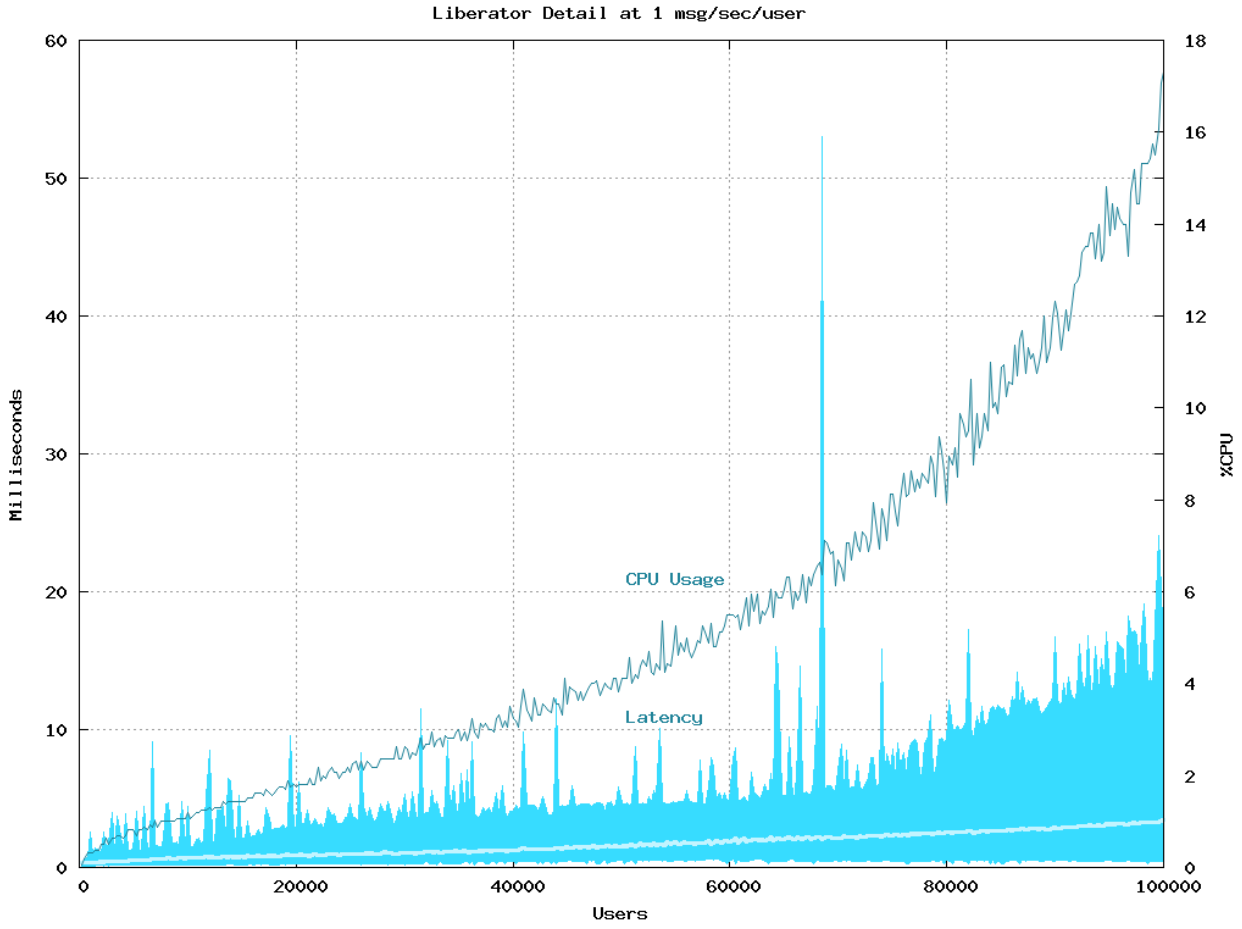


Figure 4–2: Details for low updates

4.2 Medium updates

The medium updates scenario increases the update rate for each client to 10 messages/second; this could represent a low end financial application. Each client receives 540 bytes/second.

The following graph shows that this test run reached 32,000 clients, with a mean latency of under 4 milliseconds. That is a total of 320,000 messages/second being published by Liberator, at a bandwidth of 138 Mbit/second.

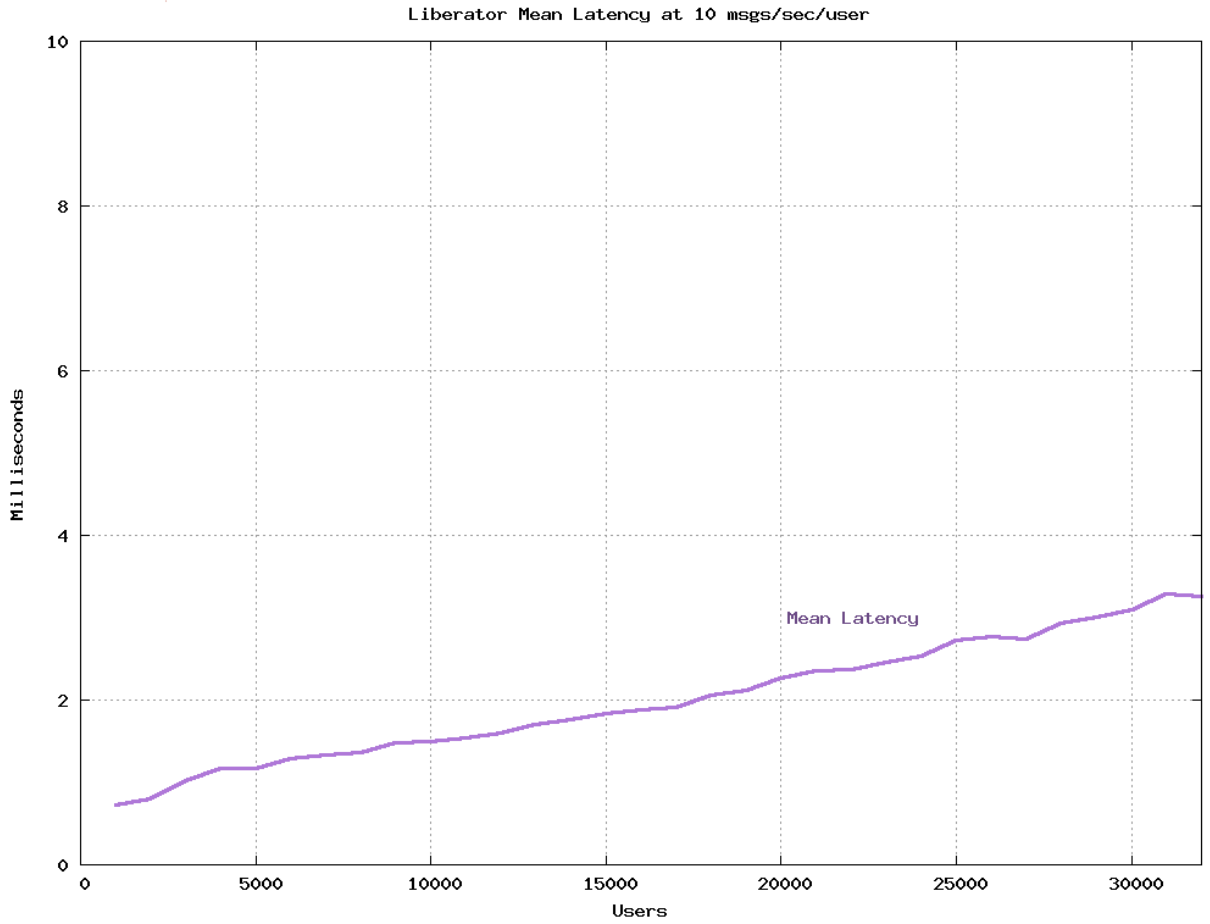


Figure 4–3: Medium updates – mean latency

The following details graph shows that maximum latencies stay below 20 milliseconds for the duration of the test. CPU usage reaches 18% of the 8-core (16 threads) server on which Liberator is running.

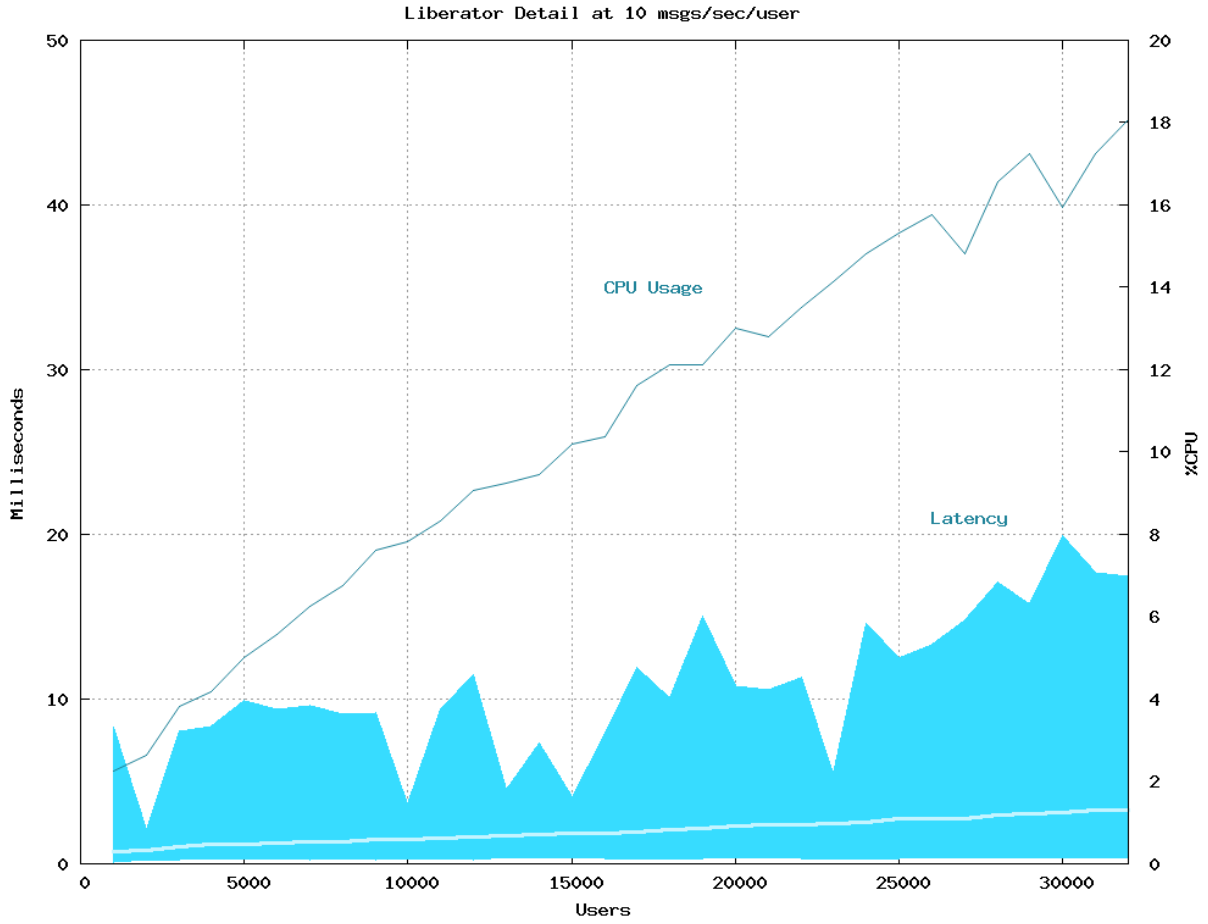


Figure 4–4: Details for medium updates

4.3 High updates

The high updates scenario is more typical of a trading application. Each user receives 50 messages/second, using a bandwidth of 21 kb/second.

The following graph shows that under this scenario Liberator manages to serve over 17,000 clients at a mean latency of about 5 milliseconds. At the end of the test Liberator is publishing 650,000 messages/second using a bandwidth of 281 Mbit/second.

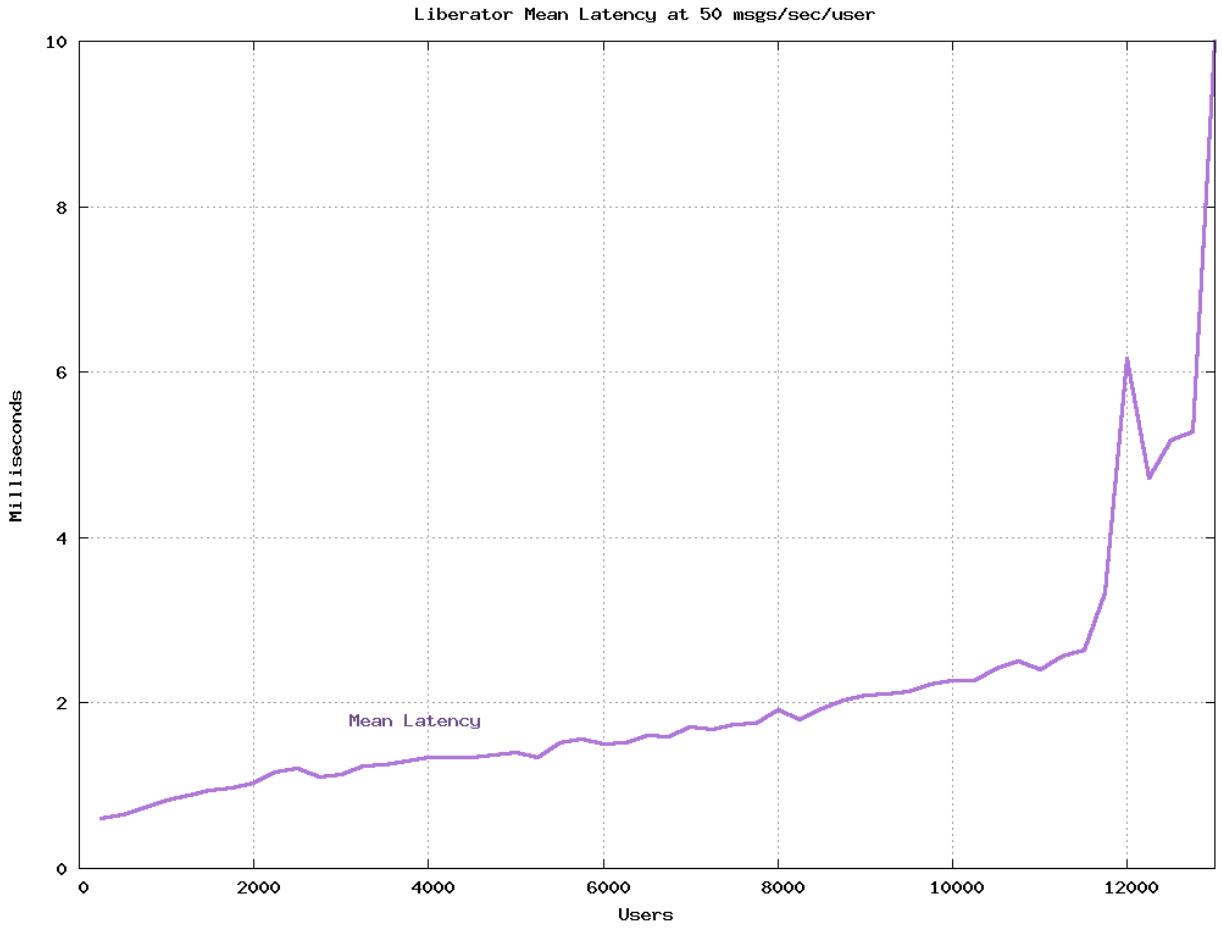


Figure 4–5: High updates – mean latency

The next graph shows that the maximum latency is pretty low throughout the test and only rising towards the end. The CPU usage is 28% of the machine's capacity.

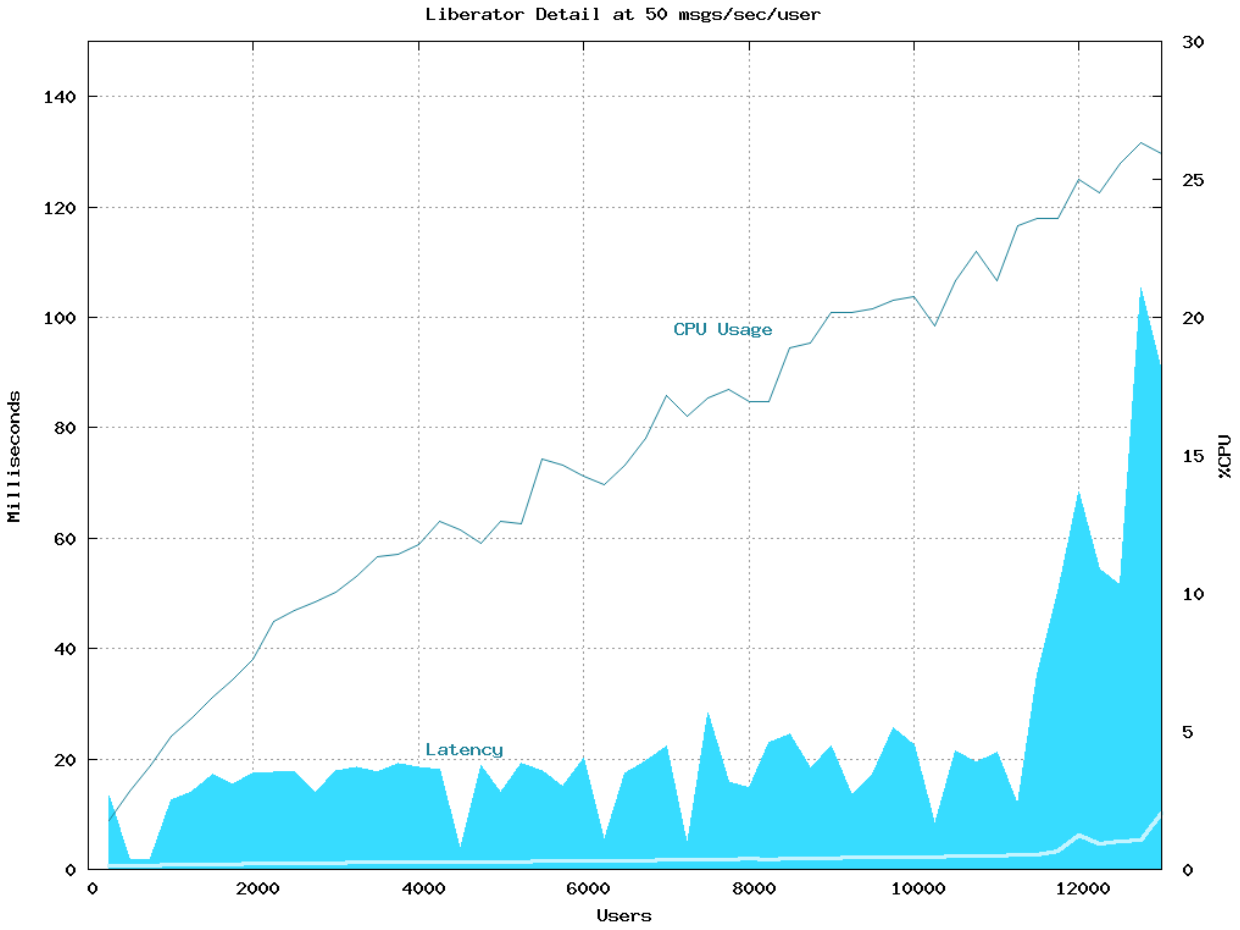


Figure 4–6: Details for high updates

4.4 High updates - batching

Liberator can be configured to batch messages together at the point they are sent to the client. This does not affect the data being sent, just how it is sent. Instead of sending a lot of small packets containing one message each, Liberator sends larger packets, less frequently, containing multiple messages.

The batching feature will only batch messages together when the update rate is over a configured amount, therefore it can be used to handle peak data rates. However, the scenarios tested here have a more uniform update rate so the batching is always active, when configured.

The following graph shows the mean latency of multiple test runs with different configurations for batching. The blue line shows the mean latency with no batching configured; this is the same run as the graphs above. The other three lines show how adding batching increases mean latency, as expected since we are delaying sending messages, but allows Liberator to support many more clients.

With 100 milliseconds batching, giving a mean latency of just over 50 milliseconds, the test run reached nearly 30,000 clients receiving 50 messages/second each. In that configuration Liberator is publishing about 1.5 million messages per second which is 648 Mbit/second, which could be approaching the limitations of the gigabit network. At 100 milliseconds batching, each batch contains 5 messages on average. At 25 milliseconds batching, the benefit is minimal. This because 25 millisecond intervals means 40 batches per second, which is not much benefit to the 50 messages per second test, as most of the time each batch will actually only contain 1 message.

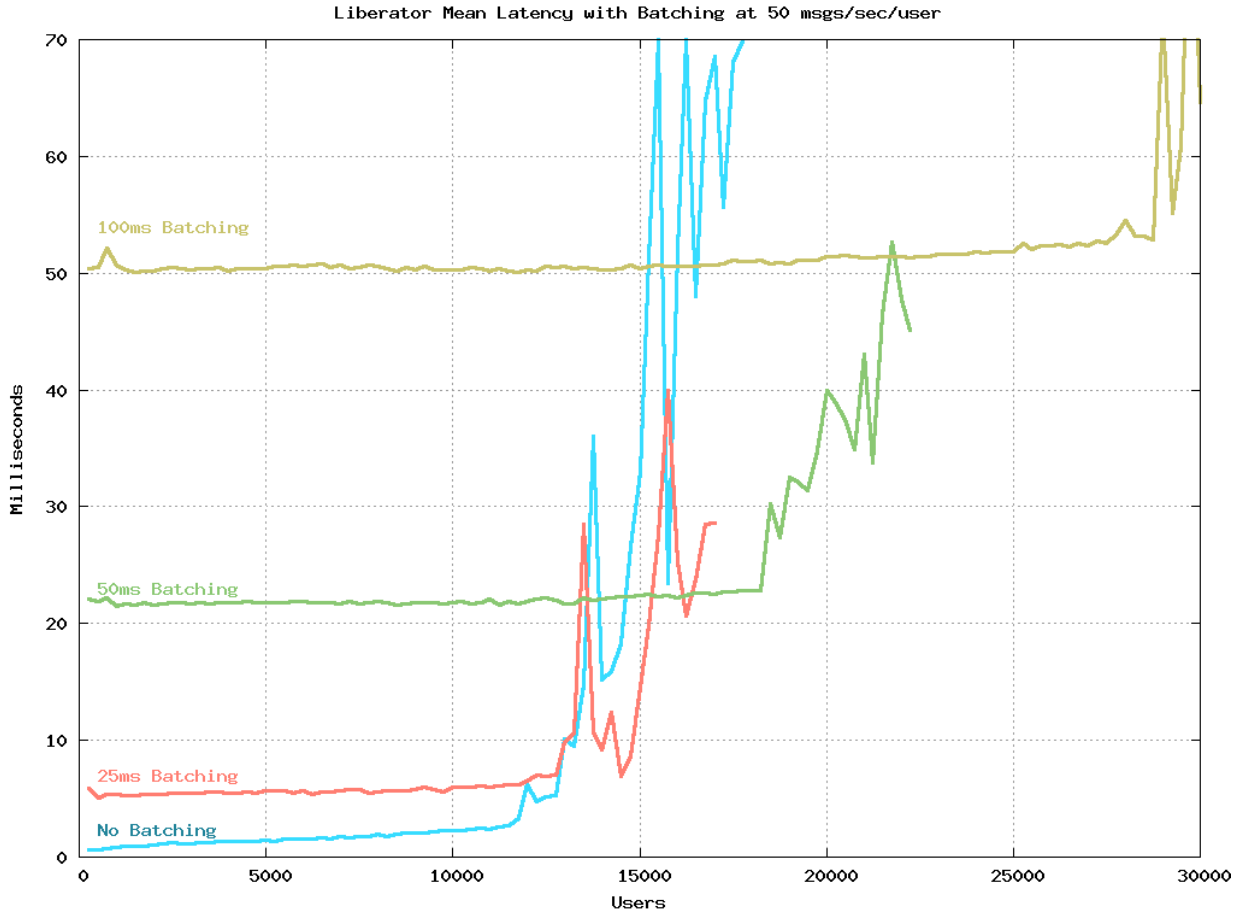


Figure 4–7: High updates – batching

4.5 Very high updates

The very high updates scenario represents a high end trading system. There is a much higher update rate at the source of the data, due to the larger number of subjects available to subscribe to. Each client receives 100 messages/second, using a bandwidth of 43 kb/second.

The following graph shows a maximum number of 4,000 clients, with a mean latency of 9 milliseconds, and the majority of the test run achieving 2-4 milliseconds latency.

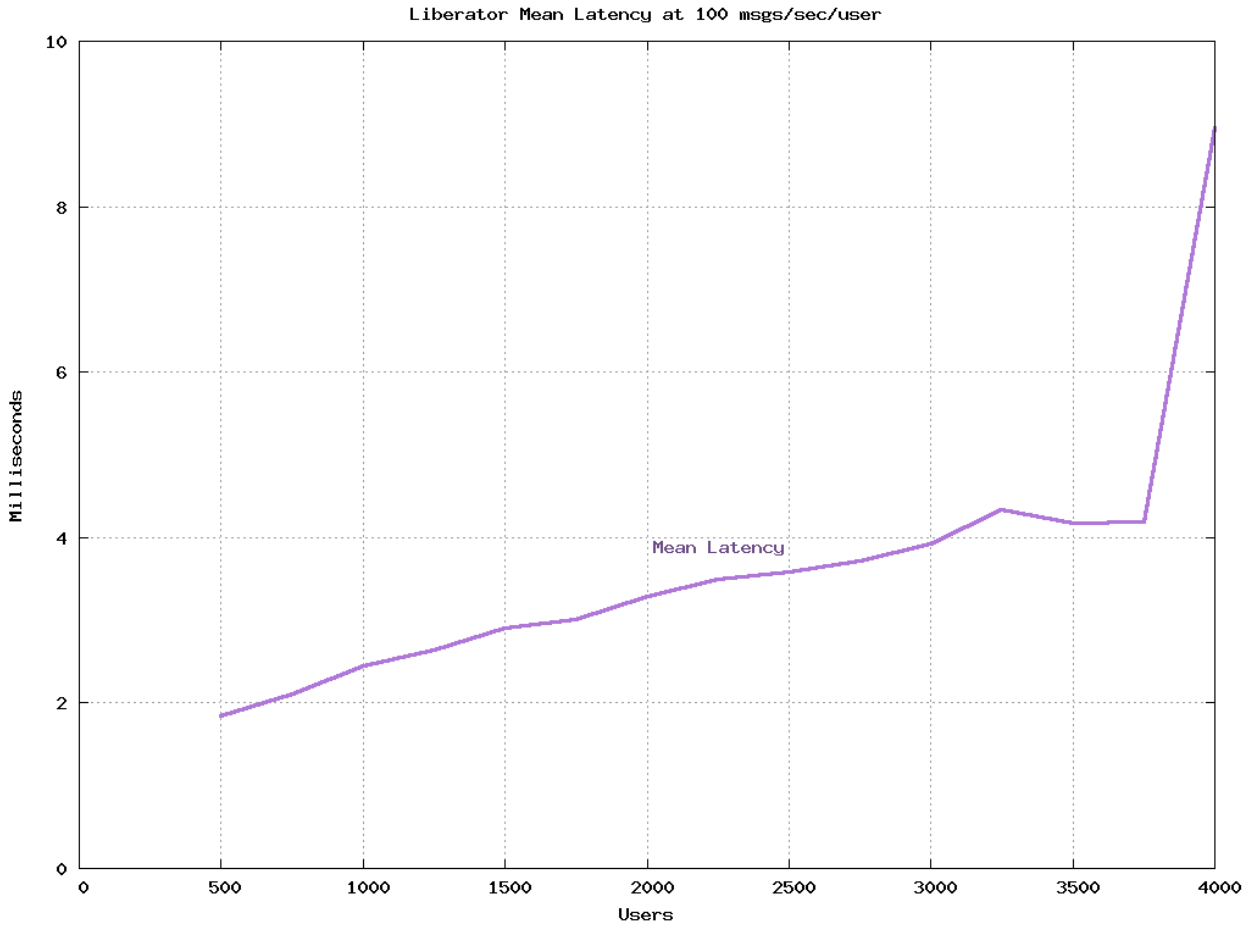


Figure 4–8: Very high updates – mean latency

The next graph shows slightly higher maximum latencies, around 100 milliseconds at some peaks of the test run. CPU usage hits 30% at the 4,000 client mark.

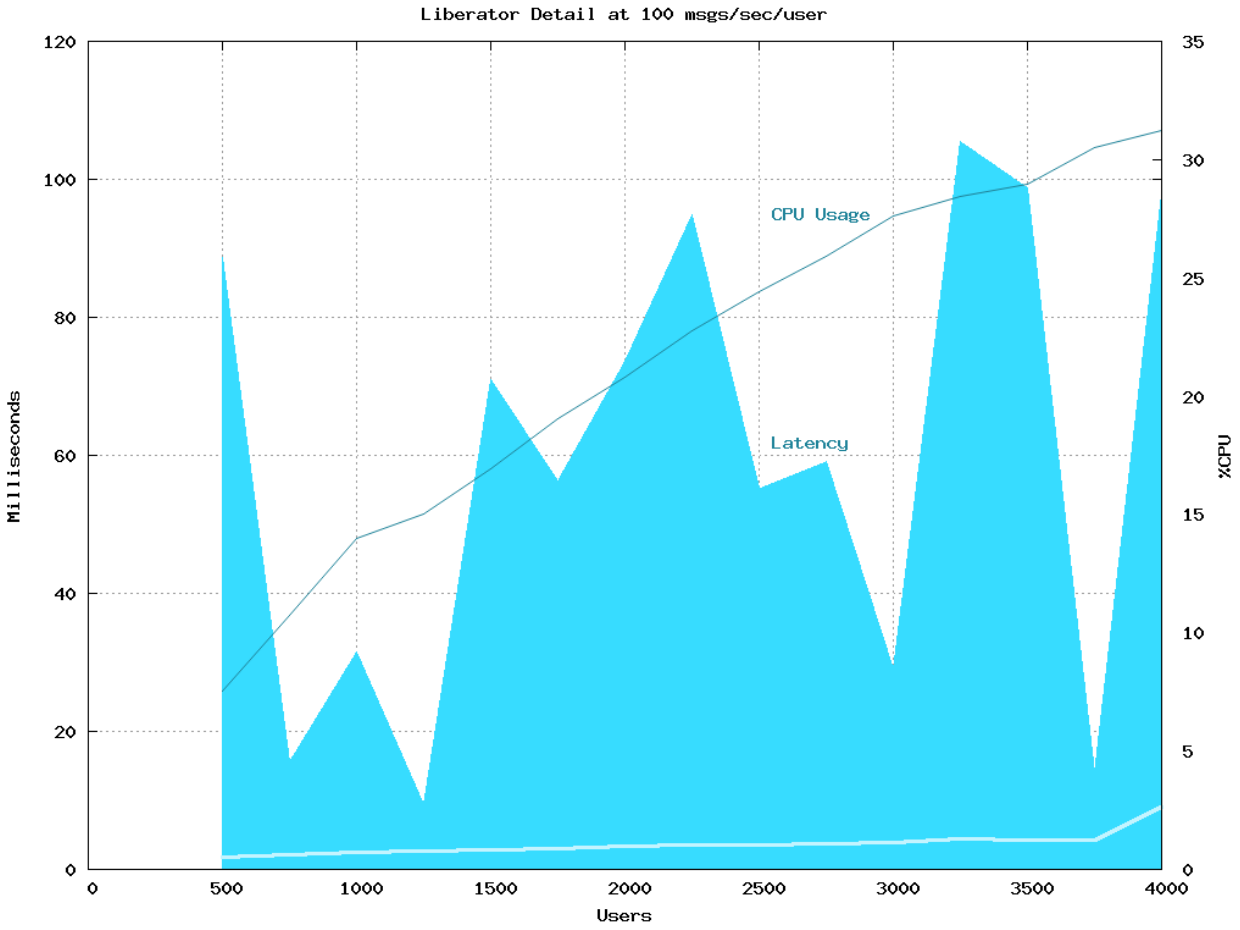


Figure 4–9: Details for very high updates

4.6 Very high updates - batching

Liberator can be configured to batch messages together at the point they are sent to the client. This does not affect the data being sent, just how it is sent. Instead of sending a lot of small packets containing one message each, Liberator sends larger packets, less frequently, containing multiple messages.

The batching feature will only batch messages together when the update rate is over a configured amount, therefore it can be used to handle peak data rates. However, the scenarios tested here have a more uniform update rate so the batching is always active, when configured.

The following graph shows the mean latency of multiple test runs with different configurations for batching. The blue line shows the mean latency with no batching configured; this is the same run as the graphs above. The other three lines show how adding batching increases mean latency, as expected since we are delaying sending messages, but allows Liberator to support many more clients.

With 100 milliseconds batching, giving a mean latency of just over 50 milliseconds, the test run reached over 16,000 clients receiving 100 messages/second each. In that configuration Liberator is publishing over 1.6 million messages per second which is 691 Mbit/second, which could be approaching the limitations of the gigabit network.

At 100 milliseconds batching, each batch will contain 40 messages. This is clearly a benefit to the number of network packets being sent, but at the expense of latency.

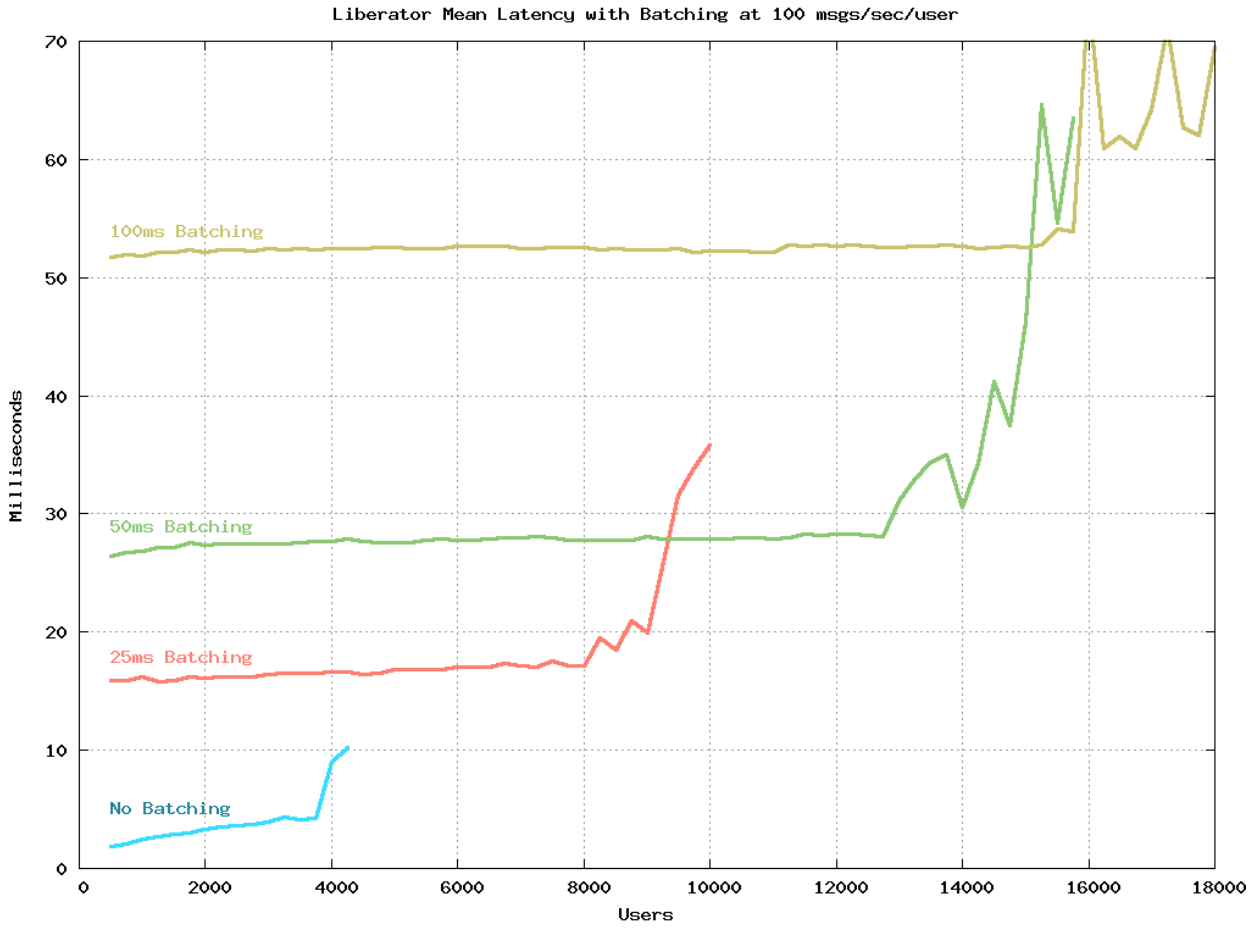


Figure 4–10: Very high updates – batching

4.7 Message sizes

So far all the tests have been with messages of 54 bytes, this may seem like a small message, but it contains 5 fields, typical of a financial application. RTTP, the protocol used between Liberator and clients, is optimized to keep message sizes as small as possible.

The following graph shows a re-run of the high updates scenario, but with different sizes of message. The results for the original 54 byte message are plotted, along with runs using 108, 162 byte, and 216 byte messages.

The graph shows that as message size increases, latency is not affected much, but the ultimate number of clients Liberator can support is reduced.

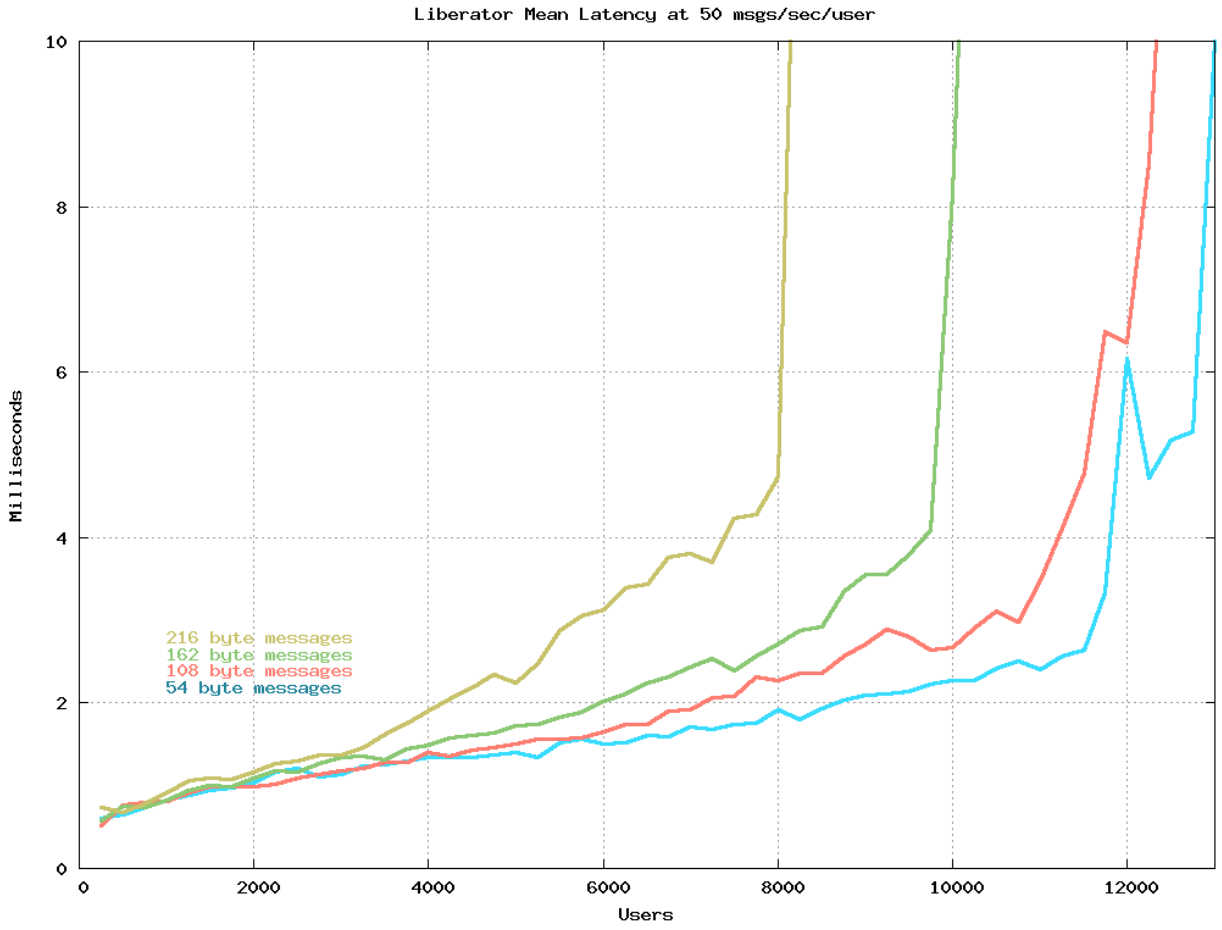


Figure 4-11: Message sizes

5 How Caplin's benchmark tests were conducted

The following sections describe the test method used, give information about the test configurations, and detail the test software, test hardware, and the network used.

5.1 Test method

Approach

Although the benchmark consisted of several different tests, they all followed a similar method. Each test consisted of one or more DataSources publishing messages into a Liberator which pushed the messages out to a set of subscribing clients through RTTP connections. Each subscribed object was updated at a regular rate by the supplying DataSource. Additional clients were logged on to the Liberator throughout the test run to determine the effect of increasing the load on the Liberator.

Test setup

The multiple RTTP client connections were simulated using a specially written application called Benchrttp. The DataSource application supplying the Liberator (Benchsrc) was also specially written. Both Benchrttp and Benchsrc are controllable using a command protocol, thus allowing message rates and number of clients to be remotely managed using scripts. The following diagram shows the hardware configuration used for the tests, and how the test software was distributed across the hardware.

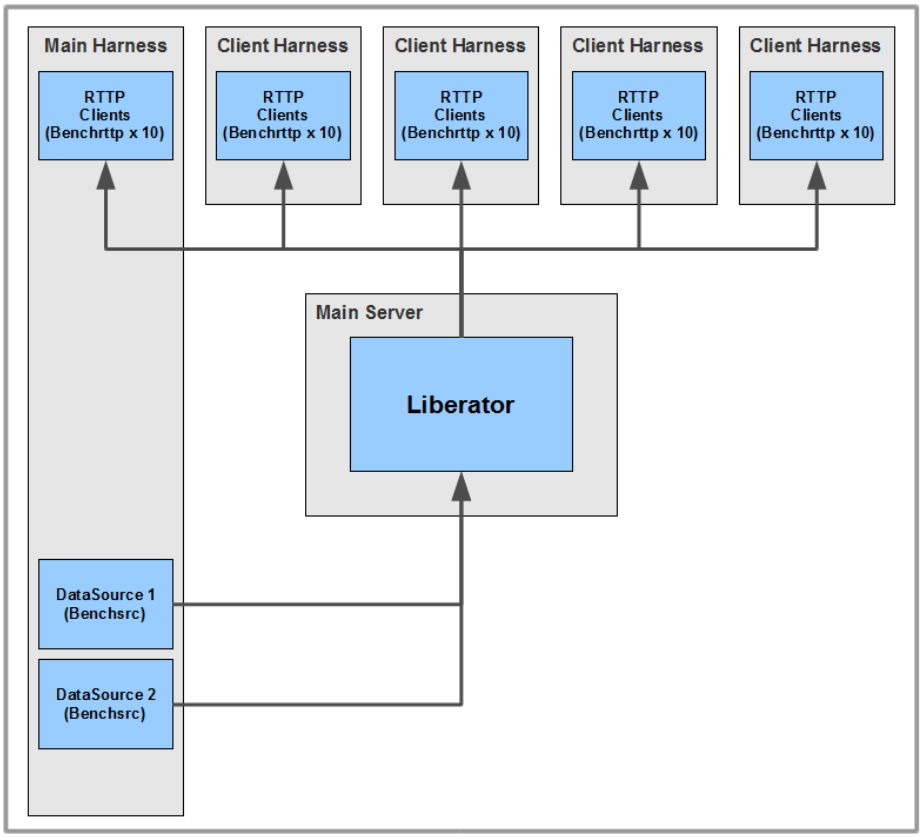


Figure 5–1: Benchmark test setup

The Liberator server was hosted on a machine of its own (Main Server). A separate machine (Main Harness) was used to host up to four DataSources (Benchsrc) feeding the Liberator. Main Harness and four further machines (Client Harness's), hosted up to ten instances each of Benchrttp.

The number of Benchrttp instances required for each test was determined by the maximum number of simulated clients needed to run the test; enough Benchrttp resource was required to ensure that Liberator limits could be reached before any limits imposed by Benchrttp.

For more detailed information about the hardware and software used to run the tests, see the following sections.

5.2 Test configurations

Most of the tests determined the average latency of update messages delivered to clients, against the number of subscribing (logged on) clients. This gave a measure of how the Liberator performed as the overall client update rate increased, for the following reason.

Each client subscribed to a number of objects (the number varied according to the test). Because each subscription required Liberator to update those objects on the client in line with the updates from the DataSources, increasing the number of logged on clients had the effect of proportionally increasing the Liberator's client update load.

For example, if each client subscribed to 100 objects and there were 2,000 clients logged on to the Liberator, then the Liberator would have to supply 100 x 2,000 = 200,000 object updates per second to all the clients.

The latency of the client update messages was measured at the client end. In all cases measurements were taken from the application log files and the results were plotted.

The most important Liberator parameters affecting performance are:

- The number of DataSource threads.
- The number of session threads (**threads-num** configuration parameter).
- The value of the **burst-max** configuration parameter.

Many of the tests were performed with varying numbers of DataSource threads and/or session threads, in order to determine the impact of the thread settings on Liberator performance. The test results show the optimum configuration for each scenario.

5.3 Test software

Caplin Liberator server

The tests were run against a Caplin Liberator 6.1 server. The server configuration only had a few changes from the default settings, as follows:

- Log Cycling. This was modified to prevent the very large amounts of data being processed from using up too much disk space.
- System Max Files. This was increased to 131072 to allow high numbers of clients to connect.
- Object Throttling. The default object throttling of 1 second was turned off for benchmarking. This allowed all clients to receive all the messages they were subscribed to.
- Threads. The numbers of DataSource threads and session threads were adjusted according to the needs of the individual tests. Many of the tests were repeatedly run with different numbers of threads in order to determine the impact of the thread settings on Liberator performance.
- Burst Settings. The default setting for **burst-max** is 0.5. This was altered to 0 for most of the tests. The “Very high updates - batching”test (page 18) used several values for this parameter.

Operating system

The operating system used on all the server hardware was Linux – CentOS 6.3 (Final) or Red Hat® Enterprise Linux® version 6. This was a standard configuration with only one significant change; the number of open file descriptors was increased to allow Liberator to support high numbers of client connections. This is detailed in the **Caplin Liberator Administration Guide**.

Test DataSource application (Benchsrc)

The Benchsrc test tool was configured to produce updates to sets of objects at a known message rate. Benchsrc is a single threaded application, so for those tests where the Liberator was configured to use multiple DataSource threads there was an instance of Benchsrc for each DataSource thread.

Test RTTP client application (Benchrttp)

The Benchrttp test tool was configured to request sets of objects that were published by the DataSource. It measured the number of messages being received and also the latency of the messages. The simulated clients all used RTTP Type 2 connections – this type of connection is HTTP tunnelled.

5.4 Test hardware

The benchmark tests used up to 6 machines as shown in the diagram in Figure 5–1 on page 20. Caplin Liberator ran on a dedicated machine (Main Server). The test RTTP client processes (Benchrttp) ran on up to 5 separate machines to spread the load (Main Harness and Client Harnesses). The test DataSources (Benchsrc) ran on one of the Benchrttp machines (Main Harness).

Main server

The main server runs Liberator, which is the component being tested.

Components	Liberator
Vendor	Dell
Model	PowerEdge R620
Processors	2 x Four-Core (8 thread) Intel Xeon E5-2643 3.3GHz
Memory	32 GB
Operating System	CentOS 6.3 (Final)
Network Card	Broadcom NetXtreme II BCM5720

Main harness

This machine is used to provide the data for the tests. It also runs some of the client harnesses so that latency can be measured using the same clock from source to destination.

Components	Benchsrc, Benchrttp(s)
Vendor	Dell
Model	PowerEdge R415
Processors	2 x Six-Core AMD Opteron 4180 2.6GHz
Memory	16GB
Operating System	CentOS release 5.5 (Final)
Network Card	Broadcom NetXtreme II BCM5716 and Intel 82572EI Gigabit NIC

Client harnesses

These machines run the rest of the client harnesses. Latency is still measured on these machines, to assert that thresholds are not broken, but graphs are all based on times recorded on the main harness machine.

Components	Benchrttp(s)
Vendor	Dell
Model	PowerEdge R210
Processors	1 x Quad-Core Intel Xeon X3460 2.8GHz
Memory	4GB
Operating System	CentOS release 5.5 (Final)
Network Card	Broadcom NetXtreme II BCM5716 and Intel 82572EI Gigabit NIC

Note that the test machines all use Broadcom network cards. We found that these performed much better than Intel network cards.

5.5 The network

The test machines were set up on a Gigabit network which was used solely for transmitting the test data. Any control messages or terminal access used a separate network.

6 Frequently asked questions

The following sections discuss various issues concerning how to configure and tune Liberator and its environment to achieve the required performance.

6.1 What burst configuration should we use?

Liberator can be configured to batch messages together to improve overall performance. Liberator's default configuration uses a 0.5 second batch time (for low latency messaging this should be reduced). The configuration option for this is called **burst-max**. With this configuration, Liberator may batch together outgoing messages to a client, delaying them by up to 0.5 seconds. If the message rate is slower than the burst setting the messages will be sent immediately. Batching messages in this way means that with a fairly constant message rate an average latency of half the burst-max setting will be introduced. However, the benefit of this is that the Liberator and network can cope with higher message rates. There is clearly a trade-off between latency and message rate, which is why the tests carried out in these Benchmarks show results for different settings of burst-max (see 4.6 Very high updates - batching on page 18).

It is clear that a low **burst-max** setting (for example 0.1 sec) can improve overall average latency and achieve much higher messages rates than with no batching. In some cases, increasing burst-max to the default of 0.5 sec will allow Liberator to achieve even higher message rates, but this is at the expense of increased latency.

6.2 How many threads should we configure?

There isn't a simple answer to this question.

Liberator's configuration option **threads-num** sets the number of session threads used to service client connections. Liberator also implements multiple DataSource threads – there is one thread per DataSource connection.

How many threads you require depends on the maximum anticipated update rate from Liberator's DataSource peers, the maximum expected number of subscribing clients, the subscription profile of the clients (see "DataSource threads" below), and the hardware characteristics of the machine running the Liberator (CPU speed and number of CPUs).

> Caplin Systems can provide customers with expert guidance on configuring the optimum number of Liberator threads to meet their particular requirements.

DataSource threads

DataSource threads enable Liberator to better handle high update rates from its DataSources. Even if there is only one DataSource instance feeding Liberator, you can still configure more than one connection to the DataSource so that the Liberator's performance can benefit from using multiple DataSource threads.

There is also a relationship between the number of *clients* using the Liberator and the number of DataSource threads required, but this depends on the subscription profile of the clients, as follows.

At one extreme, if each client subscribes to a different set of objects, then the update demand on the

DataSources will increase roughly in proportion to the number of subscribing clients. So in this case, as the maximum anticipated number of clients increases, more DataSource threads will be required in order to achieve the same message latency.

At the other extreme, if each client subscribes to the same set of objects, the update demand on the DataSources will remain constant as the number of subscribing clients increases. So in this case the number of DataSource threads required will *not* depend on the number of subscribing clients.

You should configure enough DataSource connections (and hence DataSource threads) to allow Liberator to handle the maximum anticipated update rate from the DataSource(s) with acceptable message latency for the maximum number of clients expected to use the Liberator.

Don't use more DataSource threads than you need, because the additional threads will not produce a significant additional improvement in performance. There will be a limit above which adding more DataSource connections has little extra benefit, because the limiting factor becomes the number of session threads.

Session threads

Provided there are enough DataSource threads to handle updates from the DataSources, increasing the number of session threads will allow more clients to subscribe with no unacceptable increase in message latency. However there will be an upper limit on the number of session threads, beyond which performance will decrease – see "CPU resource considerations" below.

CPU resource considerations

Once CPU resource limits have been reached on the machine running the Liberator, adding more threads will not necessarily improve performance. In these tests it was usually the case that 10 session threads (on a 12 core server).gave the best results, this left CPU cores available for the DataSource thread or threads.

6.3 How do message sizes affect performance?

When high numbers of clients and messages are used, the size of the message plays a significant part in the overall performance. Larger update messages decrease Liberator's maximum effective message rate. Liberator also performs slightly better when handling update messages consisting of a small number of large fields rather than messages containing a larger number of smaller fields.

6.4 How does network latency affect performance?

The tests in this document were performed on an internal network. Over the Internet, there will be an increase in latency of anywhere between 20-250 milliseconds. The exact latency will very much depend upon the geographical locations of the sites – with 100-120 milliseconds exhibited for transatlantic communications, for example.

6.5 How much bandwidth will our Liberator use?

In the context of Liberator performance, bandwidth is the update rate delivered to all subscribed clients in bits/sec. In most of these tests each message is about 54 bytes long; this may seem small, but it is because RTTP is a very efficient protocol. The message contains five fields, a typical update in financial applications. Each test gives some details of the bandwidth used per client and overall.

6.6 How many subscriptions can Liberator handle?

The number of subscriptions a client has does not significantly affect performance directly, rather the number of messages is far more significant. This can be controlled using throttling.

6.7 How much disk space will our Liberator need?

Liberator uses approximately 60 MB of disk space when it is installed. Running Liberator requires extra disk space for log files.

The amount of disk space needed for the log files depends entirely on messages rates and client activity. Liberator supports configuration options to control the cycling of log files, so it is possible to limit how much disk space is used and how much information is saved in log files. Log files can grow to high single-digit gigabytes per day in some setups.

7 Glossary of terms and acronyms

This section contains a glossary of terms and acronyms relating to the Liberator benchmark.

Term	Definition
Benchsrc	A Caplin benchmark test tool that provides DataSource messages as input to a Caplin Liberator server. It is primarily intended to be used in conjunction with Benchrttp and the control scripts from the Caplin Benchmarking kit.
Benchrttp	A Caplin benchmark test tool that connects to a Liberator server and simulates a configurable number of clients and contributors of streamed RTTP data. It is primarily intended to be used in conjunction with Benchsrc and the control scripts from the Caplin Benchmarking kit.
burst-max	The efficiency of Caplin Liberator can be increased by writing user output in defined "bursts", particularly in a system with a large number of clients where bursting batches together small messages before outputting them to a client. burst-max is a Liberator configuration parameter that controls bursting. It is the maximum time in seconds of client update buffering before Liberator will send updates to a client. For more information see http://www.caplin.com/developer/component/liberator.
Caplin Liberator	Caplin Liberator is a real-time financial internet hub that delivers trade messages and market data to and from subscribers over any network.
Caplin Platform	A framework for building single-dealer platforms that enables banks to deliver multi-product trading direct to client desktops.
DataSource	DataSource is the internal communications infrastructure used by Caplin Platform's server components such as Caplin Liberator, Caplin Transformer, and DataSource adapters.
DataSource adapter	A DataSource application that integrates with an external (non-Caplin) system, exchanging data and/or messages with that system.
DataSource application	A Caplin Platform application that uses the Caplin DataSource APIs to communicate with other Caplin Platform applications via the DataSource protocol.
DataSource peer	Alternative name for a DataSource application.
Mbit	Megabit

CAPLIN
THE WEB TRADING TECHNOLOGY COMPANY

Contact Us

Caplin Systems Ltd
Cutlers Court
115 Houndsditch
London EC3A 7BR
Telephone:+44 20 7826 9600

Caplin Systems, Inc.
7 World Trade Center
46th Floor
New York, NY 10007
Telephone: +1 (212) 266 0198

www.caplin.com