

Federating AMQP 1.0 message brokers

David Joe Wade
4/17/2014

Background

- AMQP - Advanced Message Queuing Protocol
 - Open-standard (current version 1.0)
 - Application layer (Internet model - FTP/DHCP/DNS)
 - Message Oriented Middleware (MOM)
 - OASIS standard (Approved October 2012)

History

- Started in 2003 by John O'Hara at JPMorgan Chase to support messaging needs of financial industry.
- Initial implementation contracted to iMatix - developed a broker in C and documented protocol (OpenAMQ).
- Versions
 - 0-8, 0-9, 0-9-1, 0-10, 1.0
- Current working Group includes 23 companies.
 - Cisco, Red Hat, Bank of America, VMware...
- AMQP 1.0 specification was a major change from 0-* specifications.

AMQP features

- Layered Protocol (5 Layers).
- Defines encoding scheme for common types.
- Symmetric, asynchronous protocol.
- Defines a standard, extensible message format.
 - Message contents are immutable.
 - Allows end-to-end signing and encryption.
 - Annotations supported, but not part of message.
- Defines standardized but extensible messaging capabilities.

Architecture/Terminology

Term	Concept	Example
Exchange	Broker connection point (location)	An airport (Bradley)
Queue/Topic	Message destination	RPI
Binding	Rule that determines path of message from exchange to queue	Flight route to Bradley
Virtual Hosts	Isolated instance of exchanges/queues/bindings	Virtual machine?
Connections	Network connection (TCP session)	POTS Circuit
Channels	Pooled path over connection	Connection pool

AMQP Message brokers

AMQP Brokers

	Version	License	Language	Release date	AMQP version
Apache ActiveMQ	5.9.0	ASL2.0	Java	October 2013	1.0
Apache Apollo	1.7.0	ASL2.0	Java/Scala	February 2014	1.0
Apache Qpid	0.26	ASL2.0	Java	February 2014	1.0
Apache Qpid	0.26	ASL2.0	C++	February 2014	1.0
RabbitMQ	3.2.4	Mozilla 1.1	Erlang	March 2014	0-9-1
SwiftMQ	9.5.0	Proprietary	Unknown	Unknown	1.0
HornetQ	2.4.0	ASL2.0/ LGPL	Java	December 2013	1.0
StormMQ	2010.05. 20	Client - Mozilla 1.1	Java	May, 2010	1.0
Microsoft Service Bus	Unknown	Proprietary	Unknown	Unknown	1.0

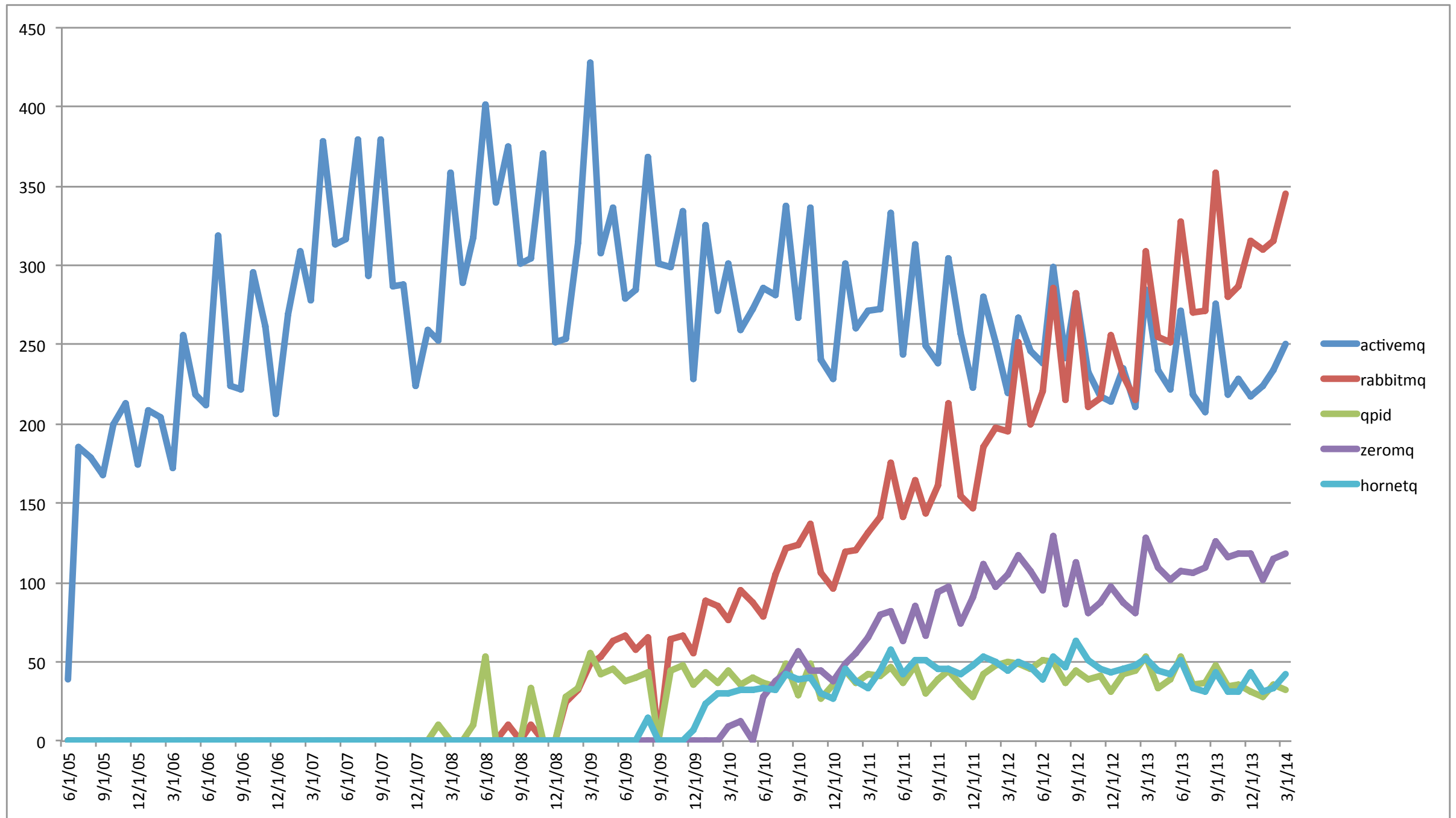
AMQP Clients

Implementation	AMQP Versions					Language bindings			
	0-8	0-9	0-9-1	0-10	1.0	Java	C/C++	.NET	Python
Qpid-JMS	✓	✓	✓	✓	✓	✓			
QPID Proton	✓	✓	✓	✓	✓	✓	✓		
RabbitMQ	✓	✓	✓			✓	✓	✓	✓
.NET Service Bus					✓			✓	
SwiftMQ					✓	✓		✓	

This list is certainly not exhaustive - RabbitMQ lists 175+ RabbitMQ clients in 21 different programming languages.

Most broker implementations that support AMQP 1.0 seem to use the Qpid Proton libraries

Google search popularity

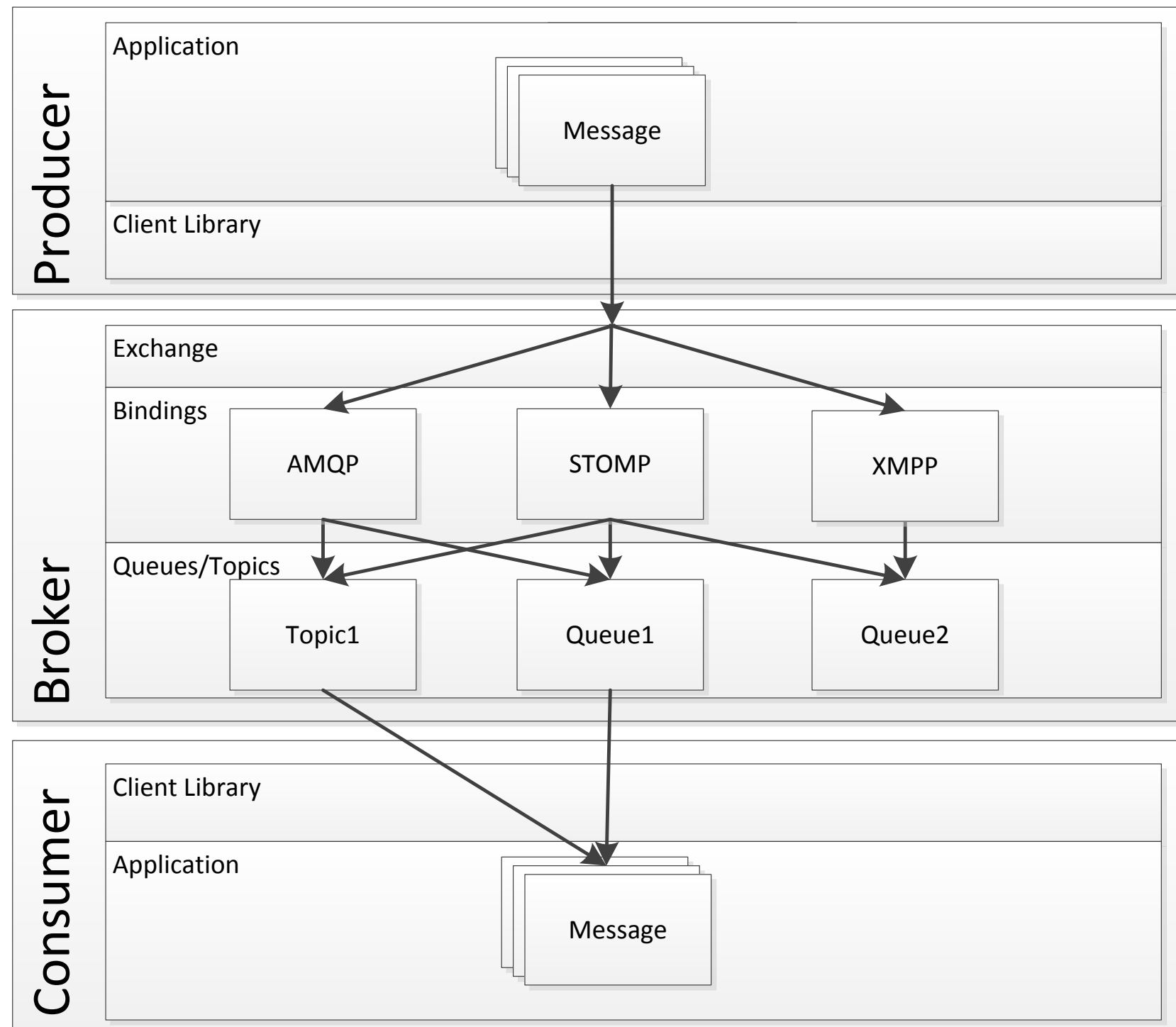


data from: <http://www.google.com/trends/explore#q=activemq%2C%20rabbitmq%2C%20qpid%2C%20hornetq%2C%20Apache%20Apollo&cmpt=q>

Broker features

- Automatic reconnect.
- Failover/High Availability.
- Persistence.
- Clustering.
- Federation.

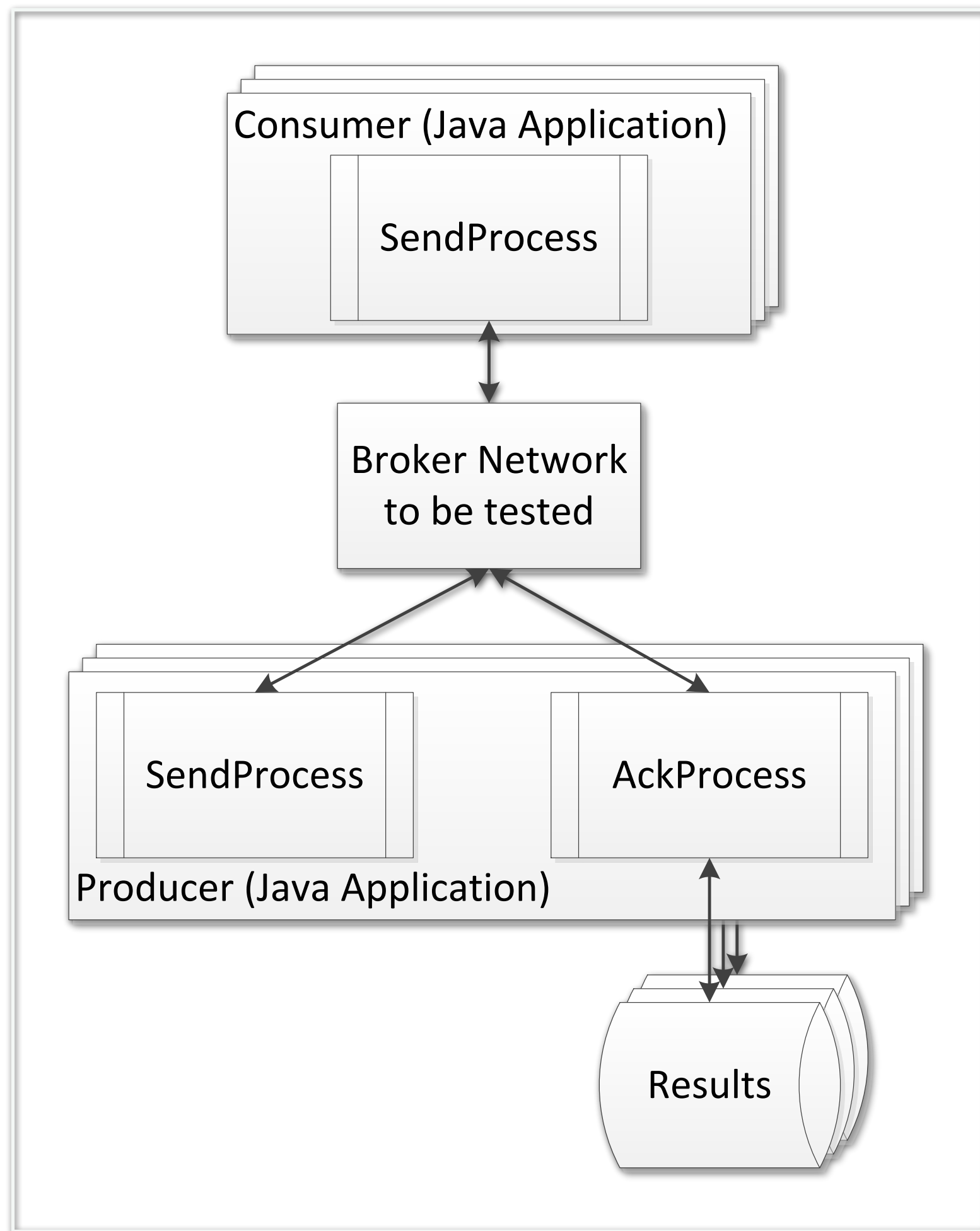
Broker Architecture

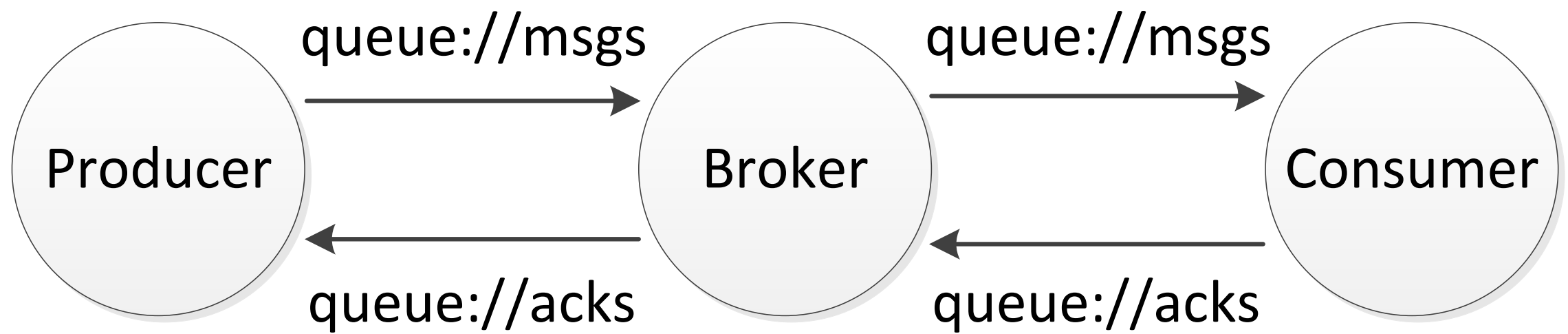


Project design

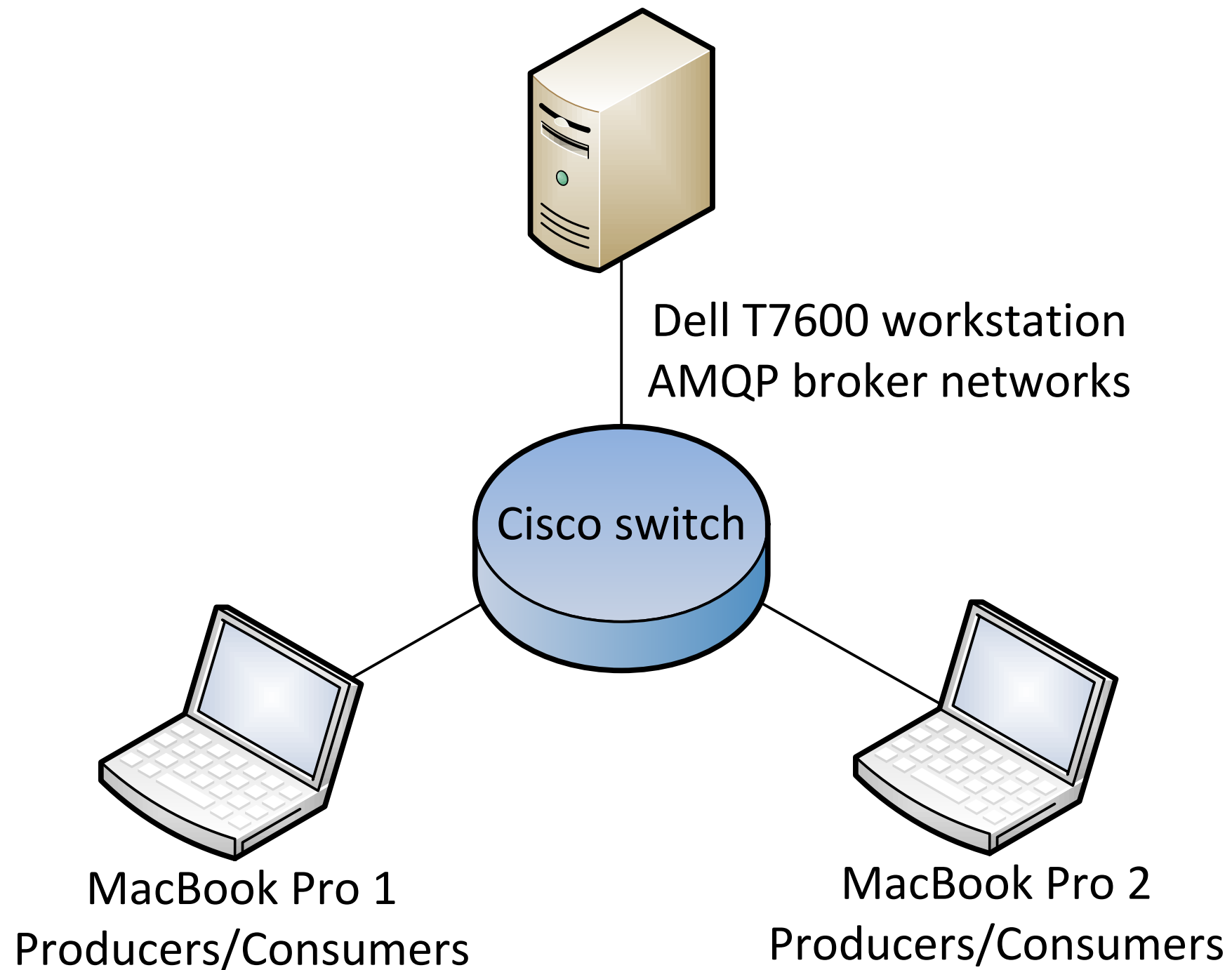
Methodology

- Create single node broker networks.
- Collect performance data from all AMQP 1.0 brokers.
- Create federated broker networks with homogeneous broker implementations.
- Collect performance data.
- Create federated broker networks with heterogeneous broker implementations.
- Collect performance data.





Architecture



Hardware specifications

	T7600	Client 1	Client 2
CPU	Index Xeon 3.1GHz E5-2687W	Intel Core i7 2.6GHz	Intel Core i7 2.5 GHz
RAM	32 GB 1600MHz DDR3	8GB 1600 MHz DDR3	8 GB 1333 MHz DDR3
OS	Ubuntu 12.04.4 x64	OS X 10.9.2	OS X 10.9.2
JVM	1.7.0_51-b13	1.7.0_51-b13	1.7.0_51-b13

Network Switch	Cisco SG100D-08P Gigabit ethernet
----------------	-----------------------------------

Project Software

Software

- Develop simple producer/consumer in Java (portability was obviously important).
- Used latest version of Qpid-JMS (v 0.26) client library.
 - One of the more popular clients.
 - Maintained by Redhat - part of Apache foundation.
 - Java allowed for portability
 - Clean API

Broker Configuration

- Disable persistence (memory persistence only).
- Minimum authentication.
- Tune memory usage to minimum while ensuring no message loss (in general - use defaults).
- Isolate test network, minimize other processes running on test computers.

Results

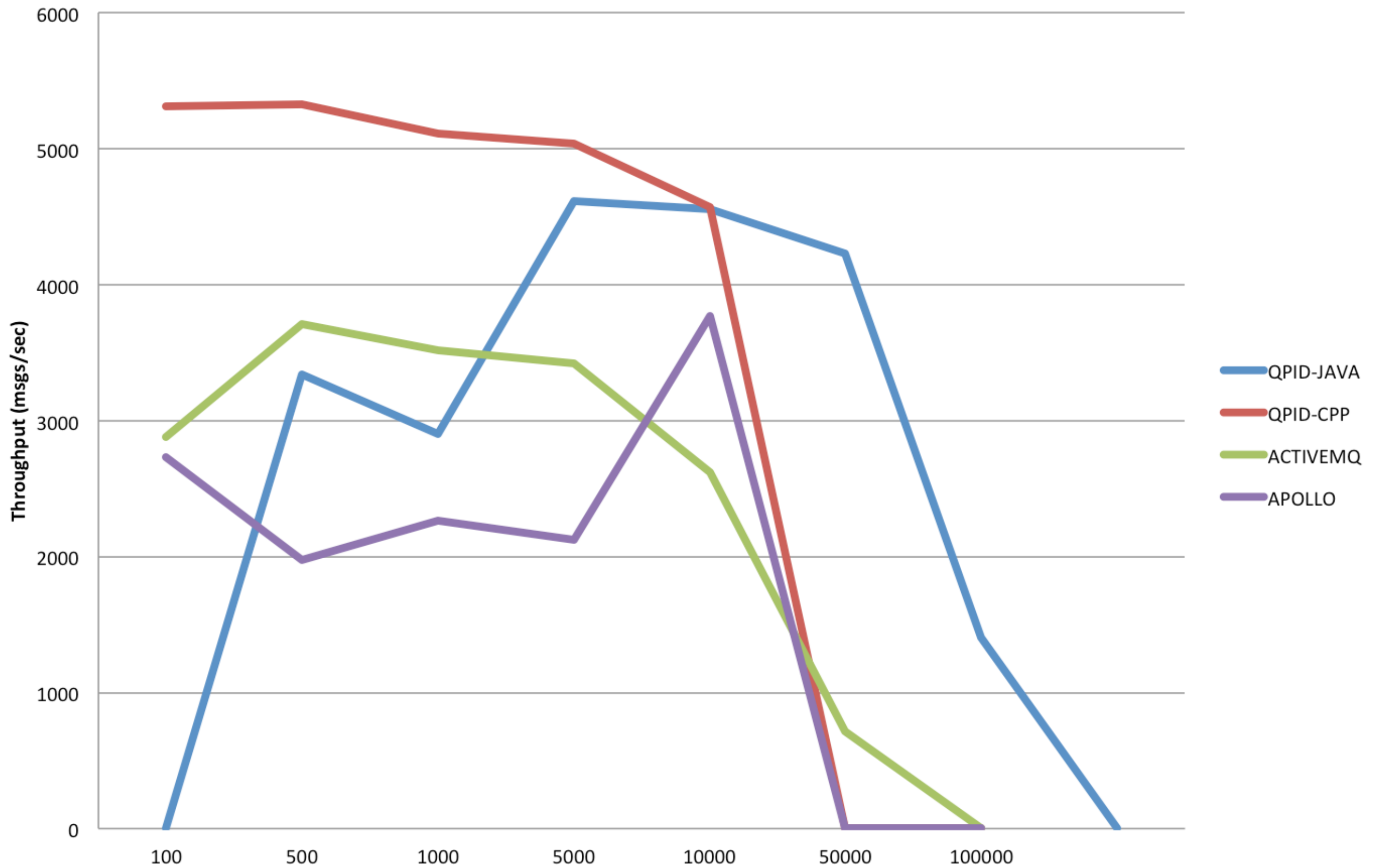
Single broker node results

	100 bytes	1000 bytes	10000 bytes	100000 bytes	1000000 bytes
Apache ActiveMQ	✓	✓	✓	✓	☹
Apache Apollo	✓	✓	✓	☹	☹
Qpid (CPP)	✓	✓	✓	☹	☹
Qpid (Java)	✓	✓	✓	✓	✓
RabbitMQ	N/A	N/A	N/A	N/A	N/A
SwiftMQ	N/A	N/A	N/A	N/A	N/A
HornetQ	☹	☹	☹	☹	☹
StormMQ	N/A	N/A	N/A	N/A	N/A
Microsoft Service Bus	N/A	N/A	N/A	N/A	N/A

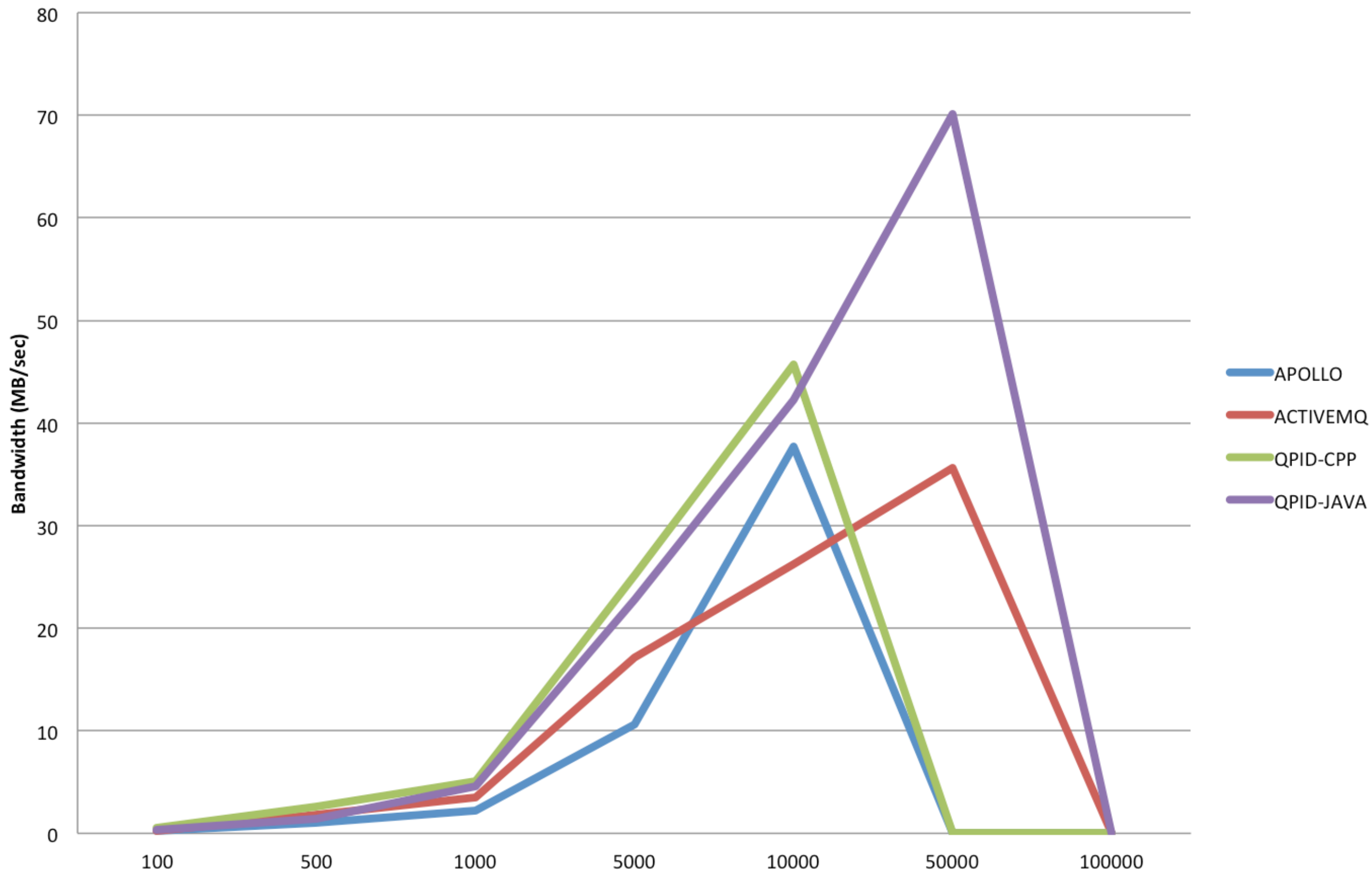
What happened?

- HornetQ never worked. Broker would start, but AMQP clients crashed - suspect is Qpid-JMS client - complaint was about session id in JMS session - not part of AMQP specification.
- RabbitMQ doesn't support AMQP 1.0 natively. It uses a plugin which didn't work.
- SwiftMQ - doesn't allow the publication of benchmark data.
- StormMQ - not possible to instantiate a local broker - totally cloud based.
- Microsoft Service Bus - Doesn't run on Ubuntu.

Comparison of broker throughput



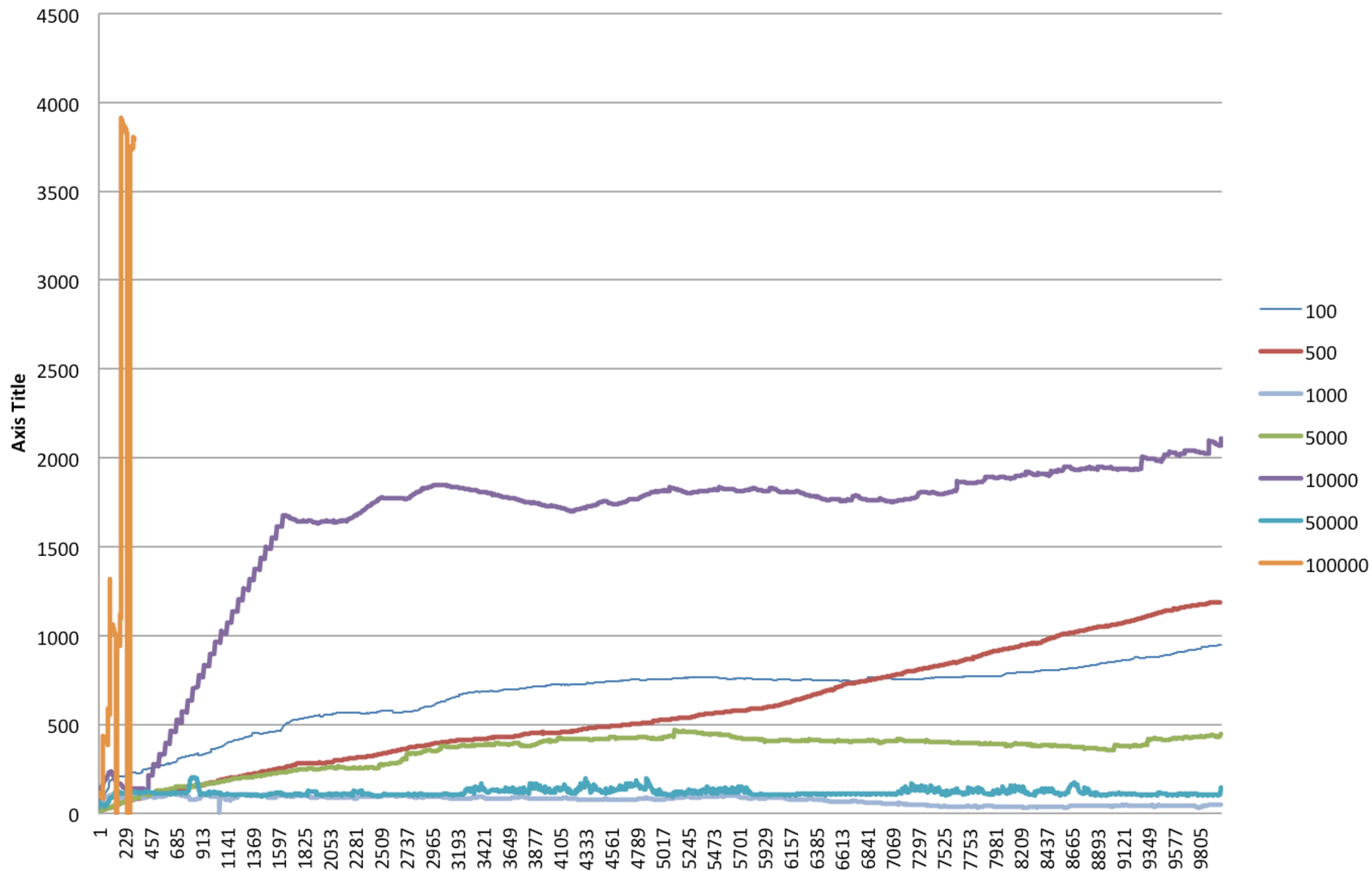
Comparision of broker bandwidth



Results

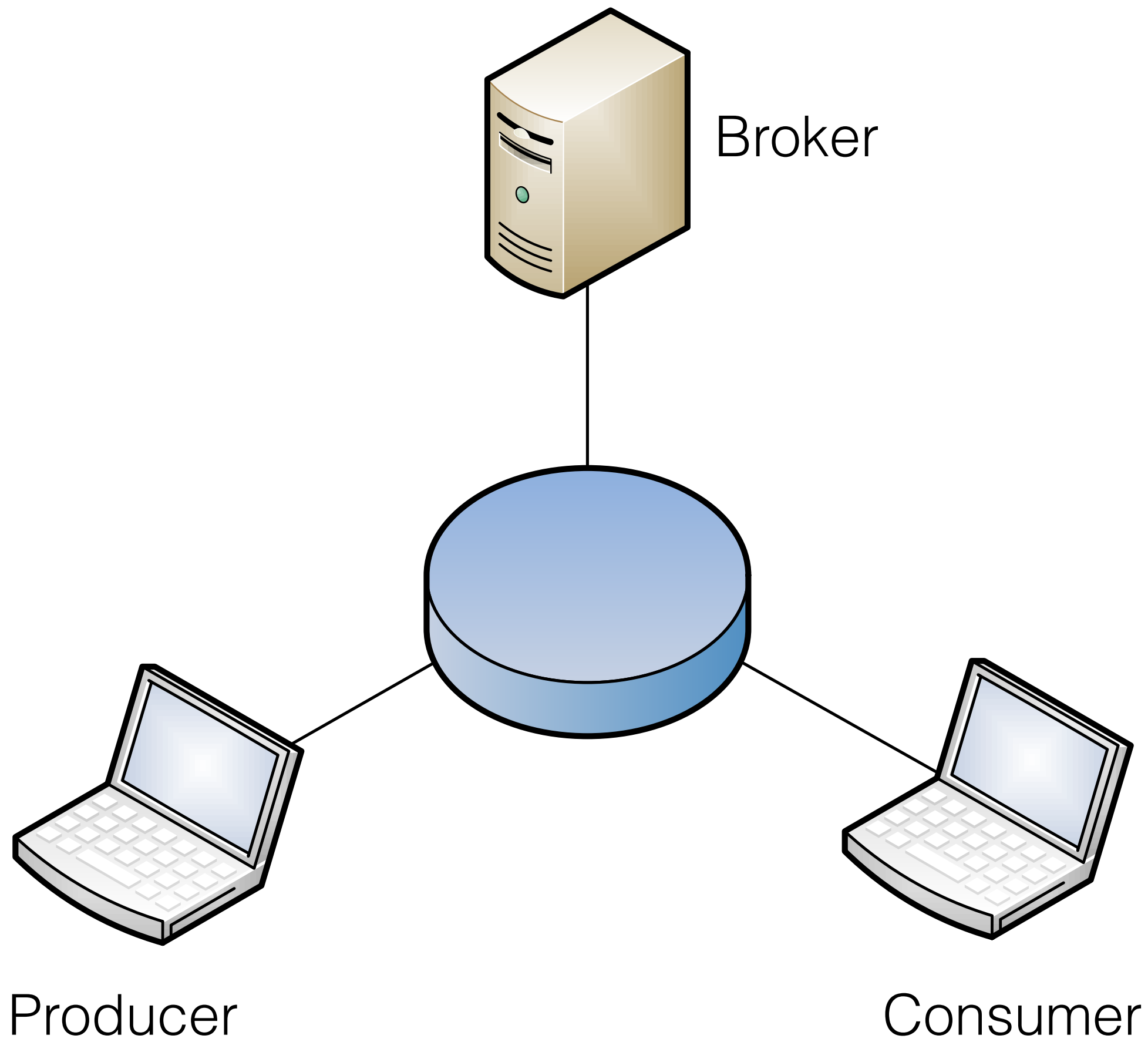
- Bandwidth and Throughput look pretty good and resemble other publicly available results.

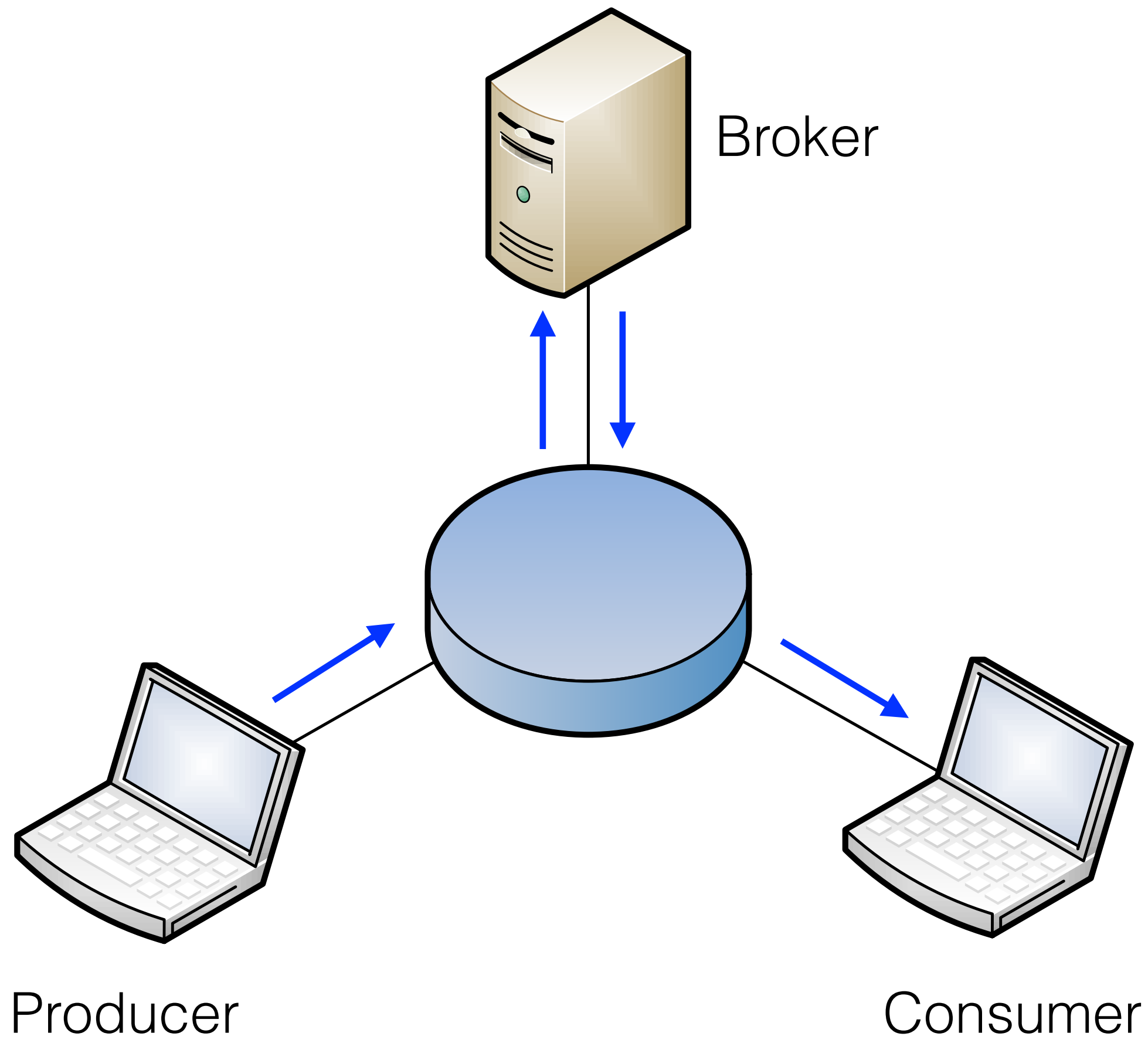
Message latency for ActiveMQ

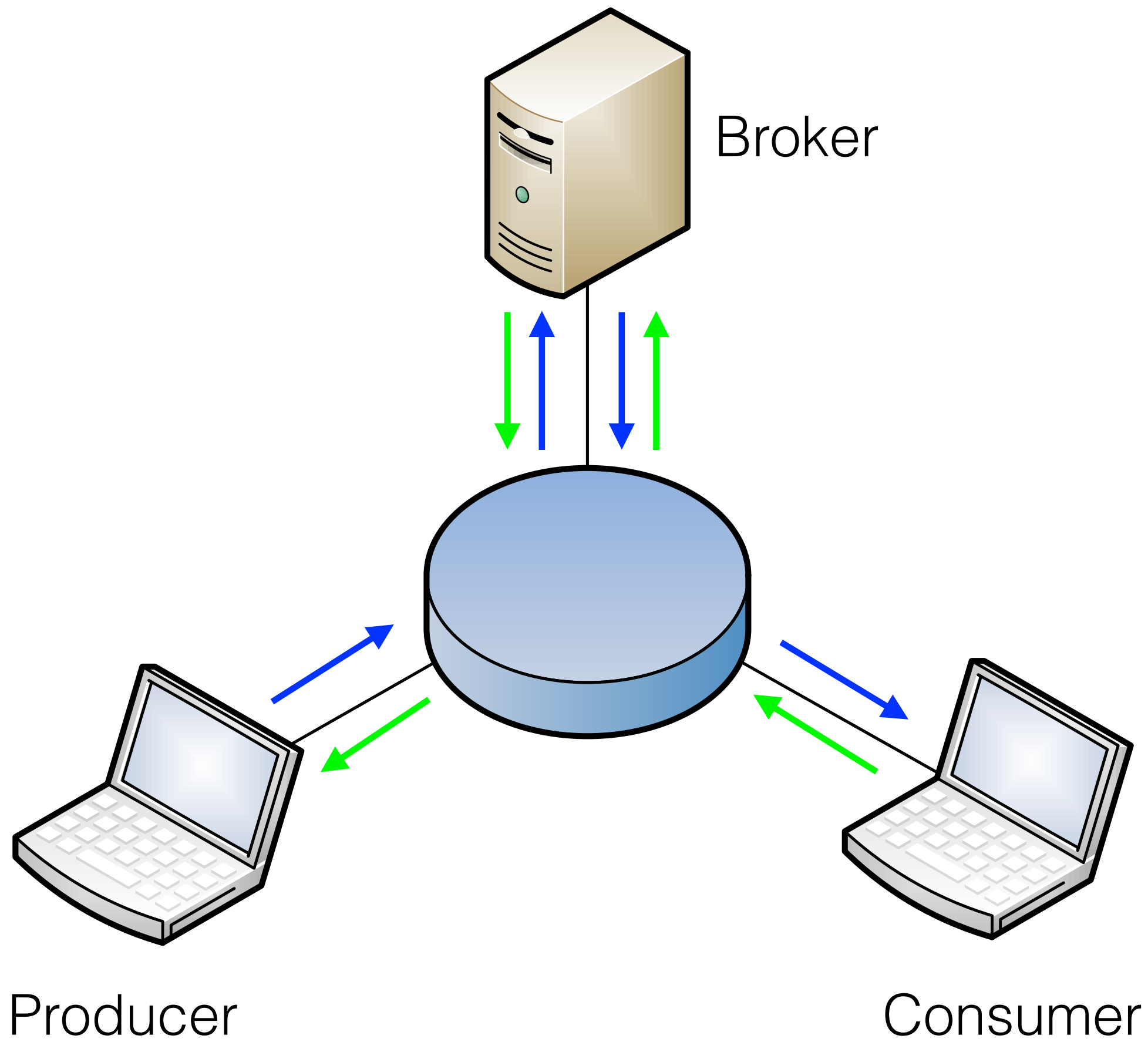


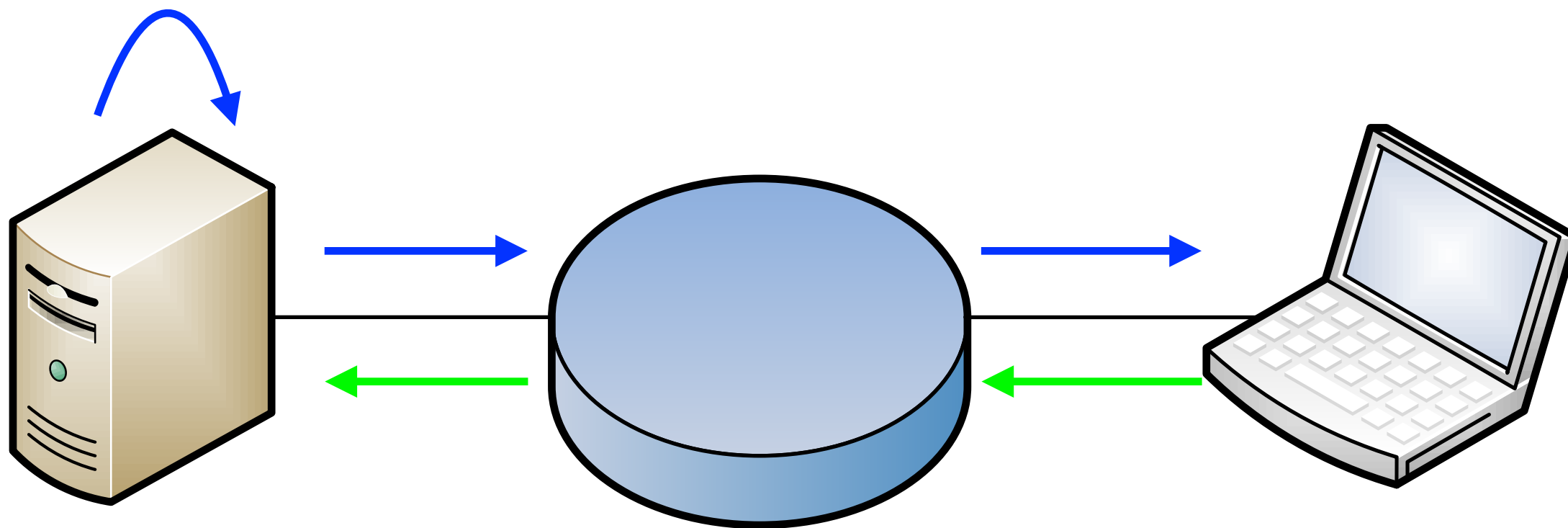
Results - Issues

- A lot going on. JMS client, broker working to maximize performance, not benchmarking accuracy.
- Latency for small messages not accurate.
- Large messages cause the broker/client to fail.
 - Recoverable in some cases, other cases fatal.
- Out of order messages - should not happen with queue exchange.
- Redesign network to simplify measurements and increase accuracy.









Broker + Producer

Consumers

Redesign

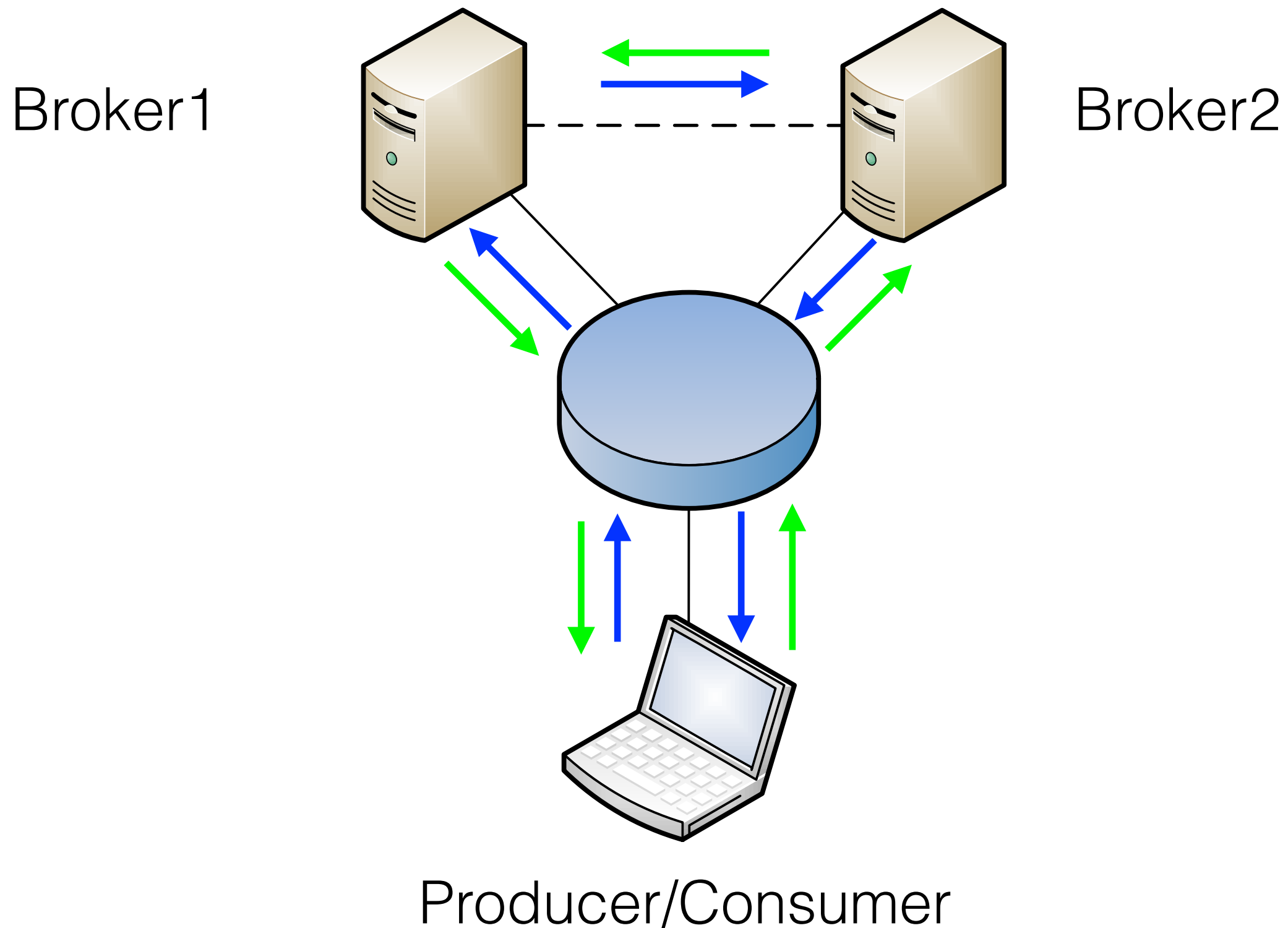
- Software - ended up requiring two separate benchmarks
 - One for pure bandwidth.
 - One for message latency.
- Needed explicit configuration in clients to support tested brokers.

Federation

Federation support

	Version	Supports Federation
Apache ActiveMQ	5.9.0	✓
Apache Apollo	1.7.0	☹
Apache Qpid - CPP	0.26	✓
Apache Qpid - Java	0.26	☹

Federated Architecture

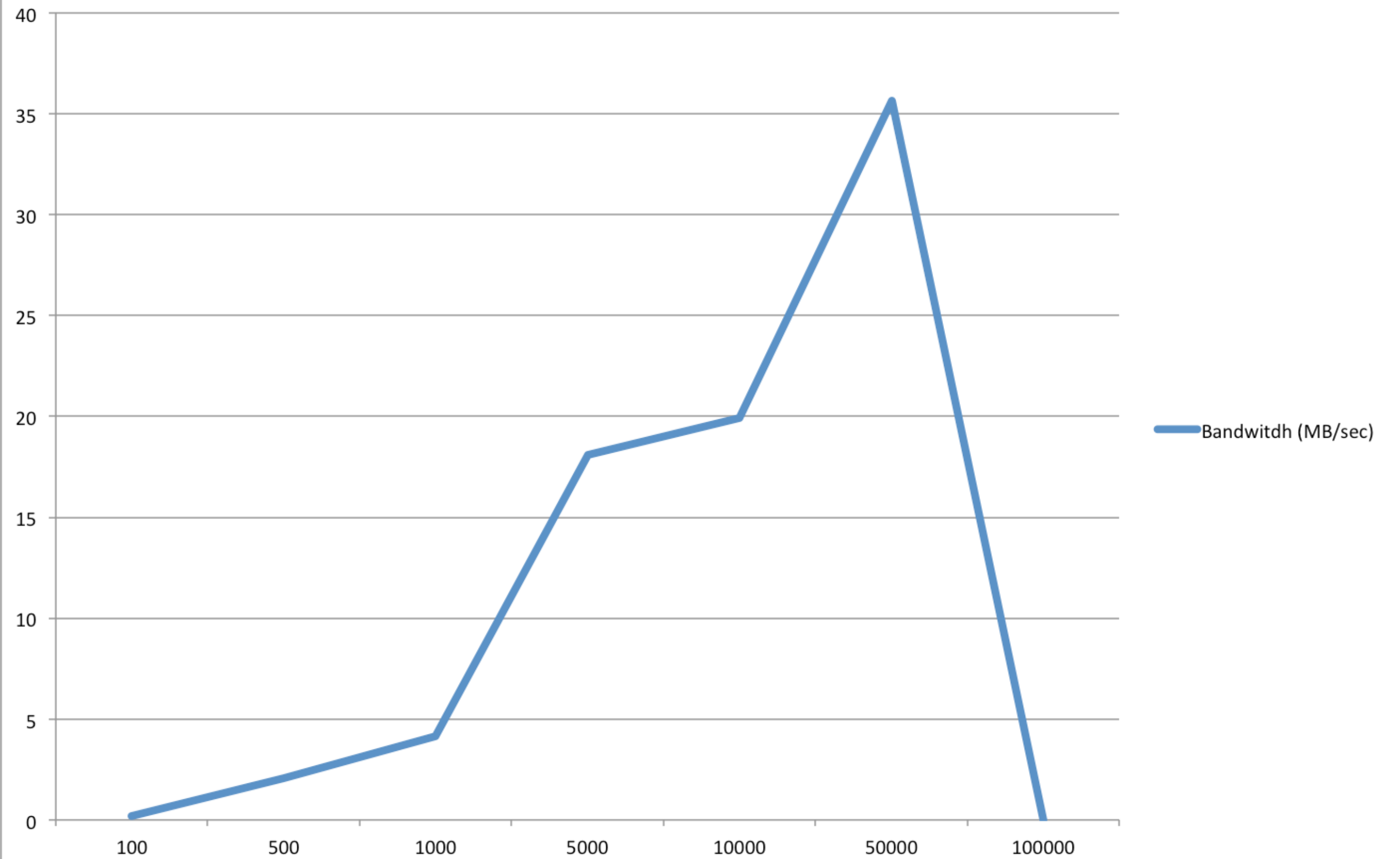


Issues

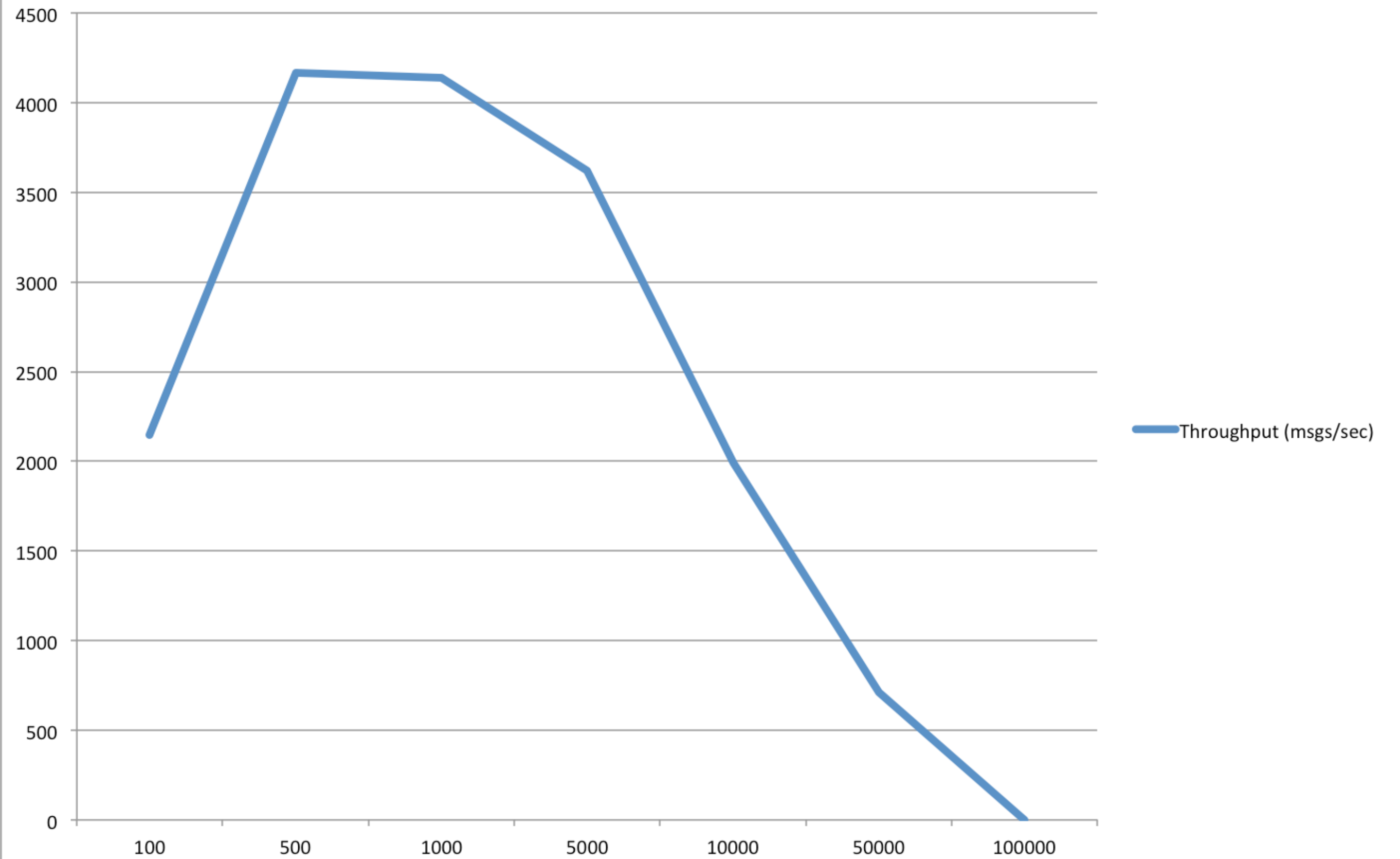
- ActiveMQ worked out of the box
 - **Uses OpenWire not AMQP protocol for federation.**
- Apache Qpid-CPP very difficult to setup - had significant problems with qpid-tools.

ActiveMQ - ActiveMQ

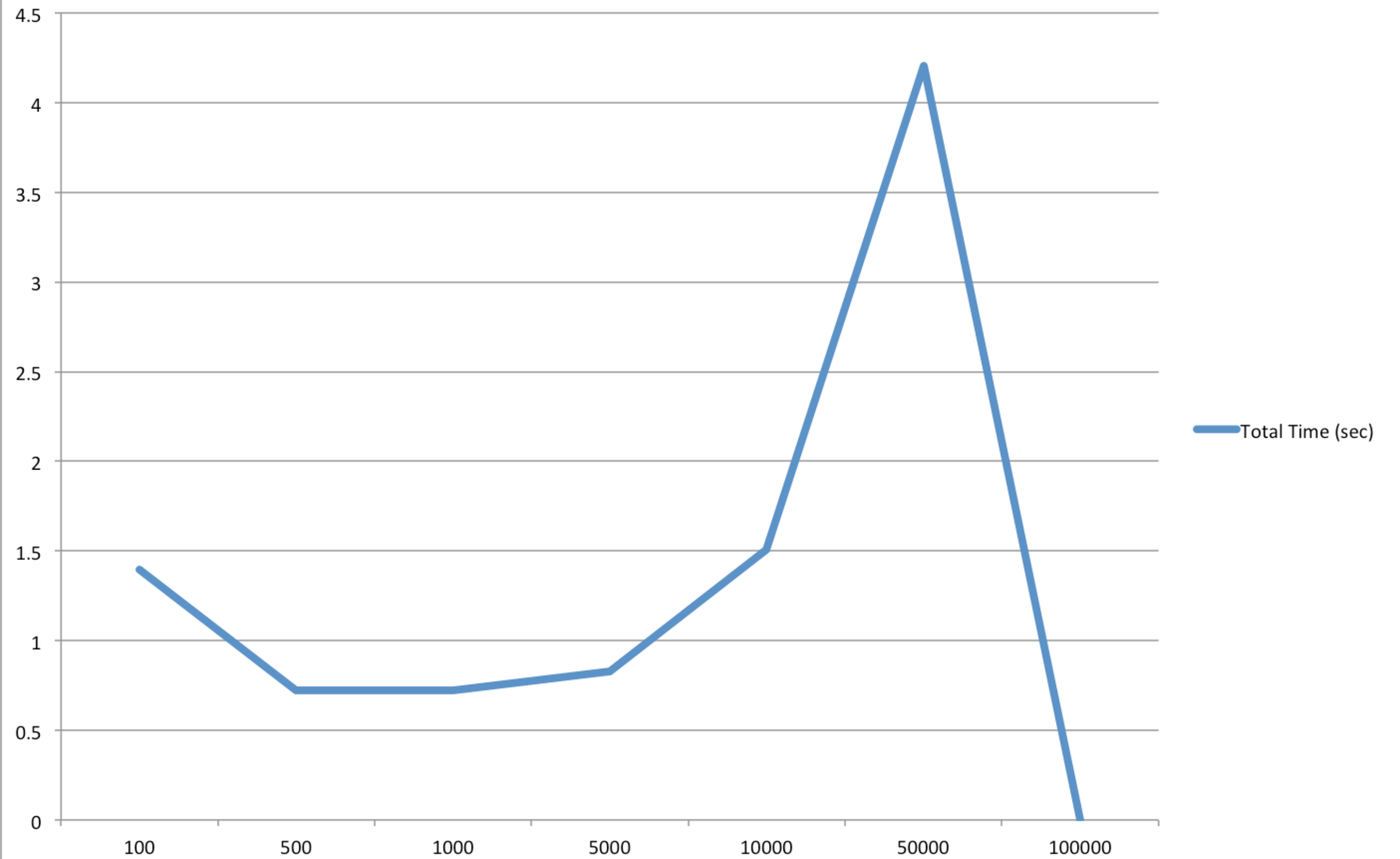
Bandwitdh (MB/sec)



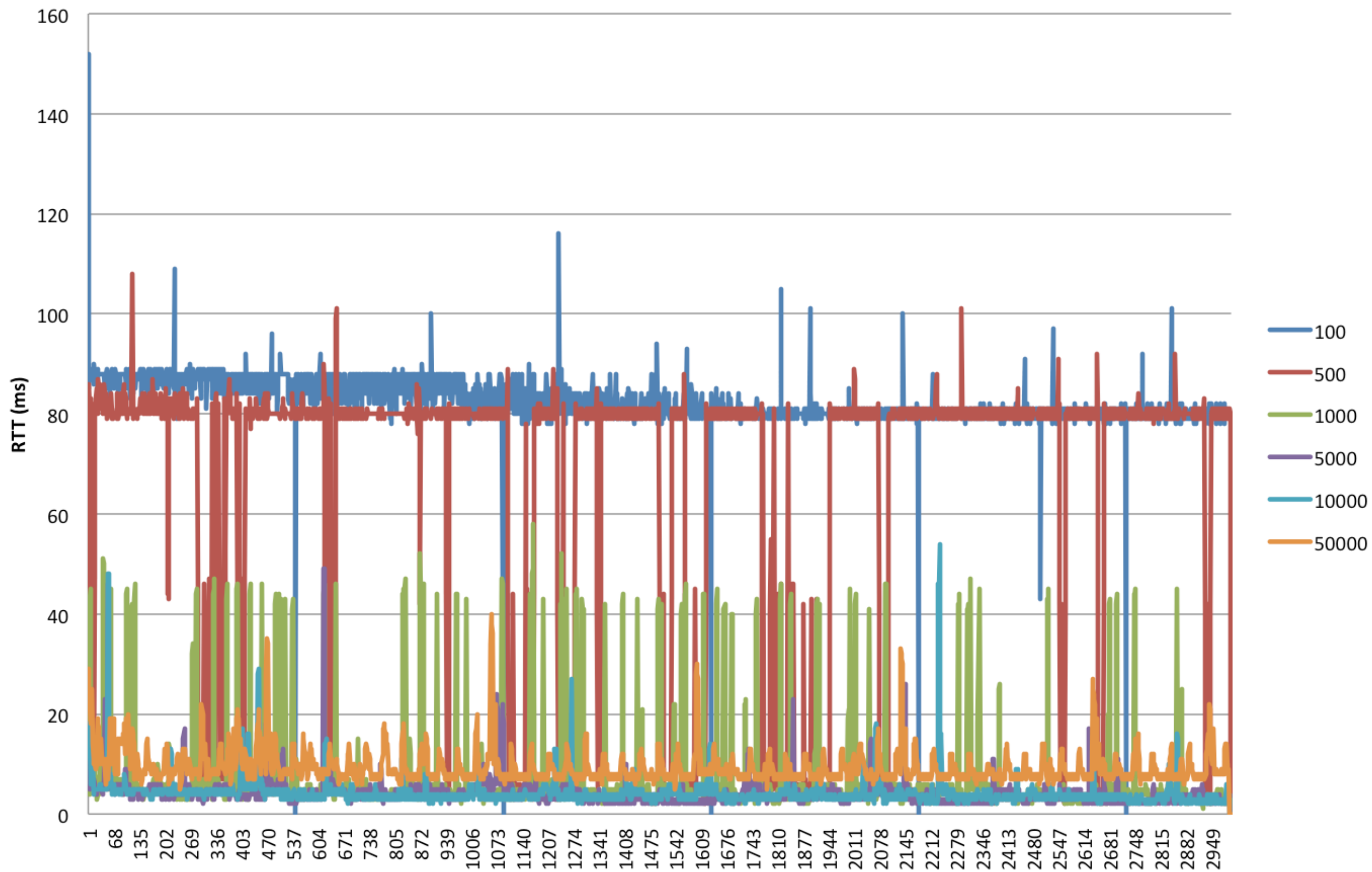
Throughput (msgs/sec)



Total Time (sec)

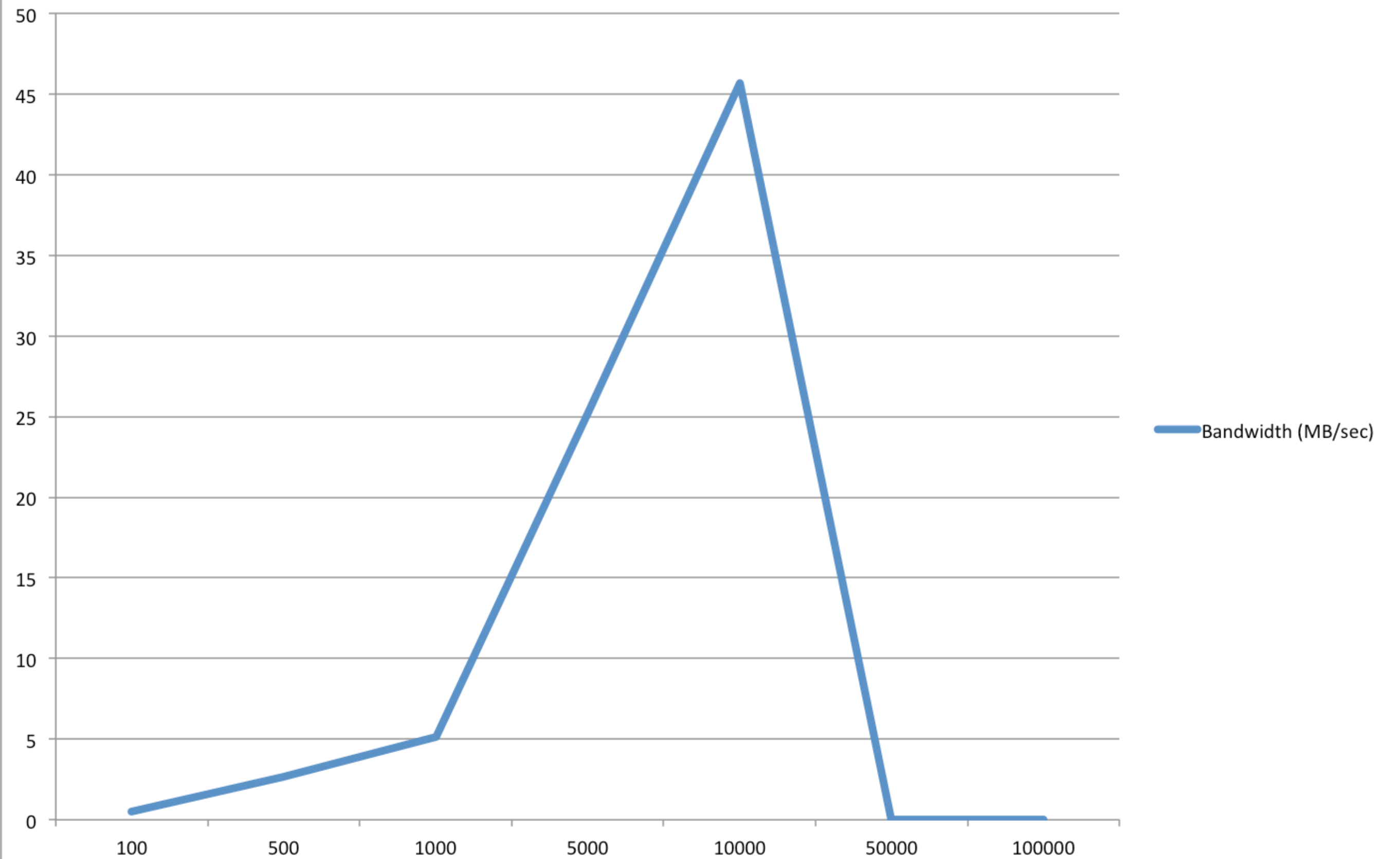


ActiveMQ-ActiveMQ Latency

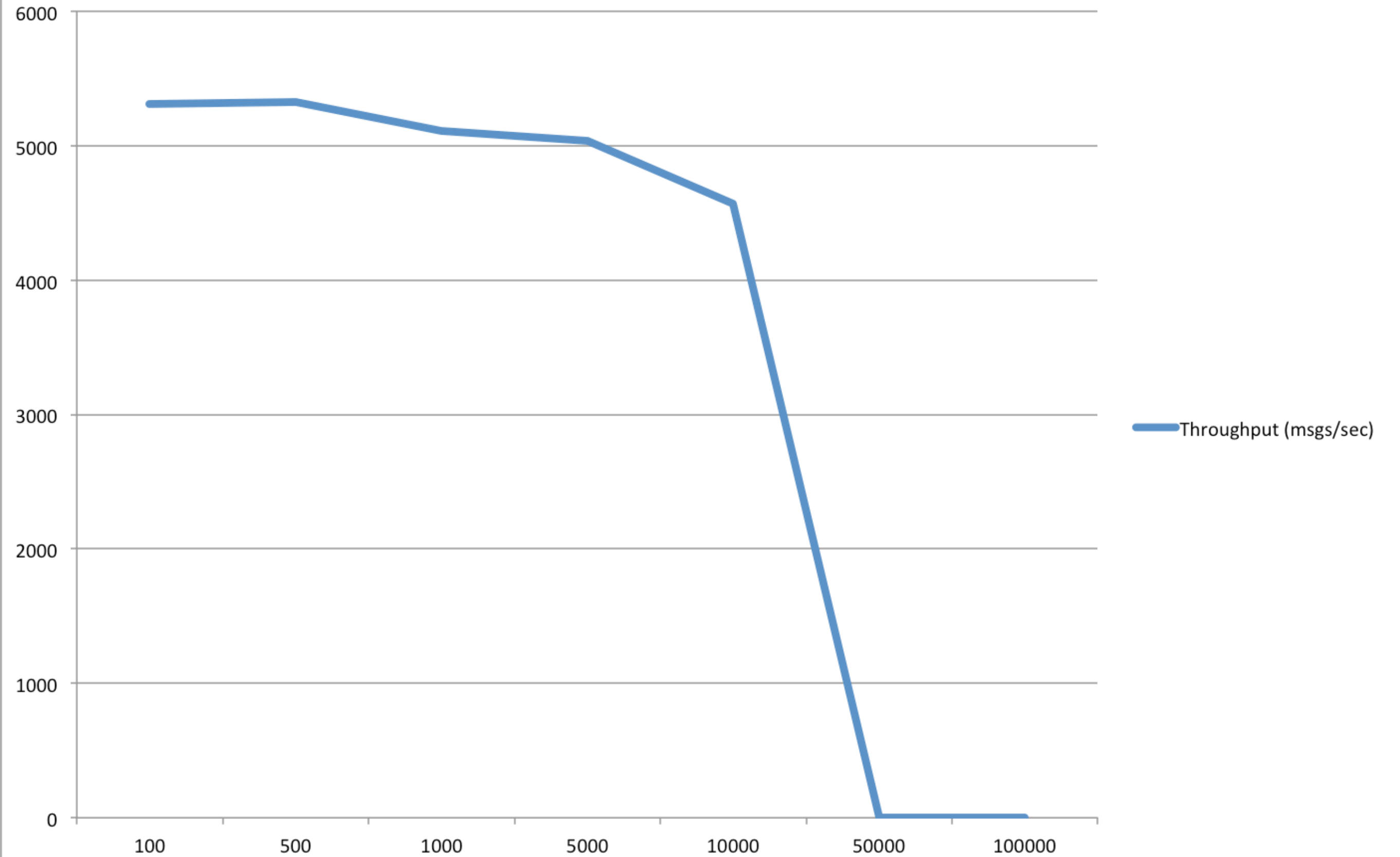


QPID(CPP) - QPID(CPP)

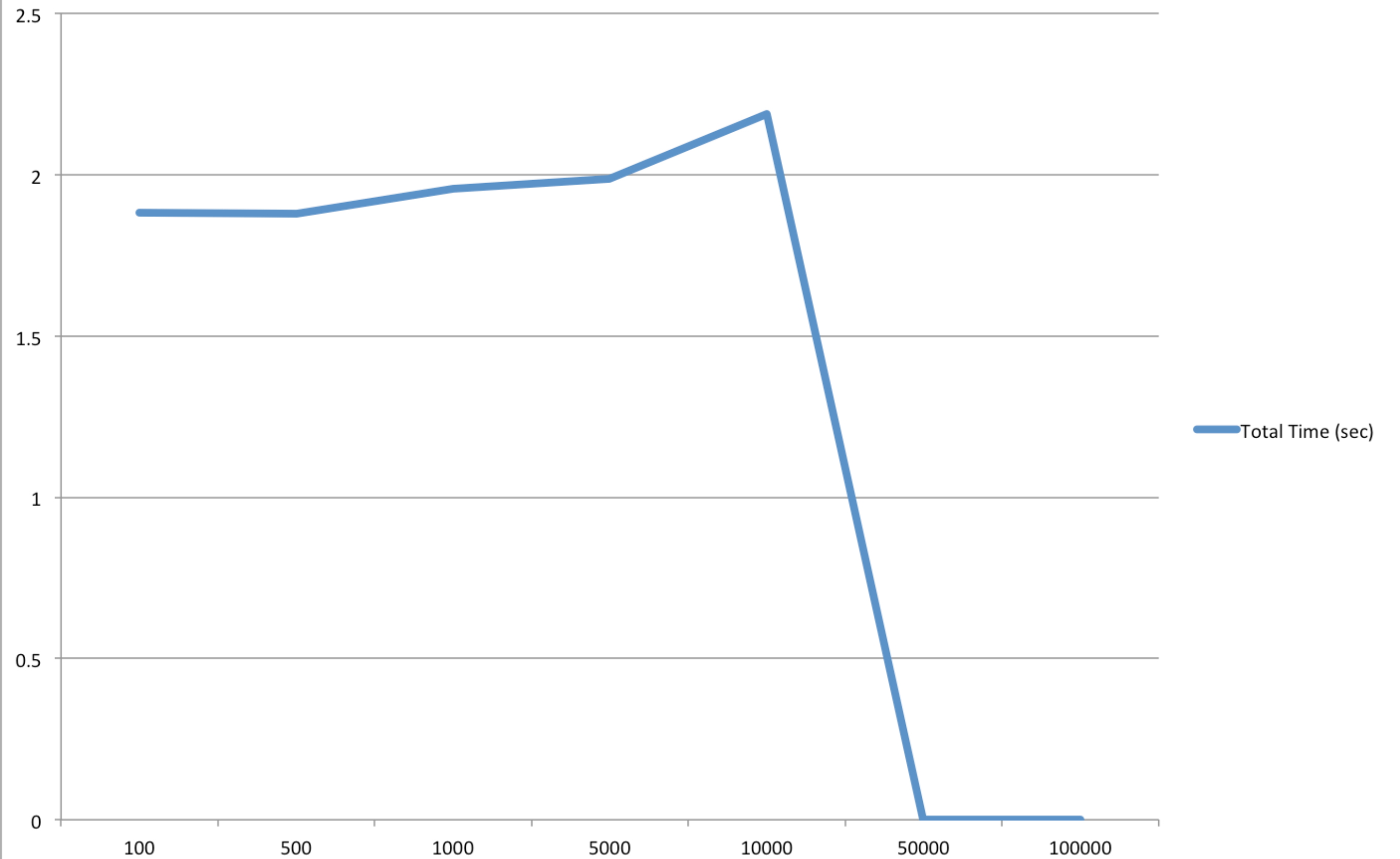
Bandwidth (MB/sec)



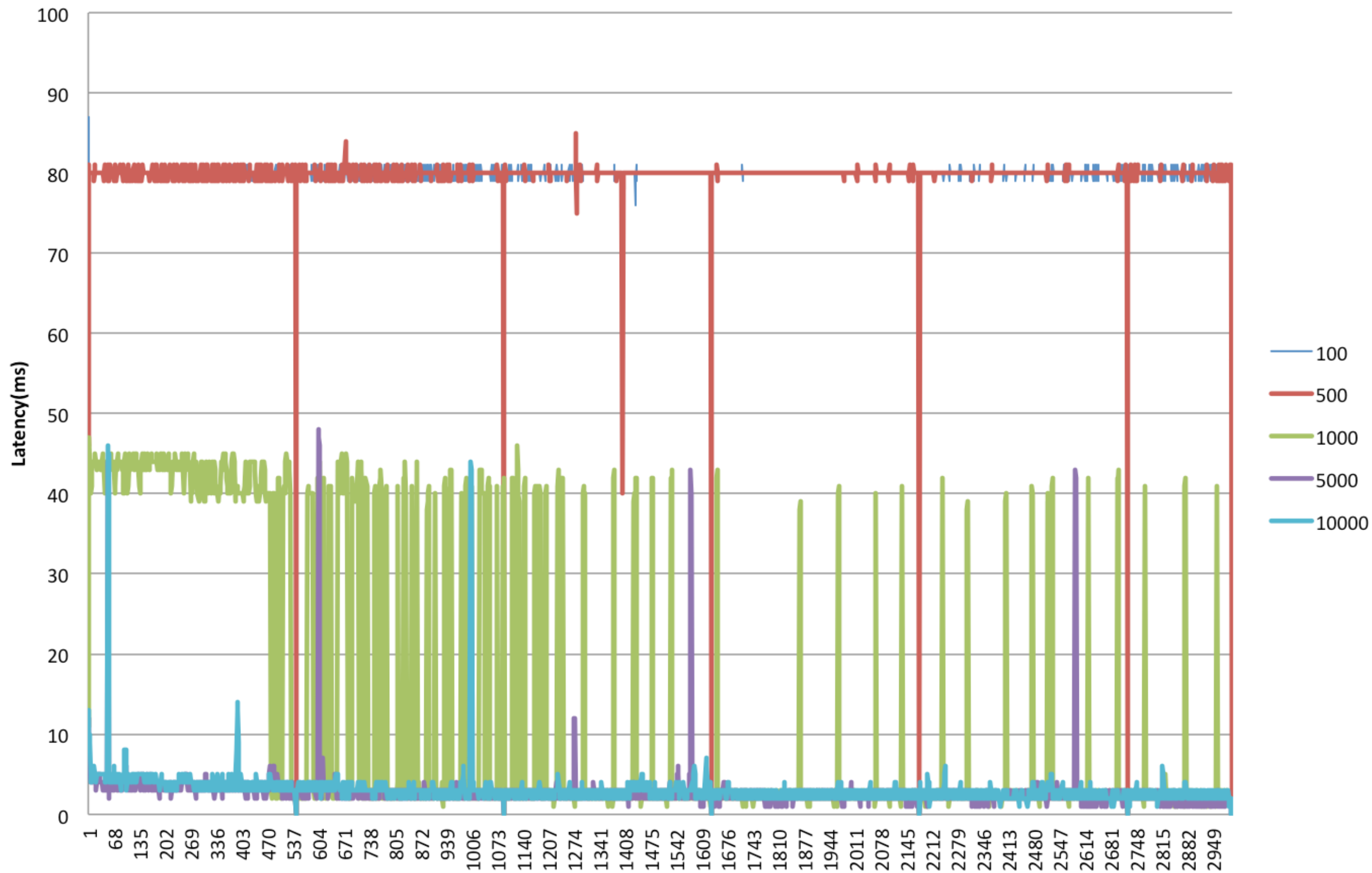
Throughput (msgs/sec)



Total Time (sec)



QPID-QPID



General observations

- All broker implementations had significant weaknesses - a lot less mature than I expected.
- Don't work well with large messages.
 - Hunch is that broker settings are playing a significant part here.
- To get something to work quickly and be forgiving of configuration - use ActiveMQ.
- If you don't care about interoperability with other brokers - use RabbitMQ (really annoyed with this one).
- If you are running a production environment and need interoperability - Apache Qpid-cpp is a good choice.
- AMQP - not an inter-op panacea, actually kind of a mess, was the inspiration for a lot of these message brokers, but is no longer needed - STOMP probably best option for interoperability.

Questions/Comments/ Death Threats?

sorry it's so late...