Initial Results from an early prototype of the Grid Event Service

Shrideep Pallickara
Department of Electrical Engineering &Computer Science
Syracuse University

Geoffrey C. Fox
Department of Computer Science
Indiana University

Abstract

The Grid Event Service (GES) is a distributed event service designed to run on a very large network of server nodes. Clients interested in using this service can attach themselves to one of the server nodes. Clients specify an interest in the type of events that they are interested in and the service routes events, which satisfy the constraints specified by the clients. Clients can have prolonged disconnects from the server network and can also roam the network (in response to failure suspicions or for better response times) and attach themselves to any other node in the server node network. Events published during the intervening period, of prolonged disconnects and roams, must still be delivered to clients that originally had an interest in these events. The delivery constraints must be satisfied even in the presence of server failures. Server nodes can fail and remain failed forever. Clients need not wait for the failed server nodes to recover. Affected clients can then roam to a new location and thus not experience any denial of service. GES provides a hierarchical dissemination scheme for the delivery of events to relevant clients. The system provides for an efficient calculation of routes to reach relevant destinations. This paper presents the results obtained in an early prototype of GES.

1 Introduction

In this section we present results pertaining to the performance of the Grid Event Service (GES) protocols. We first proceed with outlining our experimental setups. We use two different topologies with different clustering coefficients. The factors that we measure include latencies in the delivery of events, variance in the latencies and system throughputs among others. We measure these factors under varying publish rates, event sizes, event disseminations and system connectivity. We intend to highlight the benefits of our routing protocols and how these protocols perform under the varying system conditions, which were listed earlier.

2 Experimental Setup

The system comprises of 22 broker node processes organized into the topology shown in the Figure 1. This set up is used so that the effects of queuing delays at higher publish rates, event sizes and matching rates are magnified.

Each broker node process is hosted on 1 physical Sun SPARC Ultra-5 machine (128 MB RAM, 333 MHz), with no SPARC Ultra-5 machine hosting two or more broker node processes. For the purpose of gathering performance numbers we have one publisher in the system and one *measuring subscriber* (the client where we do our measurements). The publisher and the *measuring* subscriber reside on the same SPARC Ultra-5 machine and are attached to nodes **22** and **10** respectively in the topology outlined in figure 1. In addition to this there are 100 subscribing client processes, with 5 client processes attached to every other broker node (nodes **22** and **10** do not have any other clients besides the publisher and measuring subscriber respectively) within the system. The 100 client node processes all reside on a SPARC Ultra-60 (512 MB RAM, 360 MHz) machine. The publisher is responsible for issuing events, while the subscribers are responsible for registering their interest in receiving events. The run-time environment for all the broker node and client processes is Solaris JVM (JDK 1.2.1, native threads, JIT).

3 Factors to be measured

Once the publisher starts issuing events the factor that we are most interested in is the *latency* in the reception of events. This latency corresponds to the response times experienced at each of the clients. We measure the latencies at the client under varying conditions of *publish rates*, *event sizes* and *matching rates*. Publish rate corresponds to the rate at which events are being issued by the publisher. Event size corresponds to the size of the individual events being published by the publisher. Matching rate is the percentage of events that are actually supposed to be receieved at a client. In most publish subscribe systems, at any given time for a certain number of events being present in the system, any given client is generally interested in a very small subset of these events. Varying the matching rates allows us to simulate such a scenario, and perform measurements under conditions of varying selectivity. For a sample of events received at a client we calculate the *mean latency* for the sample of received events, the *variance* in the sample of these events and the *system throughput* measured in terms of the number of events received per second at the measuring subscriber. We also measure the highest and lowest event latencies within the sample of events that have been received. Another very important factor that needs to be measured is the change in latencies as the connectivity between the nodes in a broker network is increased. This increase in connectivity has the effect of reducing the number of broker hops that an event has to take prior to being received at a client. The effects of change in latencies with decreasing broker hops is discussed in section 4.5.

3.1 Measuring the factors

For events published by the publisher the number of tag-value pairs contained in every event is 6, with the matching being determined by varying the value contained in the fourth tag. The profile

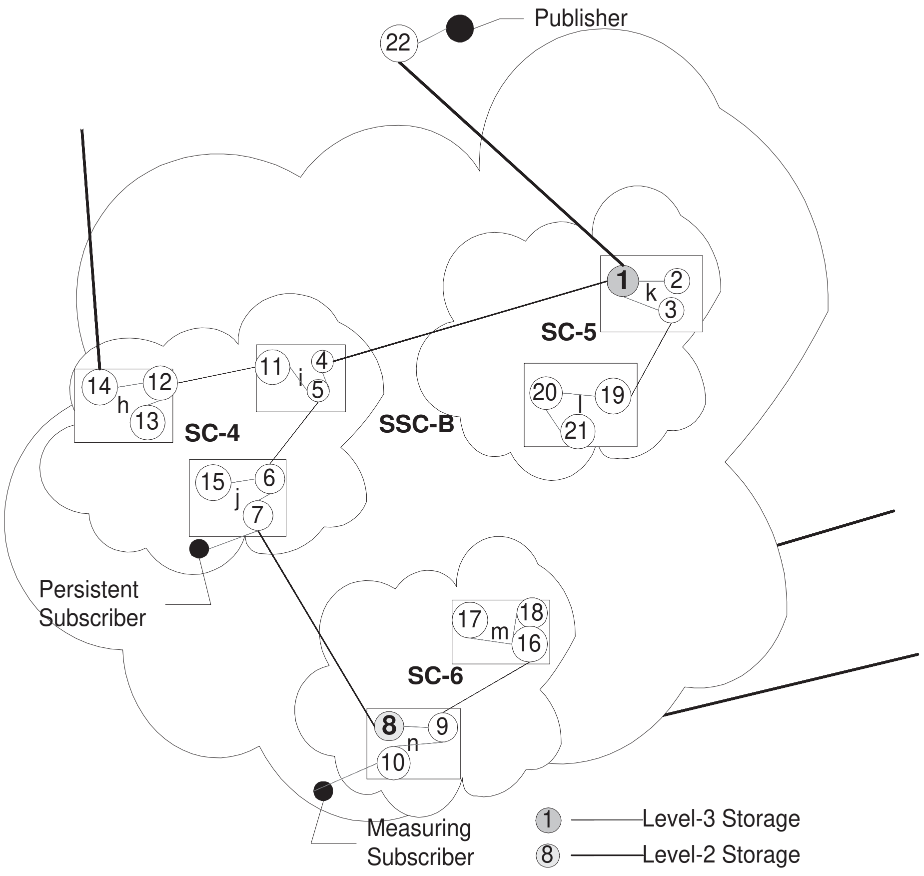

Figure 1: Testing Topology - (I)

for all the clients in the system, thus have their first 3 <*tag=value*> pairs identical to the first 3 pairs contained in every published event. This scheme also ensures that for every event for which destinations are being computed there is some amount of processing being done. Clients attached to different broker nodes specify an interest in the type of events that they are interested in. This matching rate is controlled by the publisher, which publishes events with different footprints. Since we are aware of the footprints for the events published by the publisher, we can accordingly specify profiles, which will allow us to control the dissemination within the system. When we vary the matching rate we are varying the percentage of events published by the publisher that are actually being received by clients within the system. Thus, when we say that the matching rate is set at 50%, any given subscribing client within the system will receive only 50% of the events published by the publisher. To vary the publish rates, we control the *sleep time* associated with the publisher thread, and also the number of events that it publishes at a time, once the publisher thread *wakes up*. This requires some preliminary tuning. Once the values for the *sleep time* and the number of events that are published at a time have been fixed (for the publisher and the broker node that it is attached to), we proceed to compute the real publish rates for the sample of events that we send. This is the publish rate that we report in our results.

For each matching rate we vary the size of the events from 30 to 500 bytes, and vary the publish rates at the publisher from 1 Event/Sec to around 1000 Events/second. For each of these cases we measure the latencies in the reception of events. To compute latencies we have the publishing client and the *measuring* subscriber residing on the same machine. Events issued by the publisher are *timestamped* and when they are received at the subscribing client the difference between the *present* *time* and the *timestamp* contained in the received event constitutes the latency in the dissemination of the event at the subscriber via the broker network. In case the publisher and the subscriber are on two different machines, with access to different underlying system clocks, we would need to synchronize the clocks and also account for the drift in clock rates prior to computing the latencies in event reception. Having the publisher and one of the subscribers on the same physical machine with access to the same underlying clock, obviates this need for clock synchronization and also accounts for clock drifts. It should be noted that though the publisher and the *measuring* subscriber are on the same machine, they are connected to two different broker nodes within the broker network, as depicted in figure 1. In fact it takes 9 broker hops for an event issued by the publisher to be received at the measuring subscriber.

4 Discussion of Results

In this section we discuss the latencies gathered for varying values of publish rates, event sizes and matching rates. We then proceed to include a small discussion on system throughputs at the clients. We also discuss the trends in the variance of the latencies, associated with the sample of events received at a client. The results also discuss the latencies involved in the delivery of events to persistent clients in units with different replication schemes.

4.1 Latencies for the routing of events to clients

At high publish rates and increasing event sizes, the effects of queuing delays come into the picture. This queuing delay is a result of the events being added to the queue faster than they can be processed. In general, the mean latency associated with the delivery of events to a client is directly proportional to the size of the events and the rate at which these events were published. The latencies are the lowest for smaller events issued at low publish rates. The mean latency is further influenced by the matching rates for events issued by the publisher. The results clearly demonstrate the effects of flooding/queuing that take place at high publish rates and high event sizes and high matching rates at a client. It is clear that as the matching rate reduces the latencies involved also reduce, this effect is more pronounced for cases involving events of a larger size at higher publish rates.

Figures 2 and 3, depict the pattern of decreasing latencies with decreasing matching rates. Table 1 outlines the reduction in latencies and the variance associated with the latencies corresponding to the sample of events received at a client. This reduction in the latencies for decreasing matching rates, is a result of the routing algorithms that we have in place. These routing algorithms ensure that events are routed only to those parts of the system where there are clients, which are interested in the receipt of those events. The routing algorithms are very selective about the links that are employed for event dissemination. Thus, events are queued only at those broker nodes which –

- Have attached clients interested in those events
- Are en route to broker nodes which are interested in these events. These broker nodes generally fall in the shortest path to reach the destination node.

In the flooding approach, all events would still have been routed to all clients irrespective of the matching rates.

4.2 System Throughput

We also depict the system throughputs at the client under conditions of varying event sizes and publish rates. We choose to depict the system throughputs at a matching rate of 100%. At matching rates other than 100% only the relevant events are being routed to the clients. The events received do not reveal the true throughputs that can be achieved at a client. Figure 4 depicts the system throughputs achieved at a client under conditions of different publish rates and event sizes. The maximum throughput achieved was 480.76 *events/Sec* at a publish rate of 492 *events/Sec* with the sample of events being of size 75 bytes.

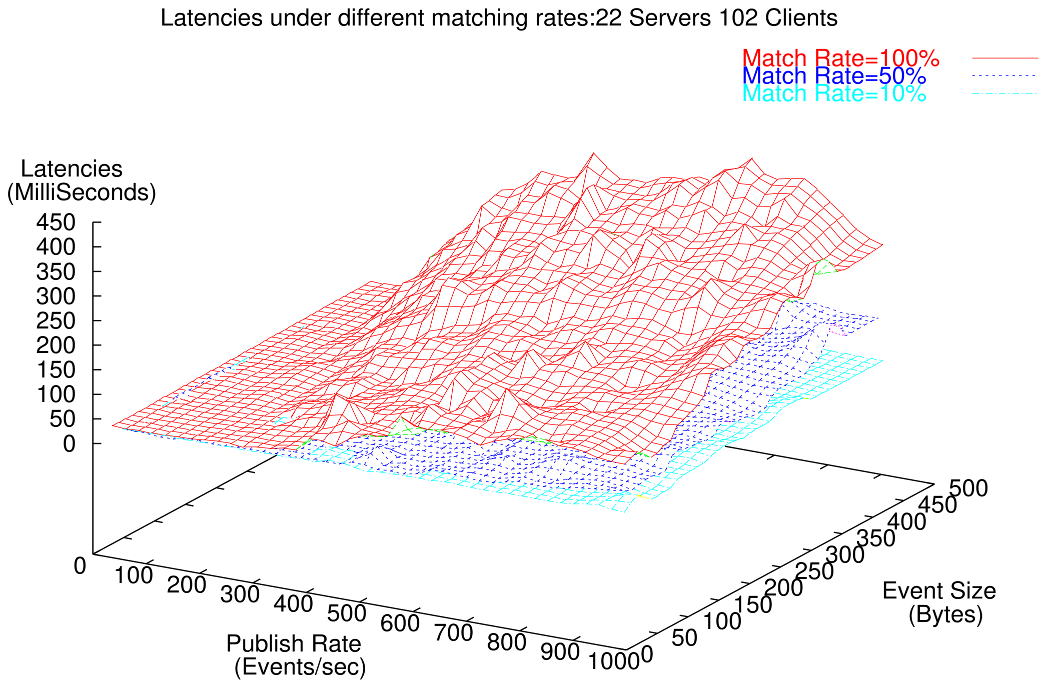

Figure 2: Latencies for match rates of 100%, 50% and 10%

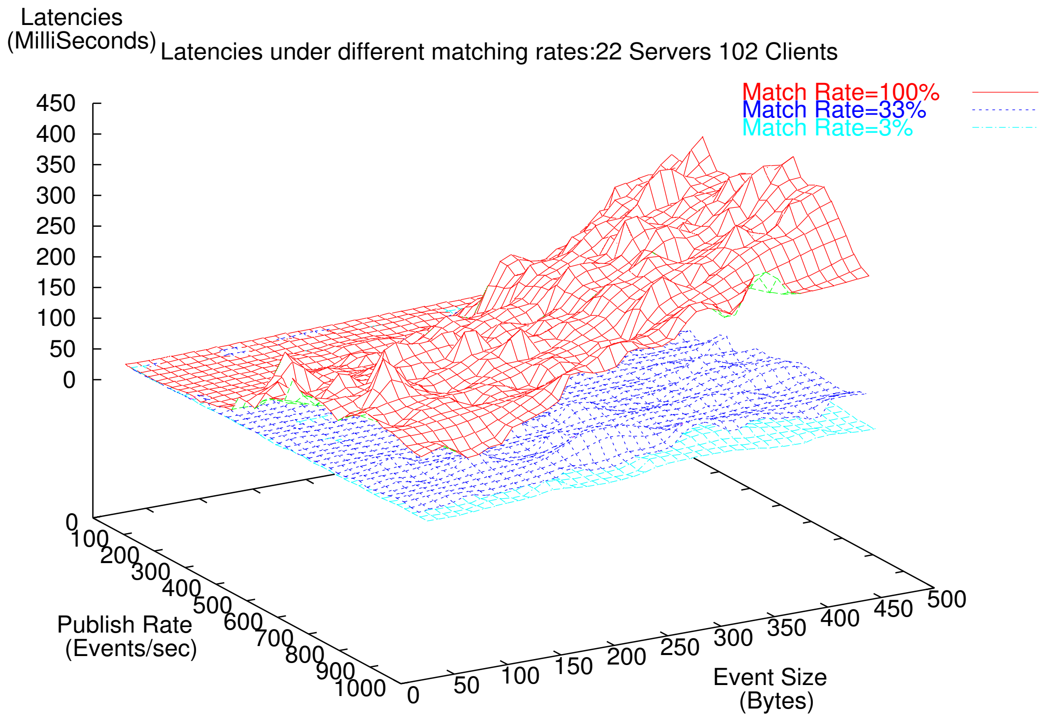

Figure 3: Latencies for match rates of 100%, 33% and 3%

Match Rate	Latency Range @ < publish Rate, event Size >		Variance Range	
	Latency $\mathcal{L}_a$	Latency $\mathcal{L}_a$	Variance $\mathcal{V}_a$ @ $\mathcal{L}_a$	Variance $\mathcal{V}_b$ @ $\mathcal{L}_b$
100%	15.54 $mSec$ @ $<1evt/Sec, 50B>$	391.85 $mSec$ @ $<952evts/Sec, 450B>$	2.3684 $mSec^2$	69,713.93 $mSec^2$
50%	13.02 $mSec$ @ $<20evts/Sec, 50B>$	178.66 $mSec$ @ $<952evts/Sec, 350B>$	56.8196 $mSec^2$	14,634 $mSec^2$
33%	15.18 $mSec$ @ $<20evts/Sec, 50B>$	95.969 $mSec$ @ $<952evts/Sec, 425\ B>$	26.57 $mSec^2$	1,263 $mSec^2$
25%	14.40 $mSec$ @ $<20evts/Sec, 50B>$	66.6 $mSec$ @ $<961evts/Sec, 400B>$	0.24 $mSec^2$	587.04 $mSec^2$
20%	15.35 $mSec$ @ $<20evts/Sec, 50B>$	62.35 $mSec$ $<952evts/Sec, 400B>$	12.02 $mSec^2$	312 $mSec^2$
10%	14.40 $mSec$ @ $<20\ evts/Sec, 50B>$	52.0 $mSec$ @ $<952evts/Sec, 400B>$	0.44 $mSec^2$	103 $mSec^2$
5 %	14.0 $mSec$ @ $<20evts/Sec, 50B>$	47.6 $mSec$ @ $<961evts/Sec, 425B>$	0.44 $mSec^2$	87.44 $mSec^2$

Table 1: Latency and Variance Range Table

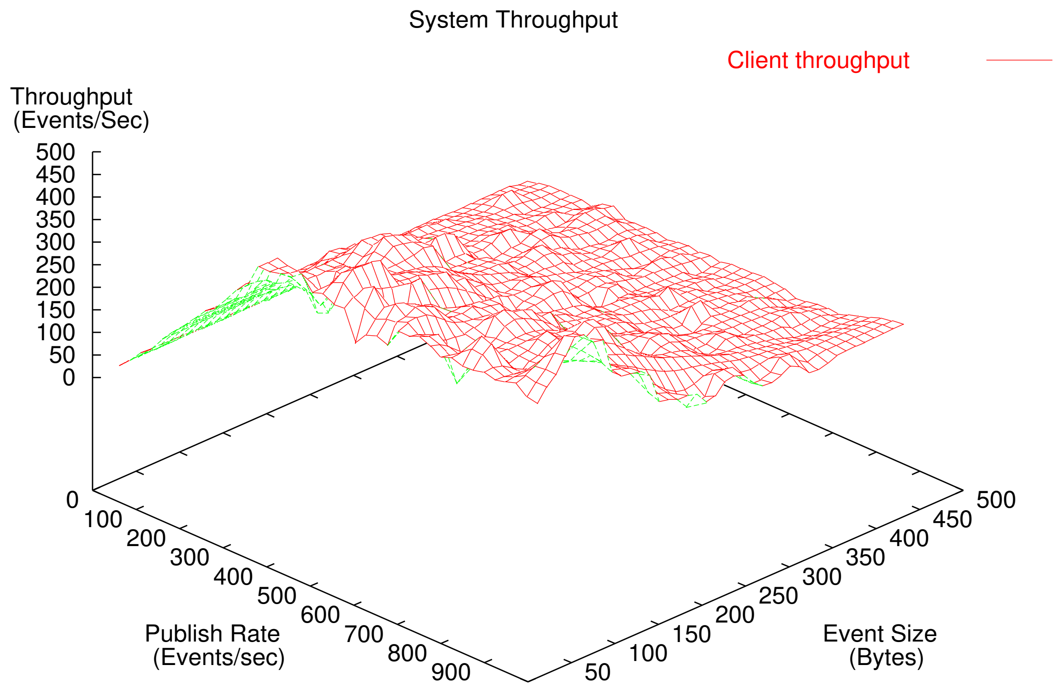

Figure 4: System Throughput

4.3 Variance

Variance for the sample of received events at a client, demonstrate how queueing delays can add up to increase the mean latency. Variance also snapshots how this mean latency has high deviations from the highest and lowest latencies contained in the sample of latencies, associated with the events that are received at a client. The variance in the sample of events varies from 69713 $mSec^2$ to 133.76 $mSec^2$ for <952 *events/Sec* , 450 *Bytes*> at matching rates of 100% to <877 *events/Sec*, 450 *Bytes*> at matching rates of 5%. Thus variance in the sample of events for higher event sizes at higher publish rates also reduces with decreasing matching rates for the published events.

4.4 Persistent Clients

In figure 1 we have also outlined the replication scheme that exists in the system. When an event arrives at node **1**, the event is first stored to the level-3 stable store so that it has an epoch associated with it. The event is then forwarded for dissemination within the unit. Clients attached to any of the nodes in super-cluster **SC-6** have a replication granularity of r_2. When the events issued by the publisher in the test topology of figure 1 are being disseminated, when clients attached to nodes in **SC-6** receive the event, that event would have been replicated twice (once at node **1** and once at node **8**). For testing purposes we set up another *measuring* subscriber at node **7** in addition to the subscriber that we would set up at node **10**. When an event is received by the subscriber attached to node **7** the event would have been replicated only once, at node **1**. These *measuring* subscribers allow us to measure the response times involved for singular and double replications experienced at clients attached to nodes **7** and **10** respectively. Every node (with the exception of nodes **7** and **10**) in the system has 5 persistent subscribing clients attached to it, for a total of 102 persistent subscribing clients. The publisher and the 2 *measuring* subscribers are all hosted on the same machine for reasons discussed earlier. Figure 5 depicts the latencies in delivery of events at persistent clients, with singular and double replications for a matching rate of 50%.

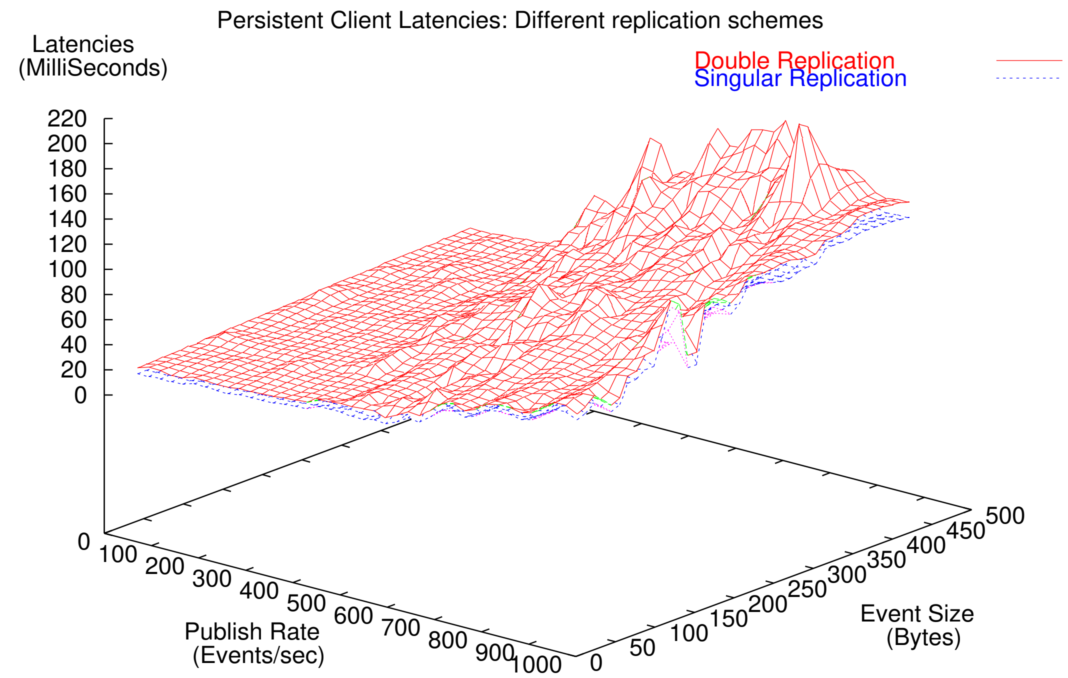

Figure 5: Match Rates of 50% - Persistent Clients (singular and double replication)

4.5 Pathlengths and Latencies

The topology in figure 1 allows us to magnify the latencies, which occur by having the queuing delays at individual broker hops add up. In that topology the number of broker hops taken by an event prior to delivery at the measuring subscriber is 9. We now proceed with testing for the topology outlined in figure 6. The layout of the broker nodes is essentially identical to the earlier one, with the addition of links between nodes resulting in a strongly connected network. We have 5 subscribing clients at each of the broker nodes. The mapping of broker nodes and subscribing client nodes to the physical machines is also identical to the earlier topology. As can be seen the addition of super-cluster link between super-clusters **SC-5** and **SC-6**, and level-0 links between nodes **8** and **10** in cluster **SC-6.n** reduces the number of broker hops, for the shortest path from the publisher to the measuring subscriber at node **10**, from 9 to 4.

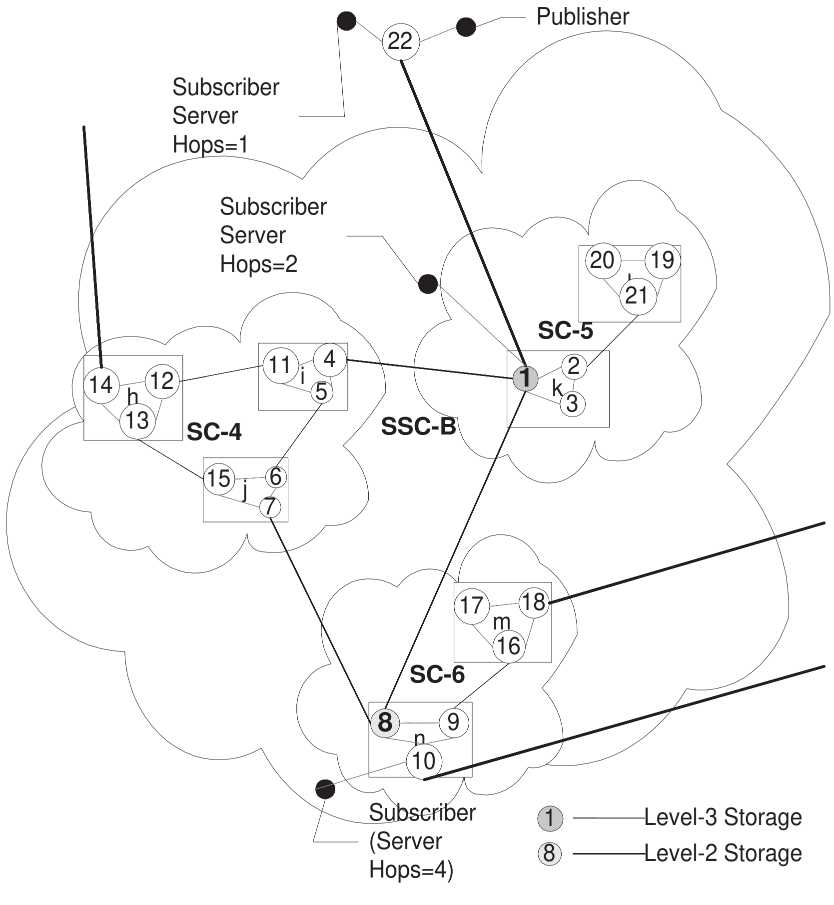

Figure 6: Testing Topology - Latencies versus broker hops

In this setting we are interested in the changes in latencies as the number of broker hops vary. We measure the latencies at three different locations, the measuring subscriber at node **10** has a broker hop of 4 while the measuring subscribers at nodes **1** and **22** have broker hops of 2 and 1 respectively for events published by the publisher at node **22**.

In general, as the number of broker hops reduce the latencies also reduce. The patterns for changes in latency as the event size and publish rates increase is however similar to our earlier cases. We depict our results, gathered at the three measuring subscribers for matching rates of 50%, 25% and 10%. The pattern of decreasing latencies with a decrease in the number of broker hops is clear by looking at figures 7 through 9.

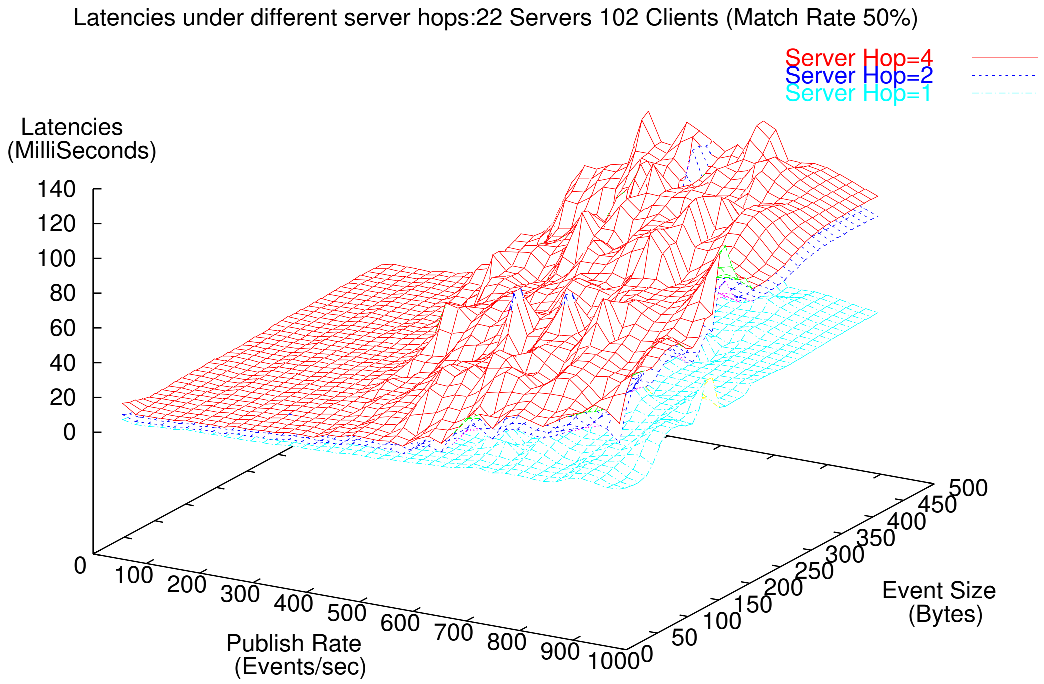

Figure 7: Broker Hops - Match Rate 50%

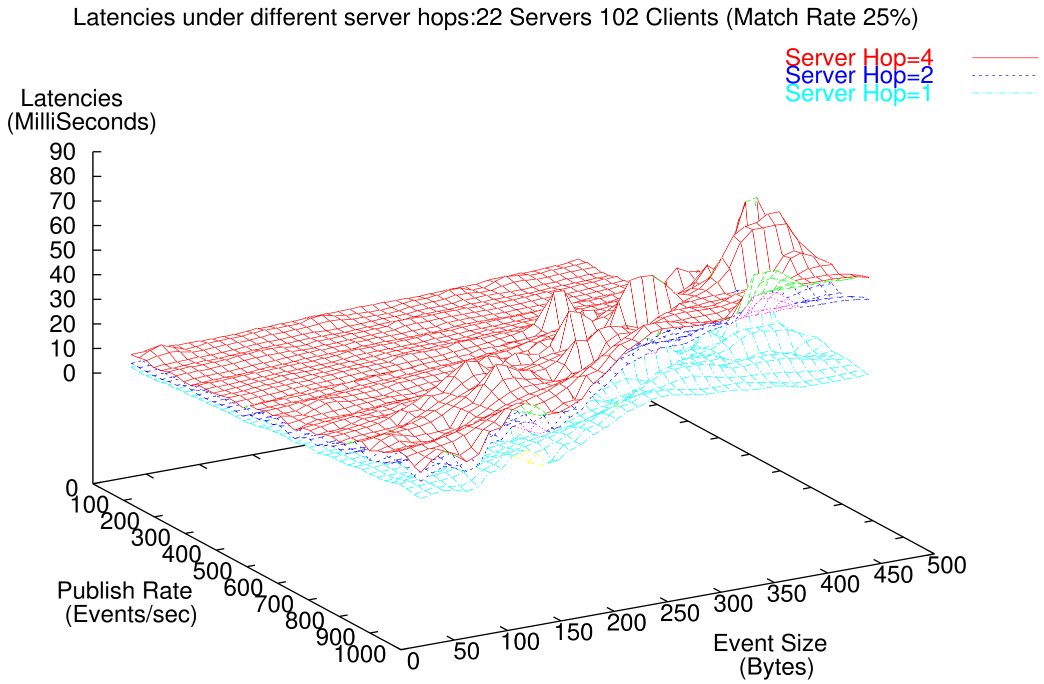

Figure 8: Broker Hops - Match Rate 25%

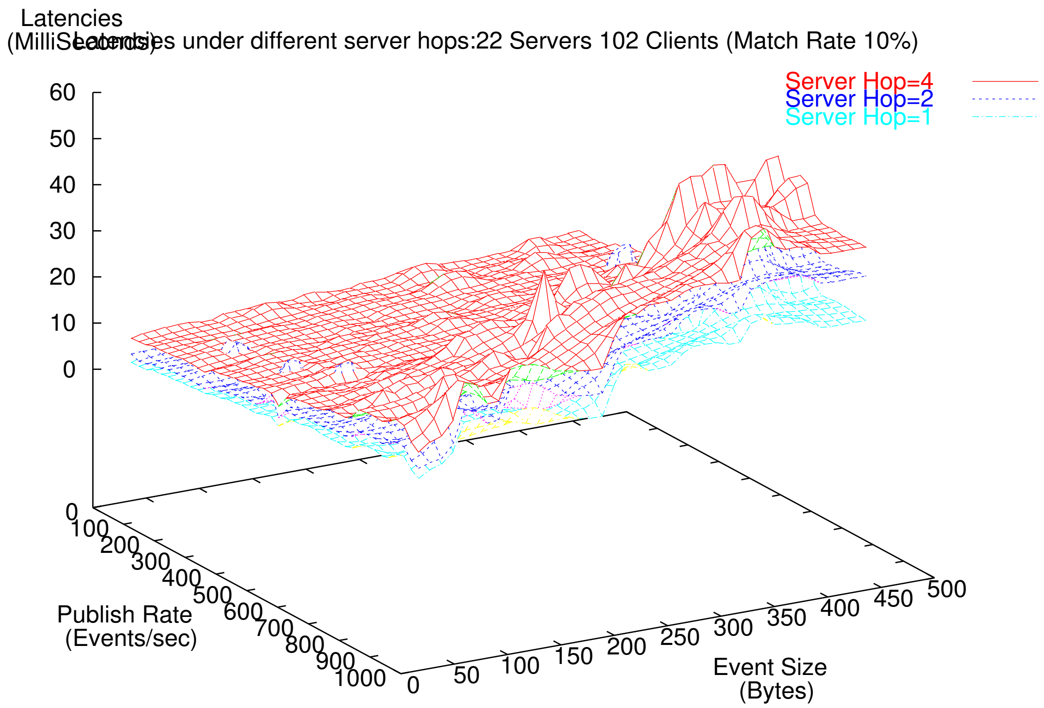

Figure 9: Broker Hops - Match Rate 10%

5 Conclusion

In this paper we have seen how the latencies vary with event sizes, matching rates, publish rates and connectivities. In general latencies decrease with increase in system connectivity, this being a result of the decrease in average pathlengths as the connectivity increases. On the other hand, increase in event sizes and publish rates result in an increase in the latency associated with event delivery. With decreasing matching rates, the latencies in event delivery decreases.