

Reuters Market Data System

RMDS 6

Performance Test Results on Sun's Blade X6270 (Two-Socket Intel Nehalem - Quad-Core dual thread) Server Module with 10GbE Ethernet Infrastructure

DISCLAIMER: The test results and recommendations contained in this report are made available for informational purposes only. By issuing this report, Sun Microsystems, Inc. does not guarantee similar performance results. This report is provided for your internal use only and may not be redistributed or published in any form without the prior written consent of Sun Microsystems, Inc. All information contained herein is provided on an "AS-IS" BASIS WITHOUT WARRANTY OF ANY KIND. Obtaining repeatable, measurable performance results requires a controlled environment with specific hardware, software, network, and configuration in an isolated system. Adjusting any single element may yield different results. Additionally, test results at the component level may not be indicative of system level performance, or vice versa. Sun Microsystems, Inc. is pleased to provide this report for our customers. We appreciate that each organization has unique requirements, and therefore may find this information insufficient for its needs. Customers wishing to obtain custom analysis for their systems are encouraged to contact their local Sun Microsystems, Inc. representative.

Issue 1.1

Date of Issue: May 08, 2009

Amjad.Khan@sun.com

Contents

1 General Information	3
1.1 Objective.....	3
1.1.1 Results Summary	3
1.2 Testing Methodology.....	3
1.3 Software Versions.....	4
1.3.1 RMDS.....	4
1.3.2 RMDS Test Tool.....	4
1.3.3 Operating Systems.....	4
1.4 Hardware.....	4
2 Preparation for Performance Test.....	4
2.1 Network.....	4
2.2 Hardware.....	4
2.3 Operating System Configuration.....	5
2.3.1 Solaris Kernel Parameters.....	5
2.3.2 TCP and UDP Buffers.....	5
2.3.3 Additional Solaris Kernel Parameters for Network Latency.....	6
2.4 RMDS Configuration.....	6
2.5 Miscellaneous Notes.....	7
3 Detailed Results.....	11
3.1 RSSL/RWF Update Throughput.....	11
3.1.1 Standalone Source Distributor.....	11
3.1.2 P2PS/LAN.....	12
3.2 Update Throughput via P2PS/POP.....	13
3.2.1 RSSL/RWF.....	13
3.3 End-to-End RSSL/RWF Latency.....	14
3.3.1 RRCP Backbone Results.....	15
3.3.2 Rendezvous Backbone Results.....	16

© Sun Microsystems, Inc. 2009. All Rights Reserved.

Sun Microsystems, Inc., by publishing this document, does not guarantee that any information contained herein is and will remain accurate or that use of the information will ensure correct and faultless operation of the relevant service or equipment. Sun Microsystems, Inc., its agents and employees, shall not be held liable to or through any user for any loss or damage whatsoever resulting from reliance on the information contained herein.

This document contains information proprietary to Sun Microsystems, Inc. and may not be reproduced, disclosed, or used in whole or part without the express written permission of Sun Microsystems, Inc.

Any Software, including but not limited to, the code, screen, structure, sequence, and organization thereof, and Documentation are protected by national copyright laws and international treaty provisions. This manual is subject to U.S. and other national export regulations.

1 General Information

1.1 Objective

The objective of this document is to report the performance test results for RMDS 6 on the Sun Blade 6000 equipped with Sun Blade X6270 (Intel Xeon CPU X5570 - Quad-Core – dual thread per core @ 2.93GHz) Server Modules. The test procedures are described in Reuters RMDS 6 Performance Test Procedures and Results document.

The X6270 blade server module is a x86 architecture based on the Intel next generation Nehalem (Xeon) processor chip. The blades used in these tests had 2 processor chips with 4 cores per chip and each core contains 2 hyper-threads (Total 8 Cores – 16 Hyper-Threads) to execute applications in parallel.

The goal of these tests is to measure throughput and latency through RMDS 6 infrastructure components, specifically the Point-to-Point Server (P2PS) and Source Distributor. The tests are grouped into three categories:

- Update throughput using RSSL/RWF data (see 3.1)
- Update throughput via P2PS/POP (see 3.2)
- End-to-end RSSL/RWF latency using embedded timestamp (see 3.3)

1.1.1 Results Summary

RMDS Components Configuration	RMDS Throughput & End-to-End Latency Using Ethernet Infrastructure
Maximum Throughput at <=1ms RMDS end-to-end Latency with RVD	850,000 updates/sec
P2PS-LAN, RRCP, Producer 50/50 Fanout (Cache Disabled)	3,535,000 updates/sec
P2PS-LAN, RVD, Producer 50/50 Fanout (Cache Disabled)	3,383,500 updates/sec
P2PS-POP, Producer 50/50 Fanout (Cache Enabled)	3,509,750 updates/sec
Src_dist, RRCP (Cache Disabled)	1,560,000 updates/sec
Src_dist, RVD (Cache Disabled)	1,360,000 updates/sec
P2PS-LAN, RRCP No Fanout (Cache Disabled)	1,330,000 updates/sec
P2PS-LAN, RVD No Fanout (Cache Disabled)	1,135,000 updates/sec
P2PS-POP, No Fanout (Cache Enabled)	715,000 updates/sec

1.2 Testing Methodology

For throughput testing, the *sink_driven_src* utility was used to generate update traffic, and the *rmdstestclient* utility was used to consume the updates. Level 1 data was used, with a Reuters Wire Format (RWF) update size of 74 bytes. Tests with no fanout of updates used a 100,000 item watchlist. The infrastructure is tuned for maximum throughput, and the update rate was increased until the CPU limit was reached with no errors reported. Where needed, and as noted, multiple Source Distributors or multiple P2PSs were used to create the load necessary to measure the component under test.

The embedded timestamp approach was used to calculate end-to-end latency for Level 1 (Quotes and Trades) data. RMDS 6 end-to-end update latency is measured by using *sink_driven_src* as

the publisher and **rmdstestclient** as the subscriber. In the embedded timestamp approach, the publisher embeds timestamps into selected updates which the subscriber uses for latency calculations. In this scenario, the publisher and subscriber must be running on the same node for accurate timestamps.

1.3 Software Versions

1.3.1 RMDS

src_dist ver. mdh6.3.2.L1
p2ps ver. p2ps6.3.3.L1
rrcp as included in p2ps6.3.3.L1
rvd 8.1.0

1.3.2 RMDS Test Tool

sink_driven_src (from MDH load above)
rmdstestclient (from P2PS load above)

1.3.3 Operating Systems

- **Solaris 10 10/08 s10x_u6wos_07a X86** (update 6)
- **Patch Level Generic_137138-09**

1.4 Hardware

- **Sun Blade X6270 Server Module** – Sun Blade Server Module powered by two-socket Quad-Core(dual thread) Intel Xeon X5500 series – **2 X 2.93GHz** Quad-Core Intel Xeon X5570 processors (total 8 cores / 16 threads)
- **Sun x8 Express Dual 10 Gigabit Ethernet** Fiber XFP NIC Cards with nxge driver
- **Arista 7124S 10GbE Ethernet switch** 24-port 10 GbE 480 Gbps, 360 Mpps, L2/3/4, 1U Switch (SFP)
- **Foundry Networks Edgellron 8x10G** (8 ports 10Mbit Fiber) 10-Gigabit Ethernet (10-GbE) switch

2 Preparation for Performance Test

2.1 Network

All the performance tests were run where the machines were connected to a private network via 10-Gigabit Ethernet(10GbE) switch Gbps. Each server blade had Sun x8 Express Dual 10GbE ports interface card where each port in a blade connected to a separate backplane and switch. One was used for the UDP traffic and the other was used for TCP communications. All the network cards and switch ports were set to Auto Negotiate and the standard MTU of 1500 was used on all of these interfaces. There was no TOE used for throughput or latency testing over 10GbE network on Solaris. Solaris default 10GbE driver(**nxge**) was used for all of the testing.

2.2 Hardware

All RMDS components were run on the same class of machines. Five Sun blade X6270 server modules were available for these tests. For all tests, two X6270 blade modules were always assigned to the **src_dist** and **p2ps** RMDS components. The third X6270 blade system was used as a second **p2ps** system for the **src_dist** throughput tests, for the **p2ps/POP** in the POP tests, and the **sink_driven_src/rmdstestclient** system for the latency tests. In all throughput tests, additional

two blade server modules were used for the sink_driven_src and rmdstestclient tools. On all of the system the Sun x8 Express Dual 10 GbE interfaces were used for MDH backbone and TCP network traffic.

2.3 Operating System Configuration

Solaris 10 10/08 (update 6) was installed on all systems with the default Solaris 10 update 6 patch "Generic_137138-09". The nxge network driver for 10GbE was used for all the testing.

The network parameters set via ndd commands may be set from the command line or via a Service Management Facility (SMF) manifest and method file (site/ndd) available from: <http://opensolaris.org/os/community/smf/manifests/>

For Solaris 10 update 3 and beyond, the udp_smallest_anon_port parameter should be set from the network-anon manifest and method files that are also available from the above [URL](#).

Further detail on tuning Solaris 10 and RMDS can be found at <http://www.sun.com/reuters>.

The combination of MDH 6.3.2 and P2PS 6.3.3 software used in this testing is the standard Reuters distribution for Sun x86 Solaris. As such RMDS 6 was compiled with optimization specific to the AMD Opteron processor, not to the Intel Xeon. Some additional performance gains may be realizable with a custom compile for the Intel Xeon architecture.

2.3.1 Solaris Kernel Parameters

Any settings changed from the defaults are noted below:

Step	Procedure		System File
1	OS	Enter the following lines in system file noted	/etc/system
	Solaris	set dld:dld_opt=2 set hires_tick=1 (Only on the sink_driven_src and rmdstestclient systems)	

2.3.2 TCP and UDP Buffers

Any settings changed from the defaults are noted below:

Step	Procedure		System File
2	OS	Enter the following lines in system file noted	/lib/svc/method/net-init
	Solaris	/usr/sbin/ndd -set /dev/tcp tcp_max_buf 8388608	
		/usr/sbin/ndd -set /dev/tcp tcp_recv_hiwat 4194304	
		/usr/sbin/ndd -set /dev/tcp tcp_xmit_hiwat 4194304	
		/usr/sbin/ndd -set /dev/udp udp_max_buf 8388608	
		/usr/sbin/ndd -set /dev/udp udp_xmit_hiwat 4194304	
		/usr/sbin/ndd -set /dev/udp udp_recv_hiwat 4194304	
/usr/sbin/ndd -set /dev/udp udp_do_checksum 0			

2.3.3 Additional Solaris Kernel Parameters for Network Latency

Step	Procedure		
4	OS	Enter the following lines in the file noted	System File
	Solaris nxge driver	set autoup=300 (<i>latency tests only</i>)	/etc/system
		set tune_t_fsflushr=5 (<i>latency tests only</i>)	
		Add the following two entries in the /platform/i86pc/kernel/drv/nxge.conf file for better network latencies and reboot the system: (<i>latency tests only</i>)	/platform/i86pc/kernel/drv/nxge.conf
rxdma-intr-time = 1;			
rxdma-intr-pkts = 1;			

2.4 RMDS Configuration

The configuration templates—*rmds.cnf.template* and *pop.cnf.template*—were customized for the tests.

Config File	Description	Path
<i>rmds.cnf.template</i>	Configuration file	<i>\$RMDS_SW/config</i> <i>on mdh and p2ps/LAN systems</i>
<i>pop.cnf.template</i>	Configuration file to do P2PS/POP test	<i>\$RMDS_SW/config</i> <i>on p2ps/POP system only</i>

Any non-required changes (i.e., IP addresses, hostnames, etc) for **throughput** are noted below:

Config File	Parameter	Value
<i>rmds.cnf</i>	*IDN_RDF*rrmpFlushInterval	20
<i>rmds.cnf</i>	*DF_RMDS*rrmpFlushInterval	20
<i>rmds.cnf</i>	*p2ps*tcpNoDelay	False
<i>rmds.cnf</i>	*p2ps*timedWrites	True
<i>rmds.cnf</i>	*p2ps*flushInterval	20
<i>rmds.cnf</i>	*RRCP*udpRecvBufSize	8192
<i>rmds.cnf</i>	*RRCP*udpSendBufSize	8192
<i>rmds.cnf</i>	*DF_RMDS*readBias	60 (add this entry)
<i>rmds.cnf</i>	*p2ps*maxOutputBuffers	32000
<i>rmds.cnf</i>	*p2ps*guaranteedOutputBuffers	800
<i>rmds.cnf</i>	*p2ps*poolSize	32000
<i>rmds.cnf</i>	*p2ps*latencyCalcMountBased	False
<i>rmds.cnf</i>	*p2ps*latencyCalcEnabled	False
<i>rmds.cnf</i>	*src_dist*latencyCalcServerBased	False
<i>rmds.cnf</i>	*src_dist*latencyCalcEnabled	False
<i>rmds.cnf</i>	*RRCP*maxPktPoolSize	200000
<i>rmds.cnf</i>	*RRCP*pktPoolLimitHigh	190000
<i>rmds.cnf</i>	*RRCP*pktPoolLimitLow	180000

Values modified for **network latency** testing are noted below:

Config File	Parameter	Value
-------------	-----------	-------

<i>rmads.cnf</i>	*IDN_RDF*rrmpFlushInterval	0
<i>rmads.cnf</i>	*DF_RMDS*rrmpFlushInterval	0
<i>rmads.cnf</i>	*p2ps*tcpNoDelay	True
<i>rmads.cnf</i>	*p2ps*timedWrites	False
<i>rmads.cnf</i>	*p2ps*flushInterval	0
<i>rmads.cnf</i>	*p2ps*maxOutputBuffers	400
<i>rmads.cnf</i>	*p2ps*guaranteedOutputBuffers	200
<i>rmads.cnf</i>	*p2ps*poolSize	16000
<i>rmads.cnf</i>	*p2ps*latencyCalcMountBased	False
<i>rmads.cnf</i>	*p2ps*latencyCalcEnabled	False
<i>rmads.cnf</i>	*src_dist*latencyCalcServerBased	False
<i>rmads.cnf</i>	*src_dist*latencyCalcEnabled	False
<i>rmads.cnf</i>	*RRCP*maxPktPoolSize	80000
<i>rmads.cnf</i>	*RRCP*pktPoolLimitHigh	70000
<i>rmads.cnf</i>	*RRCP*pktPoolLimitLow	60000

2.5 Miscellaneous Notes

Any other significant deviations from the standard test procedures, or clarifications, are noted below (such as number/type of machines used, CPU binding policy, etc.):

For RMDS 6, the p2ps application is a multi-threaded process where one thread does a substantial amount of work (this thread is referred to as the “high runner” below) with several other threads supporting it. When running with rrcpd in a LAN configuration, thread number 4 (of 5) is the high runner. When running with rvd in a LAN configuration, thread 7 (of 8) is the high runner. When running in a POP configuration using RSSL/RWF, thread 4 (of 4) is the high runner. Bindings of threads to specific processor cores were handled via the use of processor sets and explicit pbind commands. Typically, one strand was used for the high runner, for p2ps a second for the other p2ps threads, and in the LAN configurations one or two others were used for the transport daemon (rrcpd or rvd). There are test cases where use of more strands may be desirable, and it is expected that continued development of the p2ps application will also require changes to this strategy.

Test	Deviation	Comments
All	Multicast vs Broadcast	The standard rmads.cnf.template file does not enable multicast for RRCP, but does so for RV. All of the RRCP and RV tests were run here with multicast as opposed to broadcast for UDP networking.

src_dist	CPU binding, interrupts disabling and CPU scheduling class assignment	<p>Except where noted, src_dist was bound to CPU (core) 4 & 5 via the use of a processor set. The respective transport daemon (rrcpd or rvd) was bound to CPUs (cores) 6 & 7 via another processor set to take advantage of its multi-threaded capability. Interrupts were disabled on CPU (cores) 4, 5, 6 & 7 (no-intr using psradm).</p> <p>For RRCP Transport: Commands Used for process binging: <pre>% psrset -c -F 4 5 % psrset -c -F 6 7 % psrset -f 1 % psrset -f 2 % psrset -e 2 ./rrcpd source % ./src_dist % psrset -b 1 `pgrep src_dist` % pbind -b 4 `pgrep src_dist`/1 % pbind -b 5 `pgrep src_dist`/2 % pbind -b 6 `pgrep rrcpd`/7 % pbind -b 7 `pgrep rrcpd`/1-6,8-13</pre> FX scheduling class was assigned to src_dist and rrcpd or rvd daemon processes: <pre>% priocntl -s -c FX -m 60 -p 60 `pgrep src_dist` % priocntl -s -c FX -m 60 -p 60 `pgrep rrcpd`</pre> </p> <p>For Tibco RV Transport: Commands Used for process binging: <pre>% psrset -c -F 4 5 % psrset -c -F 6 7 % psrset -f 1 % psrset -f 2 % ./src_dist % psrset -b 1 `pgrep src_dist` % pbind -b 4 `pgrep src_dist`/1 % pbind -b 5 `pgrep src_dist`/2 % psrset -b 2 `pgrep rvd` % pbind -b 6 `pgrep rvd`/3 % pbind -b 7 `pgrep rvd`/1-2,4-9</pre> FX scheduling class was assigned to src_dist and rrcpd or rvd daemon processes: <pre>% priocntl -s -c FX -m 60 -p 60 `pgrep src_dist` % priocntl -s -c FX -m 60 -p 60 `pgrep rvd`</pre> </p>
----------	---	--

<p>p2ps (LAN/POP)</p>	<p>CPU binding and interrupts disabling and CPU scheduling class assignment</p> <p>For P2PS-POP simply create single processor set out of CPU (cores) 4 & 5 and follow the same step mentioned for P2PS-LAN except that there will be no transport involved in this testing.</p>	<p>Except where noted, p2ps was bound to CPUs (cores) 4 (high runner thread) and 5 (all other threads) via the use of processor set. The respective transport daemon (rrcpd or rvd) threads were bound to CPU 6 & 7 via an additional processor set to take advantage of its multi-threaded capability. Interrupts were disabled on any CPU core that was running an application thread(no-intr using psradm).</p> <p>For RRCP Transport: Commands Used for process binging: <pre>% psrset -c -F 4 5 % psrset -c -F 6 7 % psradm -i 4-7 % psrset -e 2 ./rrcpd sink % psrset -e 1 ./p2ps</pre> <pre>% pbind -b 4 `pgrep p2ps`/4 % pbind -b 5 `pgrep p2ps`/1-3,5 % pbind -b 6 `pgrep rrcpd`/9 % pbind -b 7 `pgrep rrcpd`/1-8,10,13</pre> FX scheduling class was assigned to src_dist and rrcpd or rvd daemon processes: <pre>% priocntl -s -c FX -m 60 -p 60 `pgrep p2ps` % priocntl -s -c FX -m 60 -p 60 `pgrep rrcpd`</pre> <p>For Tibco RV Transport: Commands Used for process binging: <pre>% psrset -c -F 4 5 % psrset -c -F 6 7 % psradm -i 4-7 % psrset -e 1 ./p2ps % psrset -b 2 `pgrep rvd`</pre> <pre>% pbind -b 4 `pgrep p2ps`/7 % pbind -b 5 `pgrep p2ps`/1-6,8 % pbind -b 6 `pgrep rvd`/19 % pbind -b 7 `pgrep rvd`/1-18</pre> FX scheduling class was assigned to src_dist and rrcpd or rvd daemon processes: <pre>% priocntl -s -c FX -m 60 -p 60 `pgrep src_dist` % priocntl -s -c FX -m 60 -p 60 `pgrep rvd`</pre> </p> </p>
-----------------------	--	---

End-to-End Latency	CPU binding and interrupts disabling and CPU scheduling class assignment	<p>Tuning techniques for src_dist and p2ps servers will remain same as described above. The following tuning is applied on the sink_driven_src and rmdstestclient server:</p> <pre>% psrset -c -F 4 % psrset -c -F 6 7 % psrset -f 1 % psrset -f 2 % psrset -e 1 ./sink_driven_src % psrset -e 2 ./rmdstestclient</pre> <p>FX scheduling class was assigned to src_dist and rrcpd or rvd daemon processes:</p> <pre>% priocntl -s -c FX -m 60 -p 60 `pgrep sink_driven_src` % priocntl -s -c FX -m 60 -p 60 `pgrep rmdstestclient`</pre> <p>Please see the section 2.3.3 and above notes for setting up the src_dist and P2PS for network latency.</p>
--------------------	--	--

3 Detailed Results

3.1 RSSL/RWF Update Throughput

- All the throughput numbers quoted here are for Level 1 data.
- The data file used in these tests has 1 update, with an update (data, not including header) size of 74 bytes in RWF.
- All of the tests with no fan-out used 100,000 item watchlist.
- In most of the throughput tests the individual processes were bound to particular CPU(s).
- ***src_driven_src*** and ***rmdstestclient*** were used as the publisher and consumer of data.
- In some Source Distributor tests, two P2Ps were used to create sufficient load.

3.1.1 Standalone Source Distributor

Configuration Option	Transport	Max Throughput	Comments
Cache Disabled	RRCP	1,560,000 Note: The CPU#4 where thread 1 of src_dist was bound still left with 3% CPU idle.	See section 2.5 for performance tuning details for src_dist with RRCP transport. NOTE: Two systems were used for p2ps in the test bed. Single processor CPU utilization for thread 1 of src_dist was 98%.
Cache Enabled	RRCP	675,000	See section 2.5 for performance tuning details for src_dist with RRCP transport.
Cache Disabled	Rendezvous	1,360,000 Note: The CPU#4 where thread 1 of src_dist was bound still left with 2% CPU idle.	See section 2.5 for performance tuning details for src_dist with Tibco RV transport. NOTE: Two systems were used for p2ps in the test bed. Single processor CPU utilization for thread 1 of src_dist was only 98%.
Cache Enabled	Rendezvous	635,000	See section 2.5 for performance tuning details for src_dist with Tibco RV transport. FX scheduling class was assigned to src_dist and rvd daemon processes.

3.1.2 P2PS/LAN

Configuration Option	Mounts : Commonality	Transport	Max Throughput	Comments
Cache Disabled	No fan-out	RRCP	1,330,000	See section 2.5 for performance tuning details for P2PS with RRCP transport.
Cache Enabled	No fan-out	RRCP	622,000	See section 2.5 for performance tuning details for P2PS with RRCP transport.
Cache Disabled	100 mounts; Producer 50/50	RRCP	70,000 input 3,535,000 output	See section 2.5 for performance tuning details for P2PS with RRCP transport.
Cache Disabled	No fan-out	Rendezvous	1,135,000	See section 2.5 for performance tuning details for P2PS with Tibco RV transport.
Cache Enabled	No fan-out	Rendezvous	570,000	See section 2.5 for performance tuning details for P2PS with Tibco RV transport.
Cache Disabled	100 mounts; Producer 50/50	Rendezvous	67,000 input 3,383,500 output	See section 2.5 for performance tuning details for P2PS with Tibco RV transport.

3.2 Update Throughput via P2PS/POP

These tests were performed using a P2PS/POP with data provided by an upstream P2PS/LAN, with an RRCP transport.

3.2.1 RSSL/RWF

Configuration Option	Mounts : Commonality	Max Throughput	Comments
Cache Enabled	No fan-out	705,000	See section 2.5 for performance tuning details for p2ps/POP without any transport. The throughput number was obtained with single processor CPU utilization of 98% for p2ps/POP.
Cache Enabled	100 mounts; Producer 50/50	69,500 input 3,509,750 output	See section 2.5 for performance tuning details for p2ps/POP without any transport. The throughput number was obtained with single processor CPU utilization of 100% for p2ps/POP.

3.3 End-to-End RSSL/RWF Latency

Latency is defined as the time for a data item to propagate through one or more RMDS components. “End to end” latency is defined as the delta between the time an update is posted by the publisher application to its API and the time the same update is received by the consuming application from its API, i.e. it includes both the latency contribution from the API and the core infrastructure components.

- Caching was disabled in both the Source Distributor and the P2PS during these tests.
- Optimized binaries of the RMDS infrastructure components were used.
- NTP was disabled on the tools node, as any drifts in time will affect the reported latency.
- Tests were run with 100,000 item watchlist and RWF data update size of 74 bytes [Data file (**sample.xml**) was used].
- Latency tests were run at each update rate for at least 5 minutes, up to the maximum sustainable update rate for a given setup.
- Decode of data was turned on in these tests.

3.3.1 RRCP Backbone Results

Update Rate [74-byte RWF messages]	Mean Latency (millisec)	Std Deviation (millisec)	Maximum Latency (millisec)	Minimum Latency (millisec)	Number of Latency Points
1,000	0.200	0.014	0.254	0.159	3140
5,000	0.239	0.014	0.302	0.199	3150
10,000	0.288	0.014	0.373	0.245	3140
20,000	0.341	0.015	0.508	0.293	3150
30,000	0.373	0.017	0.570	0.312	3140
40,000	0.387	0.018	0.593	0.323	3150
50,000	0.390	0.023	0.749	0.332	3140
60,000	0.389	0.023	0.736	0.312	3130
70,000	0.397	0.024	0.830	0.334	3140
80,000	0.404	0.026	0.874	0.321	3140
90,000	0.413	0.029	0.929	0.355	3140
100,000	0.430	0.041	1.085	0.348	3150
150,000	0.447	0.072	1.434	0.345	3150
200,000	0.488	0.068	0.867	0.347	3150
250,000	0.498	0.085	0.934	0.323	3130
300,000	0.534	0.114	1.311	0.351	3150
350,000	0.579	0.146	1.592	0.335	3130
400,000	0.642	0.158	1.596	0.330	3130
450,000	0.726	0.199	1.962	0.357	3120
500,000	0.721	0.207	1.977	0.339	3130
550,000	0.802	0.245	2.261	0.351	3110
600,000	0.855	0.286	2.639	0.352	3110
650,000	0.872	0.306	2.950	0.320	3120
700,000	0.949	0.352	3.191	0.359	3130
750,000	1.005	0.405	3.224	0.316	3140
800,000	1.125	0.497	3.555	0.360	3140
850,000	1.297	0.621	4.903	0.336	3110

3.3.2 Rendezvous Backbone Results

Update Rate [74-byte RWF messages]	Mean Latency (millisec)	Std Deviation (millisec)	Maximum Latency (millisec)	Minimum Latency (millisec)	Number of Latency Points
1,000	0.227	0.014	0.275	0.173	3160
5,000	0.260	0.014	0.312	0.206	3130
10,000	0.299	0.014	0.343	0.240	3140
20,000	0.361	0.014	0.408	0.312	3150
30,000	0.391	0.016	0.453	0.297	3150
40,000	0.408	0.017	0.462	0.348	3150
50,000	0.415	0.018	0.478	0.356	3130
60,000	0.399	0.017	0.456	0.317	3120
70,000	0.416	0.019	0.483	0.359	3140
80,000	0.421	0.019	0.492	0.359	3130
90,000	0.430	0.020	0.525	0.367	3150
100,000	0.437	0.024	0.518	0.356	3120
150,000	0.451	0.044	0.563	0.350	3140
200,000	0.491	0.065	0.666	0.350	3130
250,000	0.549	0.111	0.782	0.351	3130
300,000	0.545	0.109	0.780	0.360	3120
350,000	0.589	0.133	0.876	0.350	3120
400,000	0.616	0.158	0.936	0.347	3120
450,000	0.704	0.195	1.318	0.372	3120
500,000	0.764	0.253	1.299	0.375	3120
550,000	0.709	0.210	1.158	0.327	3120
600,000	0.747	0.230	1.196	0.366	3140
650,000	0.796	0.259	1.302	0.343	3100
700,000	0.819	0.273	1.348	0.355	3130
750,000	0.871	0.305	1.475	0.360	3110
800,000	0.876	0.306	1.462	0.353	3120
850,000	0.937	0.350	1.615	0.323	3090
900,000	1.104	0.528	7.858	0.318	2980