Rational Performance Tester—robustness to network impairments

Contents

1 Introduction
- Tool used for evaluation
- Methodology of evaluation

2 Testing the Rational Performance Tester tool
- Base test case
- Packet loss test case
- Latency test case
- Network outage test case

6 Conclusion

Introduction

The IBM® Rational® Performance Tester tool is used for testing systems by generating load, and focuses on identifying performance bottlenecks. It uses a controller-agent architecture, where system load is generated by agents based on the directions from the controller.

The Rational Performance Tester tool depends on the network to:

- Communicate with the system under test.
- Communicate internally with the direct agents.
- Collect the results obtained from the test.

To evaluate the resilience of the Rational Performance Tester tool to problems in communications between the controller and agents, we introduced network impairments within a controlled environment and assessed the impact on the Rational Performance Tester tool.

Tool used for evaluation

For the evaluation, we used a tool called Dummynet [Dummmynet, DummynetRevisited]. Dummynet is a tool that allows a machine acting as a router/switch to pass packets through a virtual set of network links that can introduce delay, loss, bandwidth constraints, queuing constraints, etc.[1]

Similar tools such as Netem by Linux® and Desktop VE by Shunra are also available, but we chose to work with Dummynet because:

- Dummynet has been in use for over 10 years, and the features we planned to use are very mature at this stage.
- We had prior experience of using Dummynet at the Hamilton Institute for a research on Internet congestion control.
- Dummynet closely mimics the behavior of real links when emulating bandwidth and queue constraints and we felt that this would be significant for our tests.

Methodology of evaluation

To establish the type of communication between the controller and the agents (via a TCP connection), we introduced packet loss, latency or a network outage in the system.

Subsequently, we constructed and instrumented a simple test bed (illustrated in Figure 1) consisting of:

- A Linux-based Rational Performance Tester agent
- A Linux-based Rational Performance Tester controller
- A FreeBSD-based Dummynet switch
- A commercial switch
- A test system on the Internet

We used Wireshark or TCPdump to perform instrumentation to record packet traces. These traces were analyzed to get the results.

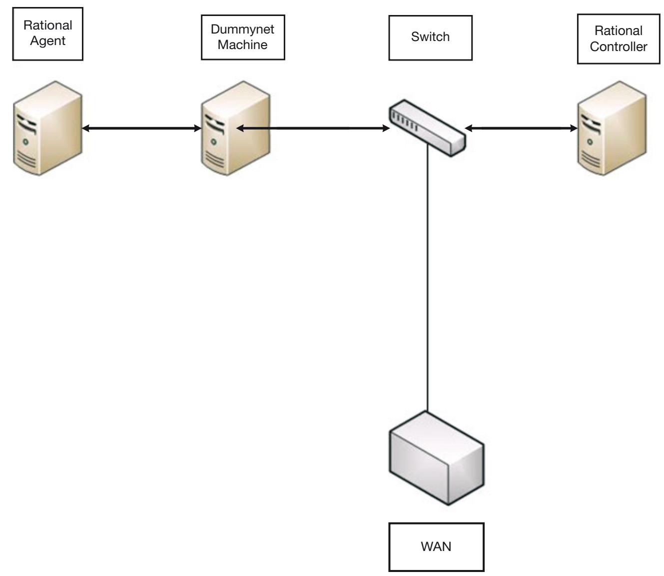

Figure 1: Test environment topology

Testing the Rational Performance Tester tool

We evaluated the behavior of the Rational Performance Tester tool using the following tests:

- Base test
- Packet loss test
- Latency test
- Network outage test

Base test case

Testing procedure

To establish the normal behavior of the Rational Performance Tester tool we conducted the initial tests without any network impairments. We conducted this test to detect any losses in the network under unimpaired network conditions and to determine the expected traffic concentration between the Rational Performance Tester controller and agent.

Observations

On performing the base test, we observed that Dummynet passed through all the packets. We compared the traffic recorded in the packet traces at the agent to the traffic recorded in the packet traces at the controller to confirm that there is no packet loss. We observed that the traffic recorded was a low 5 KBps - 10 KBps, except at the start and end of each run, where traffic spikes of 140 KBps and 40 KBps could be observed (Refer to Figure 2).

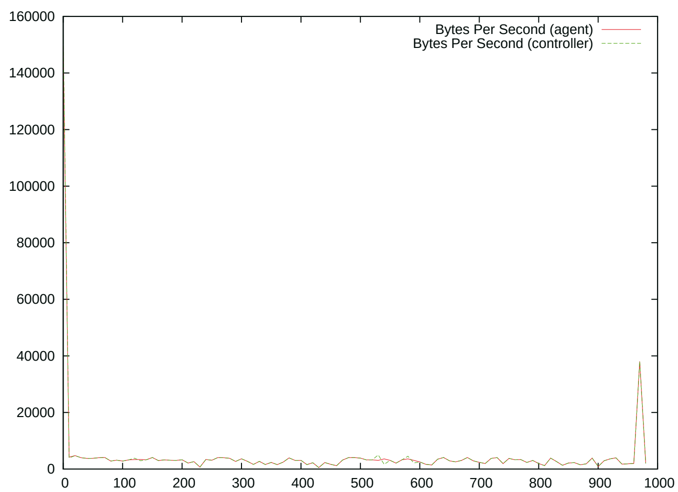

Figure 2: Network traffic during base test

We also compared the transmission and arrival times of each packet (Refer to Figure 3). After making some corrections for the differences in clocks on both systems, we could confirm that packets were delivered promptly.

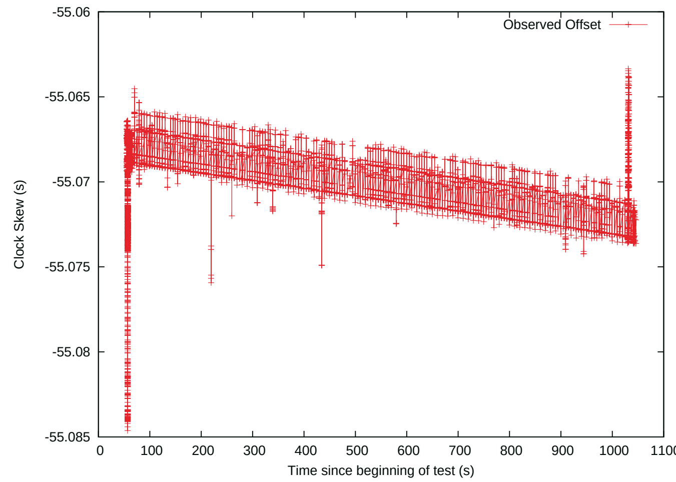

Figure 3: Clock skew for the duration of base test

Packet loss test case

Testing procedure

For performing a packet loss test case, we instructed Dummynet to randomly drop 2 percent of the packets that pass through the network. Such losses would indicate an equipment fault or a link failure. The heavy packet loss rate of 2 percent limits the throughput of the TCP.[2] However, given the low data rates between the agent and client observed in the base case, the retransmission mechanisms of TCP can cope with this loss rate.

Observations

The tests confirmed that this small loss rate does not have any negative impact on the agent and the controller. Packet losses from the packet traces are observed (Refer to Figure 4), but TCP gracefully recovers. It appears that the Rational Performance Tester software is unaware of the losses. At a higher application level, the Rational Performance Tester runs are completed normally.

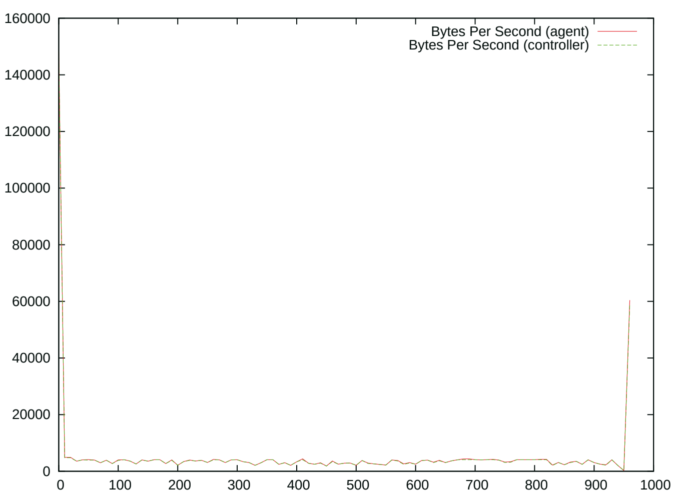

Figure 4: Network traffic during packet loss testing

Typically, a congested network does not result in random losses. Hence, the type of loss we are testing here is not representative of a congested network. While we did not conduct such a test, it can be carried out using the bandwidth limiting features of Dummynet.

Latency test case

Testing procedure

In this test, we introduced a latency of 200 ms between the agent and the controller. The typical latency between networks in Europe and the U.S. is 200 ms, so this might be representative of a Rational Performance Tester run across agents in two continents. No significant problems are expected because usually TCP is used over such delays.

Observations

The tests confirmed that the latency (on this scale) does not cause any problems for the Rational Performance Tester tool. Since the responsiveness of the TCP is slightly reduced over a longer round-trip time, the initial and final traffic spikes appear drawn out over a slightly longer period (Refer Figure 5). The packet traces showed a wider spread in packet time stamps, confirming that Dummynet is adding to the delay. The high-level Rational Performance Tester runs are not disturbed by this additional latency.

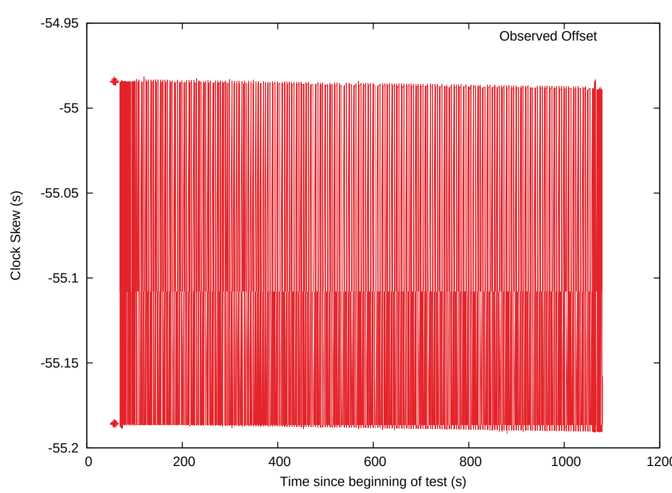

Figure 5: Clock skew during latency testing

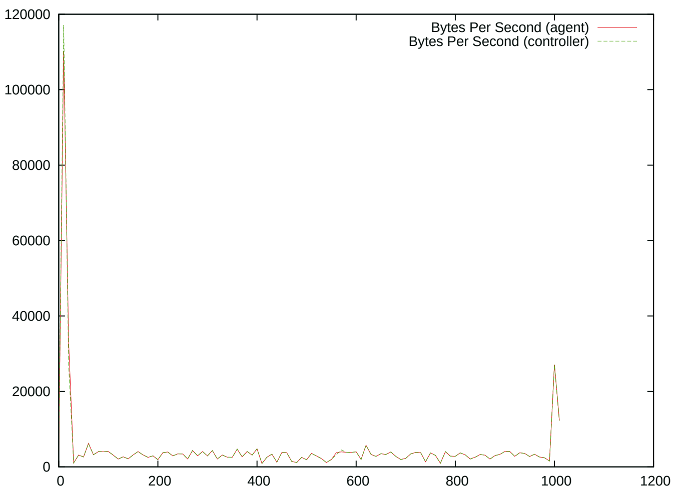

Figure 6: Network traffic during latency testing

Network outage test case

Testing procedure

For this test, we disconnected the network between the agent and controller, dropping all packets.

We did not do this by disconnecting the cable at either the agent or the controller—if the cable is removed the host will see that the link is down, and the network stack may attempt to work around the problem. We wanted to test what happened when a link in the network failed, and the end hosts had no information about this failure.

We introduced the link failure over two time periods: 60 seconds and 5 minutes. A failure time period of 60 seconds falls within the timeout settings of the TCP. Hence, for this time period, the timeout mechanism enables recovery of the connection after connectivity is restored.

A longer break of 5 minutes is beyond TCP's standard timeout period. The TCP closes the connection across which data transmission has been unsuccessful for this time period.

We performed the tests over these two time periods to establish if the Rational Performance Tester tool has any shorter timeouts and to evaluate the ability of the Rational Performance Tester tool to recover from closed TCP connections.

Observations

For the 60 second breaks, we observe the expected behavior (Refer to Figure 7). While the outage is in progress, no data (except a small number of retransmission attempts) is sent. When the link is restored, TCP detects it after a number of timeout retransmissions, and the pending data is transferred. This might happen promptly, or may take a longer time, depending on the exact duration of the break and on the time when retransmissions and timeouts occur. In either case, the Rational Performance Tester run is completed after the network is restored.

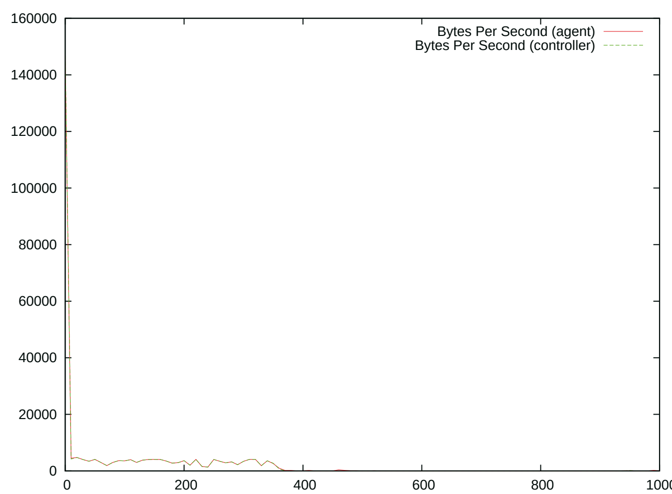

Figure 7: Network traffic during short network cut

When we introduced longer breaks of 5 minutes, the TCP connection terminated, as expected. This happens because the Rational Performance Tester tool fails to open a new connection between the agent and controller and this leads to eventual failure of the Rational Performance Tester test run. The failure is announced by an application-level message.

Conclusion

The tests we performed were very beneficial: We proved the robustness of the Rational Performance Tester tool in the event of network problems between the agent and controller. The Rational Performance Tester tool uses the reliability features of TCP. There are no short application-level timeouts that make the Rational Performance Tester tool more sensitive to network problems.

However, there are also no Rational Performance Tester-level mechanisms to reestablish the connections that fail. Considering the various uses of the Rational Performance Tester tool, an option to adjust the level of sensitivity would be suitable. For example, in an environment where the aim is to conduct a test over many days on a server, a short 5 minute outage may not be an issue, and it would be best for the Rational Performance Tester tool to continue. However, if the aim is to provide a sustained high load over a short period of time, then a 60 second outage between the agent and controller might be considered fatal. The Rational Performance Tester tool team is working towards resolving these concerns, and we expect future software releases to tackle this issue.

Overall, the tests were very useful and relatively easy to conduct. We were able to script most of the data analysis, making it easy to conduct repeated experiments and compare their results. This mode of testing can also be easily replicated in other places. For example, it would be easy to place Dummynet between the Rational Performance Tester agents and the services that they test. This would allow the assessment of a service under different network conditions. We have conducted initial tests for web-based services, and hope to assess the impact of network impairments on real-time services in the near future.

For more information

To learn more contact your IBM representative or Business Partner, or visit: **ibm.com**/software/rational/

Resources

- Dummynet: A simple approach to the evaluation of network protocols. L. Rizzo. ACM SIGCOMM Computer Communication Review. Volume 27, Issue 1, 1997.
- Dummynet Revisited, M Carbone, L. Rizzo. Technical Report University of Pizza, 2009.
- SqrtForm, Modeling TCP throughput: A simple model and its empirical validation. ACM SIGCOMM Computer Communication Review, Volume 28, Issue 4, 1998.

References

Learn more on the Rational Performance Tester product page

Explore Rational Performance Tester page on IBM developerWorks® for links to technical information for software developers and testers and to related IBM software.

Browse the Rational Performance Tester Information Center

Learn about other applications in the IBM Rational Software Delivery Platform, including collaboration tools for parallel development and geographically dispersed teams, specialized software for architecture management, asset management, change and release management, integrated requirements management, process and portfolio management, and quality management.

Visit the Rational software area on developerWorks for technical resources and best practices for Rational Software Delivery Platform products.

Explore Rational computer-based, web-based, and instructor-led online courses to hone your skills and learn more about Rational tools using introductory or advanced courses. The courses are available for purchase through computer-based training or web-based training. Additionally, some "Getting Started" courses are available free of charge.

Subscribe to the IBM developerWorks newsletter, a weekly update on the developerWorks tutorials, articles, downloads, community activities, webcasts and events.

Get products and technologies

Download the trial version of IBM Rational Performance Tester

Download these trial versions of other IBM Rational software

Download these IBM product evaluation versions and get acquainted with application development tools and middleware products from DB2®, Lotus®, Tivoli®, and WebSphere®.

Discuss

- Join the Performance Testing forum, for acquiring and sharing knowledge about IBM performance testing products, including the IBM Rational Performance Tester tool (now integrated with IBM Performance Optimization Toolkit). General performance testing, VU scripting, and load-testing topics are also discussed in this forum.

Check out developerWorks blogs and get involved in the developerWorks community

Additionally, financing solutions from IBM Global Financing can enable effective cash management, protection from technology obsolescence, improved total cost of ownership and return on investment. Also, our Global Asset Recovery Services help address environmental concerns with new, more energy-efficient solutions. For more information on IBM Global Financing, visit: **ibm.com**/financing

About the authors

Jonathan Dunne, Staff Software Engineer, IBM
jonathan_dunne@ie.ibm.com
Jonathan has an experience of seven years working in the System Test Team at Dublin, using the IBM Rational System Tester tool. Jonathan has worked on a number of major J2EE™ releases including the Workplace solution, IBM Workplace Collaborative Learning LotusLive™ Engage solution, and the IBM Lotus® Quickr™ tool. Jonathan has another two years of experience working with NUI Maynooth on network impairment research projects.

David Malone, Stokes Lecturer, NUI Ma
david.malone@nuim.ie
David Malone is an SFI Stokes Lecturer at the Hamilton Institute (NUI Maynooth) and a developer on the FreeBSD project. His interests include mathematical modeling of networks, network measurement, IPv6, and system administration. He is a coauthor of O'Reilly's "IPv6 Network Administration." His work with IBM is part of the SFI FAME project 08/SRC/I1403.

Niall Ward, Advisory Software Engineer, IBM
niall_ward@ie.ibm.com
Niall Ward joined IBM in late 2006 as a test engineer involved in the reliability testing of the J2EE version of the Lotus Quickr product, starting with version 8.0. Niall is currently an SVT team lead involved in testing the Quickr Domino® 8.5 release solution.

Ian Dangerfield, PhD student, Hamilton Institute, NUI Maynooth
ian.dangerfield@nuim.ie
Ian Dangerfield is a PhD student at the Hamilton Institute, NUIM. His interests are in network measurement, particularly in 802.11 (Wi-Fi) wireless networks, and layer 2 (MAC)-based approaches to quality of service. He is involved with IBM through the SFI FAME project.

Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
July 2010

[1] Dummynet was developed by Luigi Rizzo as an extension to the IPFW firewall of FreeBSD. It is widely used by researchers for recreating network conditions and testing protocol stacks. It is also used in production networks as a traffic shaping tool.

[2] For traditional forms of TCP, this limit can be calculated [SqrtForm].