Quagga Based Implementation of Compound Routing Metrics for Distance Vector Routing Algorithms

Spiros H. Courellis[1], Huanjun Liu, and Dimitri A. Michalopoulos[2]
Computer Science Department
California State University, Fullerton
800 N. State College Ave., Fullerton, CA 92834
USA

Abstract: - Implementation of compound routing metrics within Quagga for distance vector algorithms is presented in this paper. Quagga's RIP process has been extended to compute routing metric values composed of weighed environmental measures. Quagga has also been extended with additional processes to perform appropriate measurements contributing to the computation of the compound metric. An example is presented that extends the traditional hop count routing metric of RIP with a compound metric involving hop count and round trip time. A latency estimation module is introduced to measure round trip time between a routing node and its neighbors. The implementation was tested in the lab with an illustrative topology and testing scenarios depictive of the functionality of the implemented compound routing metric.

Key-Words: - routing, metric, latency, rtt, distance vector, quagga, rip

1 Introduction

Based on the TCP/IP infrastructure [1], numerous routing protocols have been introduced [2], [3] to address the idiosynchrarcies of various network topologies ranging from the nature and the bandwidth of the links, to the number of the nodes between source and destination, to the behavioral influence of the users on the network traffic. Often, the implementation of widely used commercial routing protocols (e.g., [4], [5], [6]) and the associated routing metrics is not open to the academic community for instructional and research purposes. Therefore, publicly available flexible and extensible routing tools have been developed to enable and facilitate research and education. Single routing protocol tools include **routed** for RIP, **OSPFd** [4] for OSPF and **pimd** [6] for PIM-SM multicast routing. Tools implementing multiple routing protocols include **GateD** [7], **MRTD** [8], **BIRD** [9], **XORP** [10], and **Zebra/Quagga** ([11], [12]). In particular, **GateD**, **MRTD**, and BIRD use a single process architecture, which has restricted extensibility, while **XORP** and **Zebra/Quagga** utilize multiple-process architecture, which enhances extensibility.

The availability of open source routing tools makes it easier to improve routing protocols. Once such improvement is the selection of an appropriate routing metric. Various routing metrics such as hop count, bandwidth and latency [13] have been implemented by all of the publicly available tools. However, not much capability exists within these tools when it comes to the use of compound routing metrics combining various topological and behavioral measurements. The use of compound metrics for research and teaching purposes is one attribute that is important to be associated with these tools as several commercial products (e.g., Cisco IGRP/EIGRP [14]) have proven that appropriate measurement combination in certain networks leads to better routing decisions.

In this paper, we present an extension to Quagga that incorporates the capability for the ripd daemon to use customized routing metrics based on topological and behavioral network measurements.

2 Design and Implementation

The implementation of compound routing metrics in Quagga's ***rip*** process involves: (1) the introduction of new and the extension of existing data structures, and (2) the introduction of new modules that (a) implement a set of rules used to compute the compound routing metrics, and (b) perform measurements to be used in the computation of the compound metrics. An example of a measurement module that we present is based on Round Trip Time (RTT).

2.1 Measurements for Routing Metrics

In general, there are two ways to measure routing metrics: actively and passively. The former perturbs the target network by generating traffic solely for making measurements [15]; the latter observes certain interested existing traffic and determines the corresponding measurement. Both of them are

capable of dynamically measuring metrics. Active measurements provide prompt response with reduced accuracy due to the affected network traffic. Thus, they are suitable for implementations that do not generate many packets, that is in lightly loaded, small networks. Passive measurements are designed for large network with heavy load**.** Since tests for this project are to be conducted in small network, active latency measurements will be utilized.

2.2 Latency Measurement

Latency measurements can be made using one of several existing approaches. Distributed and parallel router architecture [16] considers latency to be mainly caused by the route lookup delay. Traditional central router architecture (utilized in this project), considers route lookup delay not to have much weight. Instead, it views transmission delay and propagation delay as the main two causes of latency. Cisco Optimized Edge Routing (OER) passively monitors TCP SYN and TCP ACK packets for every flow to measure latency, packet loss, and reacheability. Since RIPv2 has route timer constraints to meet, this passive approach is not accurate or quick enough for this project. Cisco IGRP/EIGRP uses sum of delays configured on router interfaces to scale latency in its metric calculation [14]. This fixed setting of latency metric lacks accuracy and flexibility. Therefore it is not suitable for this project either. Active measurement approaches measure latency between hosts and servers by using mechanisms such as the "traceroute" command. Since the traceroute command is operating system dependent and not extensive, this method lacks flexibility and accuracy. In this project, we implemented our own round trip time estimator based on active measurements using ICMP packets.

2.3 The Part of Quagga that was Modified

Quagga utilizes a multiple-process structure (Fig. 1), using zebra processes to establish communication between protocol-specific processes and the OS kernel.

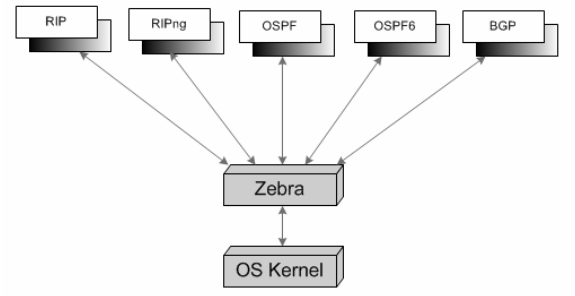

Fig. 1: Instance of Quagga using a Zebra Process as the intermediary between the OS Kernel and Protocol Specific Processes

The protocol-specific modules implement popular routing protocols. The RIP daemon, i.e. **ripd**, is the module we will perform the majority of the modifications. The structure of ripd (Fig. 2) is as follows: ***rip_main.c***: main function and main entry of ripd. ***ripd.c:*** ripd initialization and RIP logic implementation. This file contains most of the routing logic and will be the place where most of the code modification will be implemented. ***rip_zebra.c***: communication functions with Zebra daemon ***rip_snmp.c***: ripd SNMP related functions. ***rip_routemap.c***: ripd routemap management. ***rip_interface.c***: ripd interface related functions and data structures. ***rip_peer.c***: RIP peer management and communication. ***rip_offset.c***: interface offset management. ***rip_debug.c***: RIP debugging implementation.

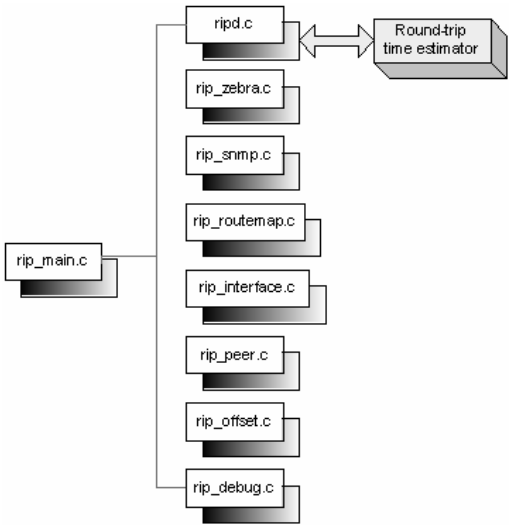

Fig. 2: The modules of Quagga's ***rip*** process.

2.4 Implementation in Quagga

Quagga's ***rip*** module was extended to accommodate the storage and propagation of data structures including more than one measurement, to apply a set of routing metric derivation rules, and to employ ***ripd*** triggered processes to perform measurements. We used this implementation framework to create an example that involved two measurements: hop count and (Round Trip Time) RTT between neighboring routers. The information propagated between routers for each destination was pairs of (hop count, RTT). The routing metric derivation rules were based on selecting the route with the smallest hop count for as long as it had the smallest RTT within a tolerance band. An RTT estimation module was developed to compute RTT between neighboring routers using ICMP packets (Fig. 2). Once triggered by Quagga's RIP daemon, this module sent out ICMP timestamp packets and waited for responses. The time

difference between sent and received ICMP packets was used to calculate the round-trip time value.

3 Results

The extended Quagga ***ripd*** we implemented was tested in the lab on the topology shown in Fig. 3. All nodes and hosts were Dell desktop PCs with 1GHz Intel Pentium III CPU and 512MB RAM. FreeBSD 5.3 was used as operating system. The workflow was as follows: (1) RIP daemons started on all routers and an RTT estimator was triggered by the RIP daemon on each router to find out the RTT to its neighbors; (2) RIP daemons propagated routing information associated with RTT and hop counts; (3) Neighboring routers received the routing information, and combined it with their own measurements; (4) Metric derivation rules were applied to compute the compound routing metric from the combined measurements; (4) All routing decisions were made with the compound metric instead of the hop count metric.

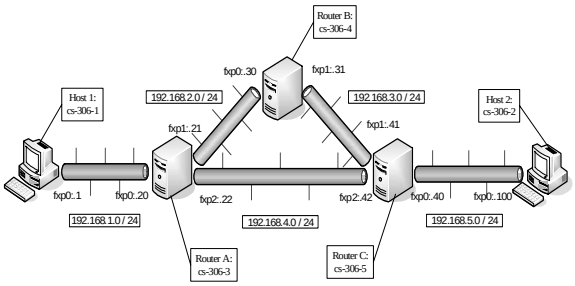

Fig. 3: Topology used in the lab to test the functionality of the compound routing metric introduced by modifying ***ripd***. All nodes marked "Router" run the modified version of Quagga.

The general idea of the topo;ogy shown in Fig. 3 was to increase the round-trip time between Router A and C such that the Quagga RIP daemon with the compound metric direct traffic between the two host PCs through route A→B→C rather than route A→C.

The network traffic generator "traffic" [17] was used to generate tcp traffic between Router A and C. Ten clients were simulated on Router A to connect to Router C simultaneously, each with ten TCP connections and 100 byte payload. Router C, running as server, was responding to the clients by echoing back the payload packets. The average network traffic generated was about 5.7 Mbps. Thus, the wire saturation was about 5.7 Mbps/10Mbps = 57%. The CPU usage of Router A and C running Zebra/ripd and trafserver/trafclient respectively was less than 30%.

The results of a specific scenario are shown in Fig. 4. In this case, (1) Host-1 pinged Host-2 for about 200 seconds at 1 ping/second rate and records the round-trip time of each ping; (2) Between the 21st second and the 170th second, the link between Router A and C was driven to 57% utilization for 150 seconds. Thus load distribution was as follows: 1- 20 seconds – light load; 21-170 seconds – heavy load; 171-200 seconds – light load. Fig.4 shows different behaviors of RIPv2 and RIP_rtt.

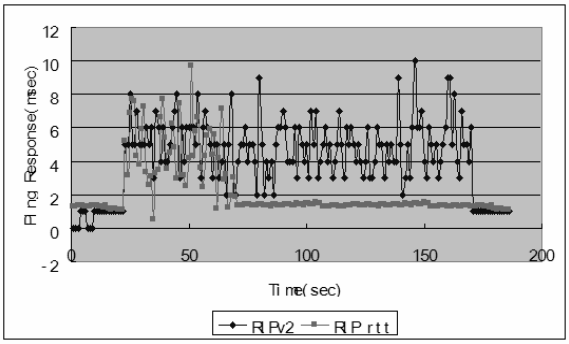

Fig. 4: Ping Response time between Host 1 and Host 2 when the link utilization between Router A and Router C is at 57%: (a) using RIPv2 with hop count as the routing metric [diamonds] (b) using RIPv2 with the compound routing metric combining hop count and rtt [squares].

During the first 20 seconds and the last 20 seconds, RIPv2 and RIP_rtt behaved similarly. In both cases, the average Ping response time was about 1.5 msec. When the link between Router A and C was overloaded, RIPv2 behaved in a random manner, with average Ping response time increased to about 5.0 msec. On the other hand, after being affected for about 60 seconds (which is the default RIP update interval), RIP_rtt managed to stabilize the Ping response time to about 1.5 msec. What actually happened is that as soon as the measurements were

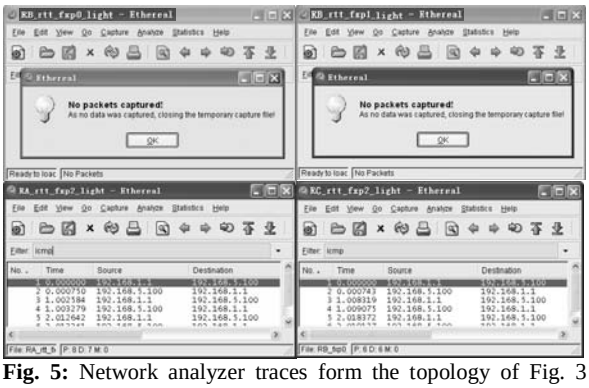

Fig. 5: Network analyzer traces form the topology of Fig. 3 using the RIPv2 with the compound routing metric under light link utilization. Path A→C is used: (a) Interface Router-B-fxp0 (top left panel), (b) Interface Router-B-fxp1 (top right panel), (c) Interface Router-A-fxp2 (bottom left panel), (d) Interface Router-C-fxp2 (top left panel).

propagated, in the case of the compound metric a switch to path A→B→C took place. Since, only path A→C experienced the increased load, ping's response time at the 80th second dropped at the same level as before the application of the load (Fig. 4, trace with squares). In the case that hop-count alone was the metric, no such switch occurred, hence the increased average response time.

The switch from path A→C to path A→B→C under heavy load during the use of the compound routing metric was confirmed by means of network analyzers (ethereal) monitoring network activity at the various router interfaces. Fig. 5 illustrates the case of light link utilization. In this case, path A→C was used when Host 1 pinged Host 2. Therefore, the network analyzers monitoring interfaces fxp0 (Fig. 5a) and fxp1 (Fig. 5b) of Router B did not record any traces, but the network analyzers monitoring interfaces fxp2 (Fig. 5c) of Router A and fxp2 (Fig. 5b) of Router C did record the trace associated with pinging.

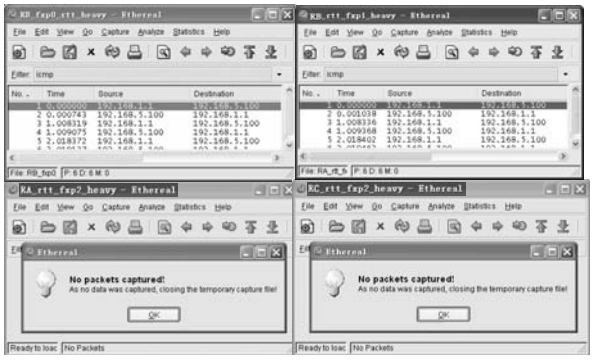

Fig. 6: Network analyzer traces form the topology of Fig. 3 using the RIPv2 with the compound routing metric with 57% link A-C utilization. Path A→B→C is used: (a) Interface Router-B-fxp0 (top left panel), (b) Interface Router-B-fxp1 (top right panel), (c) Interface Router-A-fxp2 (bottom left panel), (d) Interface Router-C-fxp2 (top left panel).

Fig. 6 illustrates the case of 57% link utilization in the A-C link. In this case, path A→B→C was used when Host 1 pinged Host 2. Therefore, the network analyzers monitoring interfaces fxp0 (Fig. 6a) and fxp1 (Fig. 6b) of Router B did record the trace associated with pinging, but the network analyzers monitoring interfaces fxp2 (Fig. 6c) of Router A and fxp2 (Fig. 6b) of Router C did not record any traces.

4 Discussion

A modification to Quagga's ***rip*** module was presented to allow the use compound routing metrics for research and education purposes. It extended the data structures of Quagga's rip module to accommodate storage and propagation of multiple measurements, it introduced a module for implementing metric derivation rules, and created a stub module with the hooks for performing additional measurements as needed. An application of this implementation was also presented using hop-count and RTT as measurements to propagate and a set of metric derivation rules to populate the routing table. The application was successfully tested using the topology of Fig. 3. This work will enable us to further pursue research and education activities in designing and testing compound routing metrics and we expect to make it available in the public domain soon.

References:

[1]Comer, D. *Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architectures (4th Edition)*. Prentice Hall, 2000.

[2]Halabi, B. *Internet Routing Architectures (2nd Edition)*. Cisco Press, 2000.

[3]Perlman, R. *Interconnections: Bridges and Routers.* Addison-Wesley, 2000.

[4]Moy, J. *OSPF Complete Implementation.* Addison-Wesley, 2000.

[5]Handley, M. *XORP: An Open Platform for Network Research.* XORP-Project, 2002 (http://www.xorp.org/papers).

[6]Helmy, A.. *Protocol Independent Multicast-Sparse Mode (pim-sm) Implementation Document*, 1996. (http://netweb.usc.edu/pim)

[7]GateD Project. (http://www.gated.org)

[8]MRTD Project. (http://www.mrtd.org)

[9]BIRD Project. (http://bird.network.cz)

[10]XOPR Project. *International Computer Science Institute, Berkeley, CA.* (http://www.mrtd.org)

[11]GNU Zebra Project. (http://www.zebra.org)

[12]GNU Quagga Project. (http://www.quagga.org)

[13] Fedyk, D., A. Ghanwani, R. Balay, J. Ash, and A. Verdrenne. *Multiple Metrics* for Traffic Engineering with IS-IS and *OSPF*. IETF Draft, November 2000.

[14]Cisco EIGRP Metrics. http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a0080094cb7.shtml#eigrpmetrics

[15]Paxon, V., G. Almes, J Mahdavi, and M. Mathis. Framework for IP Performance Metrics. *RFC2679*, September 1999.

[16]Chan, G., H Alnuweiri, and V. Leung. A Framework for Optimizing the Cost and Performance of Next-Generation IP Routers. *IEEE J. Set. Areas in Comm.*, vol. 17 (6), June 1999, pp. 1013-1029.

[17]TRAF - SourceForge traffic Project. (http://sourceforge.net/projects/traffic)