CS495P MAIN PROJECT REPORT

Tools for effective load balancing

SUBMITTED IN PARTIAL FULFILMENT OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

by

Rakesh Krishnan Y2039
Samuel John Y2050
Vinu P Y2059

Under the guidance of

Mr. Saidalavi Kalady

Department of Computer Engineering
National Institute of Technology, Calicut
2005,Monsoon Semester

National Institute of Technology, Calicut
Department of Computer Engineering

Certified that this Main Project Report entitled

Tools for effective load balancing

is a bonafide report of the work done by

Rakesh Krishnan Y2039
Samuel John Y2050
Vinu P Y2059

in partial fulfilment of the
Bachelor of Technology Degree

Mr. Saidalavi Kalady
Senior Lecturer
Dept. of Computer Engineering

Dr. M.P. Sebastian
Professor and Head
Dept. of Computer Engineering

Acknowledgement

I thank Mr Saidalavi Kalady, senior lecturer, Department of computer science and engineering, for his guidence and co-operation in the completion of this project. I also acknowledge the advice and guidence given to me by my friends and classmates.

Rakesh Krishnan Y2039
Samuel John Y2050
Vinu P Y2059

Abstract

Multihoming is a technique to increase the reliability and QoS of an internet connection using multiple network links. A network is said to be multihomed if it has more than one path to the global internet via multiple ISPs. Our aim is to develop a linux based multihoming solution that does outgoing load balancing which includes development of tools for estimating path characteristics and a userspace daemon process which provides the kernel with the necessary data.

Contents

1 Introduction

The project essentially deals with the development of a set of tools to aid in implement multi-homing solution. The outgoing load balancer routes outgoing traffic based on the characteristics of the paths to the destination. The path characteristics are prioritized based on the application level protocol of the packet while making the routing decision. The path characteristics that we consider are available bandwidth, capacity and delay. If these path characteristics are not available, then the characteristics of the first hop links to the ISPs are considered.

2 Problem Specification

To develop an effective solution for outgoing load balancing. We need to develop a tool for estimating the path characteristics like capacity, available bandwidth, delay and also a userspace daemon for communication with the kernel and also for executing the tools. The data collected by the tools is sent to the kernel by the userspace process which will then lead to the effective routing of outgoing packets.

3 Literature Survey

The following topics were studied in detail in association with the project and the best methods were arrived upon from them.

3.1 Caching

Possible methods of caching were considered. Red-Black tree was found to be the most optimum implementation method as it could search for a value with minimum complexity. There is a source-destination cache at the kernel and a two level cache at the userspace daemon, both of which are implemented using red-black trees.

3.2 VPS - Variable Packet Size Probing

Various methods for estimating path capacity were studied and a variant of a method called variable packet size probing technique (VPS) was chosen. This method is based on the assumption that the latency of each hop, known as the serialization latency, is equal to packet size divided by the link capacity of the hop. The end to end capacity of the path is the minimum per-hop capacity.

3.3 Determination of available bandwidth

The available bandwidth estimation was done using the packet train method. Here a train of packets with some initial delay is sent and later on the delay is increased and the changes in output dispersion are noted.

3.4 Netlink sockets

Netlink sockets are used for the communication of the userspace process with the kernel. Study about the various modes of communication with the kernel was done.

4 Motivation

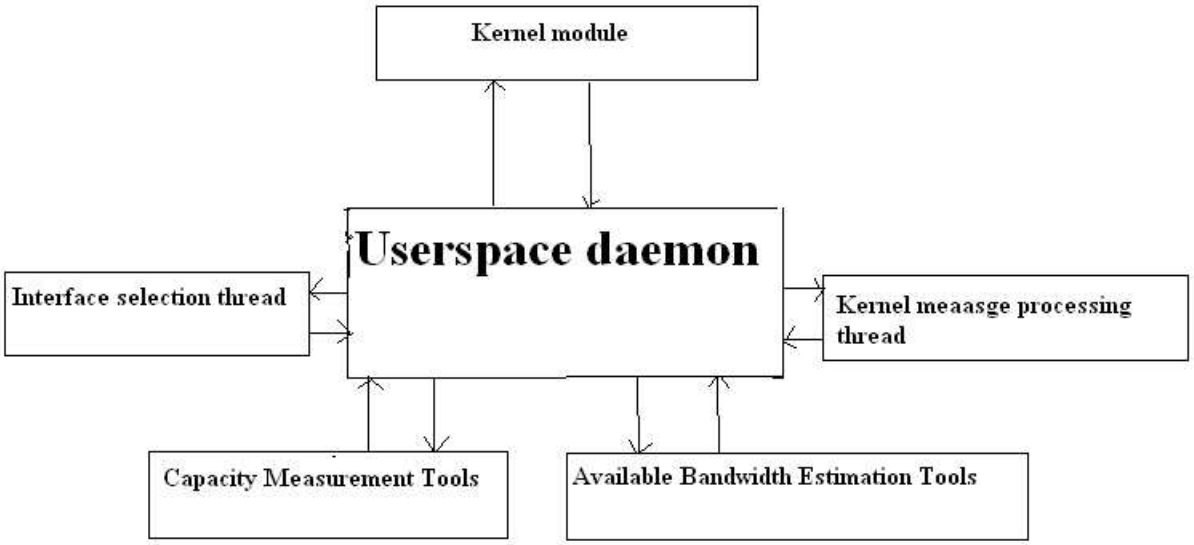
The following are some of the reasons why an enterprise might want a multihomed network:—Multihoming helps to minimize downtime due to internet connection failure and ensures reliable internet services. When connectivity to the internet through one of the upstream ISPs is lost in the event of link failure, packets can be routed over the remaining links.—Policy based routing can be used to choose the best link for a connection based on the link characteristics and the application protocol. For example, ssh and telnet traffic can be routed over the link with minimum delay to the destination, while ftp traffic can be routed over the link that provides maximum bandwidth on the path to the destination.—Having multiple broadband links to different ISPs is more economical than a single high bandwidth line. As the reliability of inexpensive broadband links improves, multihoming solutions will see widespread adoption.—A multihomed solution helps to distribute the load over multiple links. This could enable geographically distributed enterprises to route packets through the nearest gateway.

5 System Design and Architecture

The system essentially consists of two parts.–A userspace daemon process–A path characteristics estimation tools

5.1 Userspace Daemon

The architecture of the uyserspace daemon could be represented as shown below.



As it can be seen the main purposes of the userspace daemon are

1. Receive destinations from the kernel and inform the kernel about the best interface to use for a particular policy.
2. Running the path characteristics estimation tools continuously, i.e, the daemon runs the tools and finds the interface having the maximum capacity, minimum latency and maximum available bandwidth for a all the destinations that are there in the L1 cache.
3. For using the link characteristics it constantly determines the interface having the maximum bandwidth and notifies the kernel if there is any change to it

The userspace daemon receives destinations sent by the kernel and stores them in a 2- level cache. The most frequently visited destinations are stored in the level 1 (L1) cache. The daemon calculates the path characteristics (available bandwidth, capacity and delay) of the destinations in the L1 cache and sends this data to the kernel. The L1 cache is kept in sync with the kernel's destination cache. If the path characteristics for a destination in L1 cache are not available, the link characteristics are used instead.

The less frequently accessed destinations are stored in a level 2 (L2) cache. The L1 and L2 caches are implemented as red-black trees ordered on the number times the corresponding destination was visited. The size of the L1 cache is fixed, while the L2 cache expands as the number of destinations increases. When the visit count of an entry in the L2 cache becomes greater than that of an entry in the L1 cache, the entries are swapped. The cache entry for a destination is timed out if there are no packets to it in a long time. The caches are reaped periodically to remove expired entries. This ensures that the L2 cache does not become too large and that infrequently accessed destinations with high visit count don't monopolize the cache.

All entries in the caches are also stored in a red-black tree ordered on the address to enable efficient searching of entries based on destination address.

The kernel module and the daemon communicate via netlink sockets. Netlink sockets provide a full-duplex, asynchronous, flexible means of communication between the kernel and userspace via the standard socket API. This method is especially suited for high data rate communication as required by this application.

The routing decision is based on the following criteria:

1. Available bandwidth of the path to the destination.
2. Maximum capacity of the path to the destination.
3. Round trip time of the path to the destination

But when a destination is received from the kernel for the first time then we would not be having the path characteristics to that destination. So in such cases we use the link characteristics to decide in taking the routing policies. The following global parameters are considered when the per-destination path characteristics are not available:

1. Available bandwidth of the link with the upstream gateway
2. Capacity of the link with the gateway
3. Round trip time of the link

The userspace daemon calculates continuously the various path characteristics for the entries that are there in the L1 cache and sets appropriately the best interface to use when the appropriate policy is used.

5.2 Path characteristics estimation tools:

As mentioned before the path characteristics that we determine using the tools are maximum available bandwidth, maximum capacity and minimum latency to a particular destination. For the determination of each of them we studied various possible methods and finalized upon the best possible methods given the scenario.

5.2.1 Determining path capacity:

The capacity of a path is the maximum possible bandwidth that the path can deliver. Path capacity is measured using the Variable Packet Size (VPS) probing technique. This method is based on the assumption that the latency of each hop, known as the serialization latency, is equal to packet size divided by the link capacity of the hop. The end to end capacity of the path is the minimum per-hop capacity.

The VPS technique requires the measurement of latency for each hop on the path to the destination. In the VPS method, packets with increasing TTL are sent to the destination, and the time interval between sending these packets and their replies is measured.

The rtt, T_I, to a hop I and packet size L is given by:

$T_I = \sum (L/C_I + q_i + p_i + L_{REPLY}/ C + qr_i + pr_i)$ (1)

C_i, i = 1..I, is the capacity of each hop, q_i and qr_i are the queuing delays of the sent packet and the reply respectively at hop i, while p_i and pr_i are the propagation delays. L_{REPLY}, the size of the ICMP reply packet, is a constant.

From the above equation it can be seen that serialization delay (L/C_i) is the only component that is dependent on packet size L. So the equation can be written as:

$T_I = \sum(L/C_i + D_i)$ (2)

where D_i is independent of L.

ie

$T_I = a_i + b_i L$ (3)

where $b_I = \sum 1/ C_i$

For a packet of size $L^{/}$

$T_I^{/} = a_I + b_I L^{/}$ (4)

From (3) and (4),

$b_I = (T_I^{/} - T_I) / (L^{/} = L)$ (5)

For the next hop

$T_{I+1} = a_I + b_{I+1} L$ (6)

From (3) and (6)

$C_{I+1} = 1/(b_{I+1} - b_I)$ (7)

where b_{I+1} and b_I as calculated using (5).

In this method, a number of packets of increasing sizes are sent to each hop, and the value of b_I is estimated using a linear regression algorithm. For each packet size, the minimum rtt is taken as this is assumed to have resulted from a packet and a corresponding ICMP reply that did not experience any queuing delay [8]. From this, the capacity of each hop can be determined. The capacity of the path is taken to be the bottleneck capacity.

$C = \sum C_i$ (8)

This method was found to be too slow for the multihoming application as the capacity estimation tool was not found to run fast enough to keep pace with changes in the destination cache. Another problem with this approach is that Layer 2 devices present on the path cause underestimation of hop capacity [7]. So the tool was modified to calculate the end-to-end capacity rather than per-hop capacity. The advantage of not considering intermediate hops is that it is more efficient and consumes less bandwidth than the original method, which finds the capacity of all the intermediate links. Further, the accuracy of this method was found to be comparable to the traditional VPS method

In this case, the size of the reply with vary with change in the size of the request, as in the end-to-end capacity measurement, an ICMP echo reply is generated whose size is generally equal to the size of the corresponding ICMP echo request packet. We assume that the forward and return journeys are symmetric.

Hence, latency = rtt/2 (9)
where latency is the one-way from source to destination.
If rtt_1 and rtt_2 are the round trip times for packets of size s_1 and s_2 respectively,
$rtt_1/2 = L_1/C + k$
$rtt_2/2 = L_2/C + k$
$=> C = L_2 - L_1 / rtt_2 - rtt_1$ (10)
ICMP echo request packets of increasing size are sent to the destination and the rttâs are determined. The end-to-end capacity can then be determined by applying a linear regression algorithm on the capacity estimates obtained using the formula (10)

5.2.2 Determining minimum latncy:

Round trip time is the time between the sent request and the received response. The round trip time of the path is calculated along with the measurement of the capacity of the path. It is assumed to be the minimum of the round trip times of the packets sent by the VPS method. The minimum round trip time is used as this is considered to be round trip time of the path under ideal circumstances.

It is assumed that the ICMP echo request and response packets take the same path. The round trip time is thus twice the latency of the path to the destination.

5.2.3 Determining maximum available bandwidth:

The available bandwidth of a path is the maximum unused bandwidth on the path . It is the minimum of the available bandwidths of the links constituting the path. Available bandwidth calculation is based on the principle that the dispersion between back-to-back packets increases when they go from a higher bandwidth link to a lower bandwidth link, but the dispersion does not change when such packets pass from a link with lower bandwidth to one with higher bandwidth (fig1). The dispersion between two packets is the time distance between the last bits of each packet. If a train of back-to-back ICMP echo request packets is sent to a destination, the dispersion between the replies will be minimum dispersion between the packets, seen on the link with minimum available bandwidth on the path. This is the basis of the packet-train technique.

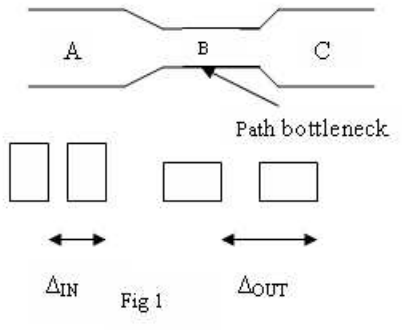


Fig 1

If d_i is dispersion between packets (of size L) in the packet train on link i of capacity B_i, then the dispersion on the next link

$d_{i+1} = \max(d_i, L/B_i)$

The dispersion measured by the receiver

$d = \max(L/B_i) = L/\min(B_i) = L/B$

$B = L/d$ (11)

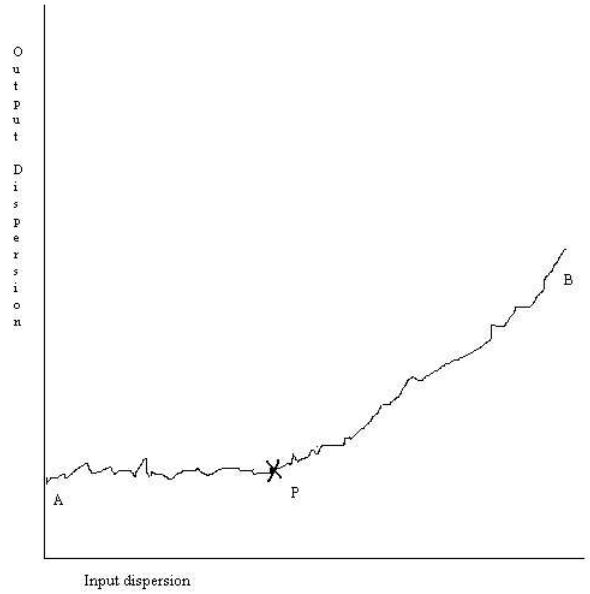
where B is the link with minimum available bandwidth.

In our implementation, we send trains of ICMP echo requests with increasing input dispersion until the output dispersion starts increasing steadily with increase in the input dispersion (fig3). Packets within a train that have abnormal dispersion and those that arrive out of order are discarded. Trains with a large number of aberrant packets are also discarded.

To find the point at which the graph starts increasing, we find the largest decreasing subsequence such that:

(length of subsequence) / (number of elements in subsequence) $<$ threshold value

This is done to differentiate between the regions AP and PB in Fig3. In AP, the output dispersion does not vary consistently with increase in input dispersion, while in PB, the output dispersion increases steadily with increase in input dispersion. The last point in this subsequence corresponds to the maximum available bandwidth on the link.



5.2.4 Determining the available bandwidth on a link:

The bandwidth usage for a link is calculated by finding the number of bytes of data sent or received in an interval. This is done periodically and the available bandwidth is calculated by subtracting the bandwidth consumed from the capacity of the link.

bandwidth used = (tx + rx - old_tx - old_rx) / interval

available bandwidth = capacity - bandwidth used

where

tx, rx: number of bytes of data that have been sent and received on the link since the link was initialized

old_tx, old_rx: previous values of tx and rx.

interval: time interval between measurements of tx and rx

6 Implementation

6.1 Estimation of Path Charecteristics

6.1.1 Estimation of latency and capacity:

A sequence of ICMP packets are sent to a destination after obtaining the ip address. For this we create a raw socket for communication. Multiple packets, of the same size are sent to the destination and as we go on, we keep on increasing the size of the packets till a threshold value which is set by the user. The packets are identified by a unique icmp sequence number which is assigned when the icmp packet is created . The send time of each packet is taken as the system time when the packet is sent and is obtained by the function gettimeofday() . The corresponding icmp replay could be identified by the sequence number and the receive time is read from the ancillary data associated with the packet . A user defined structure pckt_info is used to store the information regarding a packet. Form the send and receive times round trip time and latency are calculated. From the obtained latencies and packet sizes the path capacity is measured . The minimum latency is noted from the obtained set of latencies.

6.1.2 Estimation of available bandwidth:

Here we send a train of packets of the same size with an initial delay(input dispersion). Dispersion refers to the time distance between two consecutive packets .Then as we proceed we go on increasing the input dispersions and analyze the output dispersions. While analyzing the output dispersion we consider only the in order packets in a train. If sufficient no of in order replies are not received we attempt a fixed no of retries with the same train and input dispersion. The output dispersion corresponding to a particular train of packets is the median of the dispersion between consecutive packets in that train. We keep on sending such packet trains till we get somewhat a constant dispersion. The point at which there is a steady increase in output dispersion with input dispersion is used for calculating the available bandwidth.

6.2 Userspace daemon

There are mainly three threads(functionalities) associated with the userspace daemon. They are,

1. A thread for processing kernel messages
2. A thread for doing the ping function , i.e. for calculating path characteristics of the destinations that are in the cache.
3. A thread for selecting the interface with the maximum available bandwidth.

6.2.1 Processing kernel messages:

It receives destinations from the kernel and makes appropriate changes in the cache. The communication between the kernel and the daemon is established through netlink sockets. The messages are sent to the kernel using a function send_command. The cache at the userspace is ordered on the no of times the destinations are visited.

When a destination is received from the kernel the new entry that is to be inserted is initialized. Then this entry is inserted into the addresstree. If we are seeing this destination for the first time then we insert it into the L1 cache if there is space, else into the L2 cache. If the destination is already in the cache then the count is incremented and is inserted into the proper position in the cache, may be in L1 or L2. The 2 caches are then appropriately modified. Then caches are also reaped periodically.

6.2.2 Ping Thread:

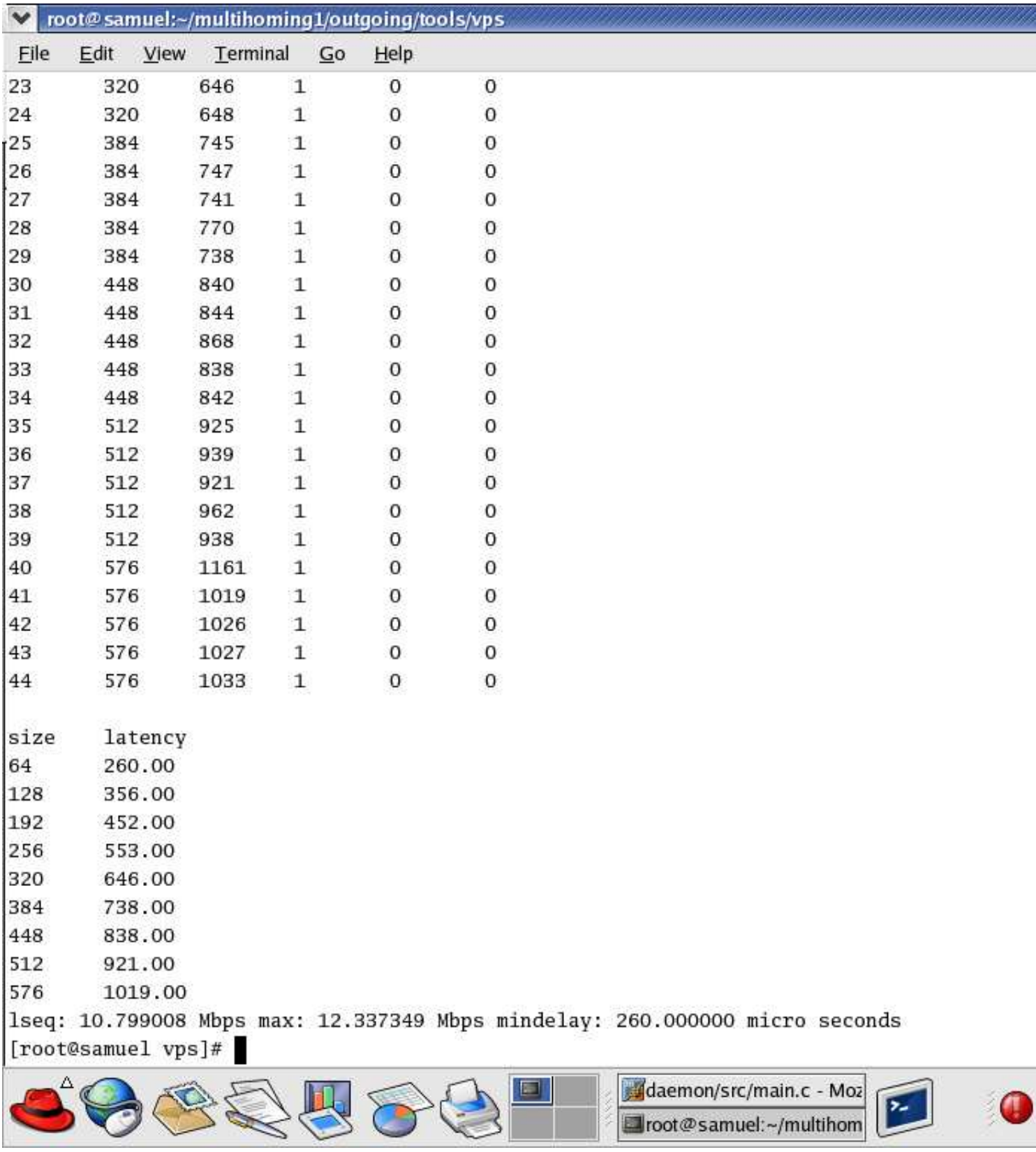
For each destination in the cache , the path characteristic measurement tools are made to run continuously on each interface in order to obtain the latest statistics . Then the minimum delay , maximum capacity and the maximum available bandwidth interfaces are stored for each entry in the cache . Depending up on the policy we select the interface for each destination.

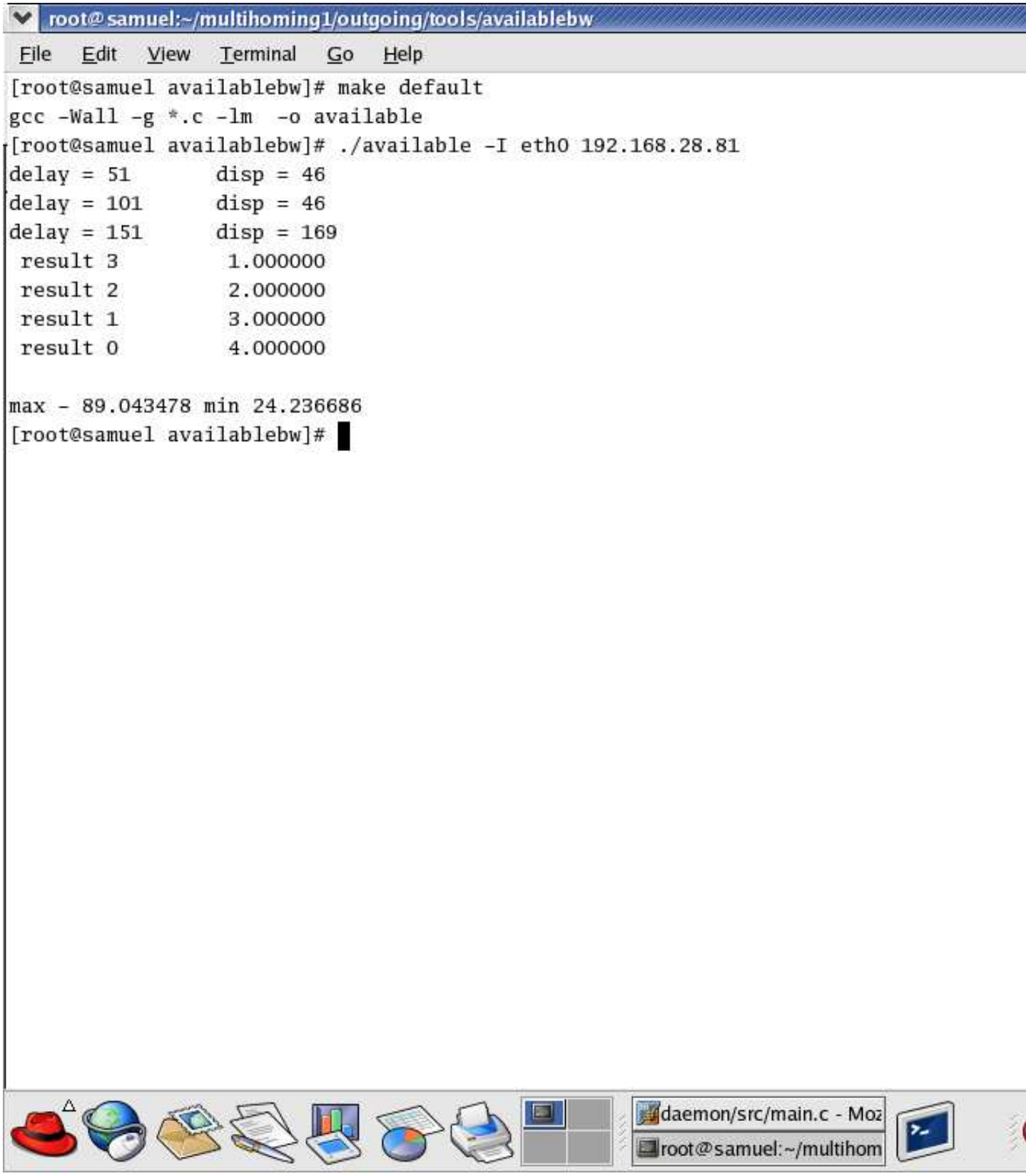
For each entry in the destination cache a function called rtt is called.i.e, it will be in an infinite loop. Then two threads are forked from the parent, and are execd, one for running the path capacity estimation tool and another for running the maximum available bandwidth estimation tool. This runs on all the interfaces and then finally the child threads are killed after getting the interfaces having the maximum values for each of the characteristics. If we find that for a particular policy an interface is not suitable then the necessary changes are made. Then finally the kernel is updated about the path characteristics of that particular destination.

6.2.3 Interface selection thread:

Since the usage is bound to change from time to time we need to estimate the bandwith for each interface regularly.This thread reads in the usage and estimate the available bandwith for the interface and selects the one with the maximum bandwith and sends it to kernel.

7 Testing


root@samuel:~/multihoming1/outgoing/tools/vps
File Edit View Terminal Go Help
23 320 646 1 0 0
24 320 648 1 0 0
25 384 745 1 0 0
26 384 747 1 0 0
27 384 741 1 0 0
28 384 770 1 0 0
29 384 738 1 0 0
30 448 840 1 0 0
31 448 844 1 0 0
32 448 868 1 0 0
33 448 838 1 0 0
34 448 842 1 0 0
35 512 925 1 0 0
36 512 939 1 0 0
37 512 921 1 0 0
38 512 962 1 0 0
39 512 938 1 0 0
40 576 1161 1 0 0
41 576 1019 1 0 0
42 576 1026 1 0 0
43 576 1027 1 0 0
44 576 1033 1 0 0
size latency
64 260.00
128 356.00
192 452.00
256 553.00
320 646.00
384 738.00
448 838.00
512 921.00
576 1019.00
lseq: 10.799008 Mbps max: 12.337349 Mbps mindelay: 260.000000 micro seconds
[root@samuel vps]#
daemon/src/main.c - Moz
root@samuel:~/multihom


root@samuel:~/multihoming1/outgoing/tools/availablebw
File Edit View Terminal Go Help
[root@samuel availablebw]# make default
gcc -Wall -g *.c -lm -o available
[root@samuel availablebw]# ./available -I eth0 192.168.28.81
delay = 51 disp = 46
delay = 101 disp = 46
delay = 151 disp = 169
result 3 1.000000
result 2 2.000000
result 1 3.000000
result 0 4.000000
max - 89.043478 min 24.236686
[root@samuel availablebw]#
daemon/src/main.c - Moz
root@samuel:~/multihom

8 Conclusion

Tools for estimating path characteristics like path capacity, latency, available bandwidth were developed and tested. A userspace daemon process capable of communicating with the kernel and executing the tools was also developed.

The outgoing load balancer does policy-based routing to choose the best link for each type of traffic. It calculates the characteristics of the paths via each of the ISPs to the most frequently accessed destinations. The best link is chosen based on the path characteristic that is relevant to the application protocol of the packet, as determined by the user-defined policy. For infrequently visited destinations, a global policy that considers the first hop connectivity to the ISPs is used.

The path characteristics considered are available bandwidth, capacity and delay. Algorithms used to calculate these parameters, and the tradeoffs involved, are discussed.

9 Future work

Possible extensions for the current project would include developing and adding a kernel module for extracting the packet of each packet and for doing policy based routing. A tool capable of checking dead gateways and notifying the caches about it also could be developed.

10 References

1. F. Guo, J. Chen, W. Li,T. Chiueh, "Experiences in Building a Multihoming Load Balancing System" IEEE Infocomm, 2004

2. Netfilter http://www.netfilter.org

3. Linux man pages

4. Akella et al âA Measurement-Based Analysis of Multihomingâ ACM SIGCOMM 2003

5. R. S. Prasad et al âBandwidth estimation: metrics, measurement techniques, and toolsâ

6. SCTP for fault tolerance and load balancing by Armado L.Caro,Janardhan Iyengar,ACM SIGCOMM Computer Communications Review, July 2002