

Scavenger performance

Cern External Network Division - Caltech

Sylvain.ravot@cern.ch

Datagrid WP7 - 24 January, 2002

Introduction

•**Introduction :**

Qbone Scavenger Service (QBSS) is an additional best-effort class of service. A small amount of network capacity is allocated for this service; when the default best-effort capacity is underutilized, QBSS can expand itself to consume the unused capacity.

•**Goal of our test :**

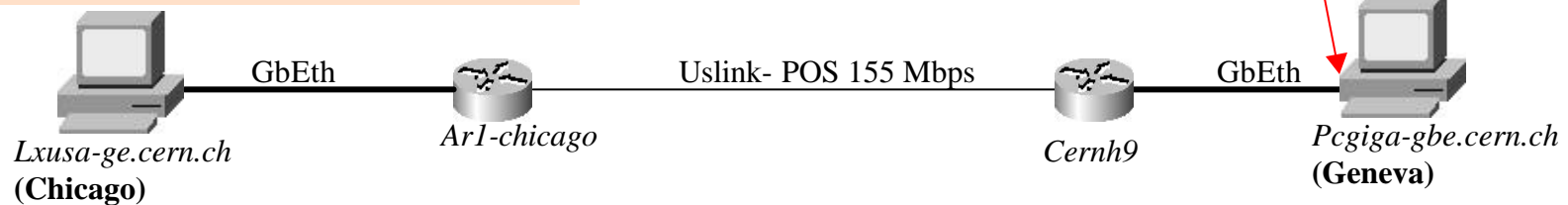
- Does the Scavenger traffic affect performance of the normal best effort traffic?
- Does the Scavenger Service use the whole available bandwidth?

Tests configuration

CERN<->Chicago

- RTT : 116 ms
- Bandwidth-delay-product : 1.9 MBytes.

QBSS traffic is marked with DSCP 001000 (⇔ Tos Field 0x20)



TCP and UDP flows were generated by Iperf.
QBSS traffic is marked using the TOS option of iperf :
`iperf -c lxusa-ge -w 4M -p 5021 --tos 0x20`

Cernh9 configuration :

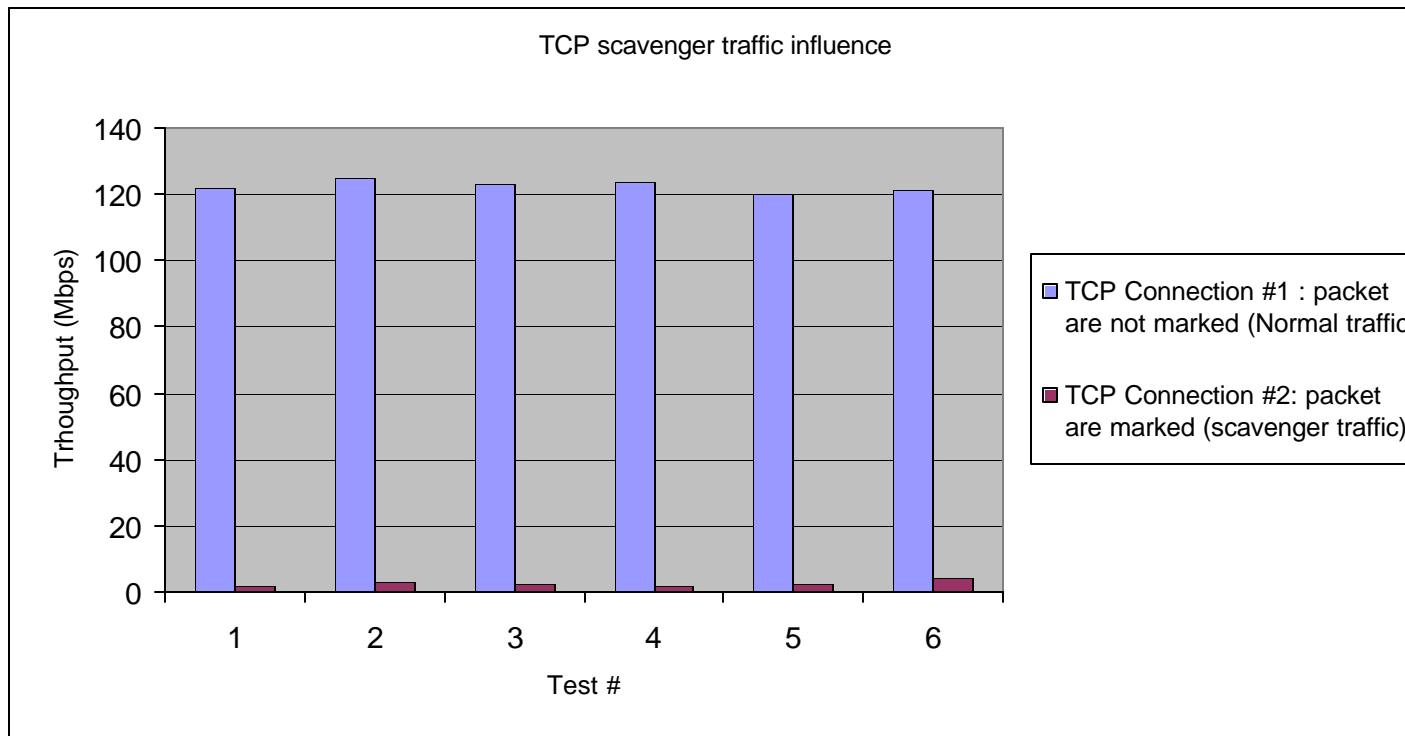
```
policy-map match-all qbss  
  match ip dscp 8
```

```
policy-map qos-policy  
  class qbss  
    bandwidth percent 1  
    queue-limit 64  
    random-detect  
  class class-default  
    random-detect
```

```
interface ...  
  service-policy output qos-policy
```

Scavenger and TCP traffic (1)

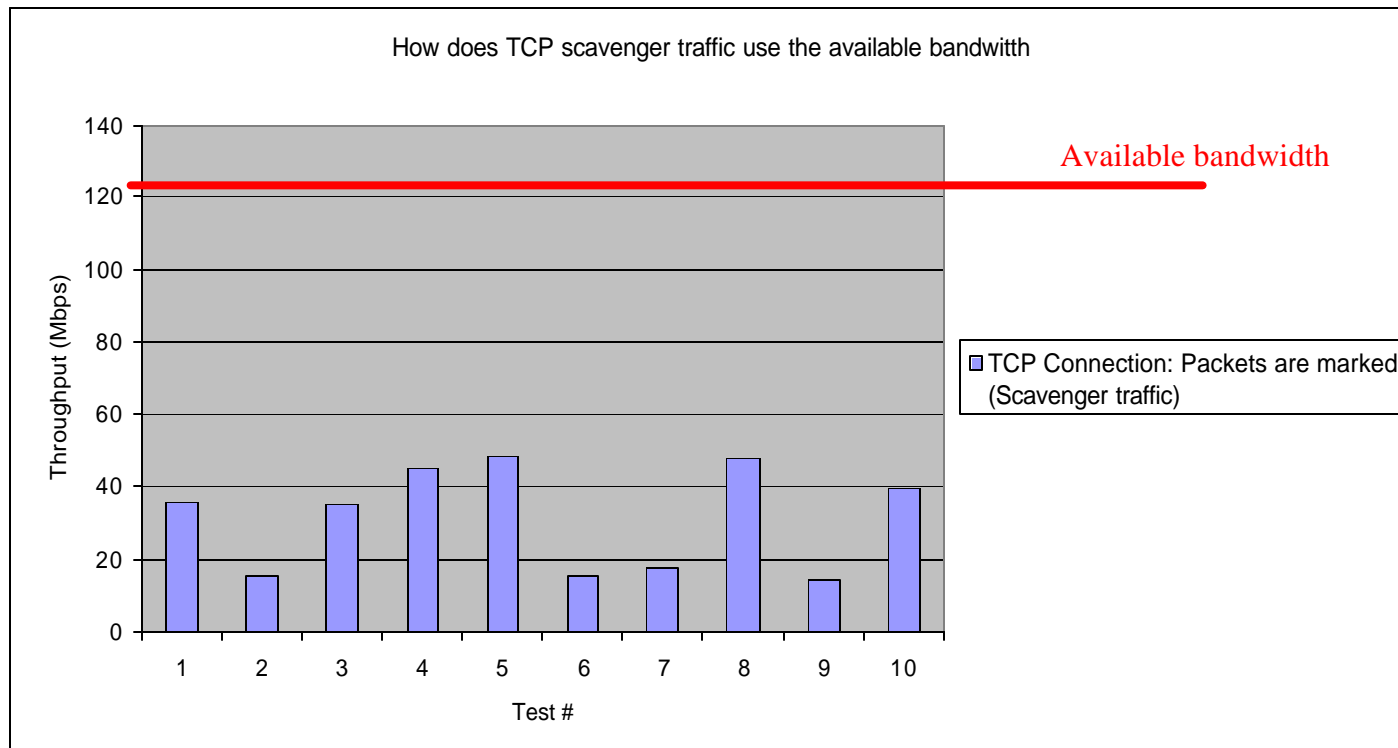
- We ran two connections at the same time. Packets of connection #2 were marked (scavenger traffic) and packets of the connection #1 were not marked. We measured how the two connections shared the bandwidth.



- TCP scavenger traffic doesn't affect TCP normal traffic. Packets of connection #2 are dropped by the scavenger service, so the connection #2 reduces its rate before affecting the connection #1. The throughput of the connection #2 remains low because the loss rate of the scavenger traffic is high.

How does TCP Scavenger traffic use the available bandwidth?

- We performed TCP scavenger transfer when the available bandwidth was larger than 120 Mbps. We measured the performance of the scavenger traffic.



- We performed the same tests without marking the packets. We had a throughput larger than 120 Mbps.
- TCP scavenger traffic doesn't use the whole available bandwidth. Even if there is no congestion on the link, some packets are dropped by the router. It is probably due to the small size of the queue reserved for scavenger traffic (**queue-limit 64**).

Conclusion

- TCP Scavenger traffic doesn't affect normal traffic. TCP connections are very sensitive to loss. When congestion occurs, scavenger packets are dropped first and the TCP scavenger source immediately reduces its rate. Therefore normal traffic isn't affected.
- Scavenger traffic expands to consume unused bandwidth, but doesn't use the whole available bandwidth.
- Scavenger is a good solution to transfer data without affecting normal (best effort) traffic. It has to be kept in mind that scavenger doesn't take advantage of the whole unused bandwidth.

• **Future Work**

- Our idea is to implement a monitoring tool based on Scavenger traffic. We could generate UDP scavenger traffic without affecting normal traffic in order to measure the available bandwidth.
 - Can we use the Scavenger Service to perform tests without affecting the production traffic?
 - Does the Scavenger traffic behave as the normal traffic when no congestion occurs?

Load balancing performance

Cern External Network Division - Caltech

Sylvain.ravot@cern.ch

Datagrid WP7 - 24 January, 2002

Introduction

Load balancing allows to optimize resources by distributing traffic over multiple paths for transferring data to a destination. Load balancing can be configured on a per-destination or per-packet basis. On Cisco routers, there are two types of load balancing for CEF (Cisco Express Forwarding) :

- Per-Destination load balancing
- Per-Packets load balancing

Per-Destination load balancing allows router to use multiple paths to achieve load sharing. Packets for a given source-destination pair are guaranteed to take the same path, even if multiple paths are available.

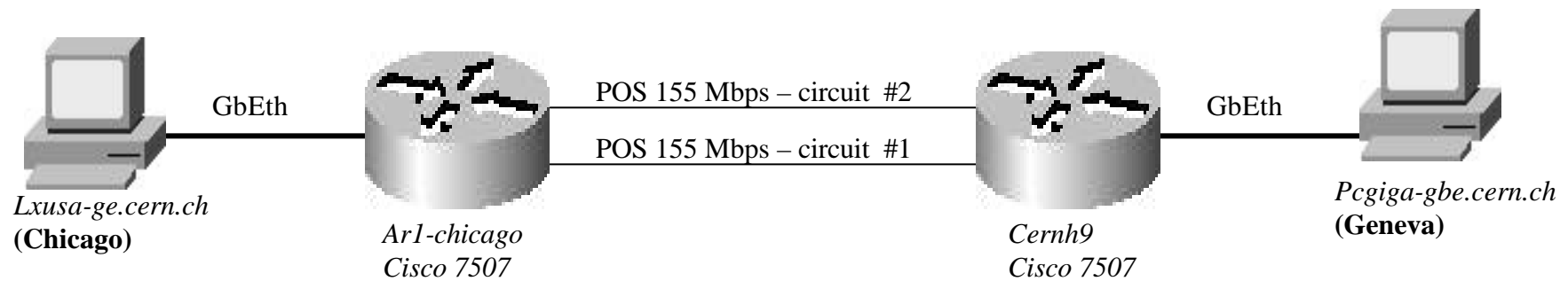
Per-Packets load balancing allows router to send successive data packets without regard to individual hosts. It uses a round-robin method to determine which path each packet takes to the destination.

We tested the two types of load balancing between Chicago and CERN using our two STM-1 circuits.

Configuration

CERN<->Chicago

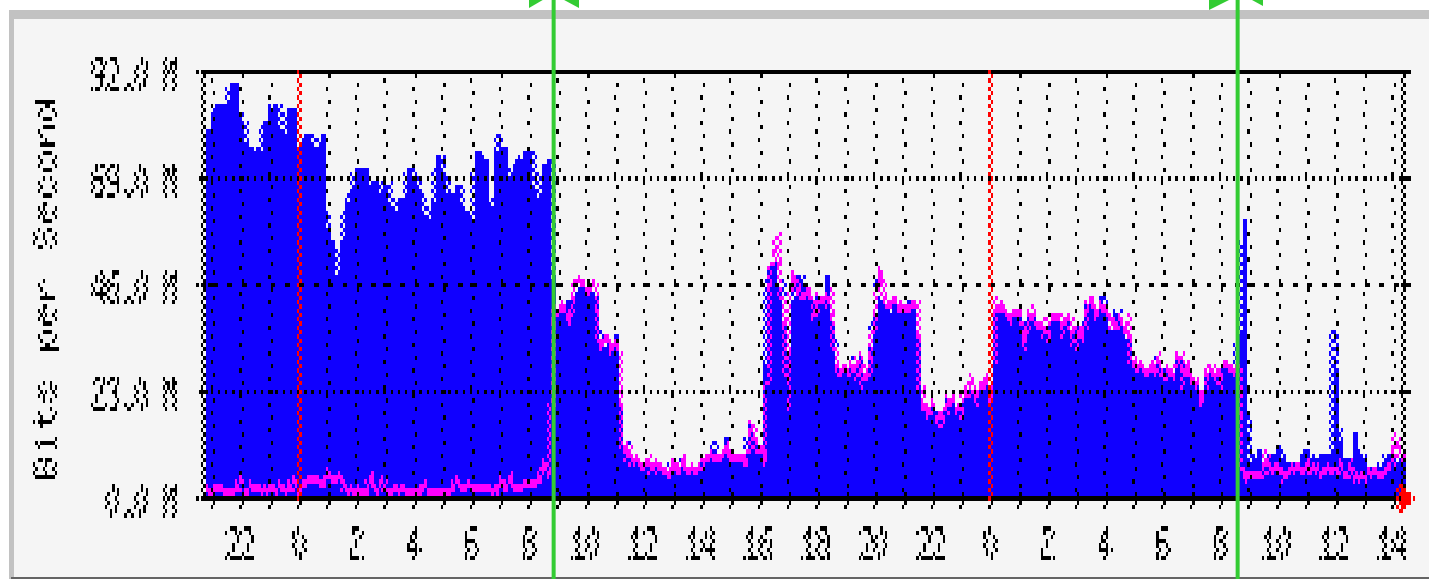
- RTT : 116 ms
- Bandwidth-delay-product : $2 * 1.9$ MBytes.



Load balancing : Per Destination vs Per Packets

- MRTG report: traffic from Chicago to CERN

Load Balancing type: Per Destination Per Packets Per Destination



- Traffic from Chicago to CERN on the link #1
- Traffic from Chicago to CERN on the link #2

When the bulk of data passing through parallel links is for a single source/destination pair, per-destination load balancing overloads a single link while the other link has very little traffic. Per-packet load balancing allows to use alternate paths to the same destination.

Load Balancing Per-Packets and TCP performance

•UDP flow (Cern -> Chicago):

Cern:

```
[sravot@pcgiga sravot]$ iperf -c lxusa-ge -w 4M -b 20M -t 20
```

```
[ ID] Interval   Transfer   Bandwidth
[ 3] 0.0-20.0 sec 50.1 MBytes 21.0 Mbits/sec
[ 3] Sent 35716 datagrams
```

Chicago:

```
[sravot@lxusa sravot]$ iperf -s -w 4M -u
```

```
[ ID] Interval   Transfer   Bandwidth   Jitter  Lost/Total Datagrams
[ 3] 0.0-20.0 sec 50.1 MBytes 21.0 Mbits/sec 0.446 ms  0/35716 (0%)
[ 3] 0.0-20.0 sec 17795 datagrams received out-of-order
```

50 % of the packets are received out-of-order.

•TCP flow (Cern -> Chicago):

```
[root@pcgiga sravot]# iperf -c lxusa-ge -w 5700k -t 30
```

```
[ ID] Interval   Transfer   Bandwidth
[ 3] 0.0-30.3 sec 690 MBytes 191 Mbits/sec
```

By using TCPtrace to plot and summarize TCP flows which were captured by TCPdump, we measured that 99,8 % of the acknowledgements are selective (SACK).

The performance is quite good even if packets are received out of order. The SACK option is efficient. However, we were not able to get a higher throughput than 190 Mbit/s. It seems that receiving too much out-of-order packets limits TCP performance.

Conclusion and Future work

- **Conclusion**

We decided to remove the load balancing per packets option because it was impacting the operational traffic. Each packets flow going through the Uslink was disordered.

Load balancing per packets is inappropriate for traffic that depends on packets arriving at the destination in sequence.