Network Bandwidth Isolation

Xen Sumit Tokyo 2008

Simon Horman <simon@valinux.co.jp>
Hiroya Inakoshi <inakoshi.hiroya@jp.fujitsu.com>

20th-21st November 2008

This work was partly funded by Ministry of Economy, Trade and Industry (METI) of Japan as the Secure Platform project of Association of Super-Advanced Electronics Technologies (ASET).

Outline

- Part I: Overview
- Part II: Identifying Packets
- Part III: Packet Scheduling

Part I

Overview

Motivation

Fairness

- Wish to ensure that each domain received a fair share of network-related resources
  - As defined by the administrator
- Guard against malicious domains
- Guard against domains that have been infected by a virus

Network-Related Resources

- NIC Bandwidth
- Dom0 CPU
- Dom0 Kernel memory

Network-Related Resources

- NIC Bandwidth
  - How fast packets are being transmitted and received by domUs
- Dom0 CPU
  - How fast packets are being transmitted and received by domUs
- Dom0 Kernel memory
  - How many packets are held in the kernel

Packet Scheduling

- Prioritise packets based on domain
- Drop packets if a domain has too many enqueued

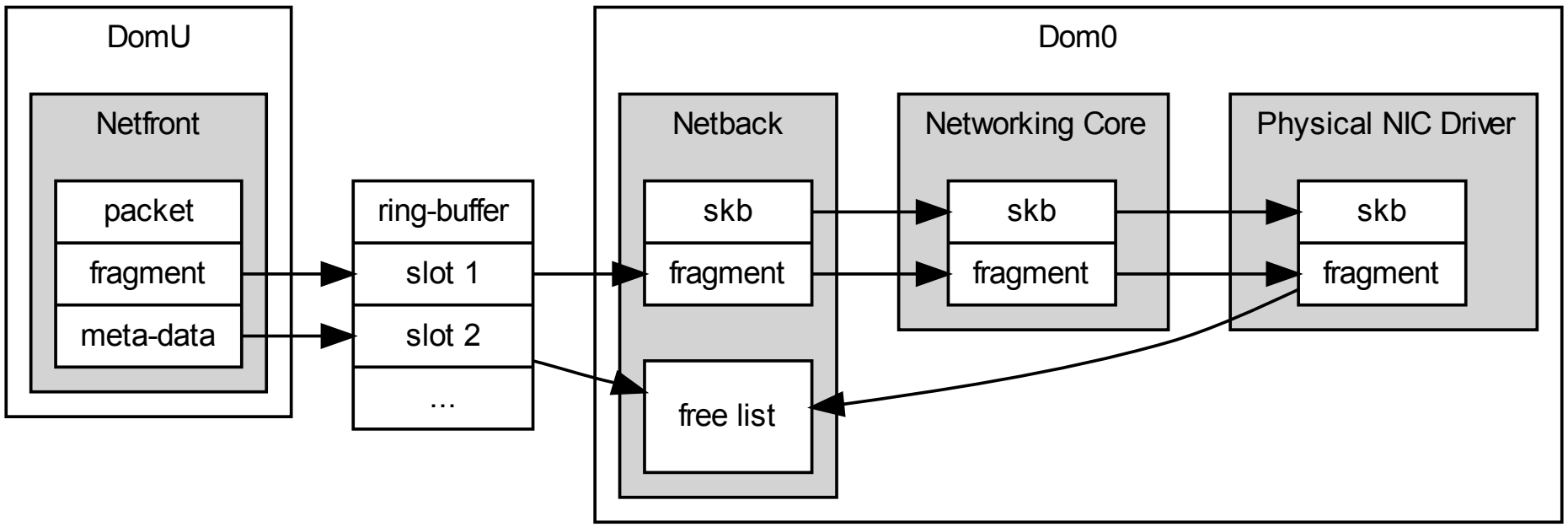
Packet Scheduling

- Prioritise packets based on domain
  - NIC Bandwidth
  - Dom0 CPU
- Drop packets if a domain has too many enqueued
  - Dom0 Kernel memory usage

Netback/Netfront Flow Control

End-to-end flow control from netfront until a packet is transmitted by the destination interface is important as it allows packet scheduling to control network-related resource usage.

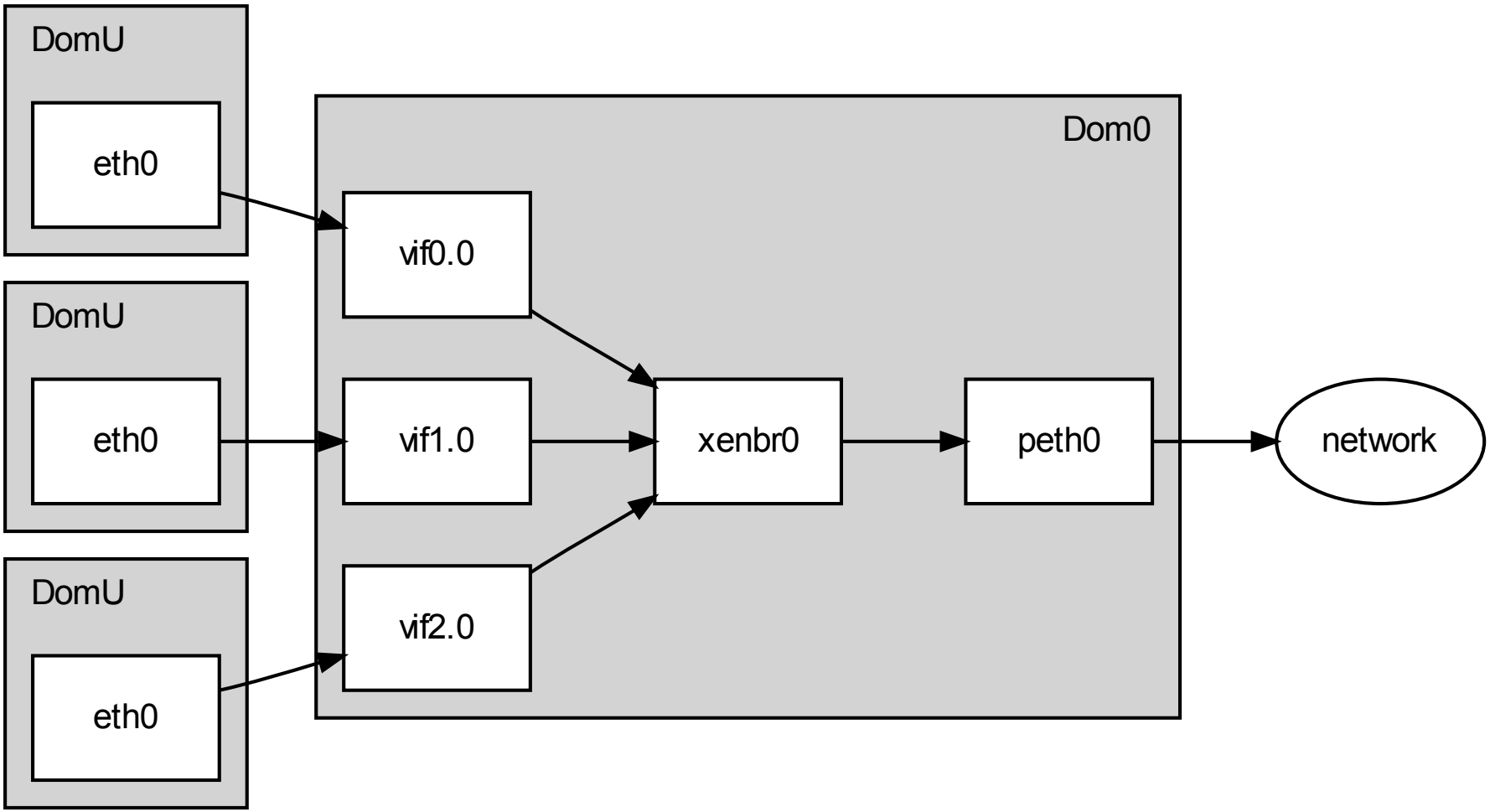
- dom0 CPU
- dom0 Kernel memory



Part II

Identifying Packets

DomU Transmit: Identifying Packets



- Match the interface from which packets enter xenbr0
- Identifies the source-domU

DomU Transmit: `iptables` Rules

Mark the packets according to which interface they arrive on

```
iptables -t mangle -A FORWARD -m physdev \
        --physdev-in vif2.0 -j MARK --set-mark 100
iptables -t mangle -A FORWARD -m physdev \
        --physdev-in vif3.0 -j MARK --set-mark 110
iptables -t mangle -A FORWARD -m physdev \
        --physdev-in vif5.0 -j MARK --set-mark 120
```

Part III

Packet Scheduling

Packet Scheduling

- Filter
  - assign to a class
- Prioritise
  - based on class assignment
  - may selective delay packets
- Queue
  - for transmission after filtering or prioritisation
- Drop
  - if a queue becomes full

Netback/Netfront Flow Control

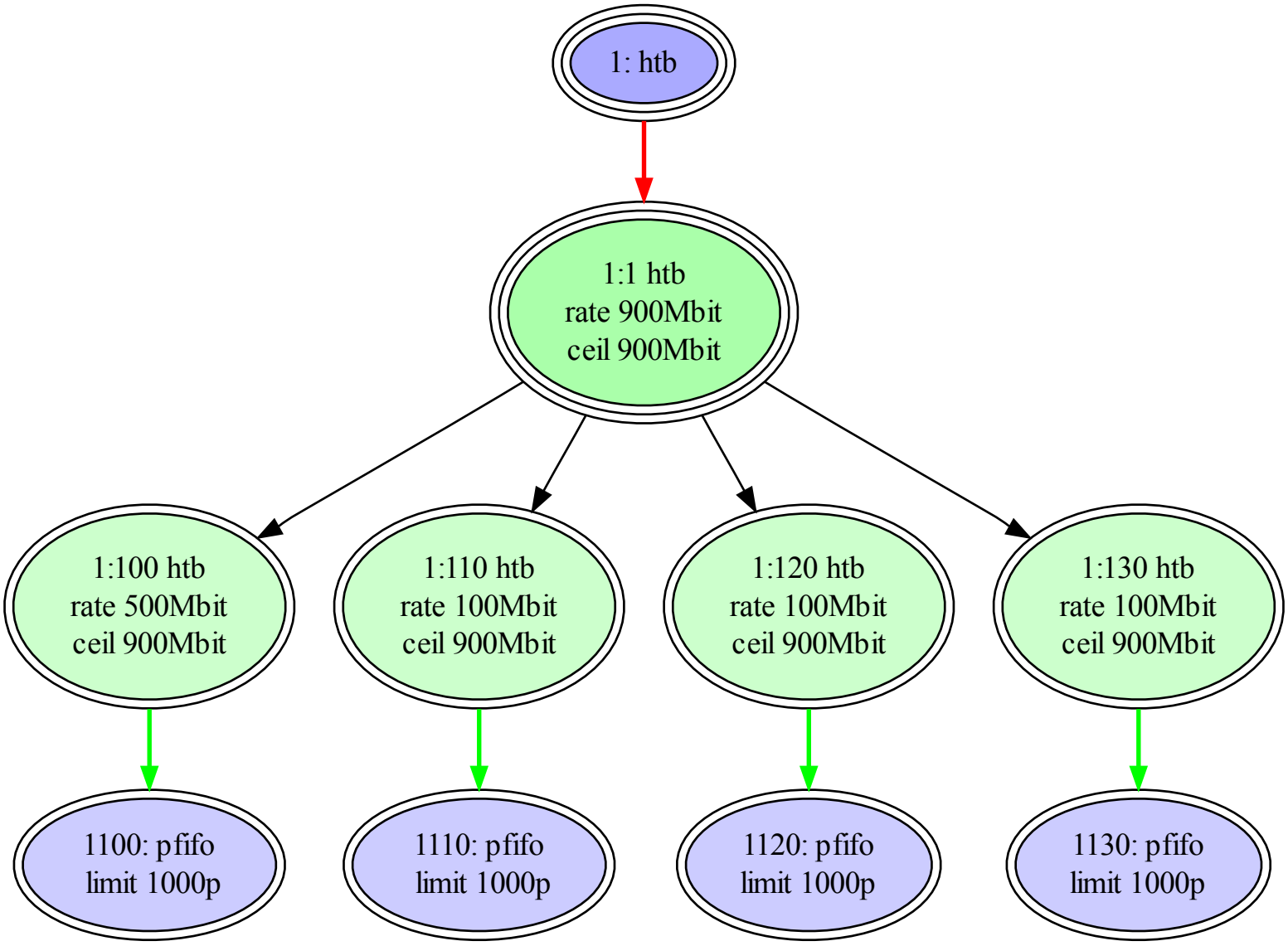
DomU Transmit

$$p \leq n$$

where: p: transmit packets enqueued in dom0 for vif$N.M$
n: netback ring-buffer slots (default $= 256$)

- Delaying packets in dom0 should be sufficient
- Dropping packets may actually be harmful
  - Holding onto packets actually slows down domU

DomU Transmit: Packet Scheduling Hierarchy



DomU Transmit: HTB Rules: Leaf Classes

Leaf Classes

- One per domain + default

```
tc class add dev peth0 parent 1:1 classid 1:100 htb \
        rate 500Mbit ceil 900Mbit
tc class add dev peth0 parent 1:1 classid 1:110 htb \
        rate 100Mbit ceil 900Mbit
tc class add dev peth0 parent 1:1 classid 1:120 htb \
        rate 100Mbit ceil 900Mbit
tc class add dev peth0 parent 1:1 classid 1:130 htb \
        rate 100Mbit ceil 900Mbit
```

DomU Transmit: Filter

Filter based on the fwmark set by `iptables`

- `handle` N is the fwmark match
- `flowid` X:Y is the queue to assign the packet to match

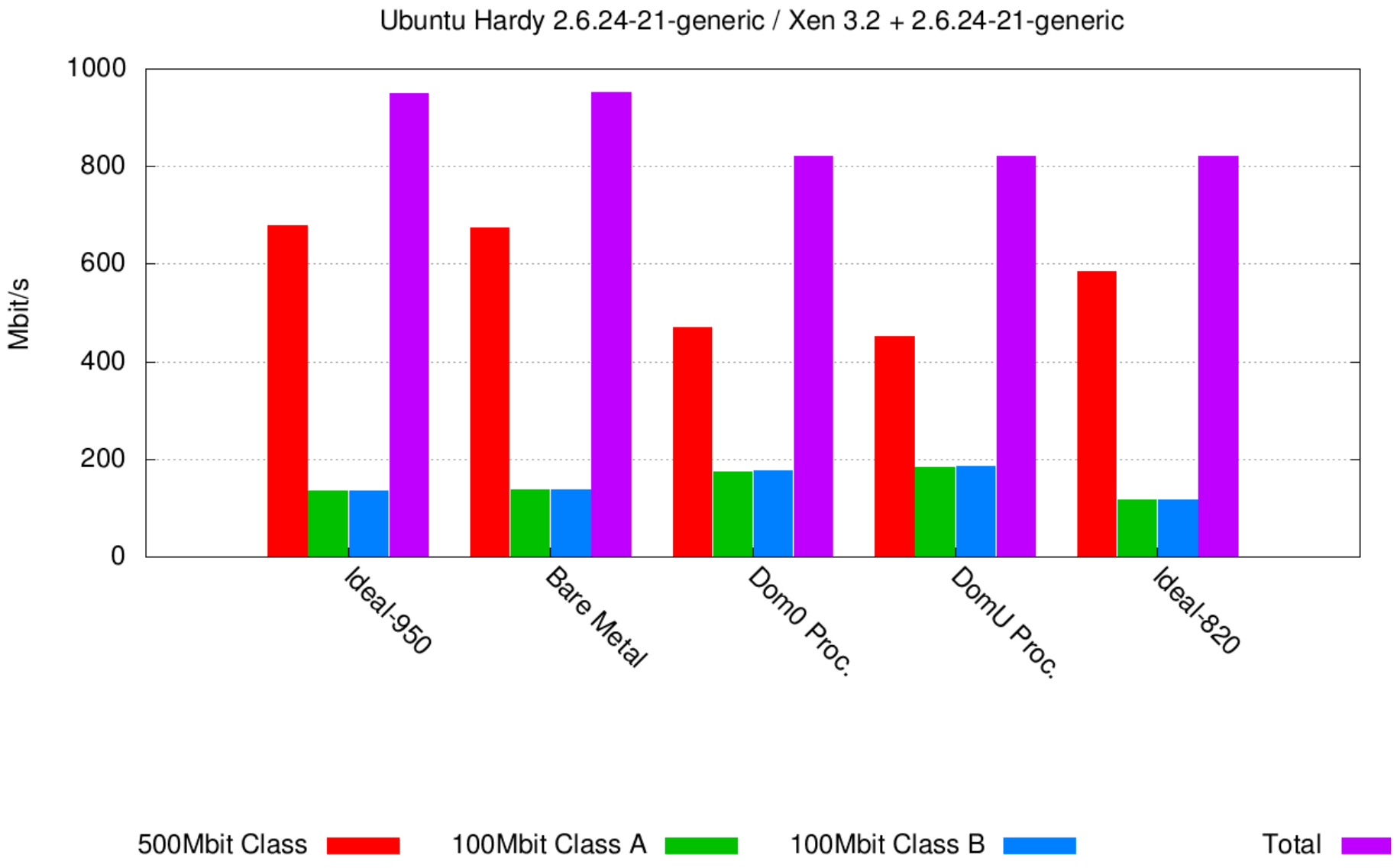
```
tc filter add dev peth0 protocol ip parent 1: \
        handle 100 fw flowid 1:100
tc filter add dev peth0 protocol ip parent 1: \
        handle 110 fw flowid 1:110
tc filter add dev peth0 protocol ip parent 1: \
        handle 120 fw flowid 1:120
```

Questions?

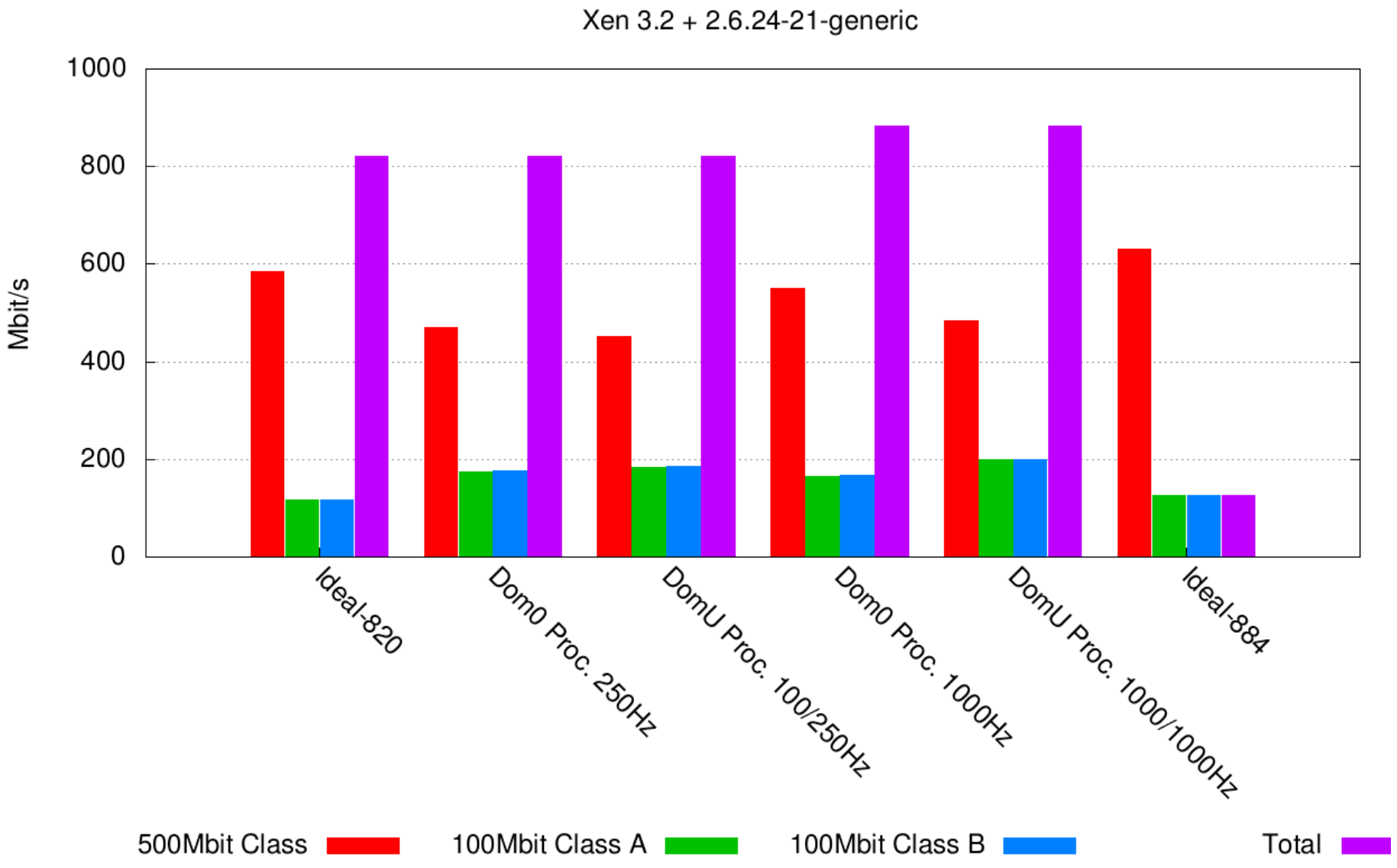
Part IV

Extra Material

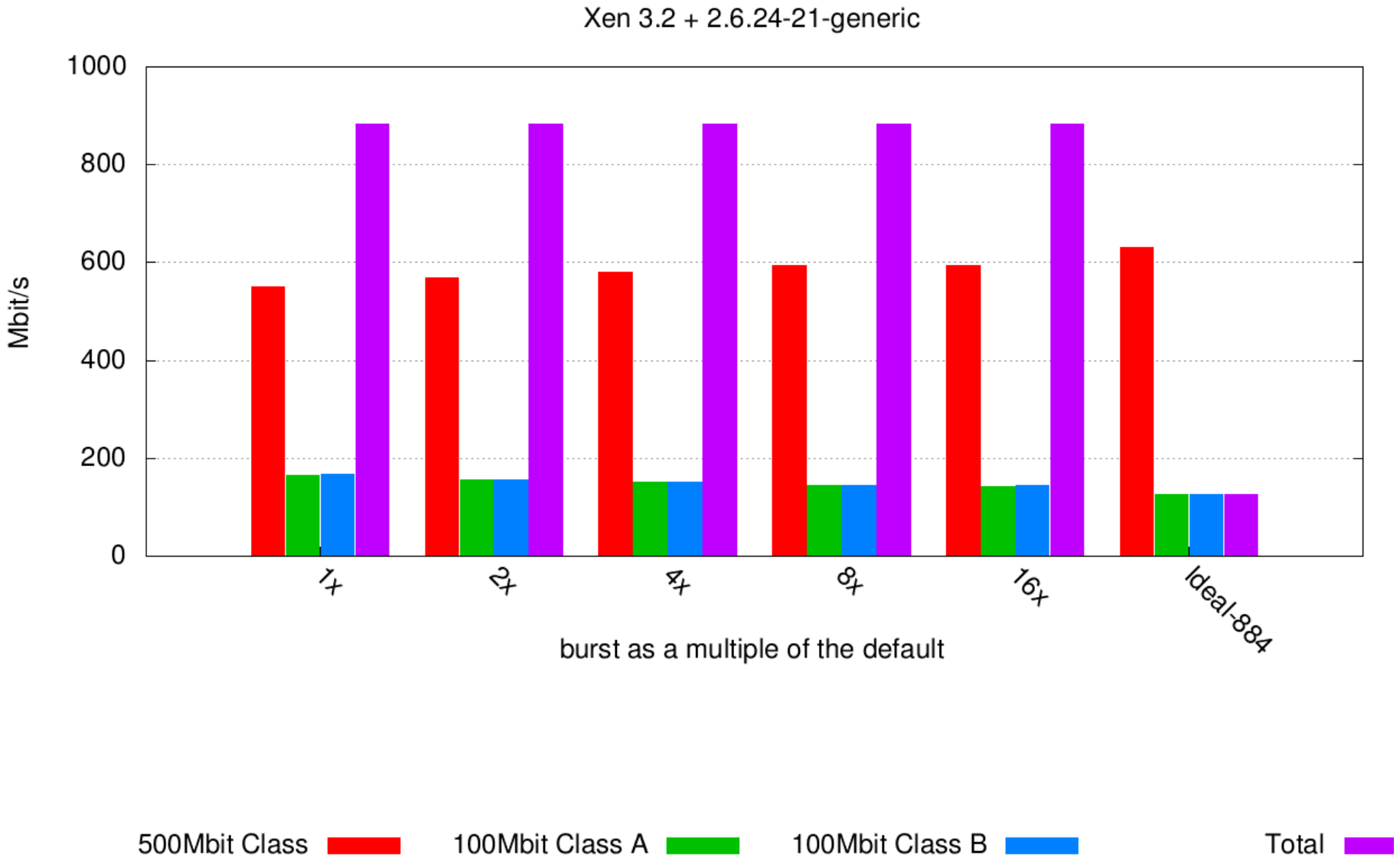
HTB Performance



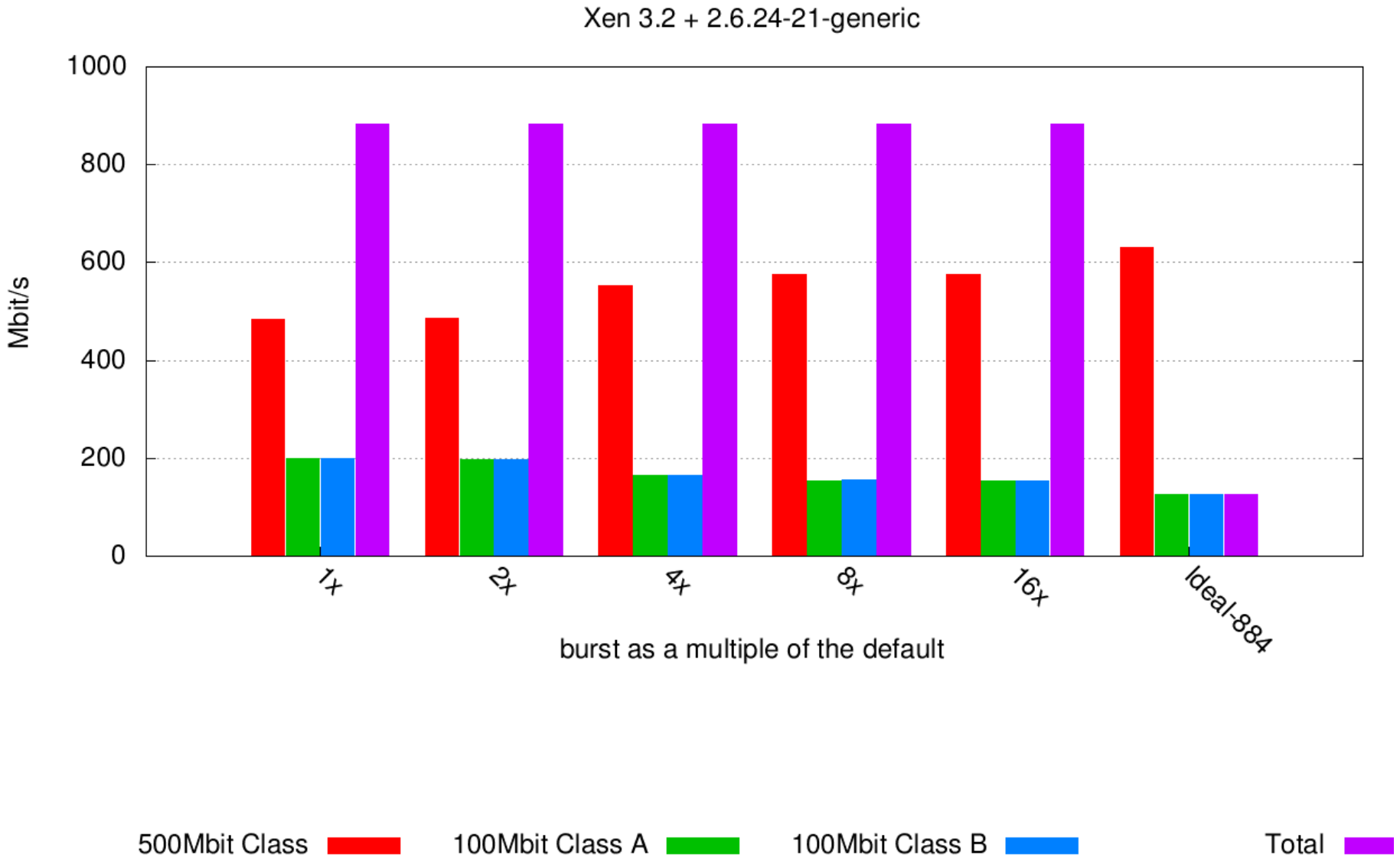
Tuning HZ


Xen 3.2 + 2.6.24-21-generic
Mbit/s
1000
800
600
400
200
0
Ideal-820
Dom0 Proc. 250Hz
DomU Proc. 100/250Hz
Dom0 Proc. 1000Hz
DomU Proc. 1000/1000Hz
Ideal-884
500Mbit Class
100Mbit Class A
100Mbit Class B
Total

Tuning Burst: Dom0 Processes



Tuning Burst: DomU Processes



End