b
UNIVERSITÄT
BERN

Seminar HS 2007

Learning module TCP congestion control

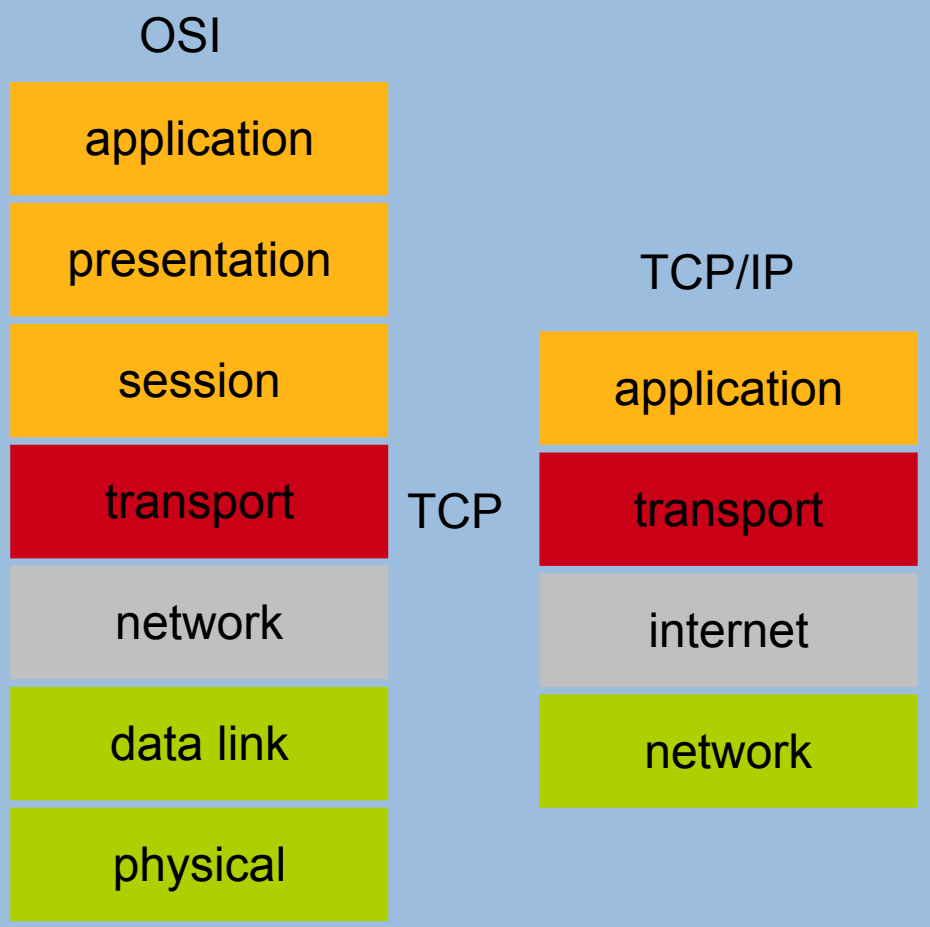
Markus Wulff
University of Bern

Outline

- Introduction to TCP
- TCP congestion control basics
- TCP configuration and monitoring with Linux
- Learning module architecture

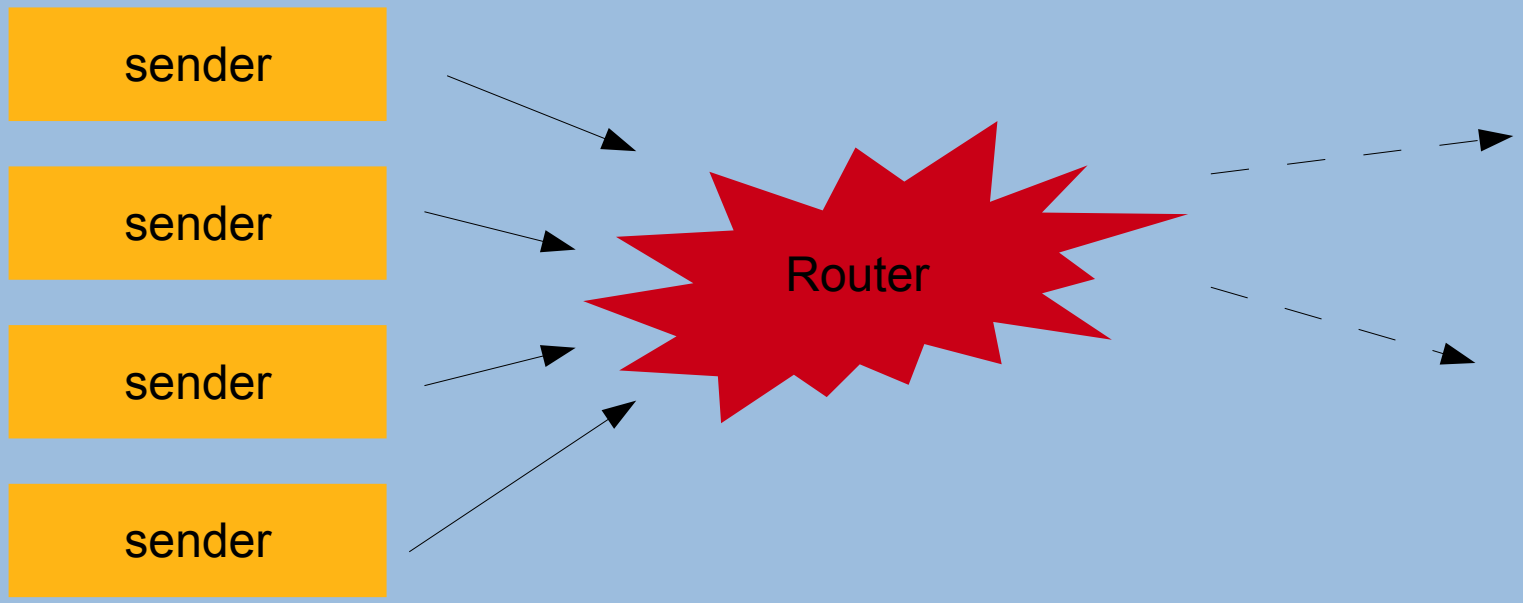
Introducing TCP

- TCP services
  - connection oriented
  - reliable data (e.g. error checking)
  - flow control
  - **congestion control**
  - **congestion avoidance**
  - stream orientation
  - Ports
- Connection
- parameters
  - Bandwidth
  - Round-Trip Time (RTT)
  - Packet loss



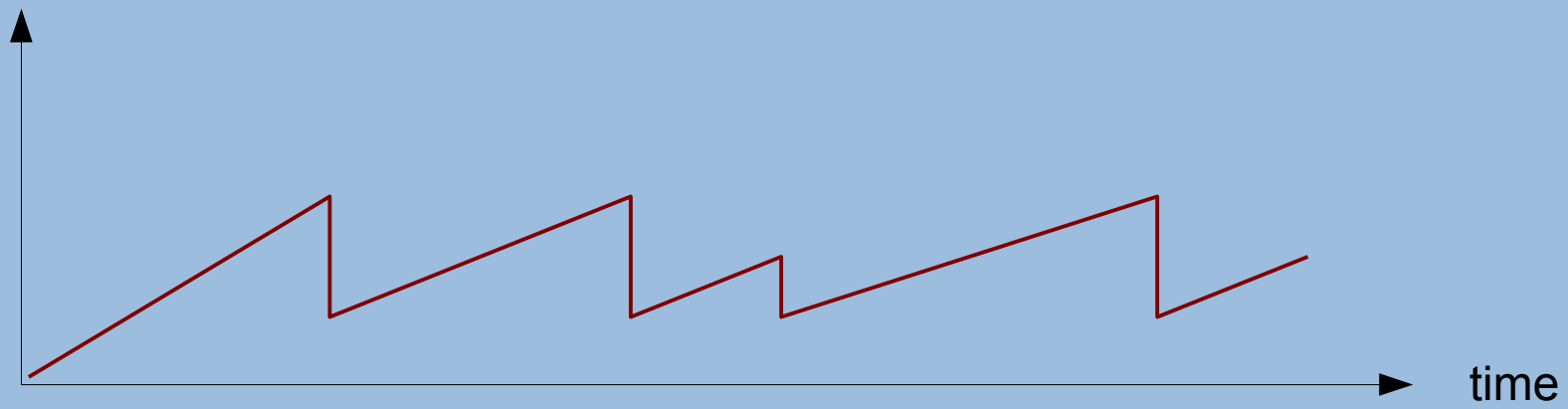
The congestion problem

- Hosts sending data to fast ⇨ router overloaded
- Router drops packets ⇨ host resends dropped packets
- Idea: congestion control (Van Jacobson)



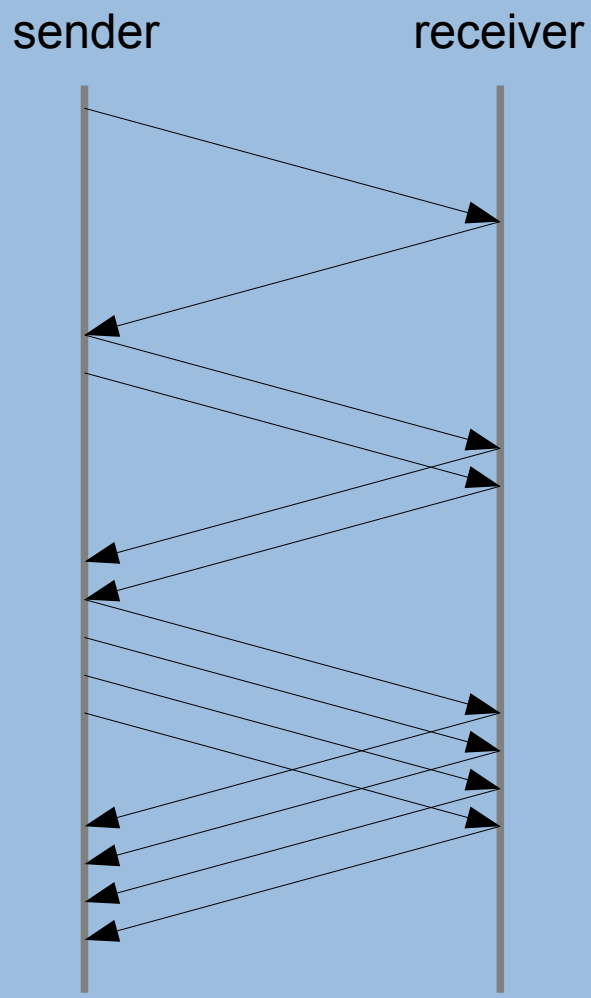
Additive increase/multiplicative decrease

- Self-clocking
  - MaxWindow = MIN(CongestionWindow, AdvertisedWindow)
  - EffectiveWindow = MaxWindow – (LastByteSent – LastByteAcked)
- Detect overload by timeouts (packet loss)
- If timeout occurs, reduce CongestionWindow (x0.5)
- If MaxWindow packets were successfully sent, increase CongestionWindow by 1
- Here: too small is better than too big



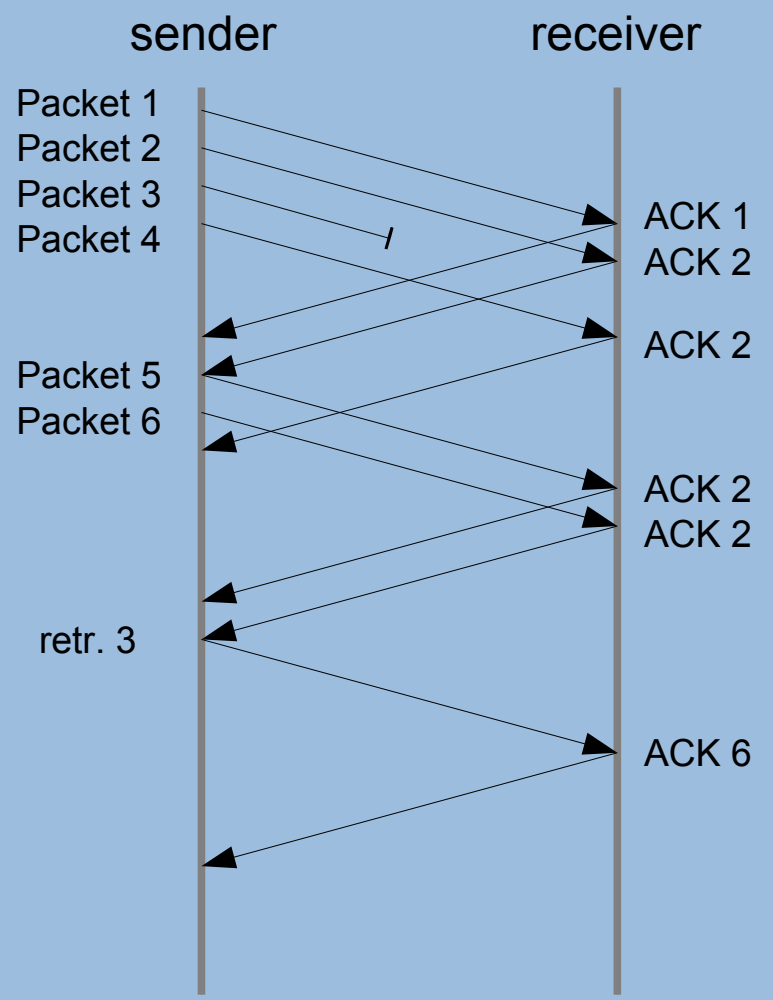
Slow start

- Earlier TCP implementations started sending at full rate
- Better: slowly approach maximum capacity of the connection
- After connection error start with last value divided by 2
- Parameter: SS-threshold



Fast retransmit

- If packet arrives out of order, receiver sends last ACK again
- Sender waits for several (usually 3) duplicate ACKs
- Sender retransmits missing packet
- Receiver sends ACK for last packet (in order) only



TCP timer management

- Retransmission timer
  - Too short: unnecessary retransmissions
  - Too long: long retransmission delay
  - Solution: determine RTT
- Constant adjustment

  RTT = a * RTT + (1-a) M

  M: when did the last ACK arrive
  a: smoothing factor (weight of old value)
- Retransmission timeout: b * RTT
  b = 2 is not the best idea

Congestion control

- Previous algorithms try to minimize the impact of congestions
  Now: try to prevent congestions
- Two approaches
  - Router based
  - Source based
- Problem: controlled setup is necessary to evaluate TCP congestion control algorithms

Congestion control example

- DECbit
  - Overload bit in packet header
  - Router sets the bit if average queue length >1
  - Sender knows how many packets caused the router to set the overload bit (overloadWindow)
    - If <50% then increase overloadWindow by 1
    - Otherwise decrease window size to 0.875 of last value

More TCP congestion control algorithms

- **High Speed TCP**
  The algorithm is described in RFC 3649. The main use is for connections with large bandwidth and large RTT (such as Gbit/s and 100 ms RTT).
- **TCP BIC**
  BIC is the abbreviation for Binary Increase Congestion control. BIC uses a unique window growth function. In case of packet loss, the window is reduced by a multiplicative factor. The window size just before and after the reduction is then used as parameters for a binary search for the new window size. BIC was used as standard algorithm in the Linux kernel.
- **TCP CUBIC**
  CUBIC is a less aggressive variant of BIC (meaning, it doesn't steal as much throughput from competing TCP flows as does BIC).
- **TCP Tahoe/Reno**
  These are the classical models used for congestion control. They exhibit the typical slow start of transmissions. The throughput increases gradually until it stays stable. It is decreased as soon as the transfer encounters congestion, then the rate rises again slowly. The window is increased by adding fixed values. TCP Reno uses a multiplicative decrease algorithm for the reduction of window size. TCP Reno is the most widely deployed algorithm.
- **TCP Vegas**
  TCP Vegas introduces the measurement of RTT for evaluating the link quality. It uses additive increases and additive decreases for the congestion window

http://linuxgazette.net/135/pfeiffer.html

UNIVERSITÄT BERN

Learning module architecture

- Linux platform, because
  - supports TCP parameter modification and monitoring
  - supports pluggable congestion control algorithms
    e.g.
    `echo "westwood"> /proc/sys/net/ipv4/tcp_congestion_control`
  - many tools for network measurement and monitoring are available
- Xen virtualization
  - machines completely autonomous
  - good performance
  - well supported
- User interface
  - Flash, Java, HTML ...?

Measuring TCP

- Several tools are available: netcat, wget, etc.
- Record network flow with wireshark

- Linux kernel module tcpprobe:

```
# modprobe tcpprobe port=5001
# cat /proc/net/tcpprobe >/tmp/data.out &
# pid=$!
# iperf -c otherhost
# kill $pid
```

  - Gives access to the congestion window and other parameters
  - Produces one line of date in text format for every packet seen
  - Using port 0 instead of 5001 allows measuring the window of all TCP connections

UNIVERSITÄT BERN

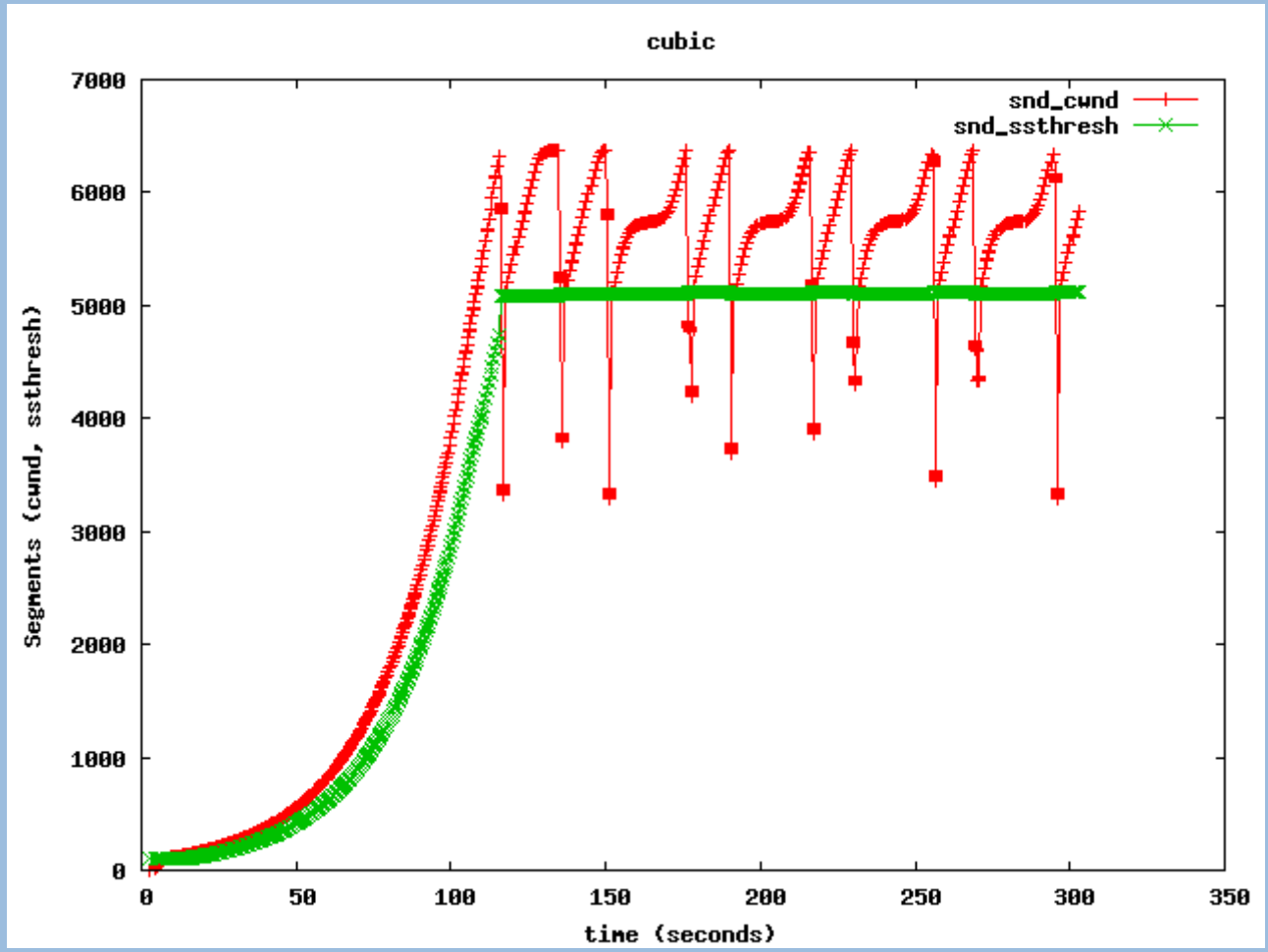
tcpprobe example I

```
0.073678 10.8.0.54:38644 192.168.1.42:5001 24 0xb6b19bb 0xb6b19bb 2 2147483647 5792
 ^        ^               ^                 ^  ^        ^        ^ ^          ^
 |        |               |                 |  |        |        | |          +- [9]
 |        |               |                 |  |        |        | +------------ [8]
 |        |               |                 |  |        |        +-------------- [7]
 |        |               |                 |  |        +----------------------- [6]
 |        |               |                 |  +-------------------------------- [5]
 |        |               |                 +----------------------------------- [4]
 |        |               +--------------------------------------------------- [3]
 |        +------------------------------------------------------------------- [2]
 +---------------------------------------------------------------------------- [1]
```

[9] Send window
[8] Slow start threshold
[7] Congestion window
[6] Unacknowledged sequence #
[5] Next send sequence #
[4] Bytes in packet
[3] Receiver address:port
[2] Sender address:port
[1] Time seconds

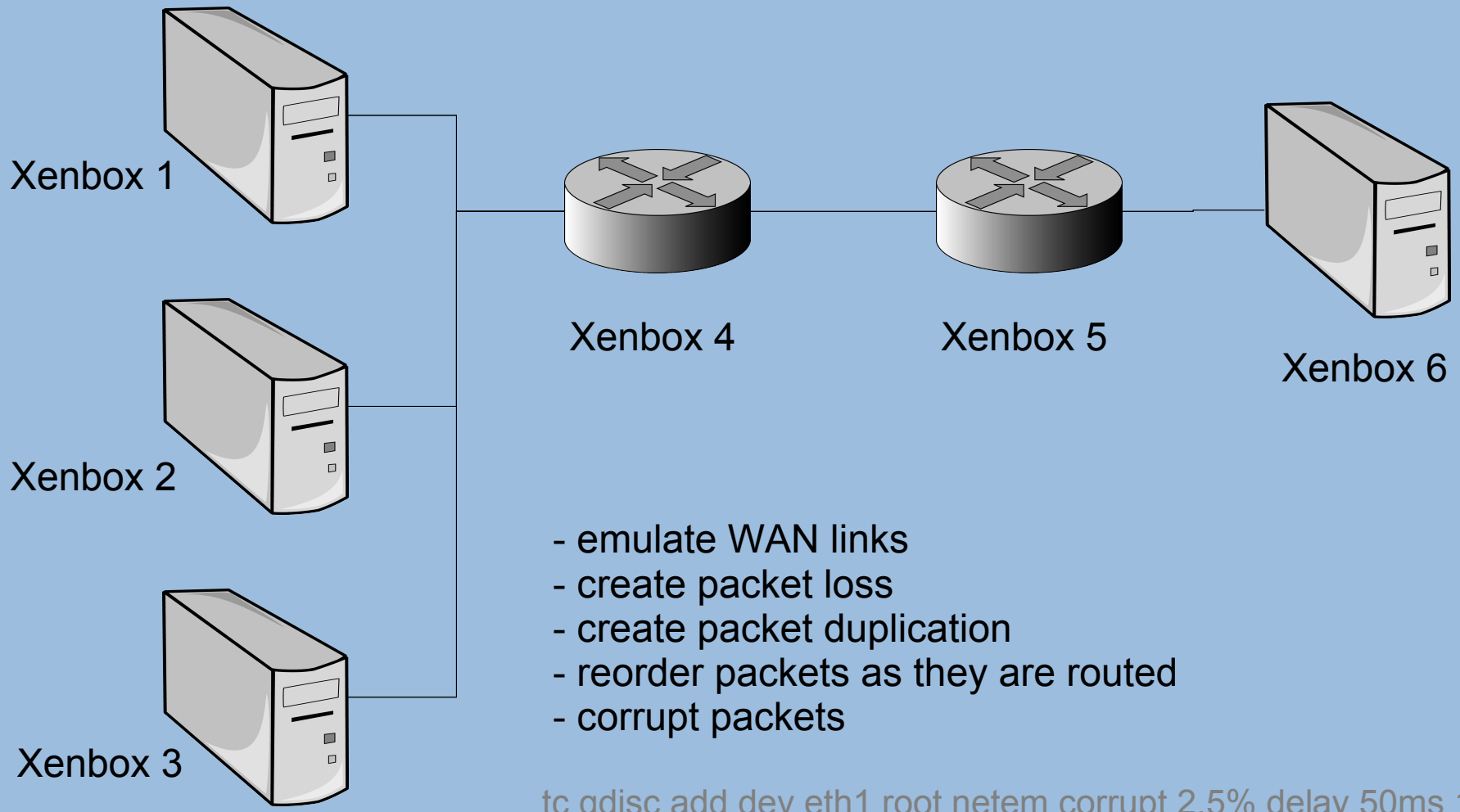
http://www.linux-foundation.org/en/Net:TCP_testing

tcpprobe example II



http://www.linux-foundation.org/en/Net:TCP_testing

Learning module setup



- emulate WAN links
- create packet loss
- create packet duplication
- reorder packets as they are routed
- corrupt packets

```
tc qdisc add dev eth1 root netem corrupt 2.5% delay 50ms 10ms
tc qdisc add dev eth0 root netem delay 100ms reorder 25% 50%
```

Conclusion

- TCP congestion control still under development.
- Congestion control algorithms are complex, many parameters.
- A interactive learning module helps to understand the basic algorithms.

Thank you!