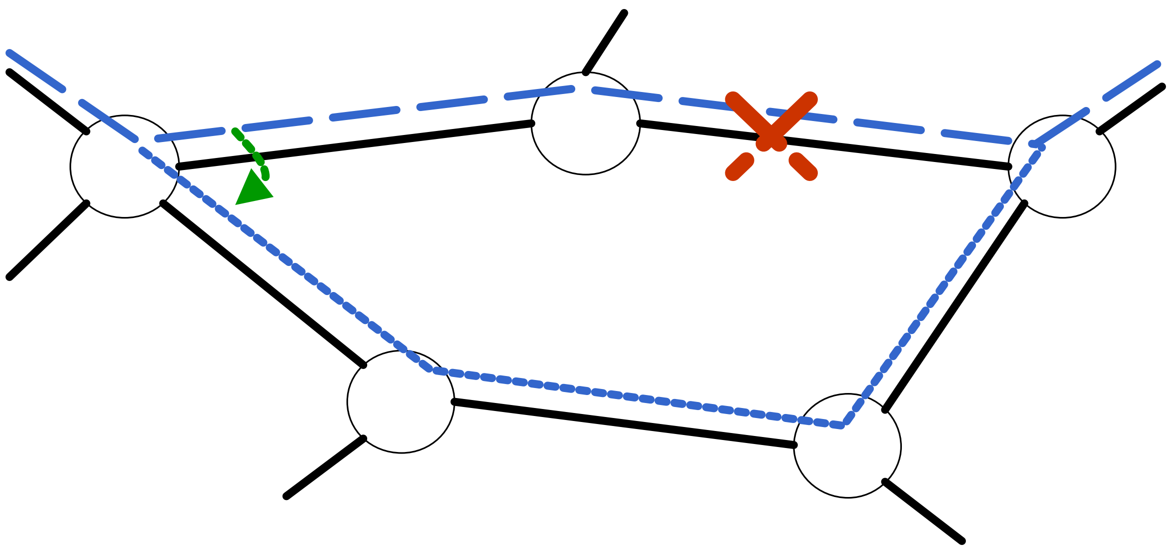
Influence of Recovery Time on TCP Behaviour

Chris Develder
Didier Colle
Pim Van Heuven
Steven Van den Berghe
Mario Pickavet
Piet Demeester

Introduction

- Network recovery: backup paths to recover traffic lost due to network failures

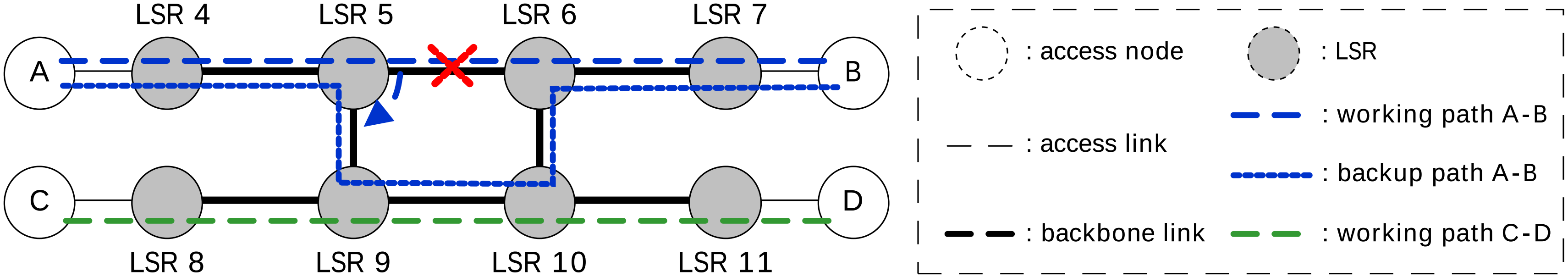


- Many questions remain to be answered:
  - How fast should this happen? Is fast protection better, or isn't it desirable? How does e.g. TCP react to protection switches?

Outline

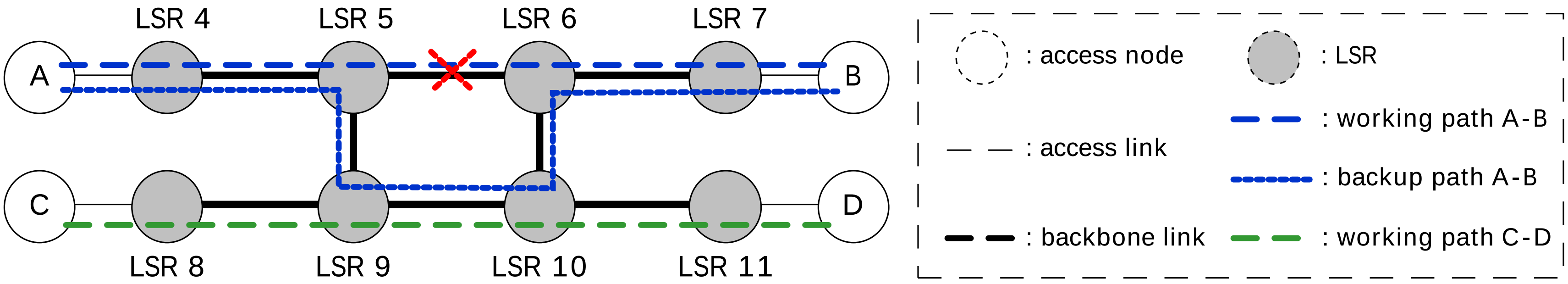
- Experiment set-up
- Qualitative discussion
- TCP goodput
- More detailed analysis
- Finding the "best" delay
- Conclusion

Experiment set-up



- Two sets of TCP flows:
  - A→B: the "(protection) switched flows"
  - C→D: the "fixed flows"
- MPLS paths and <u>pre-established</u> backup paths
  - to be able to influence exact timing
  - protection switch: "manually"

Experiment set-up



- Simulation scenario:
  - start of TCP sources: random
  - [0-10s[: link up
  - [10-20s[: link down; protection switch after delay 0/50/1000 ms
  - [20-30s[: link up again

Experiment set-up

- FYI: TCP NewReno mechanisms (RFC 2582)
  - slow start: (cwnd ≤ sstresh)
    - increase cwnd: +1 per ACK
    - set sstresh=cwnd/2; cwnd=1 after timeout
  - congestion avoidance: (cwnd > sstresh)
    - if cwnd reaches sstresh
    - linear increase of cwnd
  - fast recovery, fast retransmit:
    - if packet loss: retransmit; sstresh=cwnd/2; cwnd=sstresh
    - three duplicate ACKs: sstresh*=1/2; cwnd=sstresh+3
  - newreno: extend fast recovery and fast retr.
    - for each extra duplicate ACK: cwnd++; stay in fast recovery

Outline

- Experiment set-up
- Qualitative discussion
- TCP goodput
- More detailed analysis
- Finding the "best" delay
- Conclusion

Qualitative discussion — what will happen?

- When a failure occurs:
  - switched flows join fixed ones
  - backbone link will become bottleneck
  - due to overload, packet losses will occur
  - TCP will react by backing off

Qualitative discussion — what will happen?

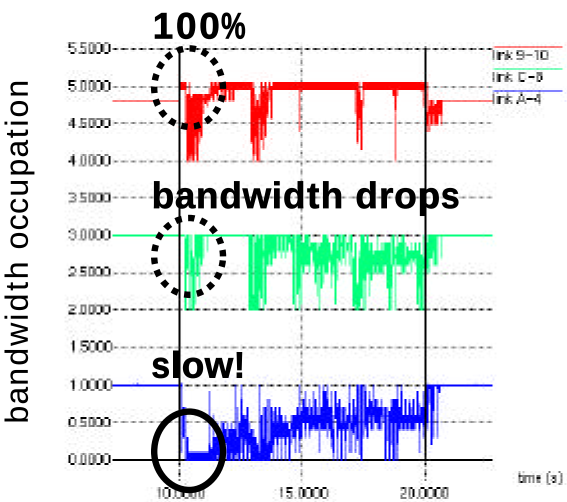
- Influence of protection switch delay:
  - no delay:
    - immediate buffer overflow on bottleneck backbone link
    - both fixed and switched flows are heavily affected
  - small delay:
    - switched flows have backed off somewhat when joining the fixed ones
    - fixed flows are less affected
  - large delay:
    - switched flows fall back to zero
    - rather smooth transition of bottleneck from access to backbone

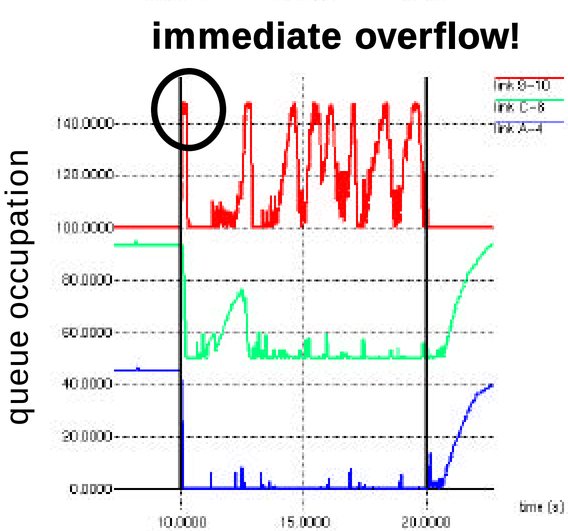
Qualitative discussion — simulation parameters

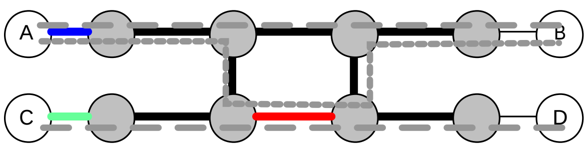
- Simulation parameters:
  - number of TCP NewReno sources:
    - 5 fixed,
    - 5 switched
  - access bandwidth: 8 Mbit/s
  - backbone bandwidth: 10 Mbit/s
  - propagation delay: 10ms/link
    - this results in a RTT of 100-150ms (+20ms in case of protection switch)
  - queue size: 50 packets
  - max. TCP window size set at 30

Qualitative discussion — bandwidth and queues

- No protection switching delay (0ms)



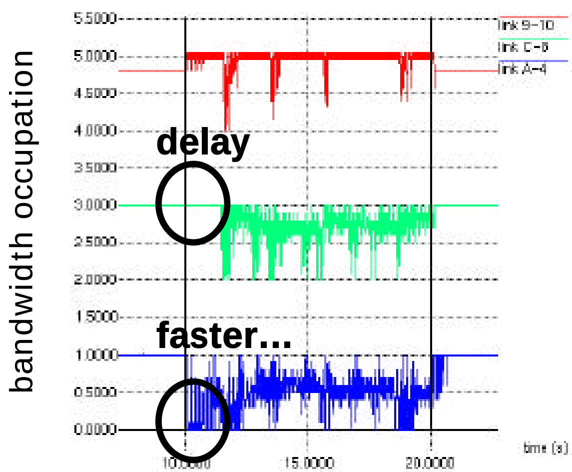




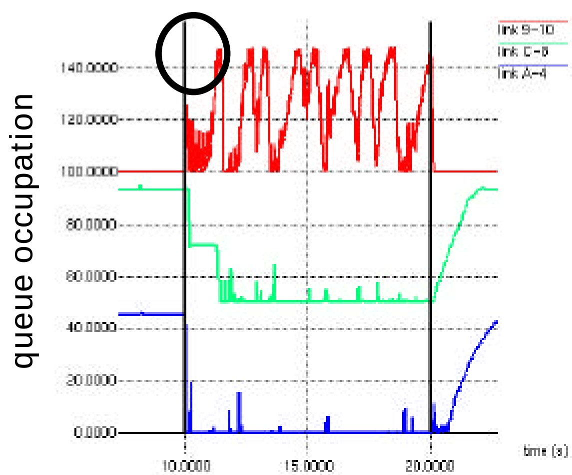
- before failure: access links are bottleneck
  - ■ link is filled for 80%; queue empty
  - ■ ■ link is filled for 100%; queue filled
- during failure: bottleneck shifts to backbone
  - ■ link gets filled for 100%; immediate queue overflow; oscillations due to TCP behaviour
  - ■ bandwidth drops: fixed flows are affected due to losses in backbone
  - ■ bandwidth seriously drops; recovery is rather slow!
- after failure: access links are bottleneck (queues in access are being filled again)

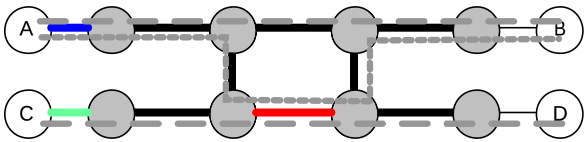
Qualitative discussion — bandwidth and queues

· Small protection switching delay (50ms)



NO immediate overflow!

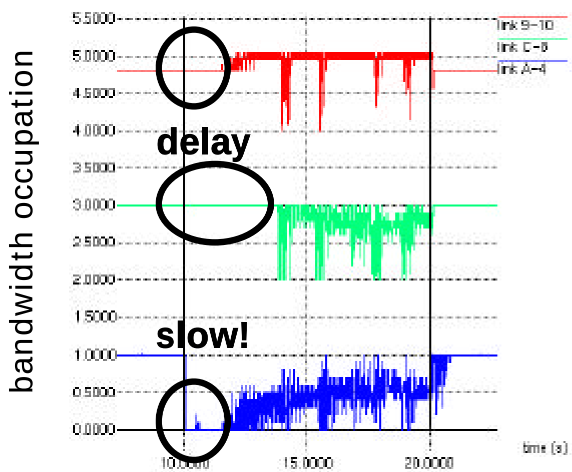


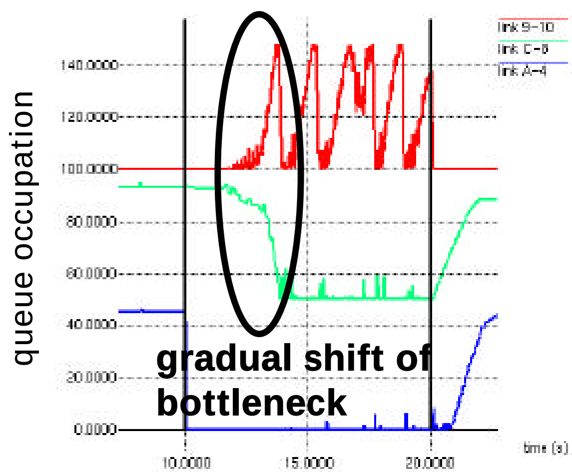


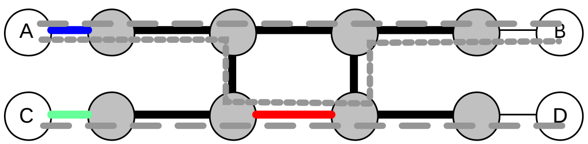
- before failure: access links are bottleneck
  - link is filled for 80%; queue empty
  - link is filled for 100%; queue filled
- during failure: bottleneck shifts to backbone
  - link gets filled for 100%; NO immediate queue overflow; oscillations due to TCP behaviour
  - bandwidth drops: fixed flows are affected AFTER CERTAIN DELAY
  - bandwidth drops less; recovery apparently is faster
- after failure: access links are bottleneck (queues in access are being filled again)

Qualitative discussion — bandwidth and queues

- Large protection switching delay (1000ms)







- before failure: access links are bottleneck
  - link is filled for 80%; queue empty
  - link is filled for 100%; queue filled
- during failure: bottleneck shifts to backbone
  - link gets filled for 100% after delay; NO immediate queue overflow: very gradual shift of bottleneck
  - bandwidth drops: fixed flows are affected only after rather long delay
  - bandwidth drops to zero; very gradual recovery
- after failure: access links are bottleneck (queues in access are being filled again)

Outline

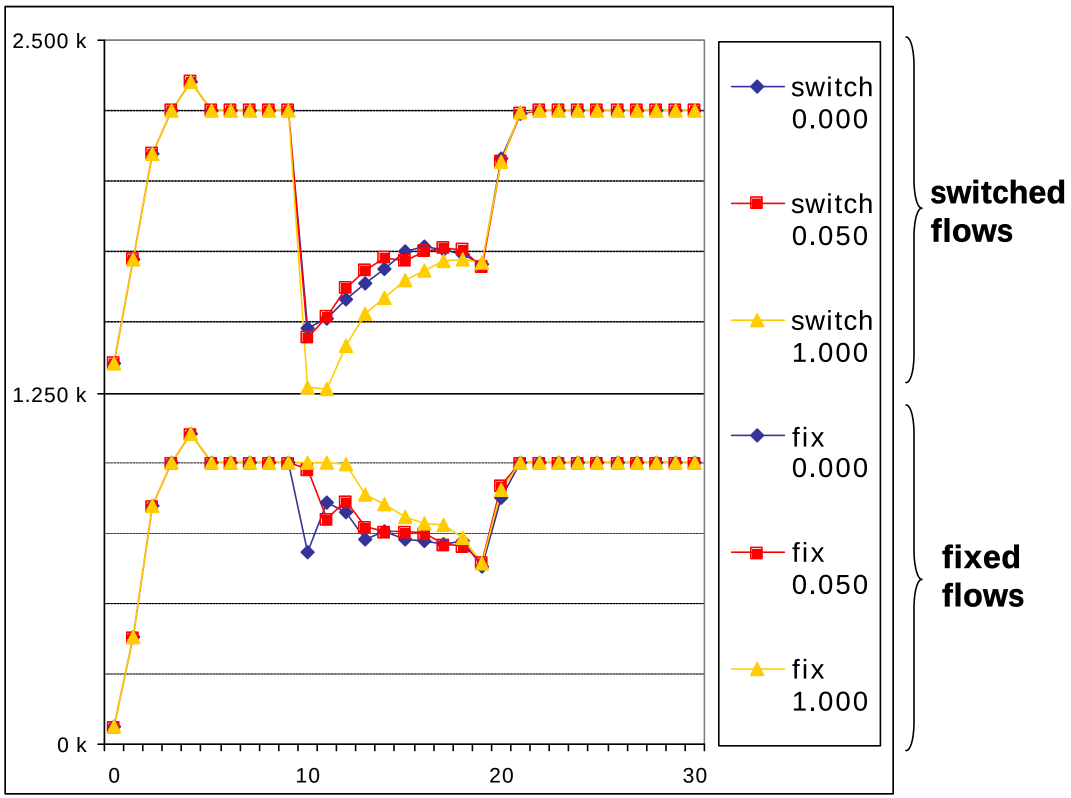
- Experiment set-up
- Qualitative discussion
- TCP goodput
- More detailed analysis
- Finding the "best" delay
- Conclusion

TCP goodput

- Previous slides showed througput, window size evolution and queue occupation:
  - this learnt something about what happens,
  - but it isn't obvious to decide what is best from these graphs
- So: what matters to end user?
  - end user of TCP only cares about how long it takes to transfer file, access webpage, etc.
  - what matters is <u>GOODPUT</u>: number of bytes successfully transported end-to-end per second

TCP goodput

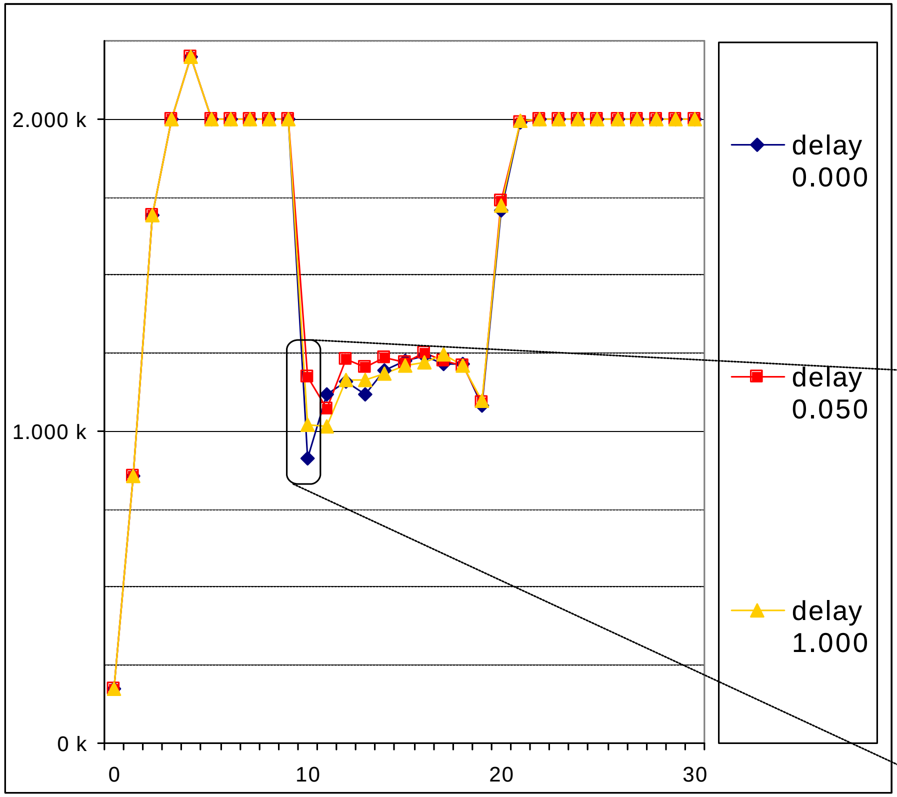
- Goodput evolution for different delays per flow category:



- no delay:
  - switched lose significantly
  - fixed show drop too
- 50 ms delay:
  - switched lose as much as for delay 0, but
  - drop in goodput for fixed is smaller
- 1000 ms delay:
  - switched lose a lot more and recover more slowly
  - drop in goodput for fixed is less (of course)

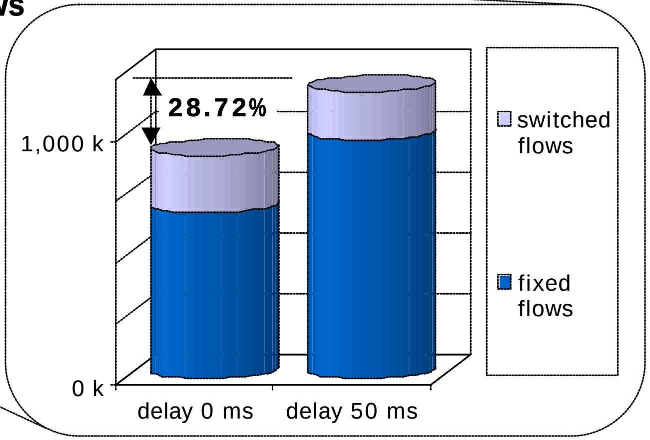
TCP goodput

- Goodput evolution for different delays over aggregate of all flows:



- The difference between the three cases is limited to the first seconds after the failure
- For the first second, the 50 ms case has 28.72% better total goodput than the 0 ms case

all flows



TCP goodput

- Preliminary conclusion:
  - extremely fast protection switching is not a must
  - it is better to have a certain delay than none at all,
  - but finding the optimal value doesn't appear to be simple
    (dependent on round trip time for TCP flows, and also on traffic load)

Outline

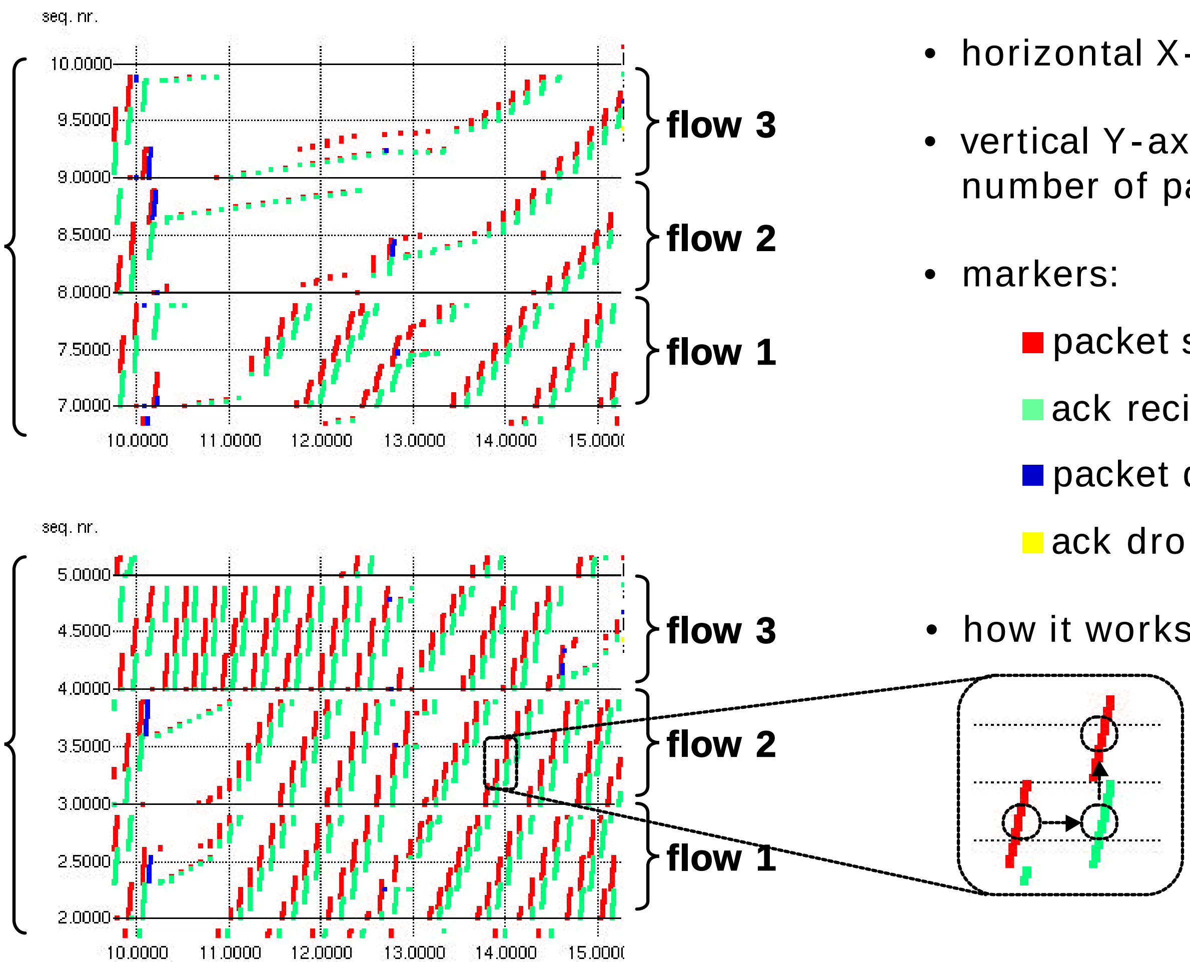
- Experiment set-up
- Qualitative discussion
- TCP goodput
- More detailed analysis
- Finding the "best" delay
- Conclusion

More detailed analysis

- Main cause for better goodput with delay 50 ms:
  - delay 0 ms: TCP sources suffering multiple packet losses recover slowly if they stay in fast retransmit & recovery phase
    $\Rightarrow$ only one packet per round trip time (RTT) is transmitted
  - delay 50 ms: some TCP flows fall back to slow start (due to timeout)
    $\Rightarrow$ this gives better goodput! (more than one packet/RTT)

More detailed analysis

- Illustration by packet traces

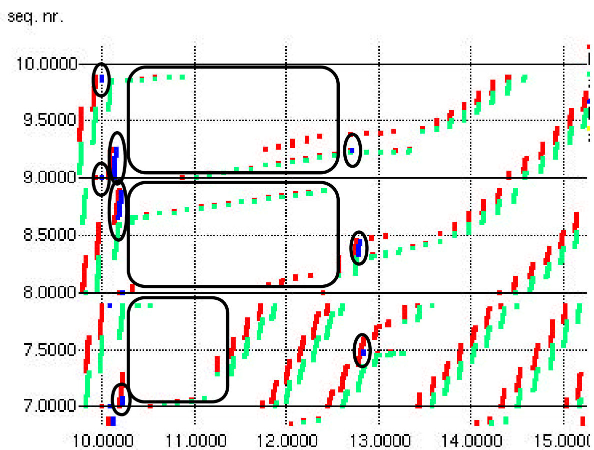


- horizontal X-axis: time (s)
- vertical Y-axis: sequence number of packet or ACK
- markers:
  - packet sent
  - ack recieved
  - packet dropped
  - ack dropped
- how it works:
  - packet is sent
  - ACK is received
  - new packet is sent

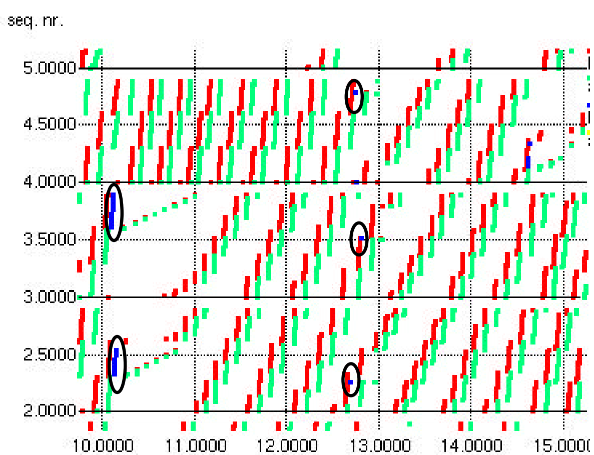
More detailed analysis

- Illustration by packet traces

switched flows



fixed flows



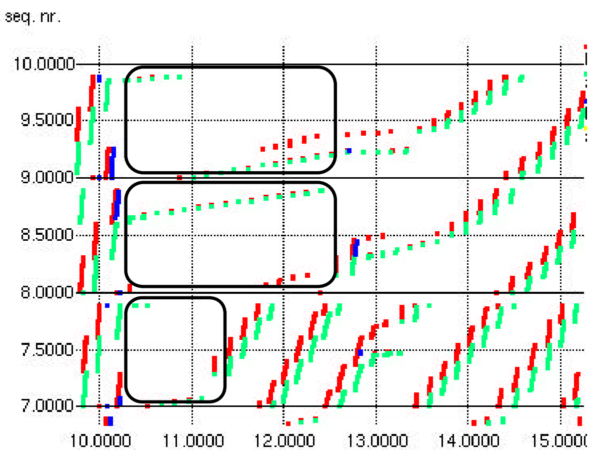
<u>Delay 0 ms</u>:

- at time of link failure: losses of packets that are being transported (switched flows only)
- almost immediately after failure: buffer overflow on bottleneck link (affects ALL flows)
- TCP algortithm: duplicate ACKs cause source to go into fast retransmit & fast recovery; only 1 packet is retransmitted per RTT
- next buffer overflows: same applies, but less packets per source are lost

More detailed analysis

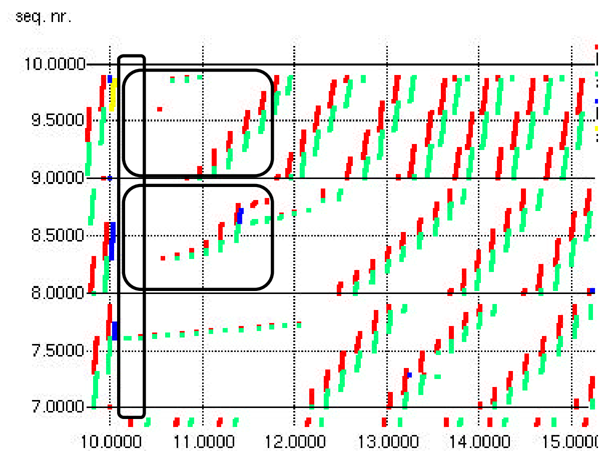
- Illustration by packet traces

switched flows

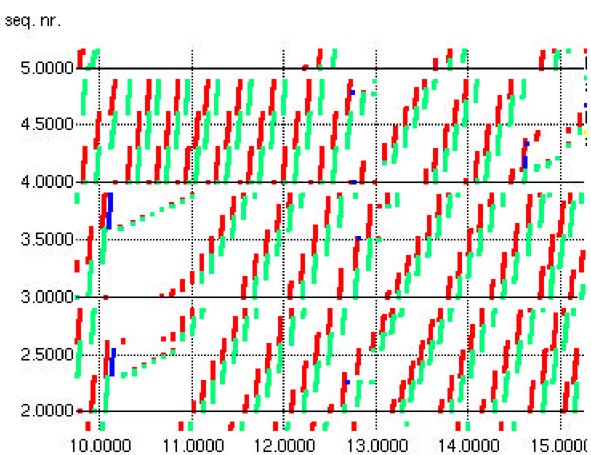


<u>Delay 50 ms</u>:

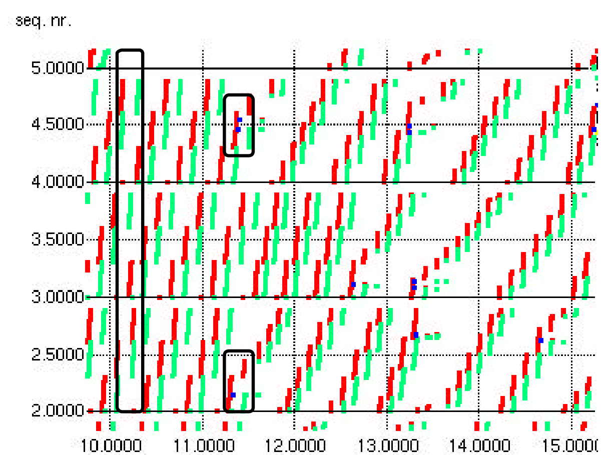
- no immediate buffer overflow
- some sources timeout and fall back to slow start ⇒ faster recovery!



fixed flows



- fixed are not affected until first buffer overflow
- overall faster recovery



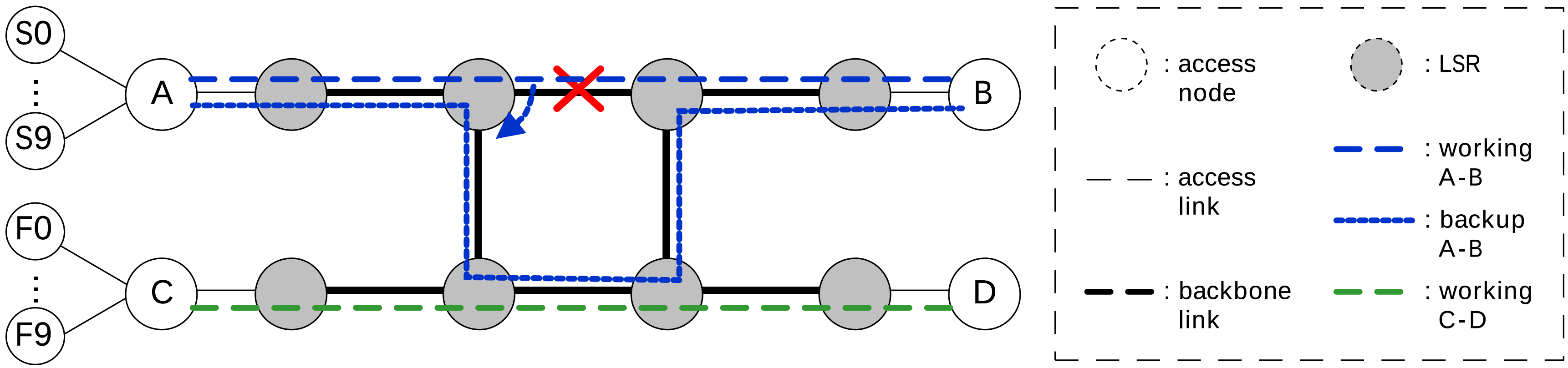
Outline

- Experiment set-up
- Qualitative discussion
- TCP goodput
- More detailed analysis
- Finding the "best" delay
- Conclusion

Finding the best delay

- Previous slides:
  - indication of importance of delay for goodput
  - "special" circumstances: same RTT for all TCP flows, all TCP sources originated at same node
- Therefore:
  - mixture of different RTTs
  - different source nodes for different flows

Finding the best delay



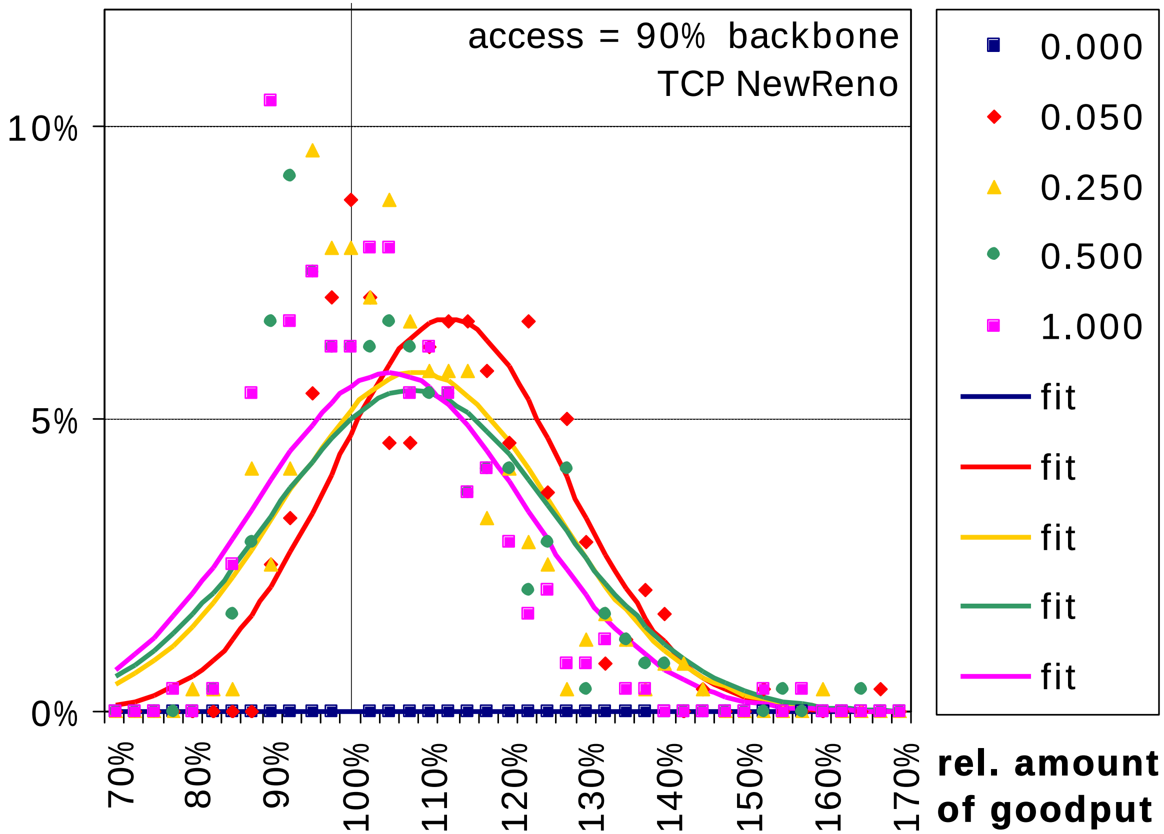
- Experiment set-up:
  - propagation delay:
    - first access link: random in [1ms,100ms[
    - all other links: 1ms
  - number of sources: 10 fixed, 10 switched
- Scenario (times in s):
  - TCP sources randomly start in [0.1,2.1]
  - [0,5[link up; [5,10[link down; [10,15[link up

Finding the best delay

- Analysis:
  - 240 different runs (other random seeds)
  - distrubution of f(x)=Good(x)/Good(0),
    - Good(x)=total goodput over all flows during first 1.5 seconds after link failure for a protection switch delay of x milliseconds
  - interpretation of f(x):
    - if f(x)>100% then delay of x results in better goodput than no delay at all
    - if f(x)<100% then delay of x results in worse goodput than no delay at all
    - e.g. f(x)=110% means delay of x gives 10% more goodput than no delay at all

Finding the best delay

- Analysis: distrubution of f(x)=Good(x)/Good(0)



- X-axis: f(x): goodput compared to goodput for delay 0 ms (same random seed)
- Y-axis: P[f(x)]: probability of finding f(x) (histogram)
- all delays result in better goodput than no delay at all:
  - delay 50ms: 11.89%
  - delay 250ms: 7.55%
  - delay 500ms: 6.91%
  - delay 1000ms: 3.98%

Outline

- Experiment set-up
- Qualitative discussion
- TCP goodput
- More detailed analysis
- Finding the "best" delay
- Conclusion

Conclusion

- Conclusions:
  - We have studied the effect of recovery on TCP flows
  - From simulation results, we have inferred that recovery time doesn't necessarily need to be as small as possible
  - For TCP traffic, introducing a protection switch delay may be useful
- Future work:
  - Pursue detailed analysis of simulation results; e.g. look at what happens after link recovery
  - Extend investigation to other (larger, more complex) topologies.

INTEC

the END

Thanks for your attention...
Please feel free to ask any questions you might have!