Issues related to TCP performance in heterogeneous networks

Chadi BARAKAT

INRIA Sophia Antipolis, France
PLANETE group

EPFL Summer Research Institute
July 16, 2002

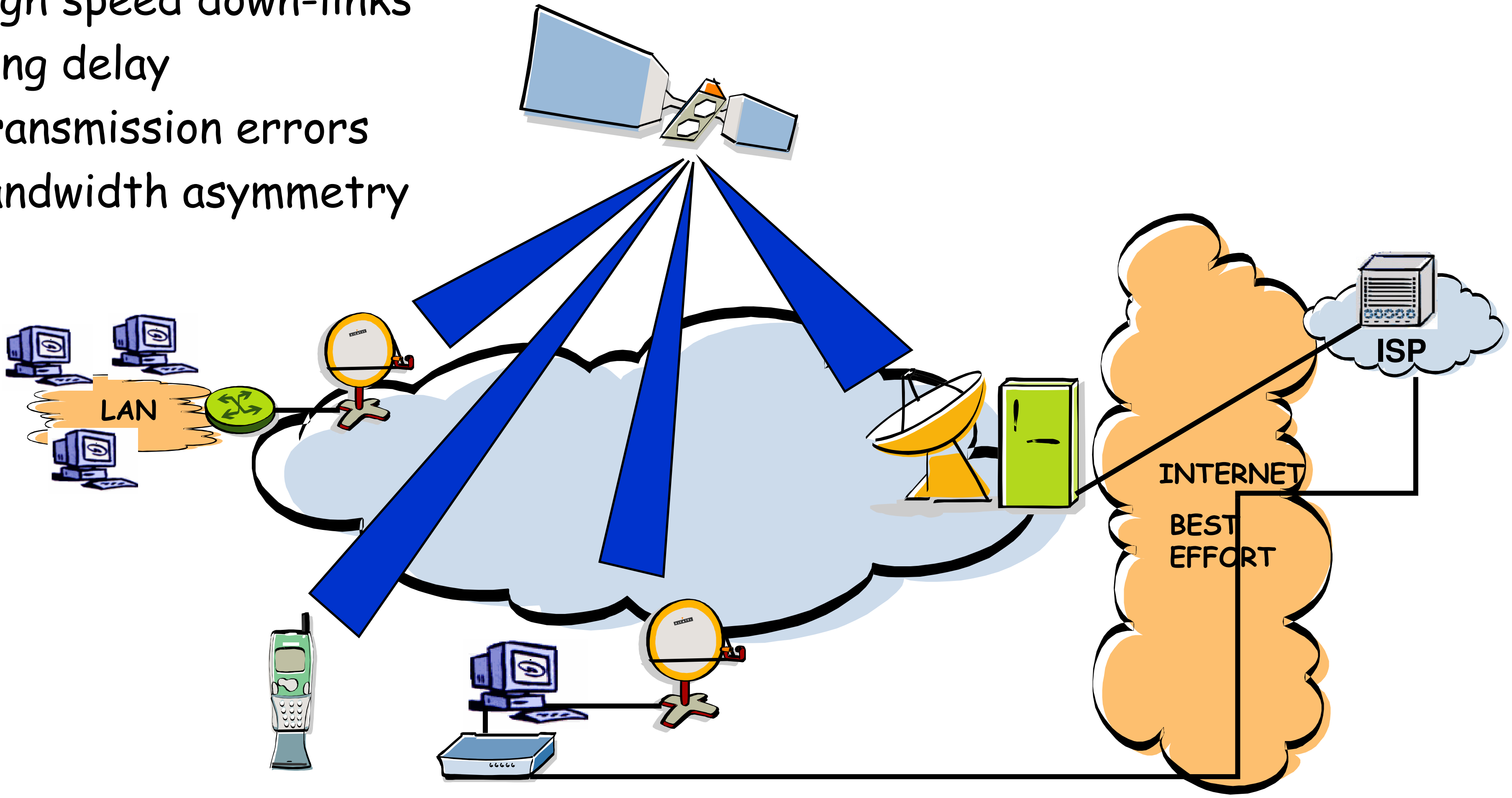
Heterogeneity of networks

- The Internet is including more and more transmission media with features challenging to TCP:

  - High speed links, e.g., optic fibers at hundreds of Gbps, high speed satellite down-links.
  - Low speed links, e.g., dial-up modem lines, small antenna satellite up-links.
  - Long delay paths, mainly due to satellite links.
  - Variable delay paths, e.g., satellite constellations, mobile networks.
  - Lossy links, e.g., noisy wireless and satellite links.
  - Asymmetric bandwidth paths, e.g., direct broadcast satellite networks, cable networks.

Typical example of a heterogeneous network

- ❑ High speed down-links
- ❑ Long delay
- ❑ Transmission errors
- ❑ Bandwidth asymmetry



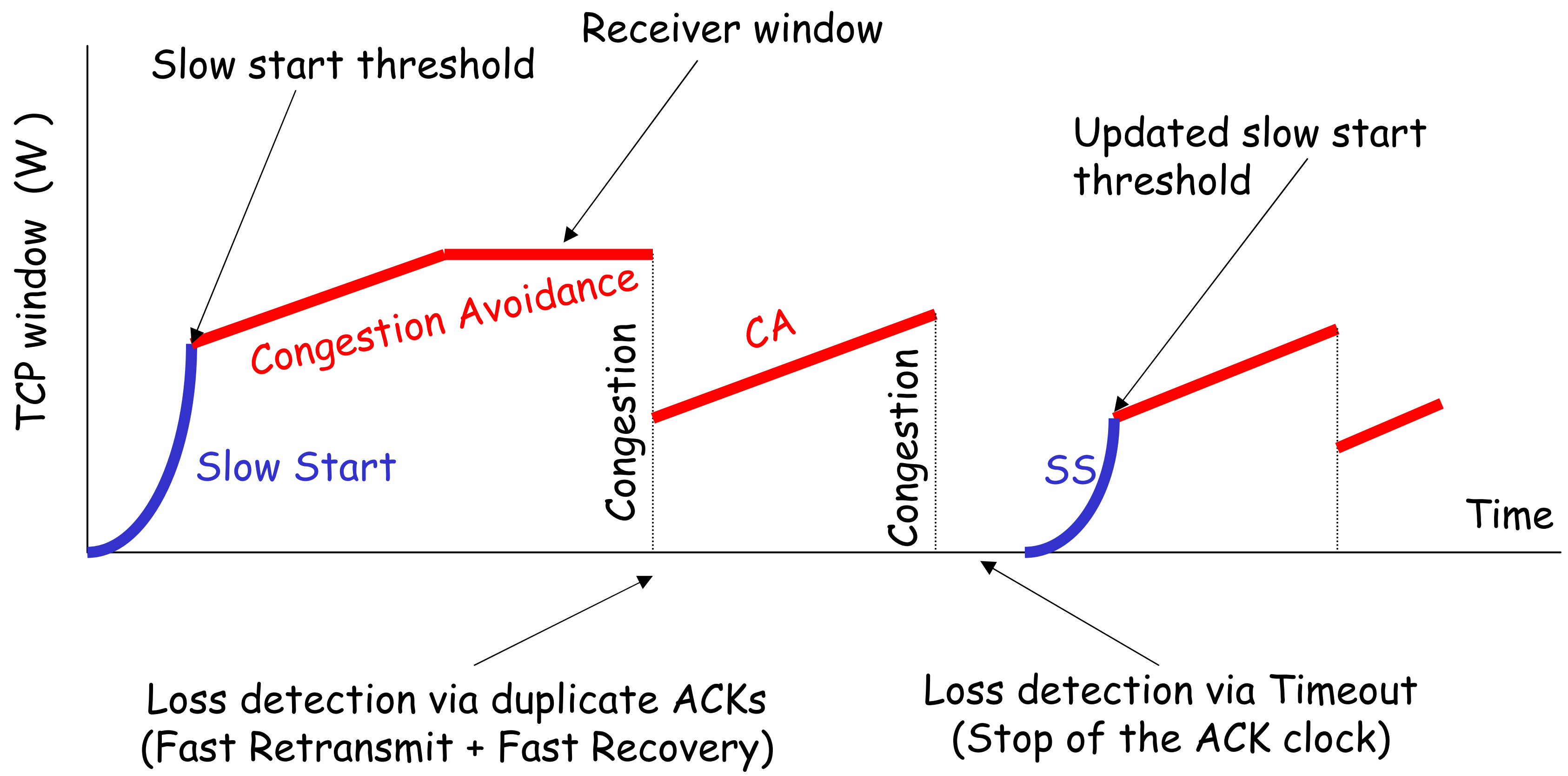
TCP performance

- This heterogeneity has a negative impact on TCP performance. Major factors impacting the performance of TCP:
  - The large bandwidth-delay product.
  - The long delay.
  - The transmission errors, or the so-called non-congestion losses.
  - The asymmetry of bandwidth.

- The impact of these factors will be studied:
  - Important performance issues.
  - Main solutions proposed in the literature.
  - Some results of our research in this field.

TCP in one slide !

- ❑ A window-based flow control protocol.
- ❑ The window size is adapted according to network conditions …



Receiver window
Slow start threshold
TCP window (W)
Updated slow start threshold
Congestion Avoidance
CA
Congestion
Congestion
Slow Start
SS
Time
Loss detection via duplicate ACKs (Fast Retransmit + Fast Recovery)
Loss detection via Timeout (Stop of the ACK clock)

Large bandwidth-delay product

- Performance issues:
  - The error recovery phase
  - TCP header
  - Buffering requirement in routers

TCP error recovery

- ❑ At large bandwidth-delay product, TCP window is large, and multiple packets can be lost in the same round-trip time.

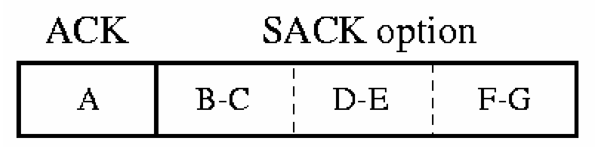
- ❑ Reno-based versions of TCP are not able to recover from many losses, without Timing-out.

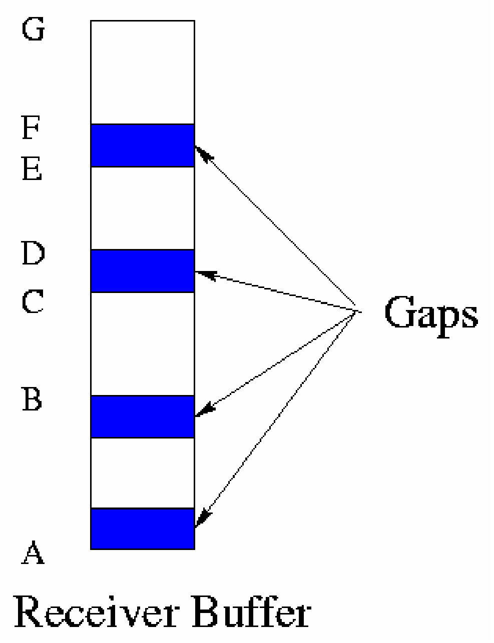
- ❑ **Ideal behavior:** Divide the window by two and recover quickly from errors, whatever is the number of packets lost in a round-trip time.

- ❑ The solution is brought by the SACK option ...

Selective ACK

- With SACK, the source has a better view of packets that have left the network, thus
  - It can retransmit quickly the packets lost.
  - While retransmitting, it injects new packets, which is necessary to keep the network well utilized.
- SACK is only the option. Algorithms at the source are needed to exploit this information, e.g.,TCP-SACK, FACK, Rate Halving, etc.





1

Problems caused by TCP header (1)

- Advertised window field (16 bits):
  - Prohibits TCP window from exceeding 64 Kbytes.
  - For a round-trip time RTT (seconds), this limits the throughput of TCP to 524288/RTT bps (934 Kbps for a satellite link of RTT = 560 ms).
  - **Solution:** A Window Scale Option that carries a number (on 14 bits) to be multiplied by the original window field. This allows the advertised window to grow up to 2 Gbytes !

Problems caused by TCP header (2)

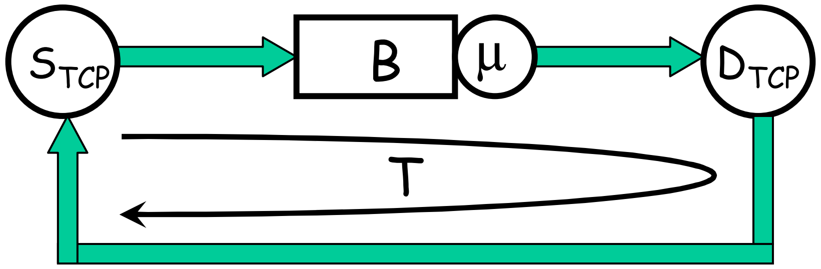
- Sequence Number field (32 bits):
  - TCP assumes that the maximum life of an IP packet is 2 minutes (MSL). It does not use the same TCP sequence number within 2.MSL. Thus,

    Throughput limited to 143 Mbps

  - **Solution:** Time-stamps option with a PAWS (Protect Against Wrapped Sequence Numbers) algorithm at the receiver. A received packet is discarded if it is sent before the last in-sequence one.

Buffering requirement in routers (1)

❑ Congestion avoidance mode:



- Congestion appears at $W_{max} = B + \mu T$
- TCP window oscillates between $W_{max}/2$ and W_{max}
- To achieve full link utilization, a buffer B larger than μT is required (larger than the bandwidth-delay product).
- This is harmful for interactive applications whose packets suffer from important queuing delays.
- Less buffer is required when:
  - A smooth version of TCP is used, as the Vegas version.
  - Multiple TCP connections share the network at the same time.

Buffering requirement in routers (2)

❑ Slow start mode:

- Buffers are also required to absorb the burstiness of the slow start phase, otherwise we get losses before TCP fully utilizes the available bandwidth. These losses are caused by what we call

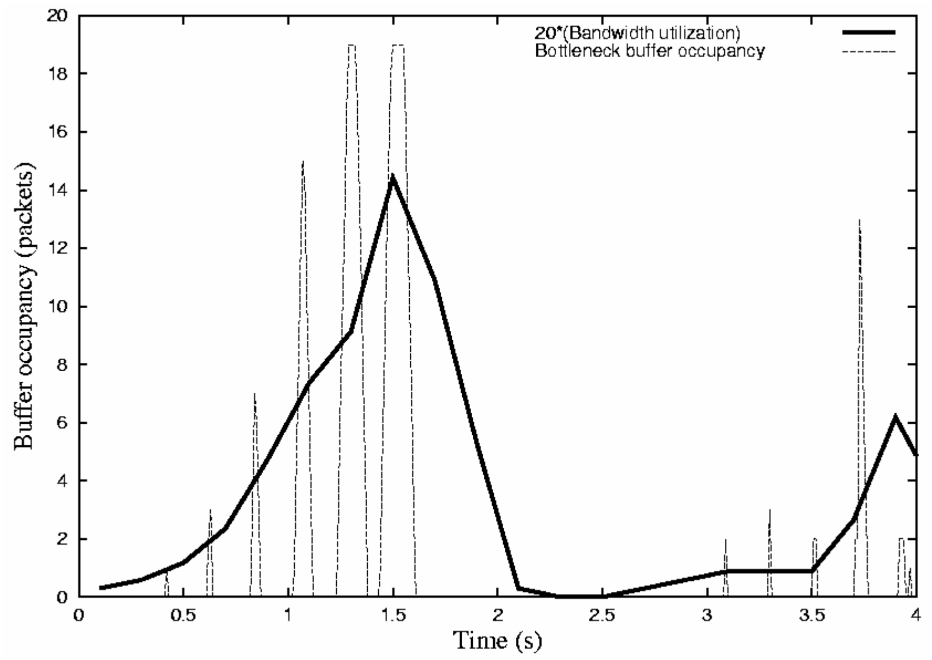
  Early Buffer Overflow.

- To reach a window W = μT, the buffer B has to be larger than
  - μT/2 for a receiver that acknowledges all packets.
  - μT/3 for a receiver than acknowledges one packet over two.

❑ Problem of Early Buffer Overflow:

- Underestimation of the available resources.
- Many packets lost at the same time, which is difficult to recover.

Buffering requirement in routers (3)



Simulation of early buffer overflow

- B = 20 packets
- m = 1.5 Mbps
- T = 200 ms

❑ Solutions:

- Set correctly the slow start threshold at the beginning (to avoid losses). The difficulty is in the estimation of B.
- Pace packets during slow start at a rate close to the link rate.
- Adapt the window increase rate in slow start as a function of the available buffer size.

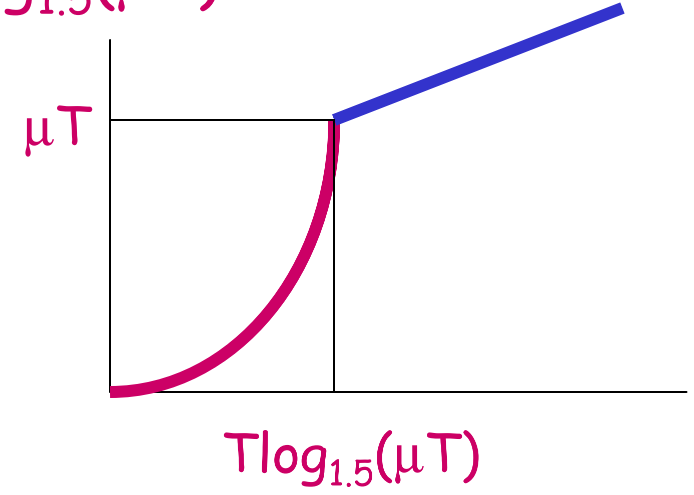
Long end-to-end delay

❑ Performance issues:

- Slow start
- Fairness

Problem of Slow Start

- Slow Start is a transitory phase that aims at gauging quickly the network resources at the beginning of the connection.
- If μT denotes the network capacity (in packets), then the duration of the slow start phase is $T\log_{1.5}(\mu T)$
- An increase in the end-to-end delay means a long slow start phase, which is dramatic since most of TCP transfers are of short duration (average transfer size = 10 Kbytes)



- Slow Start is the main problem of TCP in satellite networks !

Solutions (1)

❑ At the TCP level:

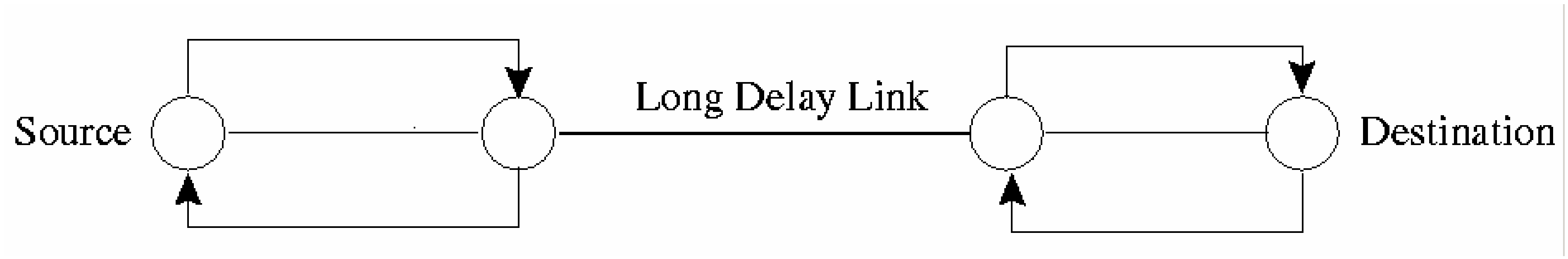
- Accelerate the window increase (results in bursts).
- Pace packets during slow start (solves the problem of bursts, and allows to reduce considerably the duration of slow start).

❑ At the application level (the easiest to deploy):

- Multiplex different transfers towards the same destination on the same TCP connection (HTTP 1.1, Ensemble TCP). This increases the total size of the transfer and reduces the impact of slow start.
- Use of proxies (bring the Internet to the edges). A long slow start phase is avoided when there is a cache hit.

Solutions (2)

- Isolate the long delay link by splitting the TCP connection:



  - A transport protocol optimized to the long delay is used in the middle.
  - TCP is used on both sides of the long delay link. The loop is now smaller and the slow start phase is no longer a bottleneck.

- Problems of this solution:
  - Breaks the end-to-end semantics.
  - Requires fixed and symmetric routing.
  - Does not work when the content of IP packets is encrypted.

Problem of Fairness

- The bias of TCP against connections of long round-trip time is well known. The reason is that the window is increased regardless of the round-trip time:
  - During slow start, TCP doubles W every RTT.
  - During congestion avoidance, W increases by 1 packet every RTT.
- Solutions:
  - Change the TCP code so that the windows of all TCP connections are increased in the same way as a function of time (this results in important burstiness for connections of long round-trip time).
  - Help TCP connections inside the network (by Active Queue Management, or by QoS architectures as DiffServ).

Non-congestion losses

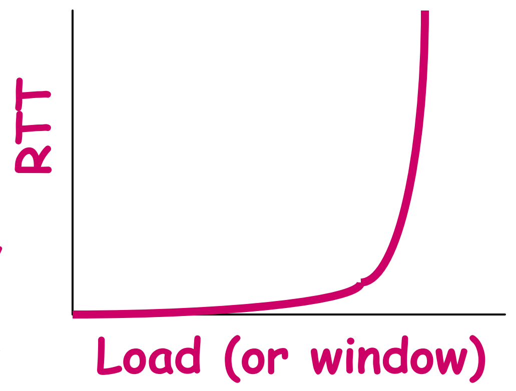
- ❑ Mainly caused by wireless networks (GSM, WLAN, Satellite).
- ❑ Different sources: Signal attenuation, Interference, multi-path fading, shadowing, rain, handoff, etc.
- ❑ Performance issues:
  - Impact on TCP and overview of the solutions.
  - Particular attention to FEC (Forward Error Correction).

TCP and non-congestion losses

- TCP misinterprets them as congestion signals and reduces its window unnecessarily.
- TCP-friendly network: Packets are only lost in routers !
- Three categories of solutions:
  - Clean links by correcting non-congestion losses locally (FEC, ARQ).
  - Help TCP to distinguish between congestion and non-congestion losses.
  - Split the TCP connection, isolate the noisy link, and transmit data over the noisy link using an optimized transport protocol.

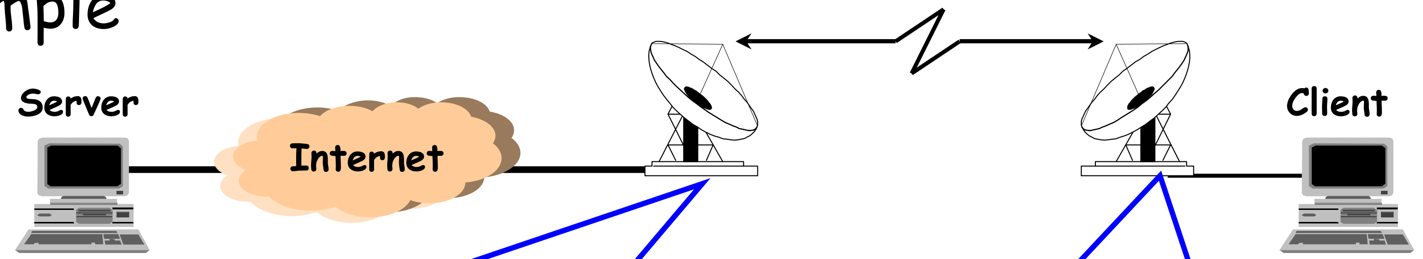
Making TCP more intelligent

- Noisy links send Explicit Loss Notification messages to TCP:
  - Very hard to implement in a wide network, and requires many changes.
  - Can be a good solution when the TCP source is directly connected to the noisy link (case of a mobile sending data).
- Distinguish between losses without the help of the network:
  - Correlation between window and Round-Trip Time is usually used.
    - If RTT increases with the window, losses are then caused by congestion.
    - Otherwise, they are caused by transmission errors.
  - Problems:
    - Impossible to make a perfect distinction.
    - Bad interaction with routers such as RED which early drop packets to signal the imminence of a congestion.

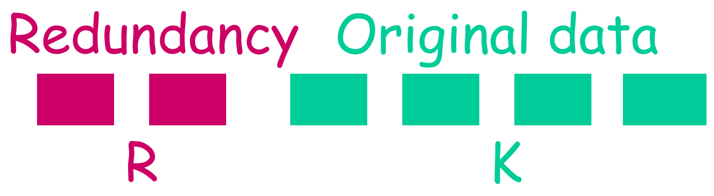


TCP and link-level FEC

- FEC shields TCP from transmission errors by correcting them locally without the intervention of TCP.
- Example



Division of a TCP packet into K link-level units (e.g., ATM cells) and addition of R units of redundancy



N= K + R
Relative amount of FEC = N/K

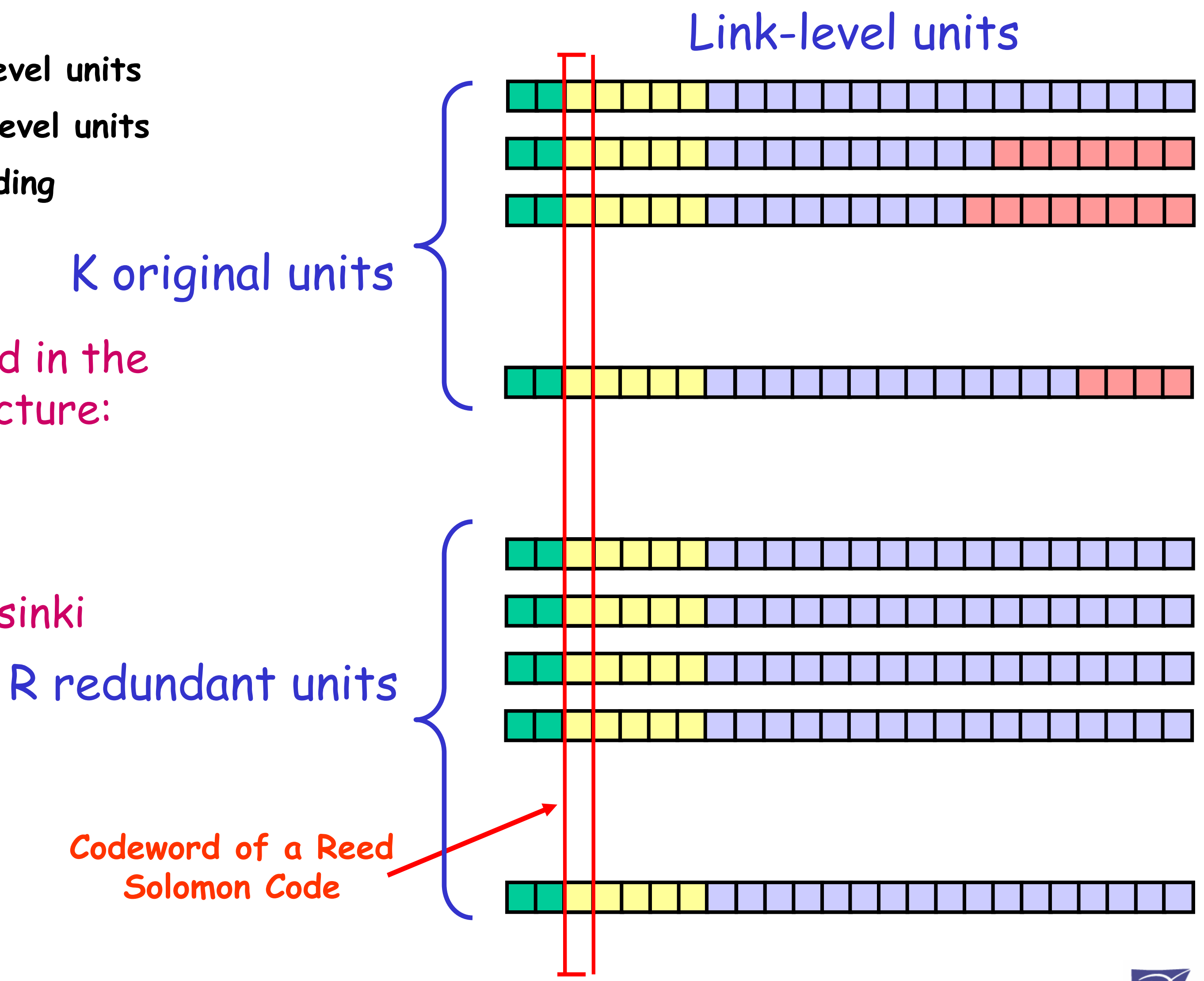
Delivery of a packet to higher layers if at least K of its N link-level units are correctly received.

A TCP packet can now resist to the loss of up to R units without being discarded.

Implementation

- 🟩 FEC header
- 🟨 Header of link-level units
- ⬜ Payload of link-level units
- 🟥 Virtual zero padding

FEC as implemented in the TRANSAT architecture:
- ALCATEL Space
- INRIA
- ENSICA
- University of Helsinki



Pros and Cons

- FEC incomes:
  - Reduces the packet loss rate.
  - Correct packets on the fly, which eliminates any interaction with TCP retransmission timer as in the case of ARQ.
  - FEC is of particular interest on long delay links and at high loss rates.
- FEC cost:
  - Processing overhead (it may not be the real problem).
  - The redundant information consumes bandwidth, which may reduce the throughput of TCP if added in large amounts.
- What is the amount of FEC that leads to the best TCP throughput?
  - Case study: The wireless link is the bottleneck for ***C*** identical long-lived TCP connections.

Modeling TCP / FEC

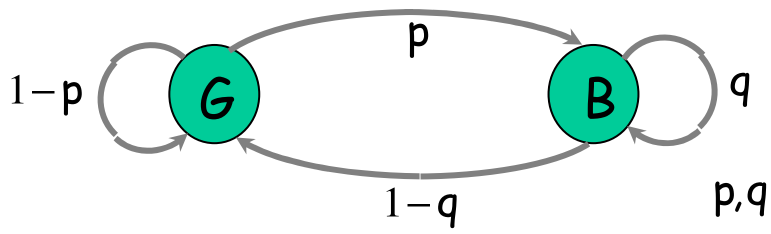
- **P** = Loss probability of a TCP packet over the wireless link.
- μ = Bandwidth of the wireless link in terms of link-level units per second.
- ***f*** **(P,RTT)** = TCP throughput with infinite μ (e.g. square root formula $1/\mathrm{RTT}\sqrt{3/(2\,P)}$).
- When μ is finite

$$\text{Total TCP throughput} = \min\left(C.f(P,RTT), \tfrac{K}{N}\mu\right)$$

- **Problem:** Calculate P as a function of the amount of FEC and the loss process of link-level units (two-state Gilbert model to account for burstiness).

p = Prob{Unit **n** lost | Unit **n-1** not lost}

q = Prob{Unit **n** lost | Unit **n-1** lost}



$p,q \in (0,1)$

INRIA SOPHIA ANTIPOLIS

Analysis

- Non-bursty case (Bernoulli):

  P = Prob{more than R losses in a frame of N units} = $\sum_{i=0}^{K-1} \binom{N}{i} (1-p)^i p^{N-i}$

- Bursty case: Some assumptions are needed

  - Only one burst of losses can hurt a TCP packet at a time.
  - The wireless link converges quickly to its stationary regime.

  P = Prob{a burst of more than R losses hurts somewhere the packet}

  $$\approx q^{N-K} L ((1-q)K+q)$$

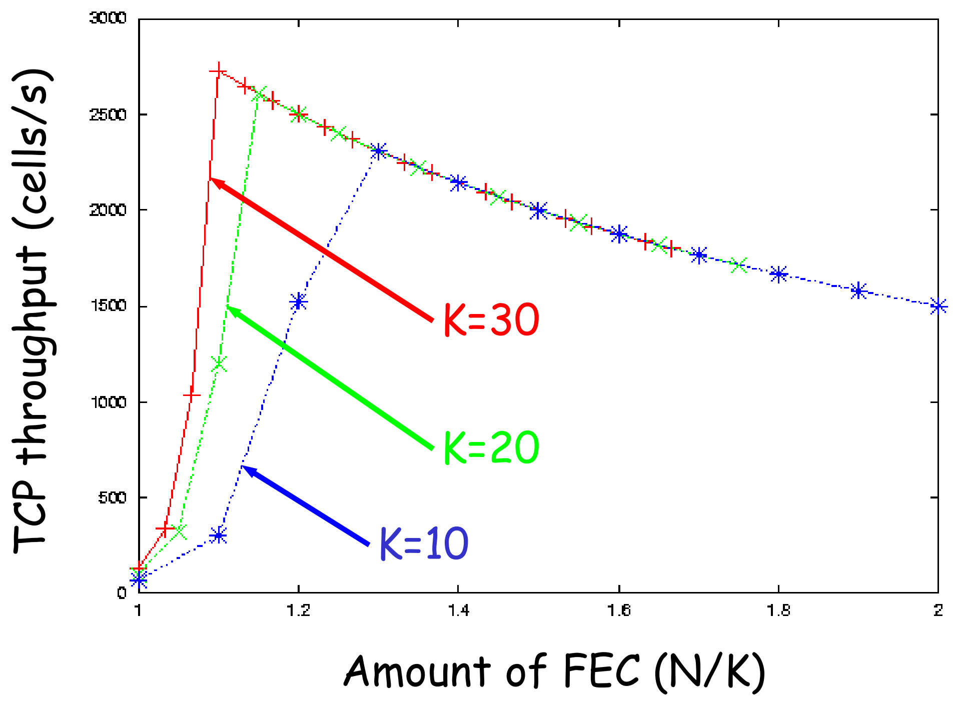
  with **L** being the average loss rate equal to $p/(p+1-q)$.

P to be plugged in the expression of the throughput

Numerical results: Non-bursty case (1)

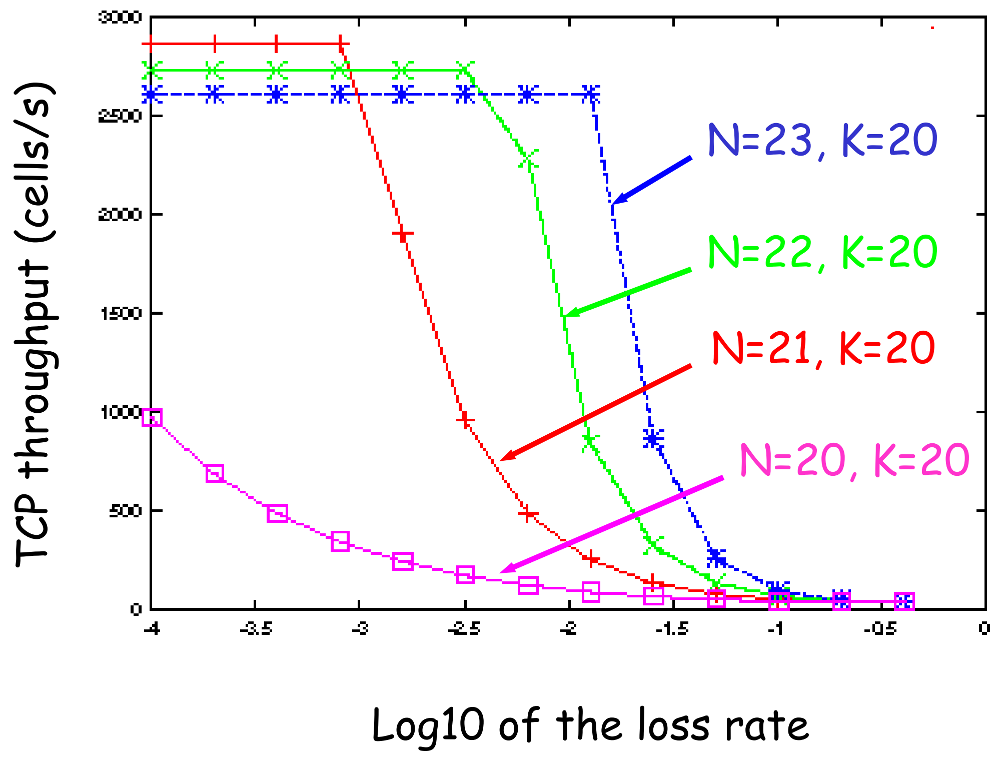
Case study: One long-life TCP connection over a 1.5 Mbps wireless ATM link. Loss rate 1%.

- TCP throughput increases with the amount of FEC until the optimal point, then starts to decrease.
- For the same amount of FEC, large frames (large packets) give better performance due to a faster TCP window increase and a better protection from bursts.



Numerical results: Non-bursty case (2)

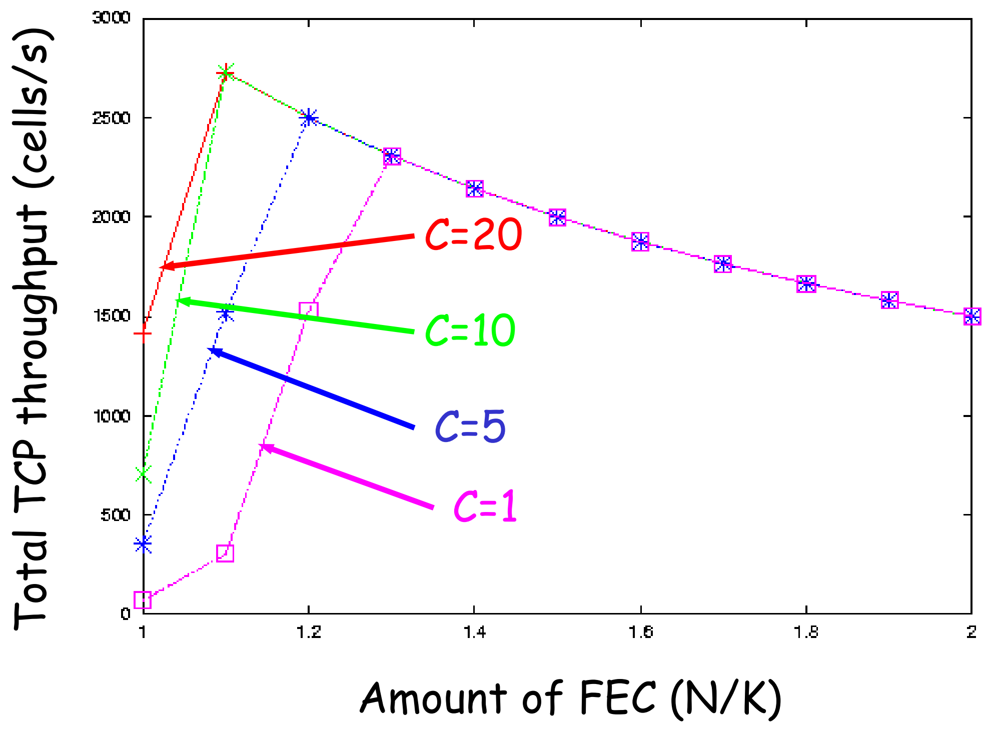
- The more the FEC, the better the resistance of TCP to an increase in the loss rate.
- The cost for a large amount of FEC is a smaller TCP throughput at low loss rate.



Numerical results: Non-bursty case (3)

- Better utilization of the link can be obtained by opening multiple TCP connections, which allows to reduce the amount of FEC without reducing the total throughput.

C = Number of TCP connections

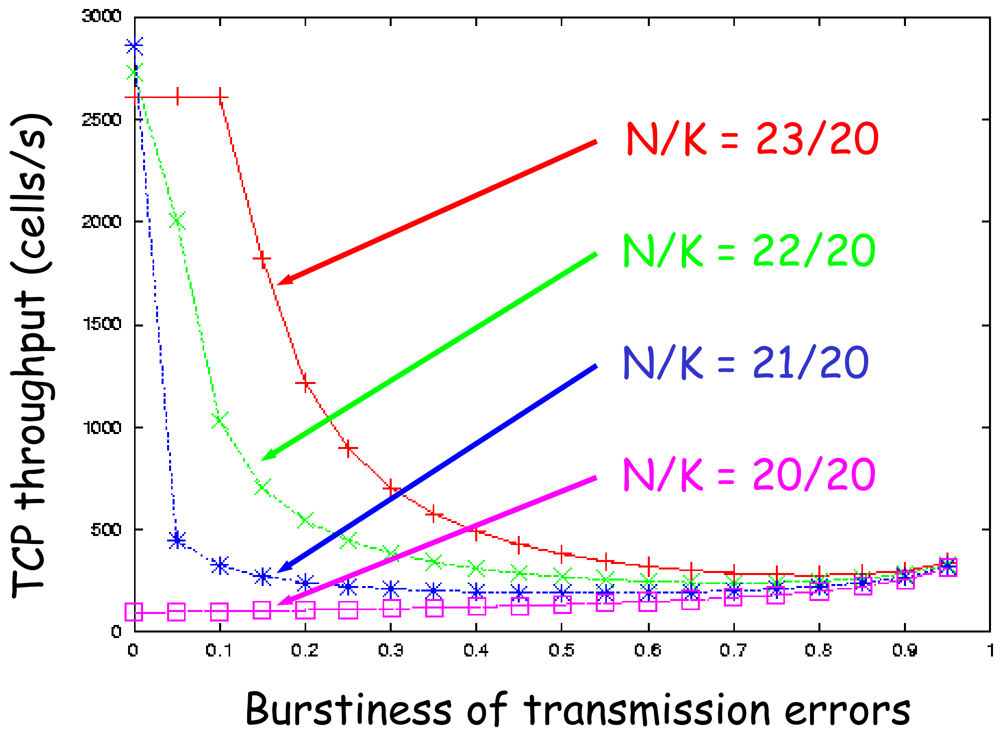


INRIA SOPHIA ANTIPOLIS

Numerical results: Bursty case (1)

For an average loss rate of 1% and for one TCP connection, we increase the burstiness of errors (by increasing **q** from 0 to 1):

- A large amount of FEC resists better to a clustering of errors, but at the expense of a smaller TCP throughput when errors stop being bursty.

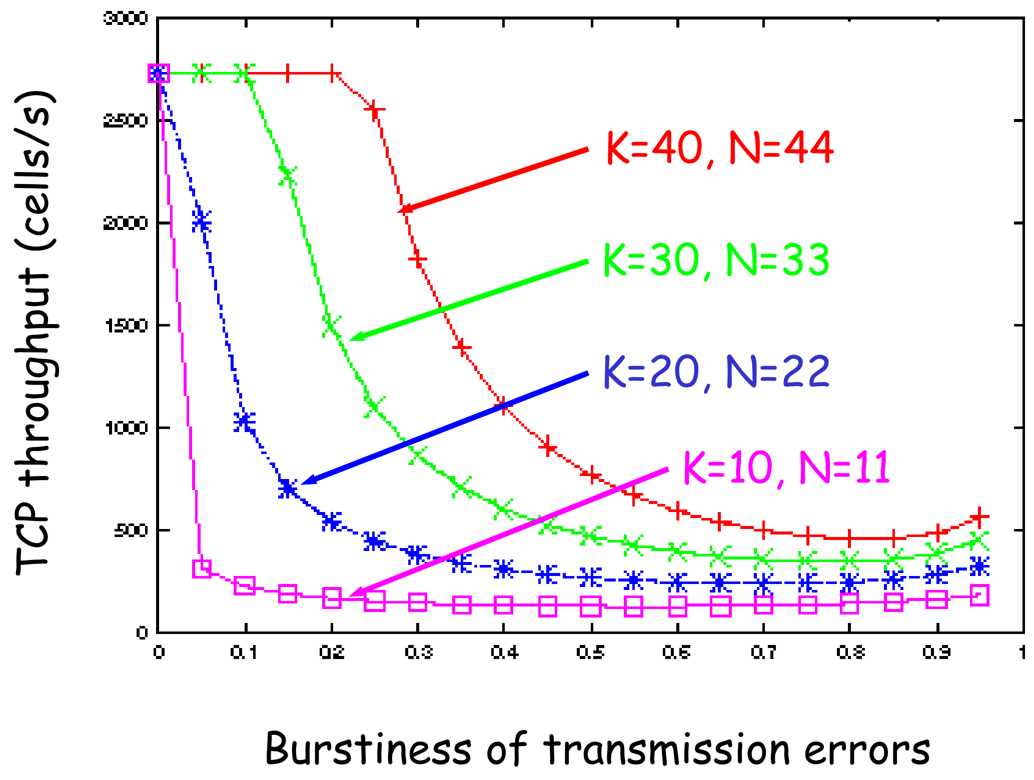


Numerical results: Bursty case (2)

For the same amount of FEC, we study the impact of the block size:

- A large frame size results in a better performance due to a faster window increase and a larger number of redundant units per block.

- The advantage of increasing the frame size is that we will not lose in performance when errors stop being bursty.

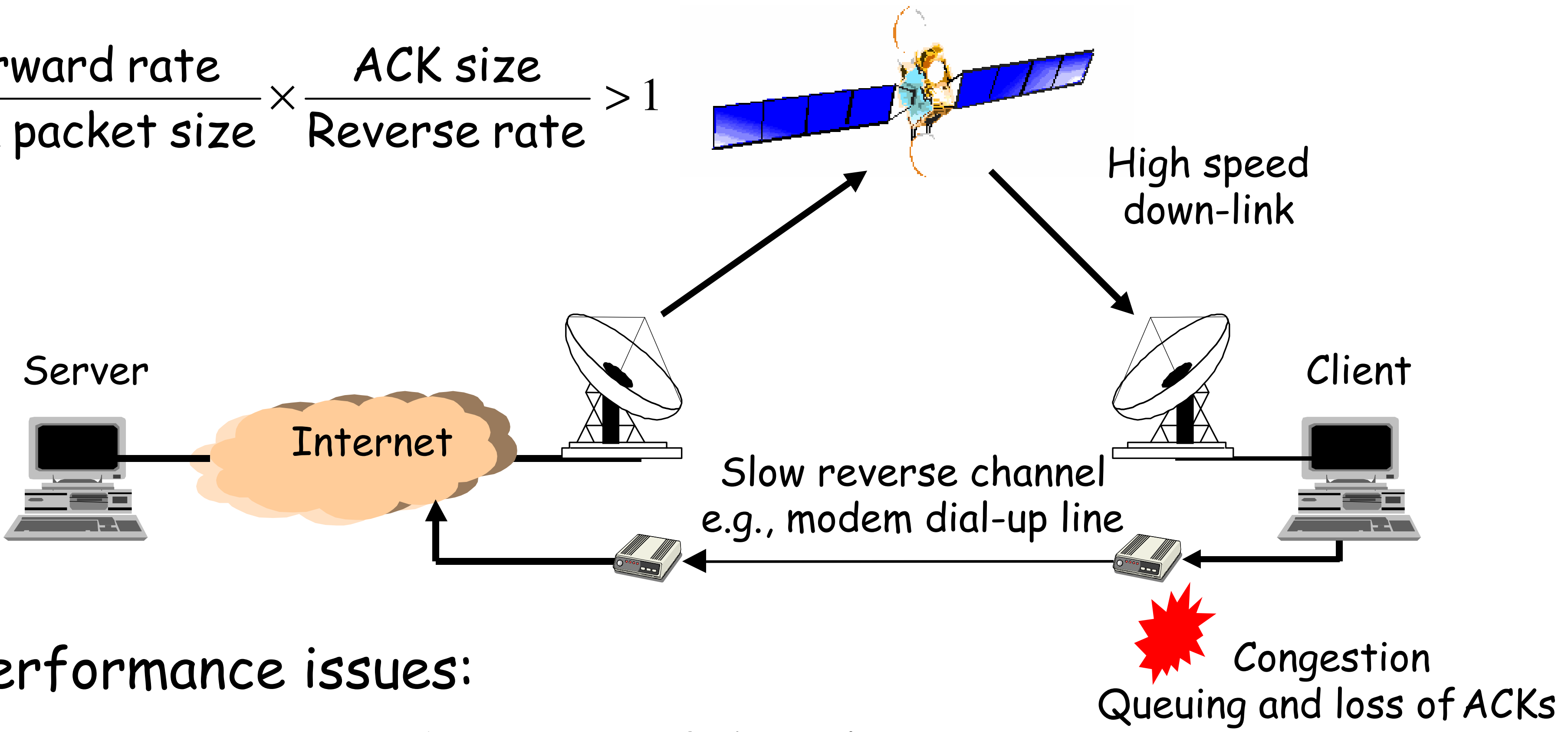


Hybrid ARQ/FEC

- ARQ is interesting on short delay links and at low loss rate:
  - Pros: Bandwidth is only wasted when packets are lost.
  - Cons:
    - Introduce jitter, which is harmful for real time applications.
    - Introduce burstiness when an in-order delivery is supported.
    - Introduce reordering when an out-order delivery is supported.
    - Interfere with TCP timeout mechanism when persistency is high.
- Better performance can be obtained by combining FEC and ARQ. For example, in TRANSAT:
  - Transmit original packets without redundancy.
  - Protect retransmissions with redundancy (given the long delay of a satellite link, only one retransmission is allowed).

Bandwidth asymmetry

$$\frac{\text{Forward rate}}{\text{Data packet size}} \times \frac{\text{ACK size}}{\text{Reverse rate}} > 1$$



❑ Performance issues:

- Impact on TCP and overview of the solutions.
- Particular attention to ACK filtering.

TCP and bandwidth asymmetry

- Problems:
  - Increase in the end-to-end delay.
  - The loss of ACKs causes burstiness and slows the window growth.
  - The non-responsiveness of the ACK stream to drops causes unfairness and blockage of new connections.
- Solutions to congestion:
  - TCP/IP header compression.
  - Generate less ACKs at receivers.
  - Drop non useful ACKs in the reverse buffer (ACK Filtering).
- Solutions to burstiness:
  - Regenerate ACKs at a regular rate in the network.
  - Byte counting + packet pacing at sources.

ACK filtering

- Erase old ACKs from a connection when a new ACK arrives (Literature: one ACK per-connection in the buffer).
- This reduces to the minimum the queue length.
  But, ACKs are also necessary for sender operation especially during slow start where they are used to increase the window.
- ACK Filtering tradeoff:

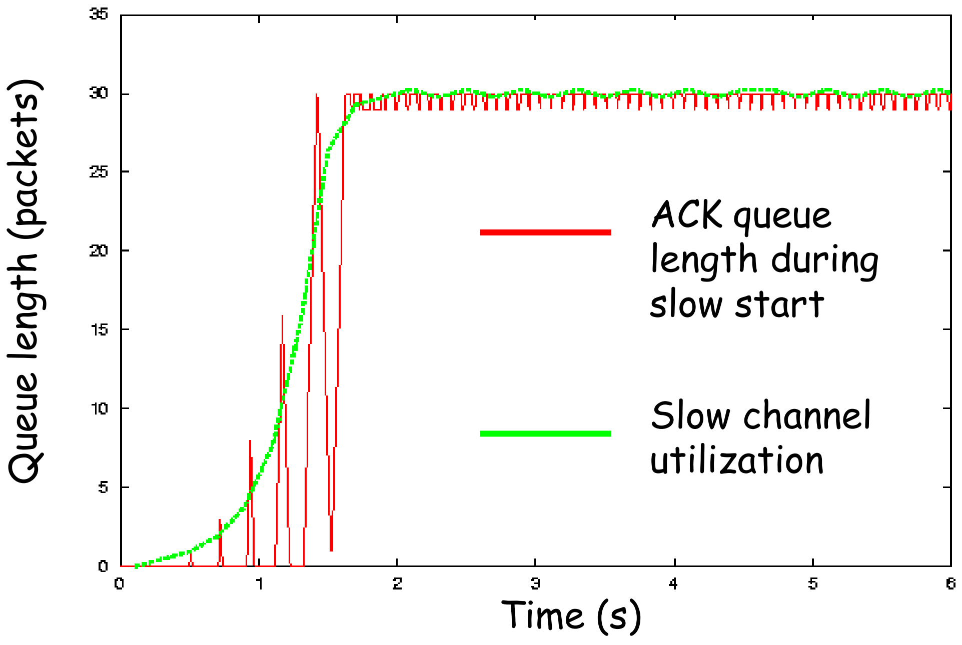
Reduce the length of the queue of ACKs to the minimum

vs.

Pass the maximum possible number of ACKs to the source

Why there is a problem?

- There is no contradiction between the two objectives if ACKs arrive smoothly at the reverse slow channel.
- But, ACKs arrive in bursts during slow start. They may also arrive in bursts due to clustering of packets in the network.
- Filtering bursts of ACKs before the full utilization of the reverse channel is not necessary since these bursts if absorbed, will not cause an increase in the round-trip time, but if filtered, they will slow considerably the window increase.



- **Optimal behavior:**

Filter ACKs just after the full utilization of the reverse slow channel

Delay ACK filtering

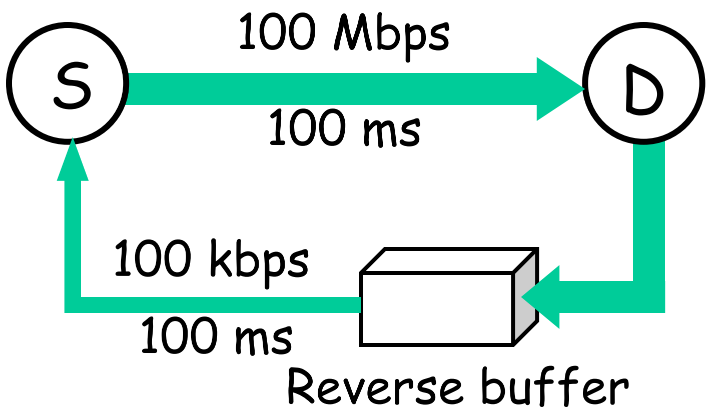
- Algorithm:
  - Measure the reverse slow link utilization.
  - Halt the filtering when the utilization is less than a certain threshold.
  - Filter ACKs if not (substitute old ACKs by new ones).
- Measuring link utilization:
  - Use the Time Sliding Window (TSW) algorithm to measure the rate of ACKs at the output of the slow reverse channel [Clark,Fang,1998]

TSW: *An Exponentially Weighted Moving Average algorithm with the advantage that the convergence rate is constant and not dependent on the frequency and the value of measurements.*

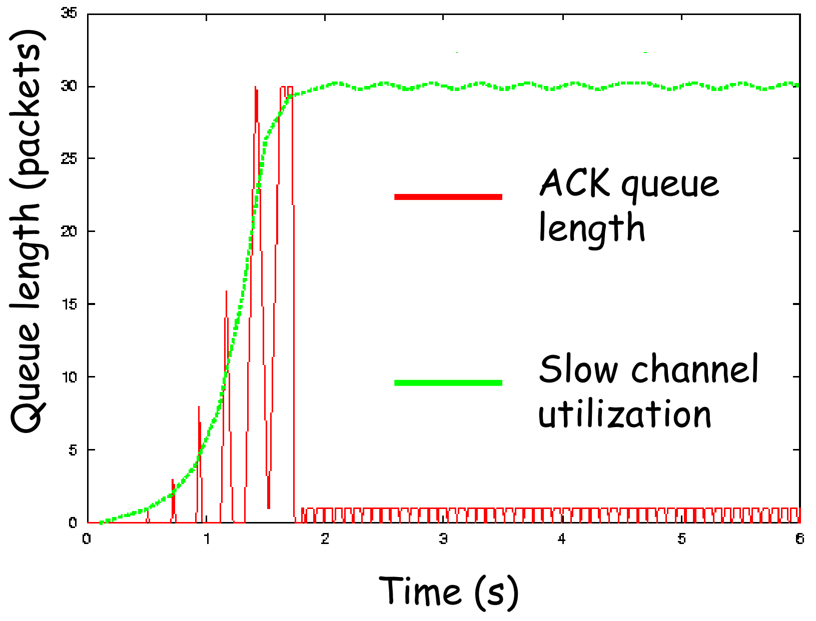
When an ACK leaves the buffer $\text{AvgRate} \leftarrow \dfrac{\text{AvgRate} \times \text{Window} + \text{ACKSize}}{\text{Window} + (\text{Now} - \text{LastTime})}$

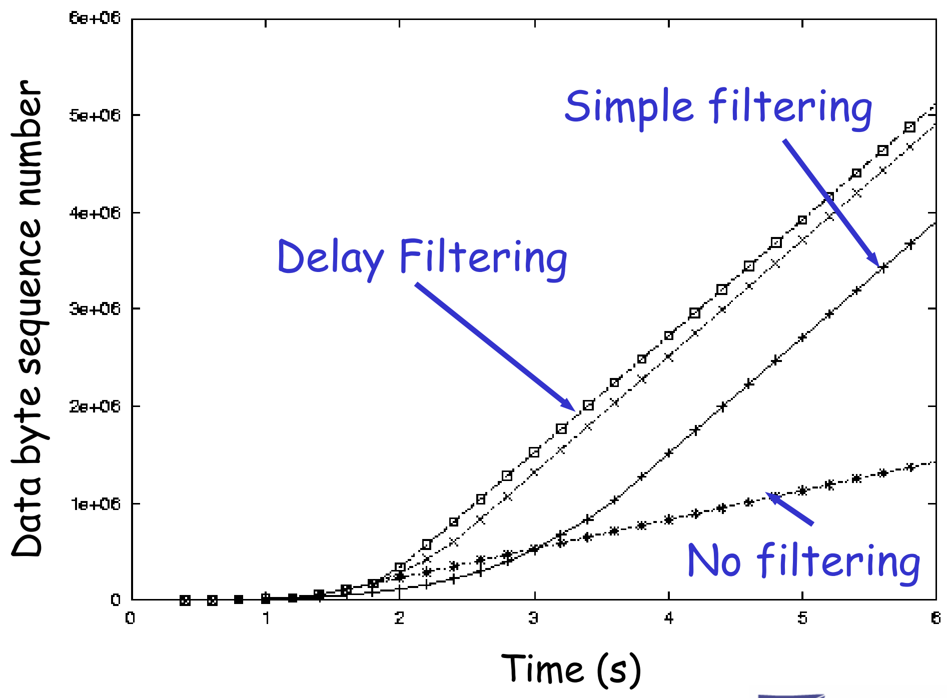
4

Validation: Case of 1 connection



- A long TCP transfer from S to D
- TCP version : Reno
- Packet size : 1000 bytes
- Very large reverse buffer



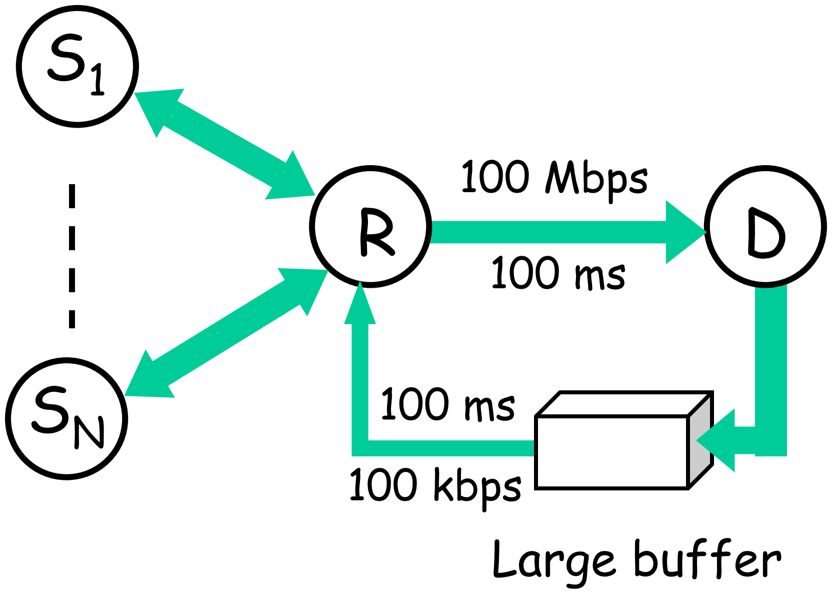


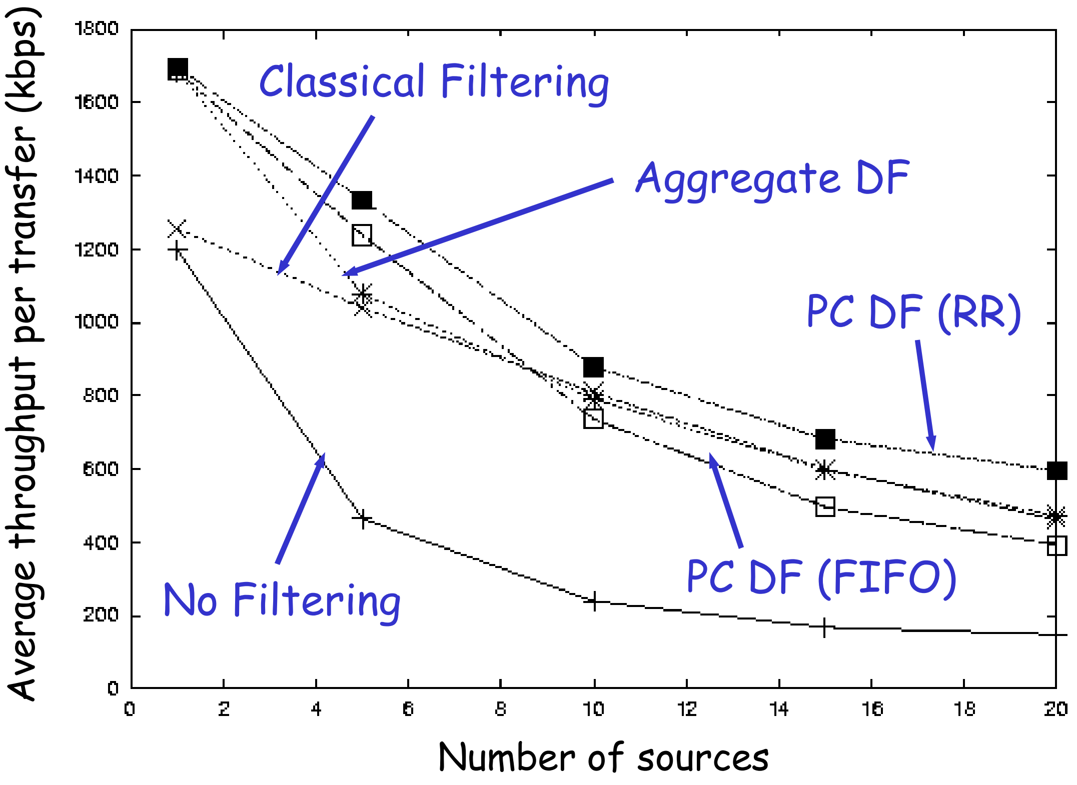
Case of multiple connections

- ❑ Need for a per-connection delay filtering
  - Using only the channel utilization to decide on ACK filtering is not enough to protect new connections from already running ones.
  - Algorithm:
    - – Measure the rate of ACKs for every active connection.
    - – Filter ACKs of a connection when their rate exceeds their fair share of the slow channel bandwidth (bandwidth / # connections).
- ❑ Need for a per-connection scheduling
  - ACKs from filtered connections need to be protected from waiting behind ACKs from unfiltered ones (by use of a Round Robin scheduler).
- ❑ Need to isolate ACKs from Data packets of reverse connections (current work)

Validation: Case of M connections

- Every source transmits successively files of size chosen randomly between 10 kbytes and 10 Mbytes





- Best performance for Round Robin (RR) and Per-Connection Delay Filtering (PC DF)
- No gain from FIFO when M is large

Conclusions

- New transmission technologies are challenging for TCP:
  - Four generic features are presented.
  - Other generic features may also have a negative impact on TCP: variable delay paths (mobility), change in routes (handoff), dynamic bandwidth (shared media), etc.
- Beliefs:
  - TCP is always able to avoid network congestion. This may not be the case if TCP does not react to packet losses, but to other forms of congestion signals (e.g. ECN, increase in round-trip time).
  - TCP can be however over-conservative (problems of fairness).
  - There is always some small improvements to be brought to TCP.
  - But, most of the work has be to done at the link-level (Spirit of PILC).

Selected publications

C. Barakat, E. Altman, W. Dabbous, "On TCP Performance in a Heterogeneous Network : A Survey", IEEE Communications Magazine, vol. 38, no. 1, pp. 40-46, January 2000.

C. Barakat, E. Altman, "Bandwidth tradeoff between TCP and link-level FEC", Computer Networks, vol. 39, no. 2, pp. 133-150, June 2002.

C. Barakat, E. Altman, "On ACK Filtering on a Slow Reverse Channel", International Workshop on Quality of future Internet Services (QofIS), September 2000.

More references at

http://www.inria.fr/planete/personnel/Chadi.Barakat