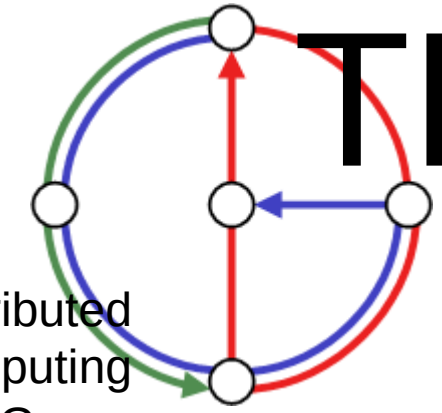
Chapter 13
TRANSPORT



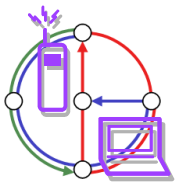
Distributed
Computing
Group

Mobile Computing

Winter 2005 / 2006

Overview

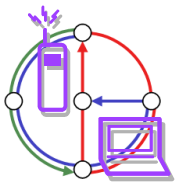
- Motivation
- Simple analysis
- Various TCP mechanisms



TCP Overview

- Transport control protocols typically designed for
  - Fixed end-systems in wired networks
- Research activities
  - Performance
  - Congestion control
  - Efficient retransmissions

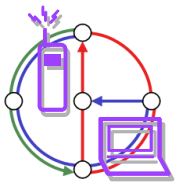
- TCP congestion control
  - packet loss in fixed networks typically due to (temporary) overload situations
  - router have to discard packets as soon as the buffers are full
  - TCP recognizes congestion only indirectly via missing acknowledgements, retransmissions unwise, they would only contribute to the congestion and make it even worse



TCP slow-start

- sender calculates a congestion window for a receiver
- start with a congestion window size equal to one segment
- exponential increase* of the congestion window up to the congestion threshold, then linear increase
- missing acknowledgement causes the reduction of the congestion threshold to one half of the current congestion window
- congestion window starts again with one segment

*slow-start vs. exponential increase: window is increased by one for each acknowledgement, that is, $1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \ldots$ In other words, the slow-start mechanism is rather a “quick-start”.

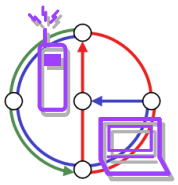


TCP fast retransmit/fast recovery

- TCP sends an acknowledgement only after receiving a packet
- If a sender receives several acknowledgements for the same packet, this is due to a gap in received packets at the receiver
- Sender can retransmit missing packets (fast retransmit)

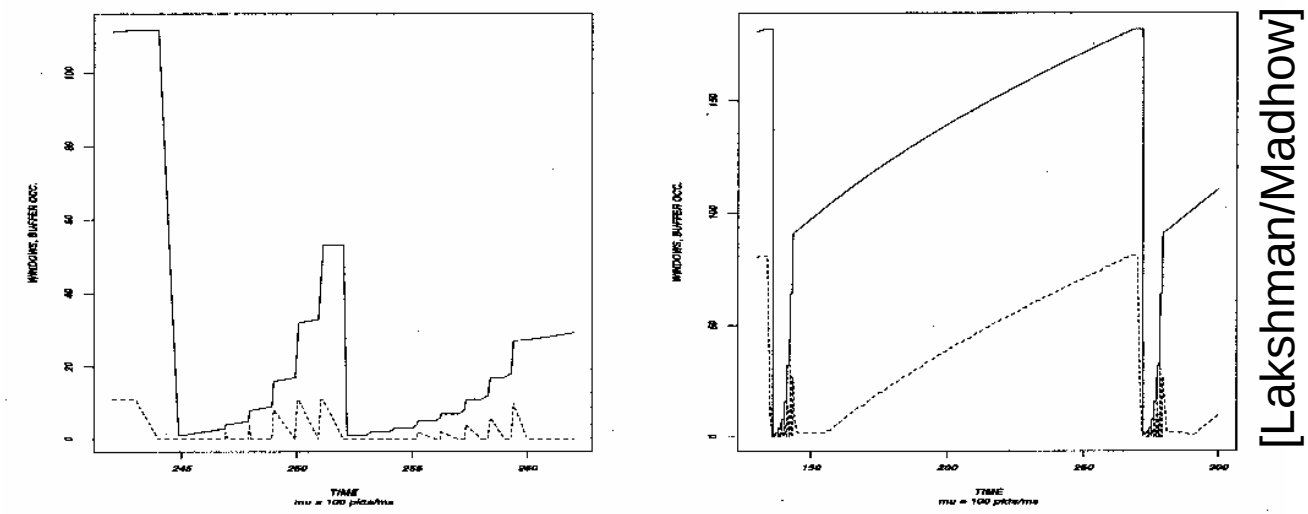
- Also, the receiver got all packets up to the gap and is actually receiving packets
- Therefore, packet loss is not due to congestion, continue with current congestion window (fast recovery)

- In the following simplied analysis, we do consider neither fast retransmit nor fast recovery.



TCP on lossy (wireless) link

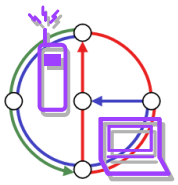
- Without fast retransmit/fast recovery



[Lakshman/Madhow]

Very high loss probability

High loss probability

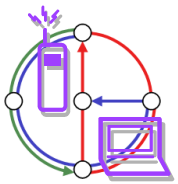


Simple analysis model for lossy TCP

- Segment loss probability q = 1–p
- We are interested in the throughput T

- If there are no losses (and all ACKs are received in time):
  - Number of segments S first doubles up to threshold W (slow start phase)
  - Number of segments S is incremented after S successful ACKs
  - At some point we reach the bandwidth of the channel B

- If there is a loss
  - We go back to S = 1



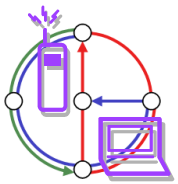
Simple analysis of lossy TCP

- For not too high error probability q we are usually in the congestion mode (that is: not the slow start mode).

- The expected number of successful transmission E before we get an error is $$E = \sum_{i=0}^{\infty} i \cdot p^i(1-p) = \frac{p}{1-p}$$

- In the equilibrium, we are in the states 1, 2, 3, …, S-1, S, and then back to 1 because we have a missing ACK, that is, we have $1+2+\ldots+S \approx S^2/2$ successful transmissions.

- With $S^2/2 = E = p/q$ we get $T = S/2 = \sqrt{1/2 \cdot p/q} = \Theta(1/\sqrt{q})$, for $p = \Theta(1)$

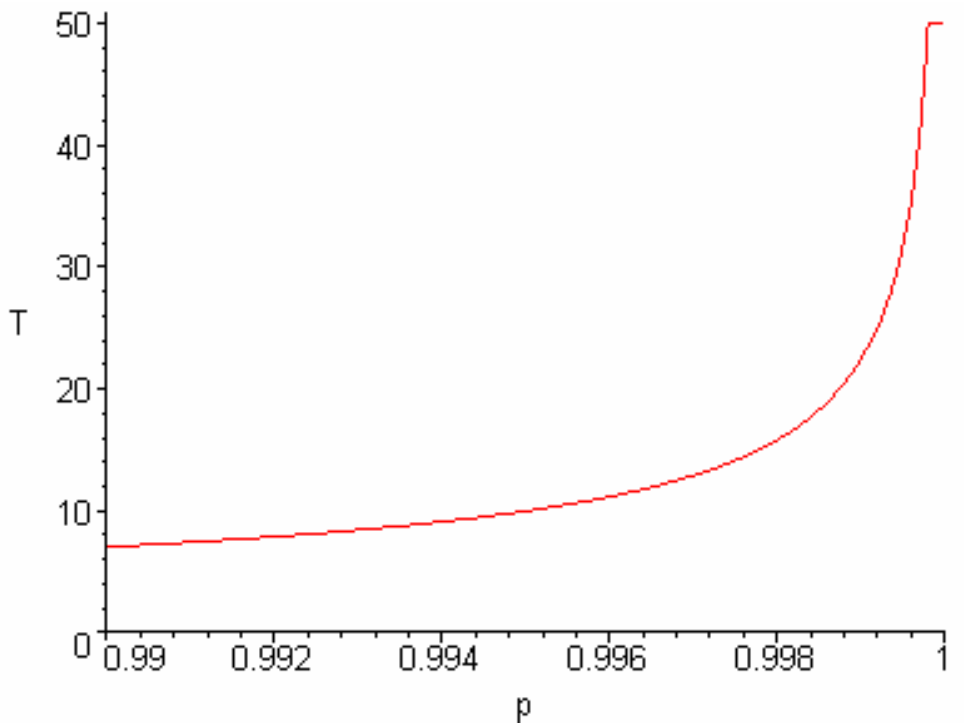


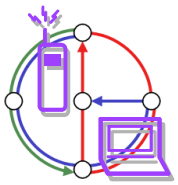
Lossy TCP: Graphical Interpretation

- $T(p) = \min\left(B, \sqrt{\frac{p}{2(1-p)}}\right)$

- Plot of $T(p)$, with $B = 50$

- Note that 1% faulty transmissions is enough to degrade the throughput to about 14% of the bandwidth
- 10% error rate gives about 4% of possible bandwidth.
- The higher the bandwidth, the worse the relative loss.

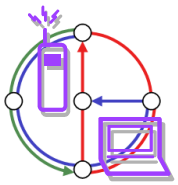




Mobility and TCP

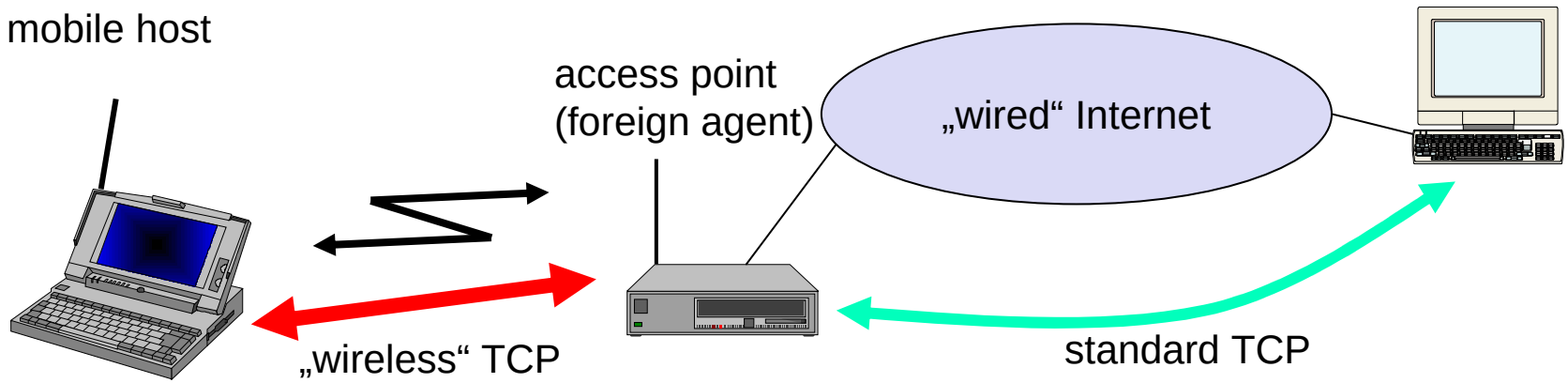
- TCP assumes congestion if packets are dropped
  - typically wrong in wireless networks, here we often have packet loss due to *transmission errors*
  - furthermore, *mobility* itself can cause packet loss, if e.g. a mobile node roams from one access point (e.g. foreign agent in Mobile IP) to another while there are still packets in transit to the wrong access point and forwarding is not possible

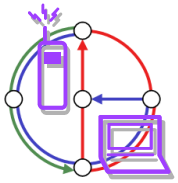
- The performance of an unchanged TCP degrades severely
  - however, TCP cannot be changed fundamentally due to the large base of installation in the fixed network, TCP for mobility has to remain compatible
  - the basic TCP mechanisms keep the whole Internet together



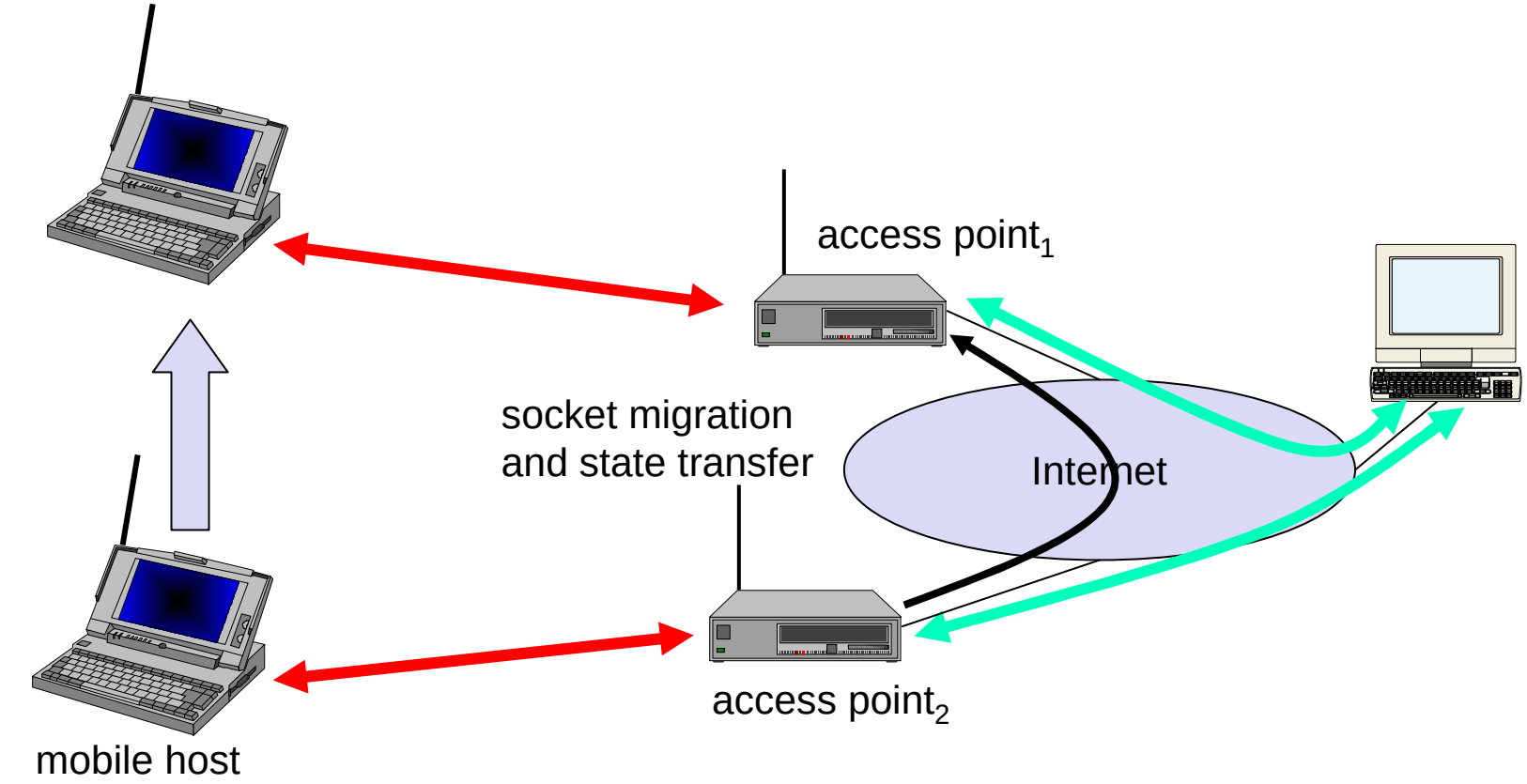
Early Approach: Indirect TCP (I-TCP)

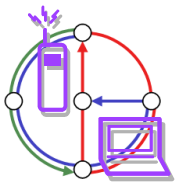
- segments the connection
  - no changes to the TCP protocol for hosts connected to the wired Internet, millions of computers use (variants of) this protocol
  - optimized TCP protocol for mobile hosts
  - splitting of the TCP connection at, e.g., the foreign agent into two TCP connections, no real end-to-end connection any longer
  - hosts in the fixed part of the net do not notice the characteristics of the wireless part





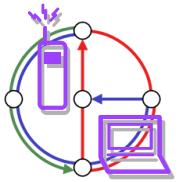
I-TCP socket and state migration





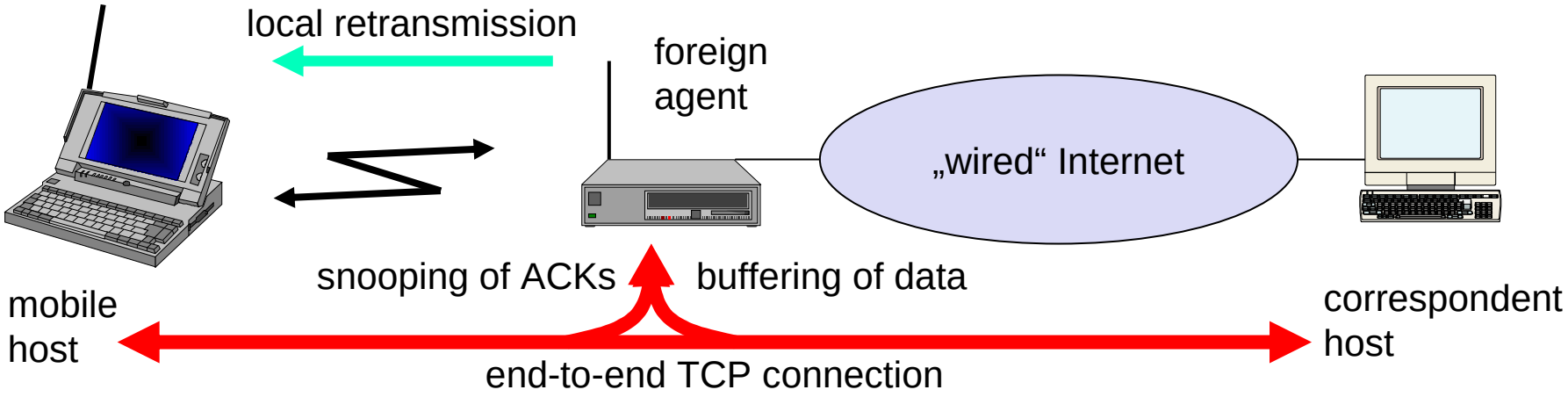
Indirect TCP Advantages and Disadvantages

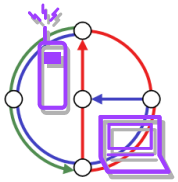
+ no changes in the fixed network necessary, no changes for the hosts (TCP protocol) necessary, all current optimizations to TCP still work
+ transmission errors on the wireless link do not propagate into the fixed network
+ simple to control, mobile TCP is used only for one hop, between a foreign agent and a mobile host
+ therefore, a very fast retransmission of packets is possible, the short delay on the mobile hop is known
– loss of end-to-end semantics, an acknowledgement to a sender does now not any longer mean that a receiver really got a packet, foreign agents might crash
– higher latency possible due to buffering of data with the foreign agent and forwarding to a new foreign agent
– high trust at foreign agent; end-to-end encryption impossible



Early Approach: Snooping TCP

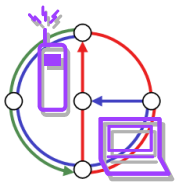
- Transparent extension of TCP within the foreign agent
  - buffering of packets sent to the mobile host
  - lost packets on the wireless link (both directions!) will be retransmitted immediately by the mobile host or foreign agent, respectively (so called “local” retransmission)
  - the foreign agent therefore “snoops” the packet flow and recognizes acknowledgements in both directions, it also filters ACKs
  - changes of TCP only within the foreign agent





Snooping TCP

- Data transfer to the mobile host
  - FA buffers data until it receives ACK of the MH, FA detects packet loss via duplicated ACKs or time-out
  - fast retransmission possible, transparent for the fixed network
- Data transfer from the mobile host
  - FA detects packet loss on the wireless link via sequence numbers, FA answers directly with a NACK to the MH
  - MH can now retransmit data with only a very short delay
- Integration of the MAC layer
  - MAC layer often has similar mechanisms to those of TCP
  - thus, the MAC layer can already detect duplicated packets due to retransmissions and discard them
- Problems
  - snooping TCP does not isolate the wireless link as good as I-TCP
  - snooping might be useless depending on encryption schemes



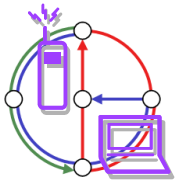
Early Approach: Mobile TCP

- Special handling of lengthy and/or frequent disconnections
- M-TCP splits as I-TCP does
  - unmodified TCP fixed network to supervisory host (SH)
  - optimized TCP SH to MH
- Supervisory host
  - no caching, no retransmission
  - monitors all packets, if disconnection detected
    - set sender window size to 0
    - sender automatically goes into persistent mode
  - old or new SH re-open the window

\+ maintains end-to-end semantics, supports disconnection, no buffer forwarding

– does not solve problem of bad wireless link, only disconnections

– adapted TCP on wireless link; new software needed

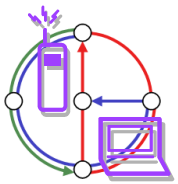


Fast retransmit/fast recovery

- Problem: Change of foreign agent often results in packet loss
  - TCP reacts with slow-start although there is no congestion
- Solution: Forced fast retransmit
  - as soon as the mobile host has registered with a new foreign agent, the MH sends (three) duplicated acknowledgements on purpose
  - this forces the fast retransmit mode at the communication partners
  - additionally, the TCP on the MH is forced to continue sending with the actual window size and not to go into slow-start after registration

\+ simple changes result in significant higher performance

– what a hack…



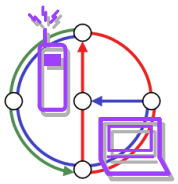
Transmission/time-out freezing

- Mobile hosts can be disconnected for a longer time
  - no packet exchange possible, e.g., in a tunnel, disconnection due to overloaded cells or multiplex with higher priority traffic
  - TCP disconnects after time-out completely

- TCP freezing
  - MAC layer is often able to detect interruption in advance
  - MAC can inform TCP layer of upcoming loss of connection
  - TCP stops sending, but does now not assume a congested link
  - MAC layer signals again if reconnected

\+ scheme is independent of data

– TCP on mobile host has to be changed, mechanism depends on MAC layer



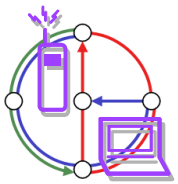
Selective retransmission

- TCP acknowledgements are often cumulative
  - ACK n acknowledges correct and in-sequence receipt of packets up to n
  - if single packets are missing quite often a whole packet sequence beginning at the gap has to be retransmitted (go-back-n), thus wasting bandwidth, especially if the bandwidth-delay product is high.

- Selective retransmission as one solution
  - RFC2018 allows for acknowledgements of single packets, not only acknowledgements of in-sequence packet streams without gaps
  - sender can now retransmit only the missing packets

\+ much higher efficiency

– more complex software in a receiver, more buffer needed at the receiver

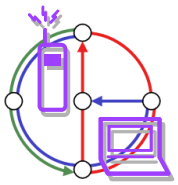


Transaction oriented TCP

- TCP phases
  - connection setup, data transmission, connection release
  - using 3-way-handshake needs 3 packets for setup and release, respectively
  - thus, even short messages need a minimum of 7 packets!
- Transaction oriented TCP
  - RFC1644, T-TCP, describes a TCP version to avoid this overhead
  - connection setup, data transfer and connection release can be combined
  - thus, only 2 or 3 packets are needed

\+ Efficiency

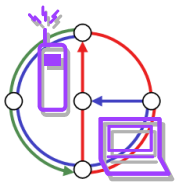
– Requires changed TCP



Comparison of different approaches for a “mobile” TCP

Approach	Mechanism	Advantages	Disadvantages
Indirect TCP	splits TCP connection into two connections	isolation of wireless link, simple	loss of TCP semantics, higher latency at handover
Snooping TCP	“snoops” data and acknowledgements, local retransmission	transparent for end-to-end connection, MAC integration possible	problematic with encryption, bad isolation of wireless link
M-TCP	splits TCP connection, chokes sender via window size	Maintains end-to-end semantics, handles long term and frequent disconnections	Bad isolation of wireless link, processing overhead due to bandwidth management
Fast retransmit/ fast recovery	avoids slow-start after roaming	simple and efficient	mixed layers, not transparent
Transmission/ time-out freezing	freezes TCP state at disconnect, resumes after reconnection	independent of content or encryption, works for longer interrupts	changes in TCP required, MAC dependant
Selective retransmission	retransmit only lost data	very efficient	slightly more complex receiver software, more buffer needed
Transaction oriented TCP	combine connection setup/release and data transmission	Efficient for certain applications	changes in TCP required, not transparent

[Schiller]

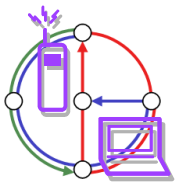


Recent work

- Initial research work
  - Indirect TCP, Snoop TCP, M-TCP, T/TCP, SACK, Transmission/time-out freezing, …
- TCP over 2.5/3G wireless networks
  - Fine tuning today's TCP
  - Learn to live with
    - Data rates: 64 kbit/s up, 115-384 kbit/s down; asymmetry: 3-6, but also up to 1000 (broadcast systems), periodic allocation/release of channels
    - High latency, high jitter, packet loss
  - Suggestions
    - Large (initial) sending windows, large maximum transfer unit, selective acknowledgement, explicit congestion notification, time stamp, no header compression
  - Already in use
    - i-mode running over FOMA
    - WAP 2.0 ("TCP with wireless profile")

$$BW \leq \frac{0.93 \cdot MSS}{RTT \cdot \sqrt{p}}$$

- max. TCP **B**and**W**idth
- **M**ax. **S**egment **S**ize
- **R**ound **T**rip **T**ime
- loss **p**robability



Recent work

- Performance enhancing proxies (PEP, RFC 3135)
  - Transport layer
    - Local retransmissions and acknowledgements
  - Additionally on the application layer
    - Content filtering, compression, picture downscaling
    - E.g., Internet/WAP gateways
    - Web service gateways?
  - Big problem: breaks end-to-end semantics
    - Disables use of IP security
    - Choose between PEP and security!
- More open issues
  - RFC 3150 (slow links)
    - Recommends header compression, no timestamp
  - RFC 3155 (links with errors)
    - States that explicit congestion notification cannot be used
  - In contrast to 2.5G/3G recommendations!