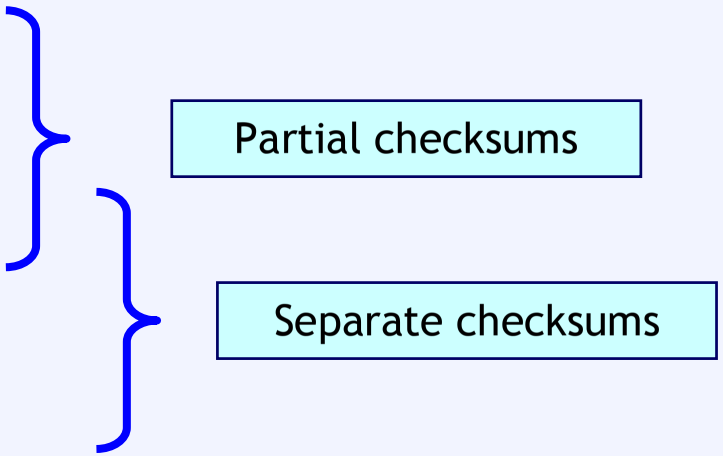
Coping with Link Noise at the Transport Level

Upperside WiMax Summit 2004

Michael Welzl
http://www.welzl.at, michael.welzl@uibk.ac.at
Distributed and Parallel Systems Group
Institute of Computer Science
University of Innsbruck, Austria

Outline

- Optional 802.16 features related to link noise
  - Link layer ARQ
  - MAC sublayer CRC
- UDP Lite
- DCCP
- TCP
- Conclusion



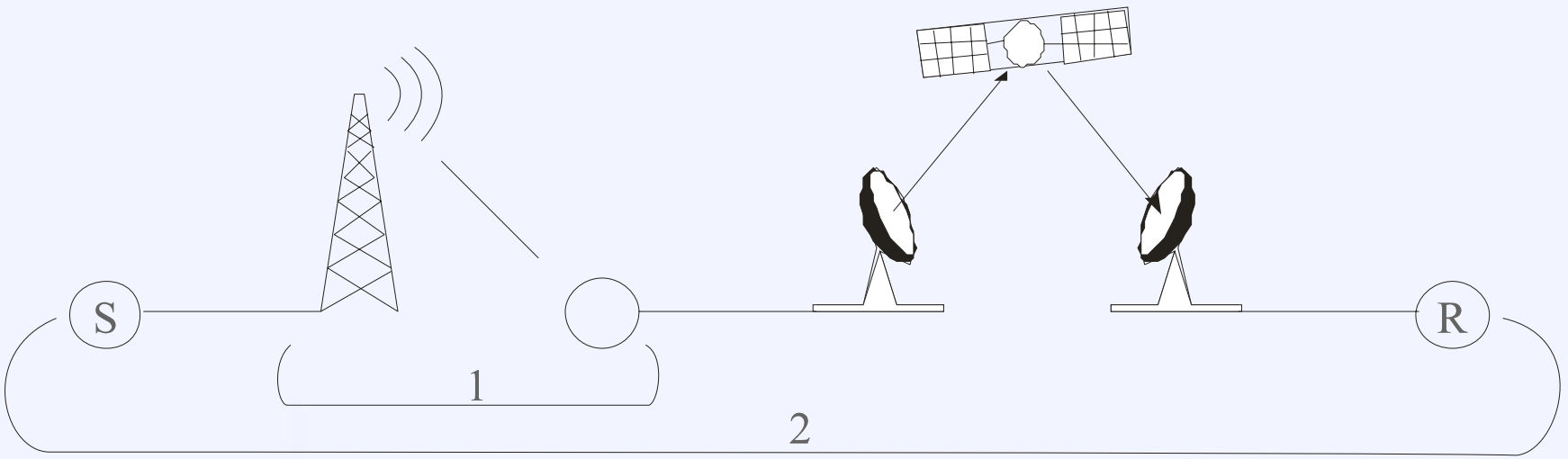
802.16 and link noise

- Large number of options
- We are concerned with two:
  1. Link layer ARQ
  2. MAC sublayer CRC
- Question: why optional?
  (What is the benefit of enabling/disabling these features?)
- Context: TCP/IP over 802.16

Link layer ARQ

- Advantages:
  - potentially faster than end-to-end retransmits
  - operates on frames, not packets
  - could use knowledge that is not available at transport end points

- example scenario: control loop 1 much shorter than 2



Link Layer ARQ /2

- Disadvantages:
  - hides information (known corruption) from end points
  - TCP: increased delay ⇒ more conservative behavior
- Link layer ARQ can have varying degrees of persistence
- So what?
- Ideal choice would depend on individual end-to-end flows
- Thus, recommendation:
  - low persistence or disable (leave severe cases up to end points)
  - Give end points means to react properly (detect corruption)

Further details:
RFC 3366

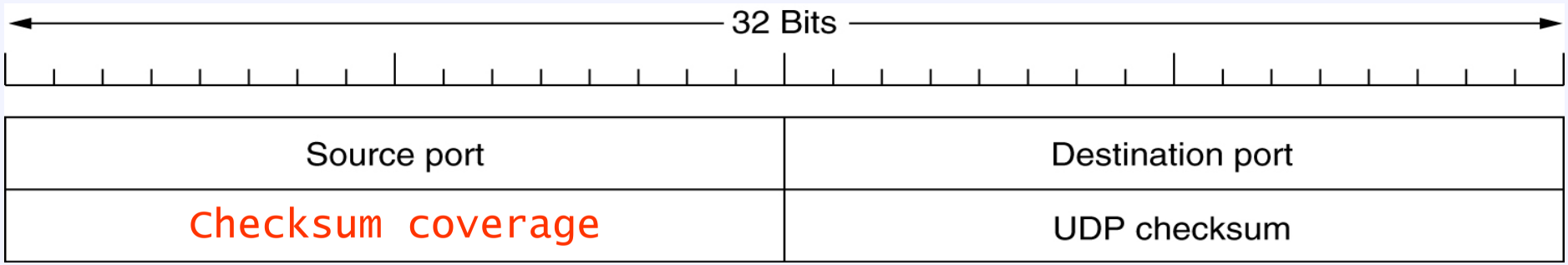
Disable the checksum !

- MAC sublayer CRC optional in 802.16
  - turn it off!
  - just do it :)

- There are ways to use erroneous data in TCP/IP
  - but only if these data are available!
  - handing over erroneous data seems stupid unless they are used by TCP/IP
  - chicken-egg type of problem ... someone has to make a start!
  - the IETF did

- Two basic approaches
  1. partial checksums (give erroneous data to apps) - UDP Lite, DCCP
  2. separate header/payload checksums - DCCP
     (avoid misinterpretation of corruption as congestion)

UDP and UDP Lite



- UDP = IP + 2 features:
  - Ports: identify communicating instances with similar IP address (transport layer)
  - Checksum: Adler-32 covering the whole packet
    - checksum field = 0: no checksum at all - bad idea!
    - ⇒ solution: UDP Lite (length := checksum coverage)
      - advantage: e.g. video codecs can cope with bit errors, but standard UDP throws a whole packet away!
      - acceptable BER up to applications (complies with end-to-end arguments)
      - critical: app's depending on UDP Lite can depend on lower layers
- Usage of UDP: unreliable data transmission
  - DNS, SNMP, real-time streams, ..

Approved by IESG for publication as Proposed Standard RFC

Datagram Congestion Control Protocol (DCCP)

- Unreliable flows (etc.) should perform congestion control
  - up to now, embedded in app (on top of UDP)
    - difficult to realize
    - often not done (properly)
- Many research efforts on CC. for streaming (smoother,..) – e.g. TFRC
- Well-defined framework for (TCP-friendly) congestion control
  - Peers negotiate appropriate mechanism
  - Core OS implementation of congestion control
- Many additional features
  - Partial checksums as in UDP Lite
  - Nonces, ..

Developed in IETF DCCP Working Group;
TFRC: RFC 3448 (Proposed Standard)

Separate header / payload checksums

- Available as feature ("Data Checksum option") in DCCP !
  - Currently discussed in IETF for TCP
  - Note: partial checksums useless in TCP (reliable transmission of erroneous data?)
- Differentiate corruption / congestion
  - Checksum covers all
    - Error could be in header
    - Impossible to notify sender (seqno, ports, ..)
  - Checksum fails in header only
    - Bad luck
  - Checksum fails in payload only, ECN = 0
    - Inform sender of corruption
    - No need to react as if congestion
    - Still react (keeping high rate + high BER = bad idea) ⇒ experimental!
  - Checksum fails in payload only, ECN = 1
    - Clear sign of congestion

Why is that so important?

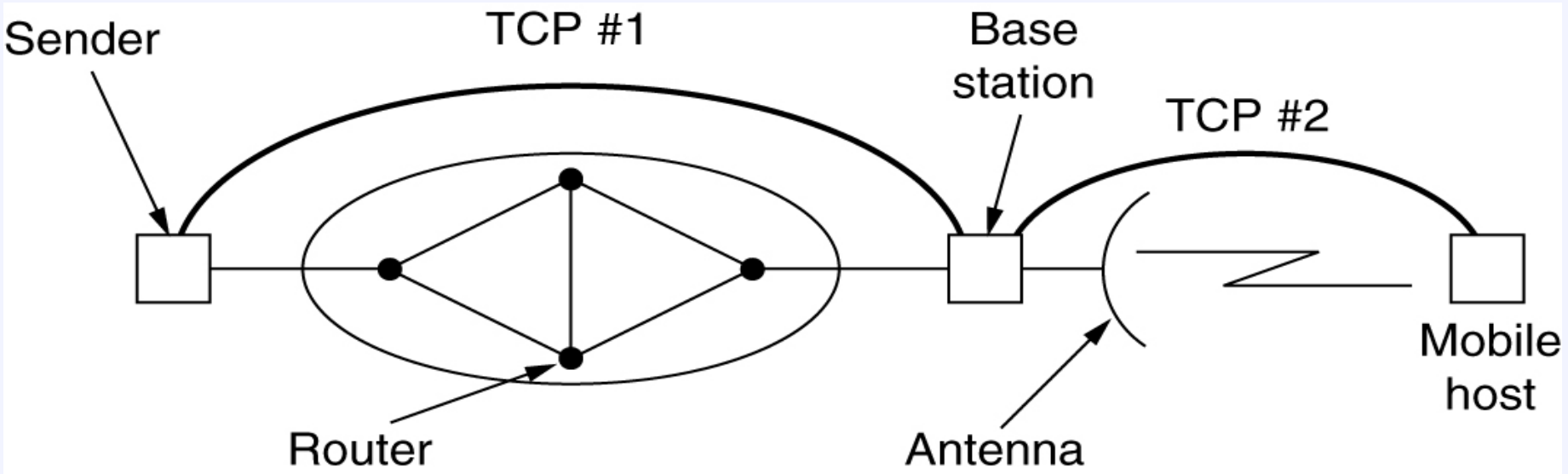
- TCP congestion control:
  - introduced in the 80s, necessary to prevent congestion collapse
- TCP interpretation of network feedback:
  - single packet loss = sign of congestion
    or
    Explicit Congestion Notification (ECN) flag = 1 = sign of congestion
    - sender halves rate (Congestion Avoidance) – exponential backoff!
  - many packets lost in a row = sign of heavy congestion (timeout)
    - sender goes back to Slow Start
  - round-trip time estimate increases / decreases
    - longer RTT = more conservative (e.g., increase by ≈ 1 segment / RTT)

MAC sublayer CRC

Link layer ARQ

Wireless links: misinterpretation of ***packet loss*** and ***RTT estimate*** ⇒ massive throughput degradation!

TCP over wireless: many hacks available



- Various possible enhancements:
  - split connection at base station
    - act like sender towards receiver + act like receiver towards sender
  - monitor connection at base station, buffer + interfere ("Snoop TCP")

Another possibility: change packet size!

- Underlying idea:
  - Frame size: 1000 bits, BER 1/1000 ⇒ each frame erroneous
  - Frame size: 100 bits, BER 1/1000 ⇒ 1 out of 10 frames erroneous
    - Tuned using link layer fragmentation (another 802.16 feature)
  - 1000-bit packet consisting of 10 100-bit frames vs. 100-bit packet consisting 1 100-bit frame: same effect!
    - Tuned at IP layer!

Project idea!

- Simple test:
  - cheap off-the-shelf 802.11b equipment, long distance at our institute
  - 1800 pings (15 minutes, 1 ping every 0.5 seconds), packet size = 100 byte

1000 byte packets	100 byte packets
rtt min/avg/max/mdev = 11.779/**28.271**/315.876/35.121 ms	rtt min/avg/max/mdev = 3.274/**12.410**/1022.437/46.585 ms

NOTE:

802.11b CRC cannot be disabled;

delay = ARQ !

The End ...

- Conclusion: disable 'em ! (unless you know it's a special case)

- **References:**

  - Michael Welzl, "Passing Corrupt Data Across Network Layers: An Overview of Recent Developments and Issues", accepted for publication in EURASIP Journal on Applied Signal Processing, special issue on "Cross Layer Design for Communications and Signal Processing Systems", Hindawi Publishing Corporation, 4th Quarter 2004.
  - Rajesh Krishnan, James P.G. Sterbenz, Wesley M. Eddy, Craig Partridge, Mark Allman, „Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks", accepted for publication in Elsevier Computer Networks, preprint available from http://www.ir.bbn.com/
  - G. Fairhurst, L. Wood, "Advice to link designers on link Automatic Repeat reQuest (ARQ)", RFC 3366
  - Eddie Kohler, Mark Handley, Sally Floyd, "Datagram Congestion Control Protocol (DCCP)", Internet-draft draft-ietf-dccp-spec-06.txt (work in progress), Feb 2004
  - L-A. Larzon, M. Degermark, S. Pink, L-E. Jonsson (Editor), G. Fairhurst (Editor), "The UDP-Lite Protocol", Internet-draft draft-ietf-tsvwg-udp-lite-02.txt (work in progress), Aug 2003