Lecture 5.

Internet Transport Layer:

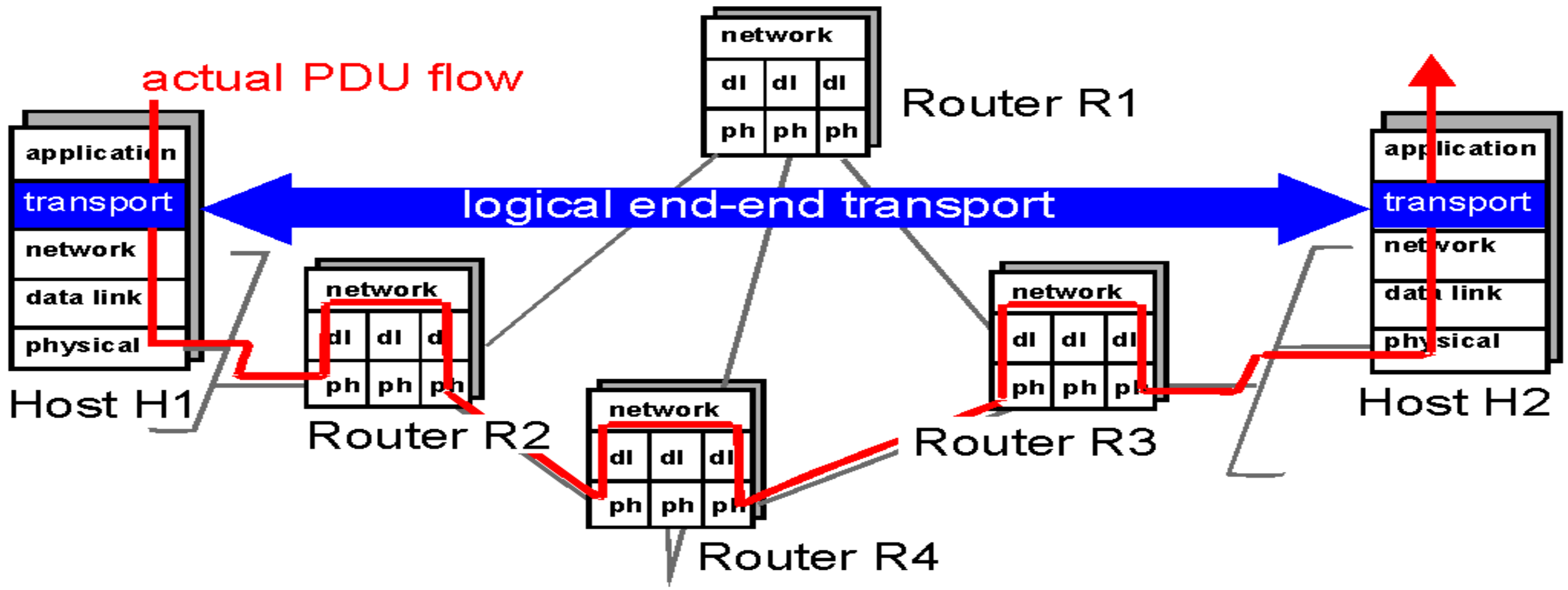
User Datagram Protocol (UDP)

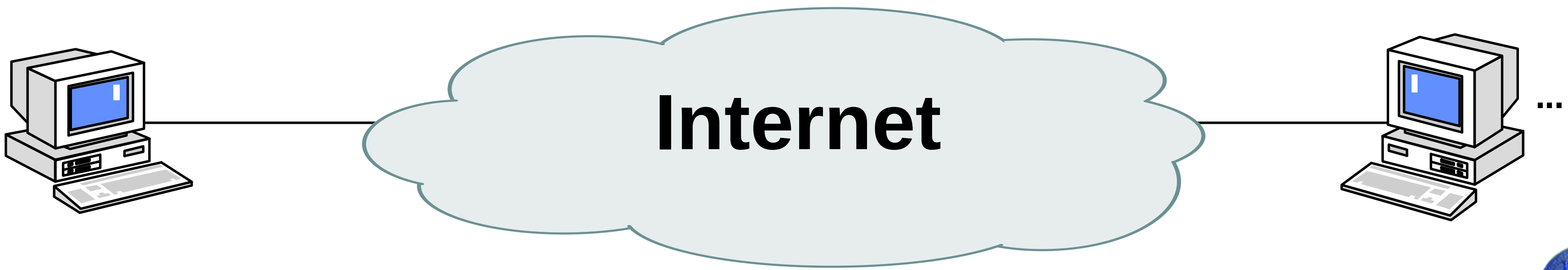
G.Bianchi, G.Neglia, V.Mancuso

Università degli Studi di Palermo

Transport Layer Protocols

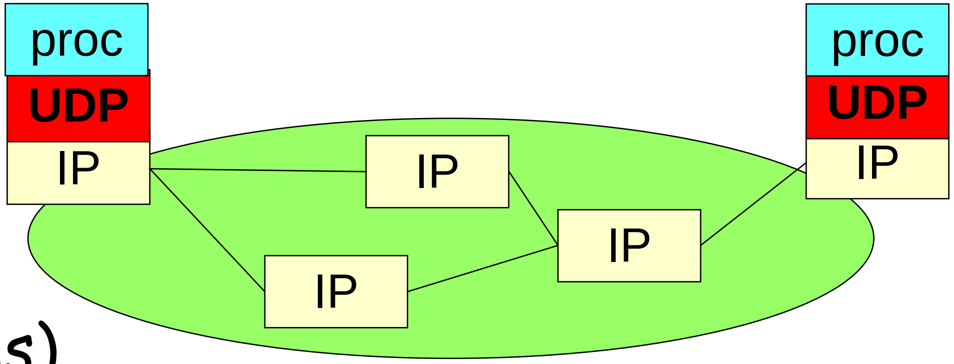
Entire network seen as a pipe



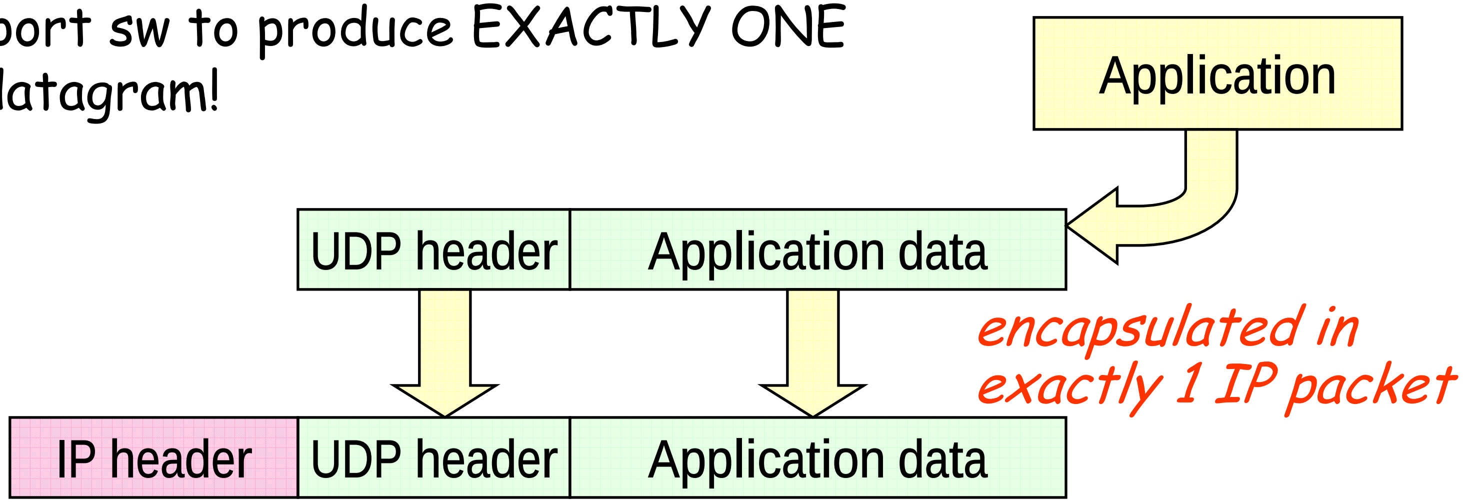


G.Bianchi, G.Neglia, V.Mancuso

UDP Packets



- ➔ Connection-Less
  - ⇨ (no handshaking)
- ➔ UDP packets (Datagrams)
  - ⇨ Each application interacts with UDP transport sw to produce EXACTLY ONE UDP datagram!



This is why, improperly, we use the term UDP packets

UDP datagram format

8 bytes header + variable payload

0 — 7 — 15	15 — 23 — 31
source port	destination port
length (bytes)	Checksum
Data	

- ➔ UDP length field
  - ⇨ all UDP datagram
  - ⇨ (header + payload)
- ➔ payload sizes allowed:
  - ⇨ Empty
  - ⇨ Odd size (bytes)

➔ UDP functions limited to:

- ⇨ addressing
  - → which is the only strictly necessary role of a transport protocol
- ⇨ Error checking
  - → which may even be **disabled** for performance

Maximum UDP datagram size

- ➔ **16 bit UDP length field:**
  - ⇨ Maximum up to $2^{16}-1$ = 65535 bytes
  - ⇨ Includes 8 bytes UDP header (max data = 65527)
- ➔ **But max IP packet size is also 65535**
  - ⇨ Minus 20 bytes IP header, minus 8 bytes UDP header
  - ⇨ Max UDP_data = **65507** bytes!
- ➔ **Moreover, most OS impose further limitations!**
  - ⇨ most systems provide 8192 bytes maximum (max size in NFS)
  - ⇨ some OS had (still have?) internal implementation features (bugs?) that limit IP packet size
    - →SunOS 4.1.3 had 32767 for max tolerable IP packet transmittable (but 32786 in reception…) – bug fixed only in Solaris 2.2
- ➔ **Finally, subnet Maximum Transfer Unit (MTU) limits may fragment datagram – annoying for reliability!**
  - ⇨ E.g. ethernet = 1500 bytes; PPP on your modem = 576

UDP: a lightweight protocol

- ➔ **No connection establishment**
  - ⇨ no initial overhead due to handshaking
- ➔ **No connection state**
  - ⇨ greater number of supported connections by a server!
- ➔ **Small packet header overhead**
  - ⇨ 8 bytes only vs 20 in TCP

- ➔ **originally intended for simple applications, oriented to short information exchange**
  - ⇨ DNS
  - ⇨ management (e.g. SNMP)
  - ⇨ Distributed file system support (e.g. NFS)
  - ⇨ etc

Unregulated send rate in UDP

- ⇨ No rate limitations
  - → No throttling due to congestion & flow control mechanisms
  - → No retransmission
- ⇨ Less overhead
- ⇨ In contrast to TCP, UDP may provide multicast support

→ **extremely important features for today multimedia applications!**

→ **specially for real time applications which can tolerate some packet loss but require a minimum send rate.**

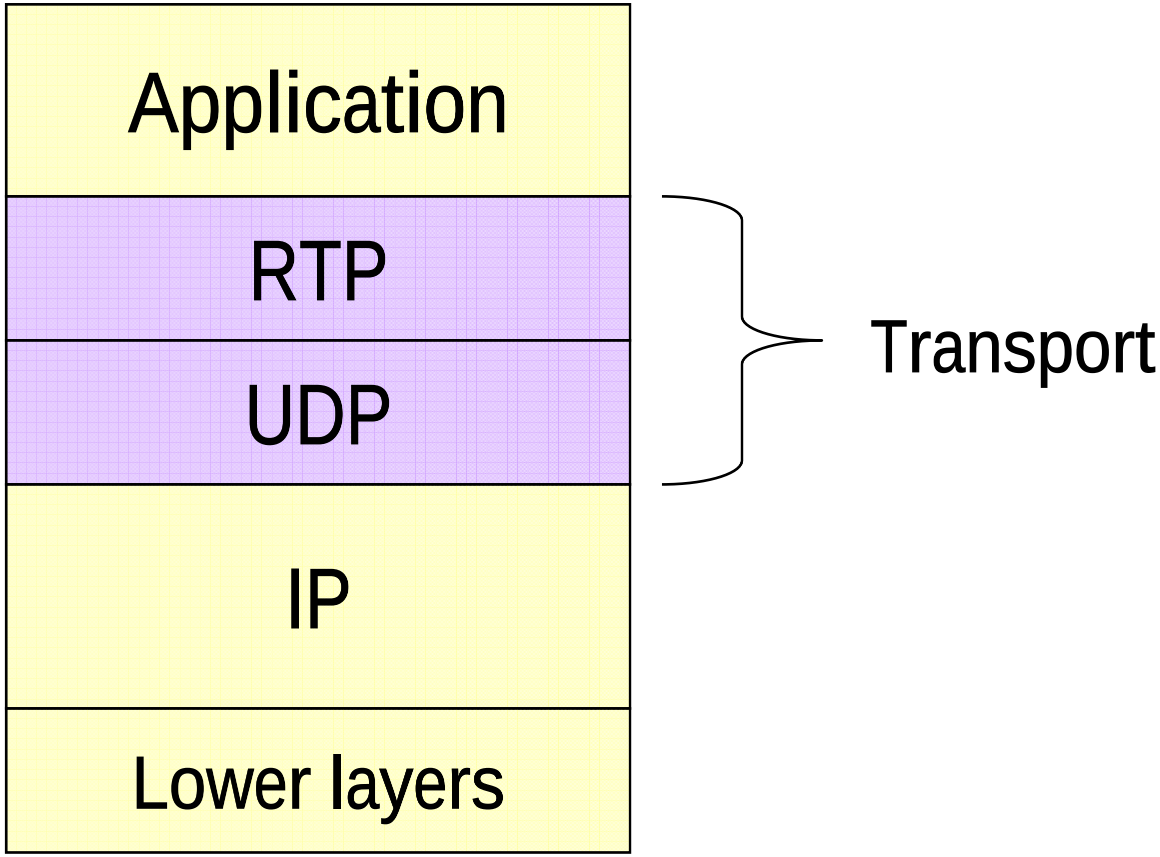
Be careful: UDP ok for multimedia because it does not provide anything at all (no features = no limits!). Application developers have to provide supplementary transport capabilities at the application layer!

Audio/Video Support

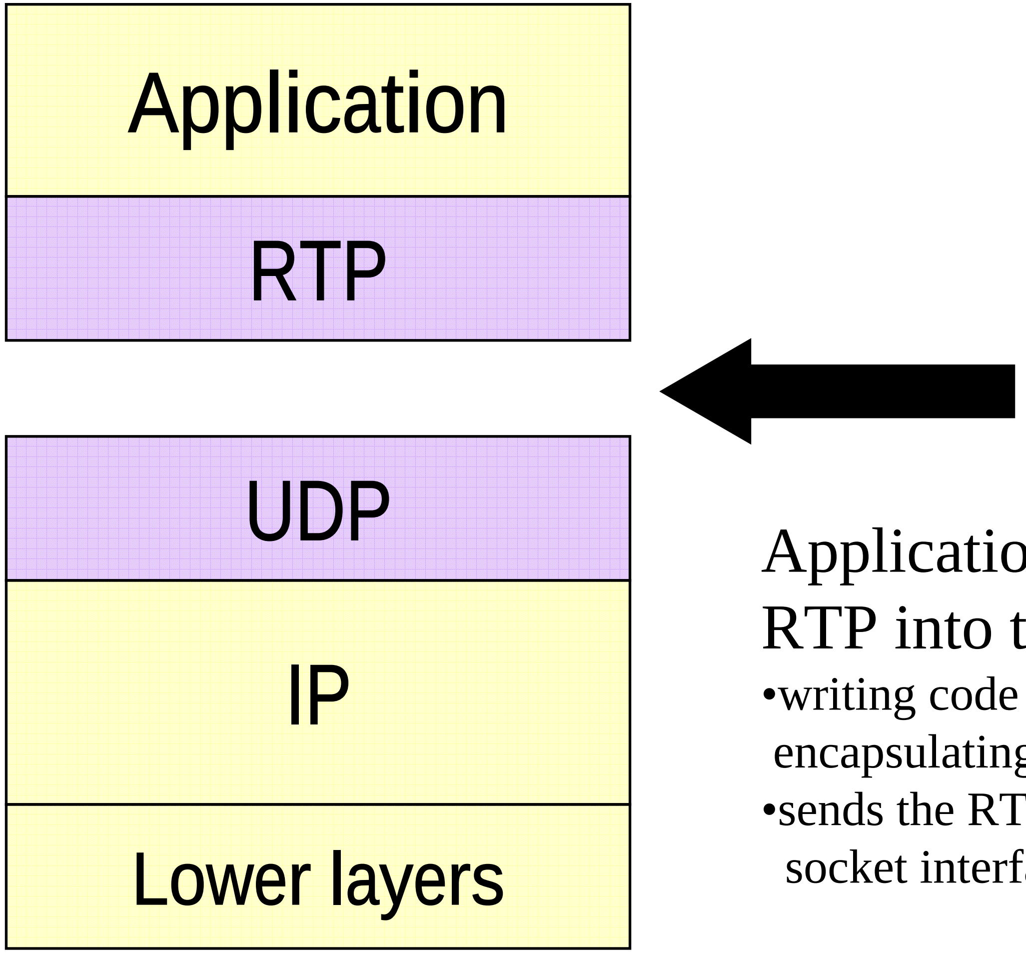
- ➔ **UDP is transport layer candidate**
- ➔ **UDP is too elementary!**
  - ⇨ No sequence numbers
  - ⇨ No timestamp for resynchronization at receiver
  - ⇨ No multicasting
- ➔ **Old solution: let application developer build their own header**
- ➔ **New solution: use an enhanced transport protocol**

Real Time Protocol (RTP, RFC 3550)

RTP: sublayer of Transport



RTP as seen from Application



SOCKET INTERFACE

Application developer integrates RTP into the application by:

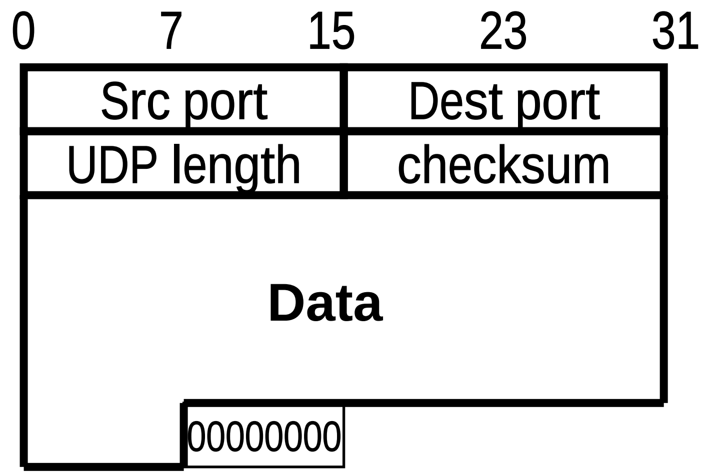
- writing code which creates the RTP encapsulating packets;
- sends the RTP packets into a UDP socket interface.

Details of RTP in subsequent courses – or see it in RFC 1889

Università degli Studi di Palermo

Error checksum

- ➔ **16 bit checksum field, obtained by:**
  - ⇨ summing up all 16 bit words in header data and pseudoheader, in 1's complement (checksum fields filled with 0s initially)
  - ⇨ take 1's complement of result
  - ⇨ if result is 0, set it to 111111…11 (65535==0 in 1's complement)
- ➔ **at destination:**
  - ⇨ 1's complement sum should return 0, otherwise error detected
  - ⇨ upon error, no action (just packet discard)
- ➔ **efficient implementation RFC 1071**



- ➔ **Zero padding**
  - ⇨ when data size is odd
- ➔ **checksum disabled**
  - ⇨ by source, by setting 0 in the checksum field

Università degli Studi di Palermo

disabling checksum

- ➔ **In principle never!**
  - ⇨ Remember that IP packet checksum DOES NOT include packet payload.
- ➔ **In practice, often done in NFS**
  - ⇨ sun was the first, to speed up implementation
- ➔ **may be tolerable in LANs under one's control.**
- ➔ **Definitely dangerous in the wide internet**
  - ⇨ Exist layer 2 protocols without error checking
  - ⇨ some routers happen to have bugs that modify bits

Pseudo header

➔ Is not transmitted!

- ⇨ it is information available at transmitter and at receiver
- ⇨ intention: double check that packet has arrived at correct destination and transport protocol
- ⇨ it violates protocol hierarcy!

<table>
<tr><th>0</th><th>7</th><th>15</th><th>23 31</th><th></th></tr>
<tr><td colspan="4">Source IP address</td><td rowspan="3">12 bytes pseudoheader</td></tr>
<tr><td colspan="4">Destination IP address</td></tr>
<tr><td>00000000</td><td>protocol</td><td colspan="2">UDP length</td></tr>
<tr><td colspan="2">Source port</td><td colspan="2">Destination port</td><td rowspan="2">8 bytes UDP header</td></tr>
<tr><td colspan="2">UDP length</td><td colspan="2">checksum</td></tr>
<tr><td colspan="4">data</td><td></td></tr>
</table>

Same checksum calculation used in TCP. UDP length duplicated.