Class Notes

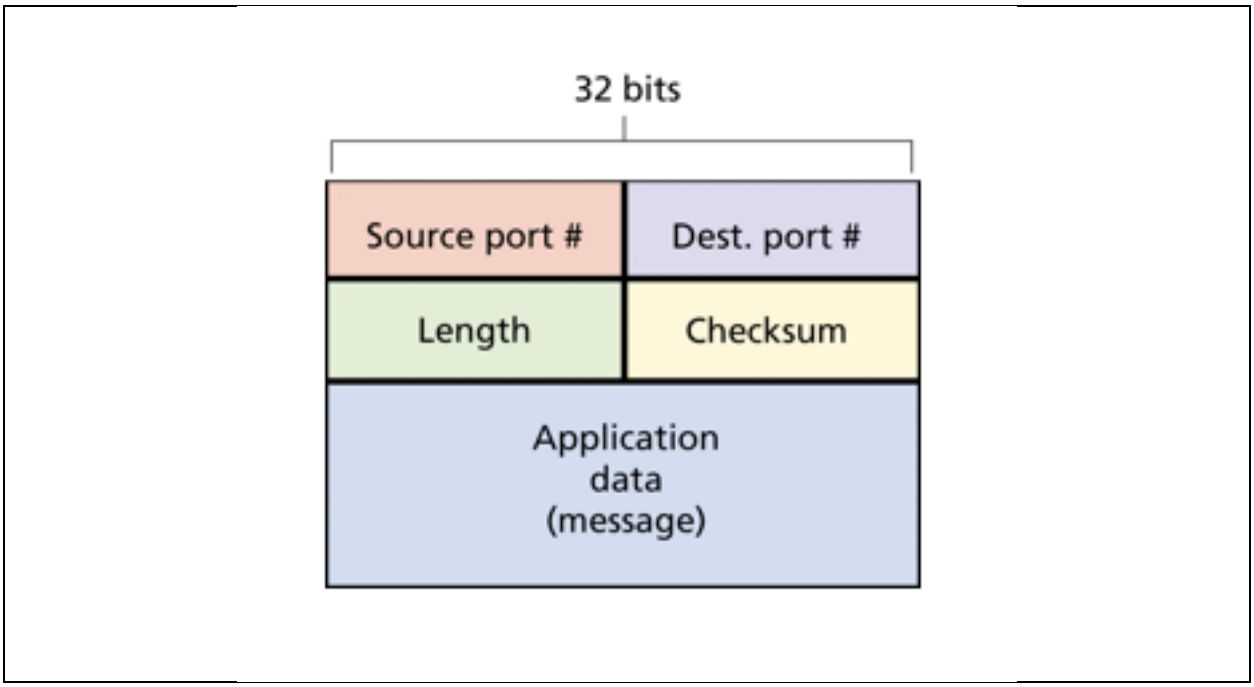
Preamble

(1) One-third of the semester is over!
(2) The syllabus of the second sessional exam will be **chapter 3** and **chapter 4** from our textbook.
(3) All exams are comprehensive, meaning everything we studied in the class starting lecture one will be included in the syllabus.
(4) This middle part of the course is relatively dense and fast-paced. It is **mandatory** to read the relevant sections from the textbook along with the lectures.

A quick revision

Let's start with a quick revision to build our context.

Connectionless Transport UDP



(1) Structure of UDP segment
    a. By convention RFC's show 32 bits in a row. We will follow that convention.
    b. Be careful which bit is the first bit and hence will be put first on the wire.
    c. Source and destination ports are of 16 bit each (possible values from 0 to 65535 i.e. 2^{16} - 1)

i. Port no 0 to 1023 are “well-known” ports and your application will not be able to use them on OS like Linux until your application has appropriate privileges

(2) Length is of 16 bits

a. Theoretical max size of a UDP packet (UDP header + payload) = 65535
b. Units of Length = Bytes
c. But IPv4 wich will carry a UDP packet as payload can not be larger than 65535. Smallest IPv4 hear is of 20 Bytes. So 65535 – 20 = 65515 Bytes
d. But most physical layer technologies has a Maximum Transmission Unit (MTU) that dictates how much data they can deliver in one packet.
e. If a UDP packet exceeds the MTU of a network segment, it will be fragmented into smaller pieces by the IP layer. Fragmentation increases the likelihood of packet loss, as the loss of a single fragment results in the loss of the entire original datagram. For this reason, many applications that use UDP, such as DNS, often limit their packet sizes to a much smaller and safer value, like 512 bytes, to avoid fragmentation.
f. Path MTU: Find the minimum MTU end-to-end and use that size to avoid data packet fragmentation at the IP layer.
g. Typical UDP size = 1500 – 20 = 1480 Bytes

Common MTU values — compact table

Technology / Link layer	Typical MTU (bytes)	Notes
Ethernet (IEEE 802.3)	1500	De-facto Internet LAN MTU (IP payload).
Ethernet — Jumbo frames	~9000 (often 9000–9216)	Vendor-dependent; must be supported end-to-end to be useful.
Wi-Fi (IEEE 802.11)	1500 (typical IP MTU)	802.11 MAC can carry larger MSDUs (up to 2304) but stacks usually use 1500 for compatibility.
PPPoE (DSL)	1492	Ethernet MTU (1500) minus 8 bytes PPPoE overhead — common WAN setting.
LTE / Cellular	~1500 (varies: often 1350–1500)	Carriers and tunneling can reduce MTU; path-dependent.
Bluetooth Classic (L2CAP)	~672 (stack-dependent)	Historic default; can be negotiated larger depending on implementation.
Bluetooth LE (ATT)	23 (default), commonly negotiated up to 247	ATT default MTU 23; modern stacks and LE Data Length Extension allow larger L2CAP payloads.
FDDI (historical)	4352	Older LAN tech — shown for reference.
ATM (AAL5)	~9180 (AAL5 maximum payload)	ATM cell network commonly used with AAL5 for IP; practical limits depend on encapsulation.
Loopback (software, e.g. lo on Linux)	65,536 (common default)	Host-local interface; very large since no physical link.

h. IPv4 header size is variable due to some reasons we will see later. Minimum IPv4 header size is 20 Bytes. IPv6 header is fixed at 40 Bytes.

i. For IPv4: 65,535 bytes (IP total length)−20 bytes (IP header)−8 bytes (UDP header)= 65,507 bytes of data

j. For IPv6: 65,535 bytes (IPv6 payload limit)−8 bytes (UDP header)=65,527 bytes of data

Key Differences Summarized

Feature	IPv4	IPv6
Field Name	Total Length	Payload Length
What's Included	**Header + Payload**	**Payload Only**
How to get total size	The value *is* the total size	40 bytes + field value

i.

(3) Checksum

(4) UDP checksum is over (pseudo header) AND (UDP header) AND (UDP Payload)

a. Optional in Ipv4, mandatory in Ipv6
b. Use 0X0000 when need to include checksum field in the checksum algo
c. In Ipv4, 0X0000 => UDP checkum not in use
d. To disambiguate, if checksum values comes out to 0X0000, change it to 0X1111
e. If checksum is in use, and on receifer side if checksum not verified, packet is usually thrown away.

(5) Pseudo headers. Why we need them?

IPv4 pseudo-header (concatenate these bytes, in order):

1. Source address (32 bits)
2. Destination address (32 bits)
3. Zero (8 bits) — one zero octet, value `0x00`
4. Protocol (8 bits) — e.g. `0x11` for UDP
5. UDP/TCP length (16 bits) — network byte order

So the bytes layout is:

`[src(4)] [dst(4)] [0x00] [protocol] [length(2)]`

IPv6 pseudo-header (concatenate these bytes, in order):

1. Source address (128 bits)
2. Destination address (128 bits)
3. Upper-layer packet length (32 bits) — length of header+payload
4. Zero (24 bits) — three zero octets `0x00 0x00 0x00`
5. Next Header (8 bits) — e.g. `0x11` for UDP

So the bytes layout is:

`[src(16)] [dst(16)] [length(4)] [0x00 0x00 0x00] [next-header]`

(6) If link-layer provides data integrity why do it again at UDP level?
    a. End-to-end principle
(7) Some discussion on error detection and error correction codes
    a. Hamming code
    b. Reed-Solomon codes
    c. … many more
(8) UDP checksum example from slides (Slide no 34)
(9) Another note: Note the DeMux key at the IP layer!