Internetworking

Internetworking

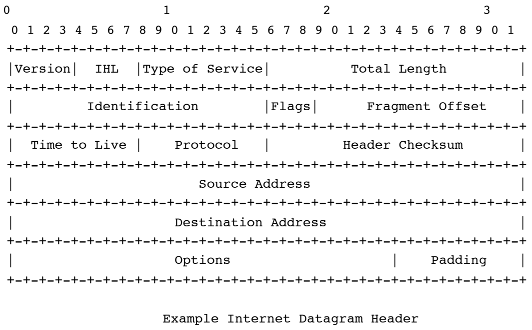
—Relationship to other protocols

```
+------+ +-----+ +-----+     +-----+
|Telnet| | FTP | | TFTP| ... | ... |
+------+ +-----+ +-----+     +-----+
      |   |         |           |
     +-----+     +-----+     +-----+
     | TCP |     | UDP | ... | ... |
     +-----+     +-----+     +-----+
        |           |           |
     +--------------------------+----+
     |    Internet Protocol & ICMP   |
     +--------------------------+----+
                    |
       +---------------------------+
       |   Local Network Protocol  |
       +---------------------------+
```

FTP = file transfer protocol
TFTP = trivial FTP
TCP = transmission control protocol
UDP = user datagram protocol
ICMP = internet control message protocol

Internetworking

- IP Header Format, the IP NPDU (see the RFC)



```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Version|  IHL  |Type of Service|          Total Length         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Identification        |Flags|      Fragment Offset    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Time to Live |    Protocol   |         Header Checksum       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Source Address                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Destination Address                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Options                    |    Padding    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                    Example Internet Datagram Header
```

IHL = header length

Internetworking

— Header Fields

- we have seen part of this already, but there are certain things I would like to address again
- Version:
  - version of IP, i.e. how fields are to be interpreted.
  - current version is IP version 4, or IPv4.
- IHL:
  - header length in multiples of 32 bit words. Minimum length (without options) is 5.
- Type of Service:
  - provides an indication of the abstract parameters of the required QOS.
  - parameters are used during route selection.
  - networks may offer service precedence; at time of high traffic, only high precedence traffic is accepted.
  - example: if high reliability service is preferred over best-try transfer, the gateway should select a connection-oriented over a connectionless network.

Internetworking

QOS Fields

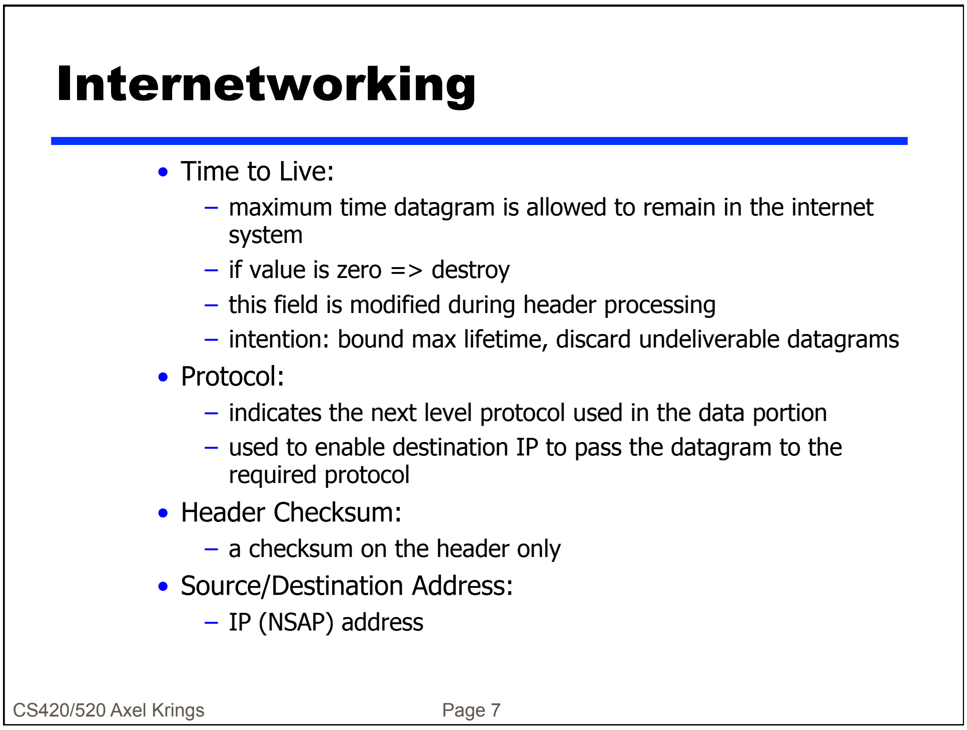
```
Bits 0-2:  Precedence.
Bit    3:  0 = Normal Delay,       1 = Low Delay.
Bits   4:  0 = Normal Throughput,  1 = High Throughput.
Bits   5:  0 = Normal Reliability, 1 = High Reliability.
Bit  6-7:  Reserved for Future Use.

   0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+-----+
|                 |     |     |     |     |     |
|   PRECEDENCE    |  D  |  T  |  R  |  0  |  0  |
|                 |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+

  Precedence:
    111 - Network Control      110 - Internetwork Control
    101 - CRITIC/ECP                  100 - Flash Override
    011 - Flash           010 - Immediate
    001 - Priority                  000 - Routine
```

Internetworking

- Total Length:
  - 16 bit => max length of 65536 bytes
- Identification:
  - an identifying value assigned by the sender to aid in assembling the fragments of a datagram, e.g. a message is transferred in multiple datagrams -- relate different datagrams to the same user message.
- Flags:

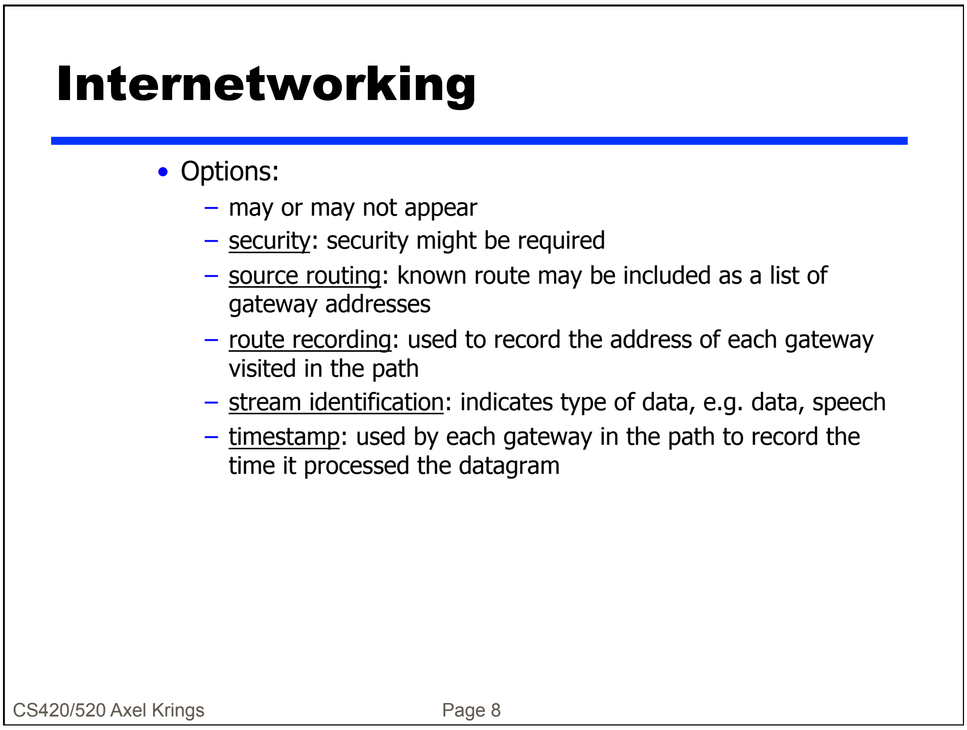
```
Bit 0: reserved, must be zero
Bit 1: (DF) 0 = May Fragment,  1 = Don't Fragment.
Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments.
             0   1   2
           +---+---+---+
           | 0 | D | MF|
+---+---+---+
```

- Fragment Offset:
  - indicates where in the datagram this fragment belongs
  - fragment offset measured in units of 8 bytes (64 bits)



Internetworking

- Time to Live:
  - maximum time datagram is allowed to remain in the internet system
  - if value is zero => destroy
  - this field is modified during header processing
  - intention: bound max lifetime, discard undeliverable datagrams
- Protocol:
  - indicates the next level protocol used in the data portion
  - used to enable destination IP to pass the datagram to the required protocol
- Header Checksum:
  - a checksum on the header only
- Source/Destination Address:
  - IP (NSAP) address



Internetworking

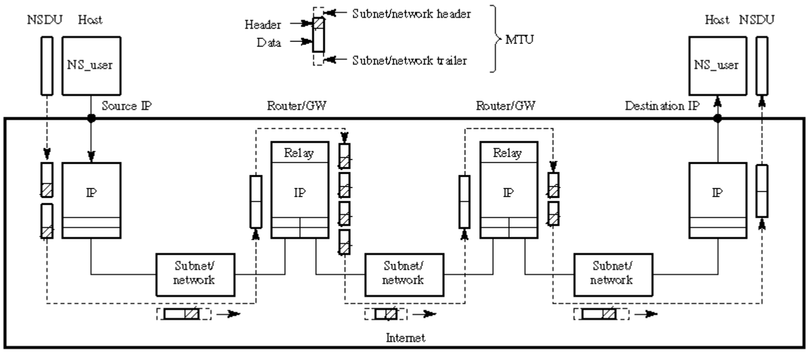
- Options:
  - may or may not appear
  - security: security might be required
  - source routing: known route may be included as a list of gateway addresses
  - route recording: used to record the address of each gateway visited in the path
  - stream identification: indicates type of data, e.g. data, speech
  - timestamp: used by each gateway in the path to record the time it processed the datagram

Internetworking

- IP protocol Functions:
  - fragmentation and reassembly
  - routing
  - error reporting
- —Fragmentation and Reassembly
  - The IP protocol permits a user to submit a TPDU (NSDU) up to 64Kbytes in size. However, member networks may not be able to handle such large packets. IP provides a mechanism for dividing the packets into smaller fragments *when needed* and reassembling them before delivery. There are two possible techniques:
    - – Intranet Fragmentation
    - – Internet Fragmentation

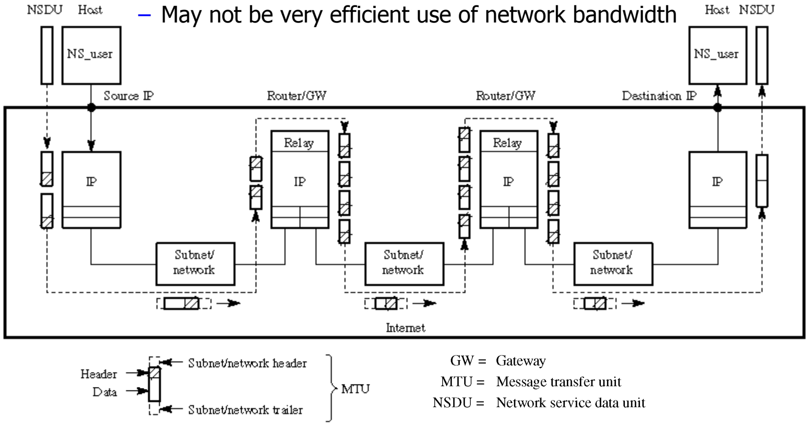
Internetworking

- **Intranet fragmentation** - This technique requires that a companion network that forces a fragmentation does so only locally and reassembles the packet before passing it on to the next member network.
  - – Always utilizes the maximum size packet possible
  - – More overhead, processing delay and buffering required.



Internetworking

- **Internet Fragmentation** - This technique keeps packets fragmented until they arrive at final destination (IP uses this with a **time-to-live** option)
  - No extra overhead or buffering in intermediate nodes
  - May not be very efficient use of network bandwidth

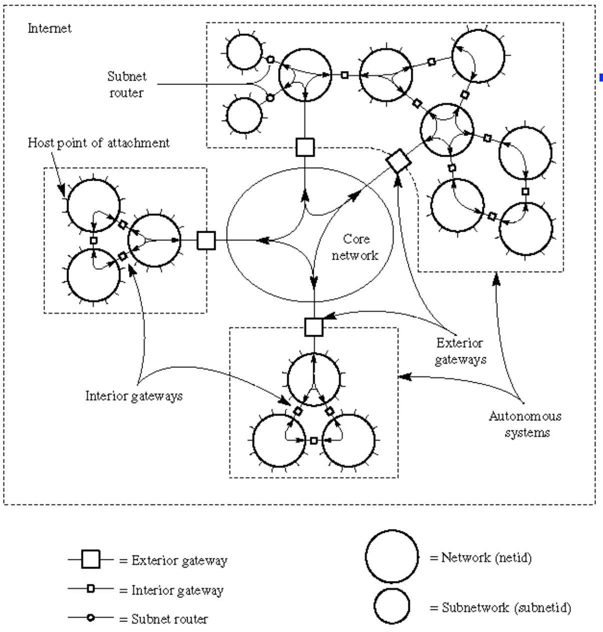


Internetworking

- Routing:
  - To move datagrams across multiple networks, from the source to the destination, we need a **routing** mechanism.
    - Centralized routing - based on the concept of a central repository that maintains routing information for all machine on the net.
    - Distributed routing - based on the concept that the routing information is strategically distributed across the network. The Internet uses this scheme.
  - The routing is performed by **gateways** (or **routers**)
    - Interior Gateways - configured to route packets within a local (or corporate-wide) network -- a single autonomous network.
    - Exterior Gateways - configured to route packets between autonomous networks to other exterior gateways and transfer information between attached autonomous network and core network

General internet architecture

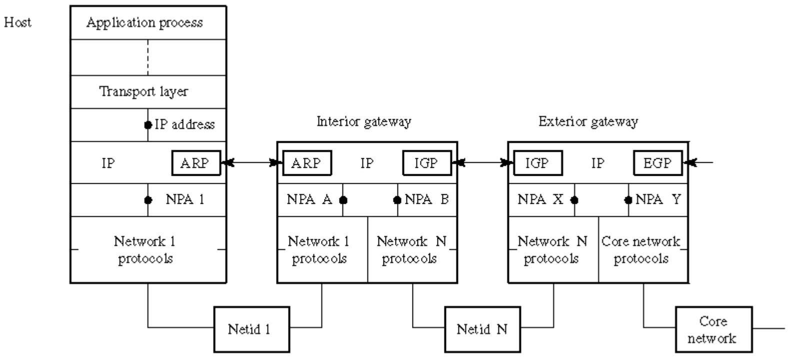
Hal96
fig. 9.12



Internetworking

Routing protocols

Hal96
fig. 9.13



ARP = Address resolution protocol

IGP/EGP = Interior/exterior gateway protocol

NAP = Network point of attachment

Internetworking

- ARP (Address Resolution Protocol)
  - —Basic routing requires that we send a message to the interior gateway attached to our subnetwork and ask it to forward the message to the destination.
    - This is horribly inefficient for messages to hosts on the same subnetwork. We want local IP agent to know that destination is on same subnetwork and transmit packet to that NPA directly.
    - ARP is a protocol that allows us to do that
      - Broadcast an ARP request packet with own IP/NPA address, and requested target IP address.
      - Receive a response from target with correct IP/NPA address.
      - This information is cached.

Internetworking

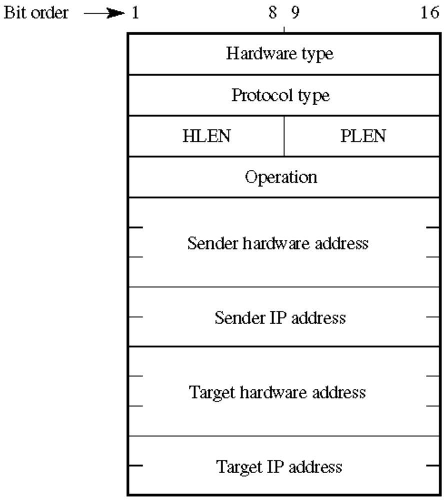
—Example run of arp (with -a “archive” option)

```
/etc/arp -a
brownlee.cs.uidaho.edu (129.101.55.175) at 8:0:9:5c:c5:fe ether
mallard.cs.uidaho.edu (129.101.55.176) at 8:0:9:41:ee:53 ether
dworshak.cs.uidaho.edu (129.101.55.177) at 8:0:9:5f:9:88 ether
laser25.cs.uidaho.edu (129.101.55.114) at 8:0:9:2d:6d:99 ether
teapot.cs.uidaho.edu (129.101.55.83) at 8:0:9:70:e:b2 ether
snake.cs.uidaho.edu (129.101.55.119) at 8:0:9:8c:dc:3b ether
payette.cs.uidaho.edu (129.101.55.89) at 8:0:9:5c:8d:4 ether
salmon.cs.uidaho.edu (129.101.55.90) at 8:0:9:7b:e1:da ether
rosebud.cs.uidaho.edu (129.101.55.91) at 8:0:9:3d:81:18 ether
```

Internetworking

- ARP and RARP message formats

Hal96
fig. 9.14



Bit order → 1 8 9 16

Hardware type

Protocol type

HLEN | PLEN

Operation

Sender hardware address

Sender IP address

Target hardware address

Target IP address

HLEN = Hardware address length
PLEN = IP address length
Operation = 1 ARP request
= 2 ARP response
= 3 RARP request
= 4 RARP response

Internetworking

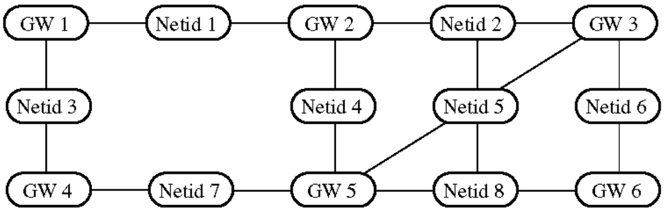
- Interior Gateway Protocols (IGP)
  - —Most commonly used is the IP routing information protocol (RIP)
    - The IP RIP is a distributed routing protocol based on a *distance vector algorithm* (DVA)
    - Distance is usually measured in terms of **hops** (or some other metric such as delay).
    - An interior gateway uses **hop counts** to determine distance between itself an all other subnetworks on the same autonomous system; information is stored in remote routing table.
    - Initially (at configuration time) the only information stored in the gateway is that of:
      - attached networks
      - adjacent gateways

Internetworking

- Remote routing table:
  - if metric is hops, the table contains
    - netid of each of its local networks
    - distance of zero
    - its own IP (as the gateway from which the distance applies)
  - if metric is delay, the delay is determined by the gateway.
    - gateway sends datagram to all gateways attached to its own network and uses half the round-trip delay.
- Periodically IGs send contents of their (remote) routing table to adjacent neighbors to dynamically update tables:
  - not all tables will have same information
  - large overhead for large networks

Internetworking

—Example using the following topology:

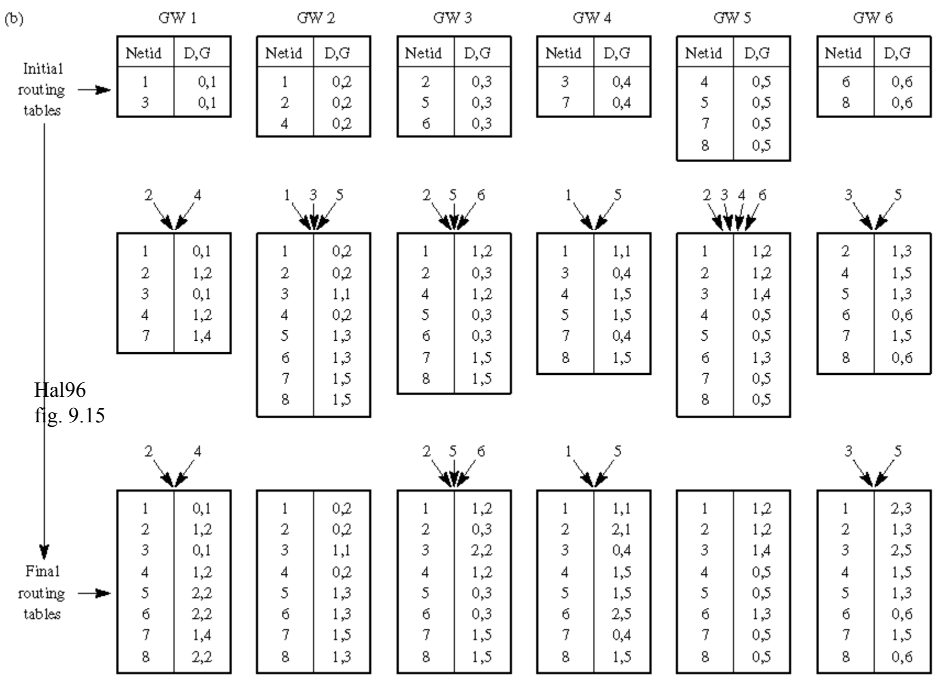


—Routing table entries

- netid
- distance to other gateway GW

Distance vectors

Destination netid	Distance, GW D, G
⋮	⋮



Initial routing tables

GW 1 Netid	GW 1 D,G	GW 2 Netid	GW 2 D,G	GW 3 Netid	GW 3 D,G	GW 4 Netid	GW 4 D,G	GW 5 Netid	GW 5 D,G	GW 6 Netid	GW 6 D,G
1	0,1	1	0,2	2	0,3	3	0,4	4	0,5	6	0,6
3	0,1	2	0,2	5	0,3	7	0,4	5	0,5	8	0,6
		4	0,2	6	0,3			7	0,5		
								8	0,5		

GW 1 Netid	GW 1 D,G	GW 2 Netid	GW 2 D,G	GW 3 Netid	GW 3 D,G	GW 4 Netid	GW 4 D,G	GW 5 Netid	GW 5 D,G	GW 6 Netid	GW 6 D,G
1	0,1	1	0,2	1	1,2	1	1,1	1	1,2	2	1,3
2	1,2	2	0,2	2	0,3	3	0,4	2	1,2	4	1,5
3	0,1	3	1,1	4	1,2	4	1,5	3	1,4	5	1,3
4	1,2	4	0,2	5	0,3	5	1,5	4	0,5	6	0,6
7	1,4	5	1,3	6	0,3	7	0,4	5	0,5	7	1,5
		6	1,3	7	1,5	8	1,5	6	1,3	8	0,6
		7	1,5	8	1,5			7	0,5		
		8	1,5					8	0,5		

Hal96
fig. 9.15

Final routing tables

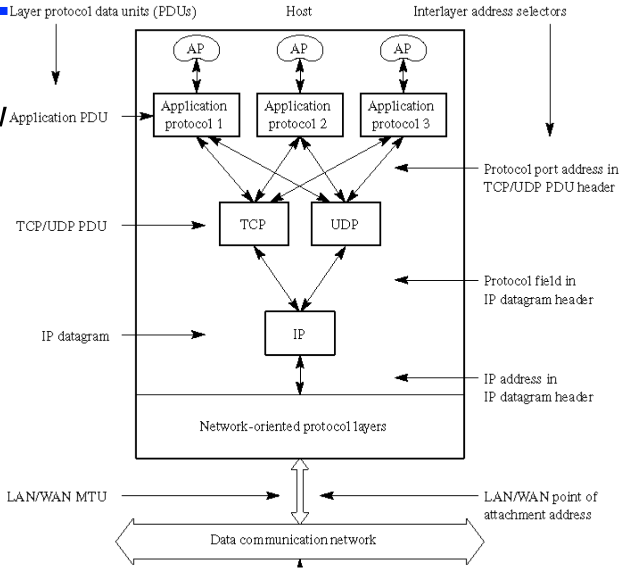
GW 1 Netid	GW 1 D,G	GW 2 Netid	GW 2 D,G	GW 3 Netid	GW 3 D,G	GW 4 Netid	GW 4 D,G	GW 5 Netid	GW 5 D,G	GW 6 Netid	GW 6 D,G
1	0,1	1	0,2	1	1,2	1	1,1	1	1,2	1	2,3
2	1,2	2	0,2	2	0,3	2	2,1	2	1,2	2	1,3
3	0,1	3	1,1	3	2,2	3	0,4	3	1,4	3	2,5
4	1,2	4	0,2	4	1,2	4	1,5	4	0,5	4	1,5
5	2,2	5	1,3	5	0,3	5	1,5	5	0,5	5	1,3
6	2,2	6	1,3	6	0,3	6	2,5	6	1,3	6	0,6
7	1,4	7	1,5	7	1,5	7	0,4	7	0,5	7	1,5
8	2,2	8	1,3	8	1,5	8	1,5	8	0,5	8	0,6

Transport Layer

- Some more bits and pieces related to TCP
- We have seen most of it, but I like to emphasize some issues

Transport Layer

- TCP/IP overview


Layer protocol data units (PDUs)
Host
Interlayer address selectors
AP
AP
AP
Application PDU
Application protocol 1
Application protocol 2
Application protocol 3
Protocol port address in TCP/UDP PDU header
TCP/UDP PDU
TCP
UDP
Protocol field in IP datagram header
IP datagram
IP
IP address in IP datagram header
Network-oriented protocol layers
LAN/WAN MTU
LAN/WAN point of attachment address
Data communication network

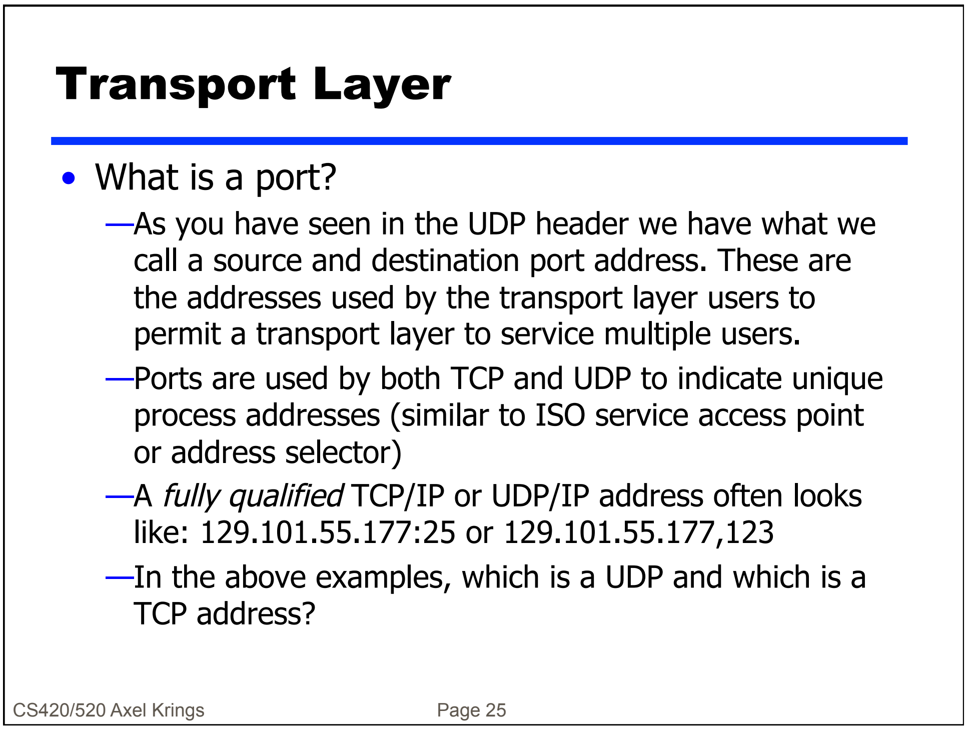
Hal96
fig. 11.2

Transport Layer

- A UDP Protocol Data Unit Header:

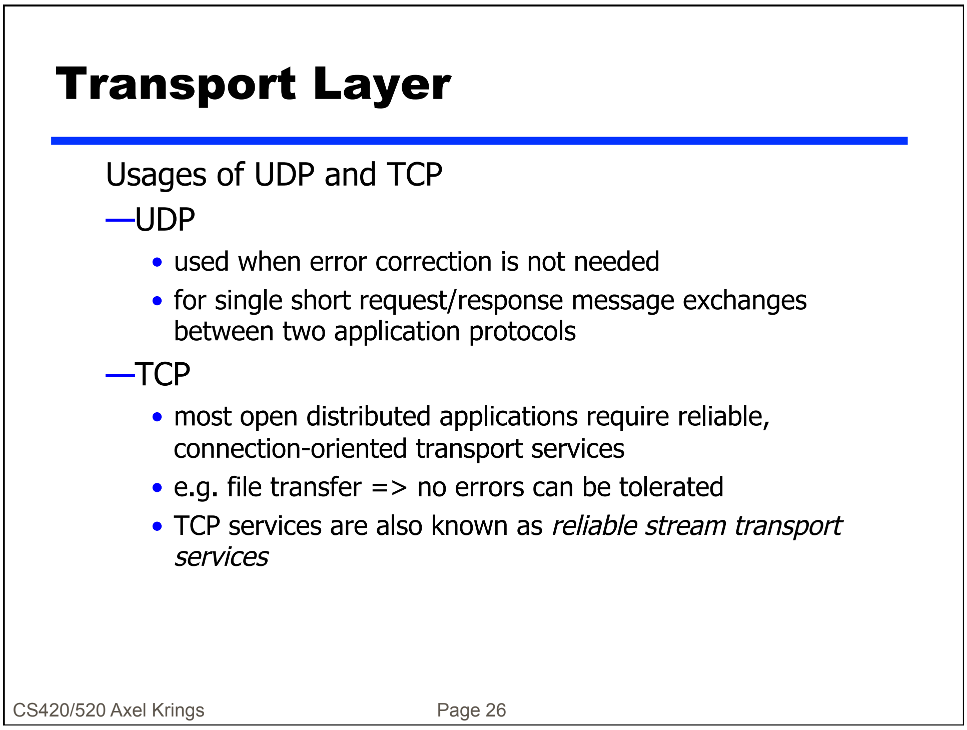
Source Port (16 bits)
Destination Port (16 bits)
Length (16 bits)
Checksum (16 bits)

- **Source Port**- an optional field, when meaningful it indicates the port of the sending process, and may be assume to be the port to which a reply should be sent. If not used, a value of zero is indicated
- **Destination Port** - has meaning within the context of a particular internet destination address
- **Length** - is number of octets of the user datagram including the header
- **Checksum** is 16-bit one complement of the one' s complement sum of a pseudo header of IP layer info and UDP header info.



Transport Layer

- What is a port?
  - —As you have seen in the UDP header we have what we call a source and destination port address. These are the addresses used by the transport layer users to permit a transport layer to service multiple users.
  - —Ports are used by both TCP and UDP to indicate unique process addresses (similar to ISO service access point or address selector)
  - —A *fully qualified* TCP/IP or UDP/IP address often looks like: 129.101.55.177:25 or 129.101.55.177,123
  - —In the above examples, which is a UDP and which is a TCP address?



Transport Layer

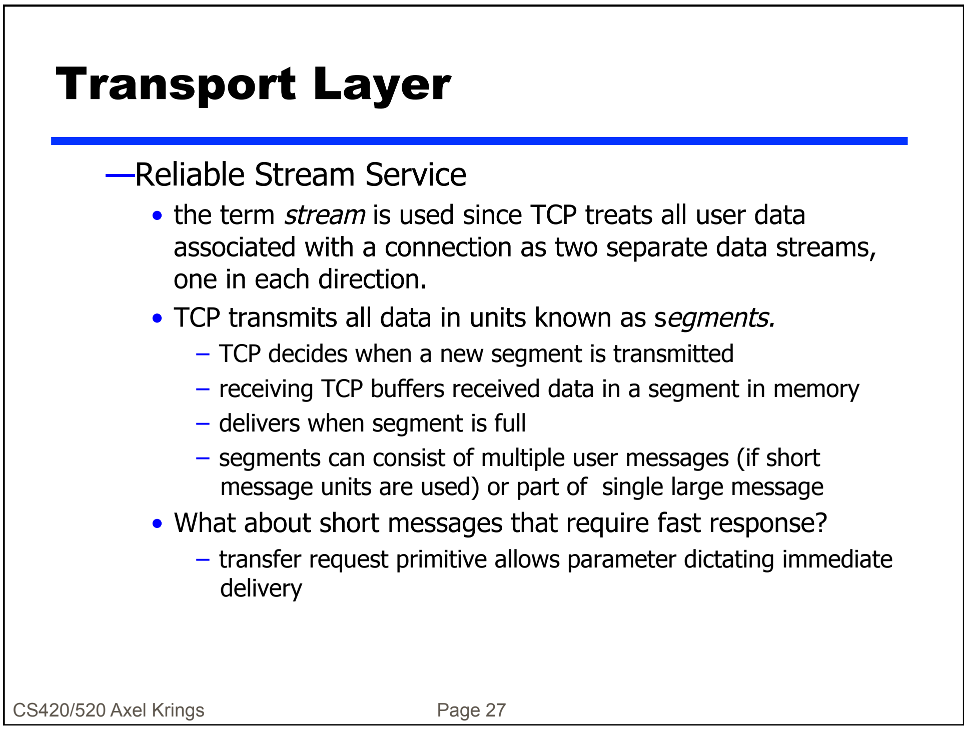
Usages of UDP and TCP

—UDP

- used when error correction is not needed
- for single short request/response message exchanges between two application protocols

—TCP

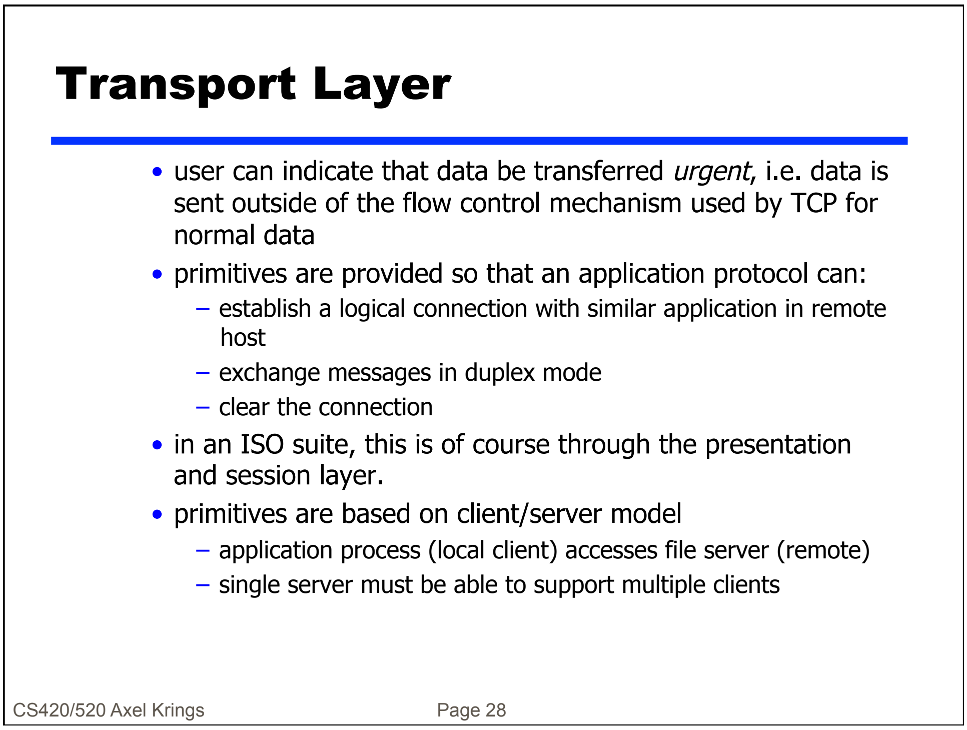
- most open distributed applications require reliable, connection-oriented transport services
- e.g. file transfer => no errors can be tolerated
- TCP services are also known as *reliable stream transport services*



Transport Layer

—Reliable Stream Service

- the term *stream* is used since TCP treats all user data associated with a connection as two separate data streams, one in each direction.
- TCP transmits all data in units known as s*egments.*
  - TCP decides when a new segment is transmitted
  - receiving TCP buffers received data in a segment in memory
  - delivers when segment is full
  - segments can consist of multiple user messages (if short message units are used) or part of single large message
- What about short messages that require fast response?
  - transfer request primitive allows parameter dictating immediate delivery



Transport Layer

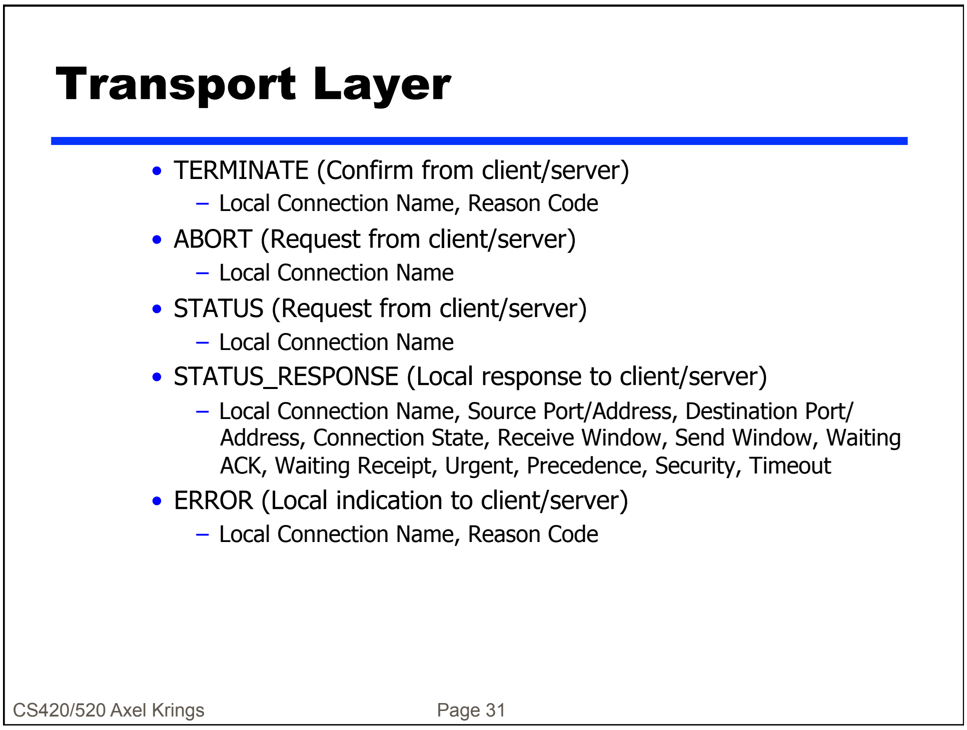
- user can indicate that data be transferred *urgent*, i.e. data is sent outside of the flow control mechanism used by TCP for normal data
- primitives are provided so that an application protocol can:
  - establish a logical connection with similar application in remote host
  - exchange messages in duplex mode
  - clear the connection
- in an ISO suite, this is of course through the presentation and session layer.
- primitives are based on client/server model
  - application process (local client) accesses file server (remote)
  - single server must be able to support multiple clients

Transport Layer

- TCP Service Primitives:
  - UNSPECIFIED_PASSIVE_OPEN (Request by server)
    - Source Port, *Timeout, Timeout-Action, Precedence, Security Range*
  - FULL_PASSIVE_OPEN (Request by server)
    - Local Connection Name , *Timeout, Timeout-Action, Precedence, Security Range*
  - ACTIVE_OPEN (Request by client)
    - Source Port, Destination Port, Destination Address, *Timeout, Timeout-Action, Precedence, Security Range*
  - ACTIVE_OPEN_WITH_DATA (Request by client)
    - Source Port, Destination Port, Destination Address, Data, Data Length, Push Flag, Urgent Flag, *Timeout, Timeout-Action, Precedence, Security Range*
  - OPEN_ID (Local Response to Client)
    - Local Connection Name, Source Port, Destination Port/Address

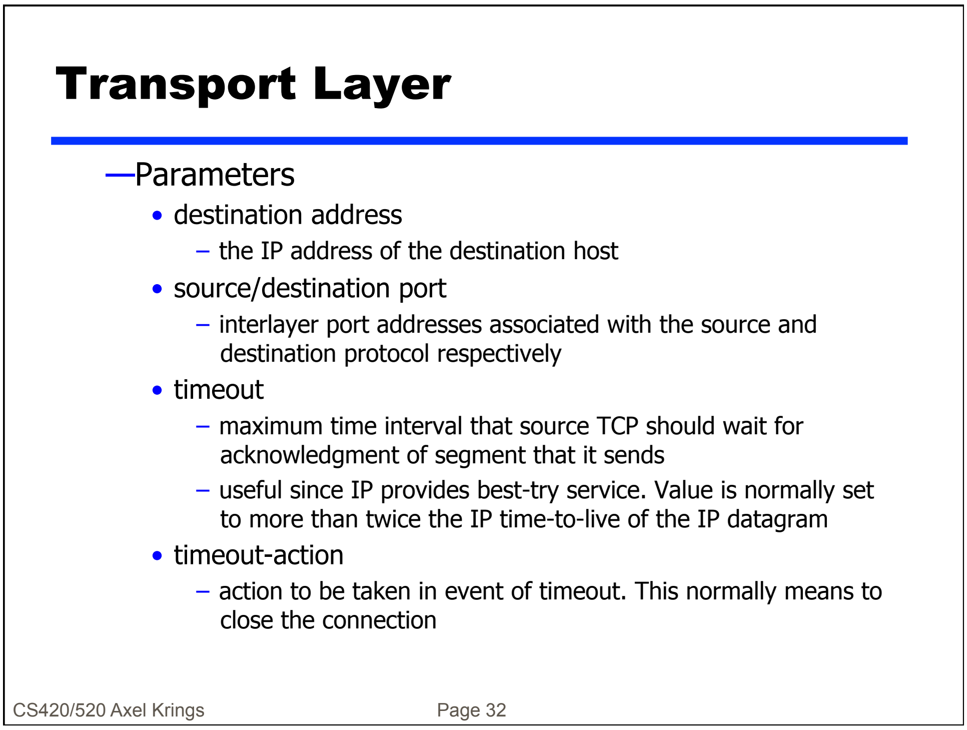
Transport Layer

- OPEN_SUCCESS (Confirm to Client)
  - Local Connection Name
- OPEN_FAILURE (Confirm to Client)
  - Local Connection Name
- SEND (Request from client/server)
  - Local Connection Name, Data, Data Length, Urgent Flag, *Timeout, Timeout-Action*
- DELIVER (Indication to client/server)
  - Local Connection Name, Data, Data Length, Urgent Flag
- ALLOCATE (Request from client/server)
  - Local Connection Name, Data Length
- CLOSE (Request from client/server)
  - Local Connection Name
- CLOSING (Indication to client/server)
  - Local Connection Name



Transport Layer

- TERMINATE (Confirm from client/server)
  - Local Connection Name, Reason Code
- ABORT (Request from client/server)
  - Local Connection Name
- STATUS (Request from client/server)
  - Local Connection Name
- STATUS_RESPONSE (Local response to client/server)
  - Local Connection Name, Source Port/Address, Destination Port/Address, Connection State, Receive Window, Send Window, Waiting ACK, Waiting Receipt, Urgent, Precedence, Security, Timeout
- ERROR (Local indication to client/server)
  - Local Connection Name, Reason Code



Transport Layer

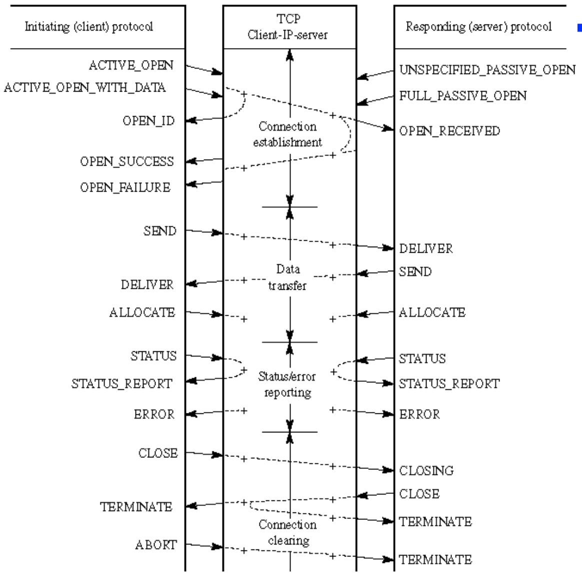
—Parameters

- destination address
  - the IP address of the destination host
- source/destination port
  - interlayer port addresses associated with the source and destination protocol respectively
- timeout
  - maximum time interval that source TCP should wait for acknowledgment of segment that it sends
  - useful since IP provides best-try service. Value is normally set to more than twice the IP time-to-live of the IP datagram
- timeout-action
  - action to be taken in event of timeout. This normally means to close the connection

Transport Layer

—Parameters
- precedence
  - collection of parameters specifying the contents of the service type field of the IP packet header
- security range
  - specifies the level of security to be applied to potential users
- push flag
  - transmit data immediately
- urgent flag
  - transmit outside of normal flow
- local connection name
  - port address enables TCP to relate a received segment to a particular application protocol
  - in case of server, however, multiple transactions (and thus logical connections) may be in progress concurrently

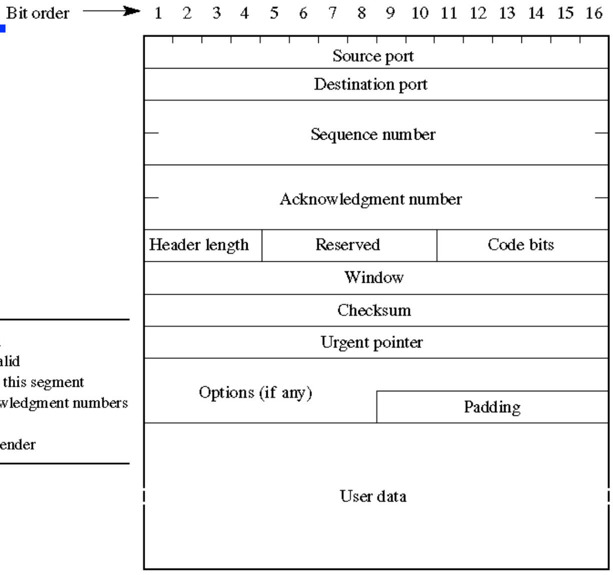
Transport Layer



Hal96
fig. 11.4

Transport Layer

- TCP Protocol Data Unit



Bit order → 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Source port

Destination port

Sequence number

Acknowledgment number

Header length | Reserved | Code bits

Window

Checksum

Urgent pointer

Options (if any) | Padding

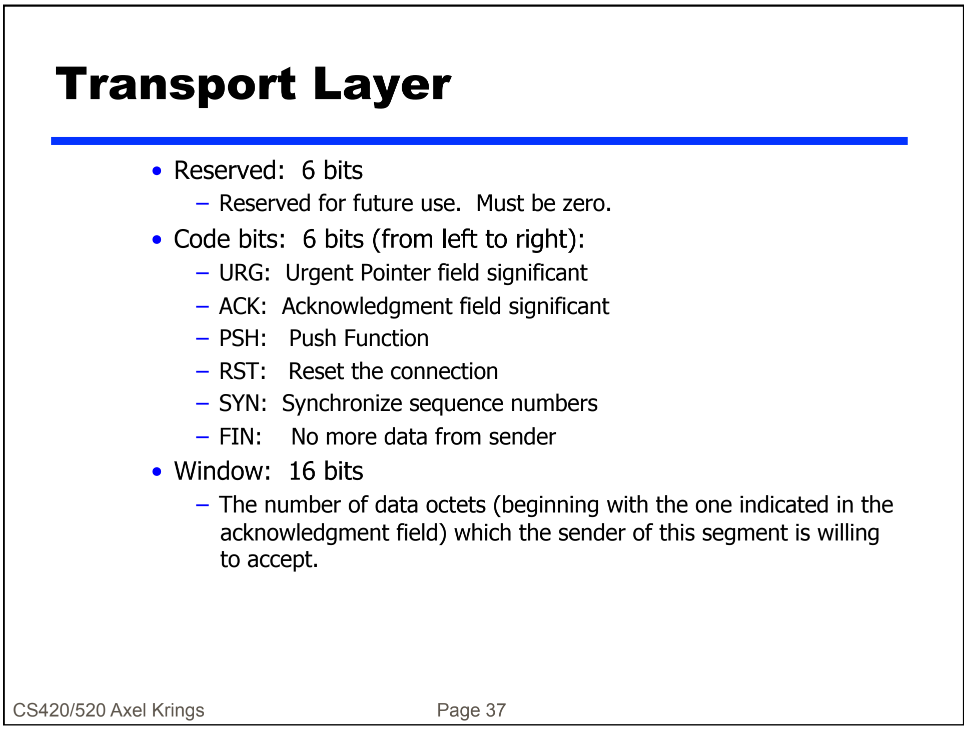
User data

Code bits:

Bit position	Name – function
11	URG – urgent pointer field valid
12	ACK – acknowledgment field valid
13	PSH – deliver data on receipt of this segment
14	RST – reset the sequence/acknowledgment numbers
15	SEQ – sequence number valid
16	FIN – end of byte stream from sender

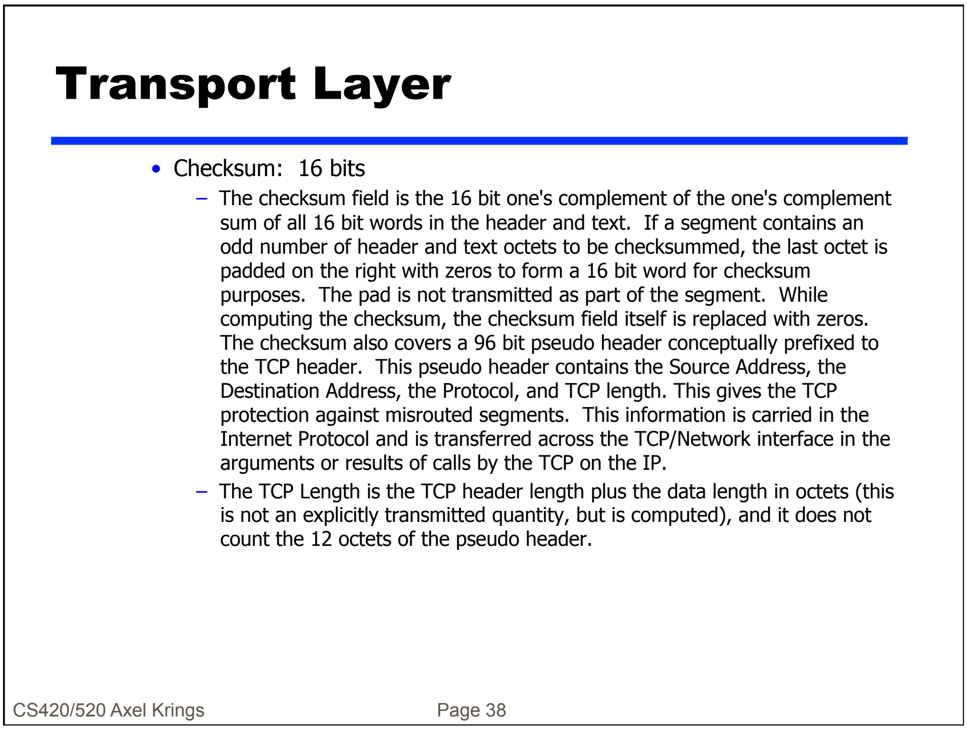
Transport Layer

- Source Port: 16 bits
  - The source port number.
- Destination Port: 16 bits
  - The destination port number.
- Sequence Number: 32 bits
  - The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.
- Acknowledgment Number: 32 bits
  - If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent.
- Header length: 4 bits
  - The number of 32 bit words in the TCP Header. This indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits long.



Transport Layer

- Reserved: 6 bits
  - Reserved for future use. Must be zero.
- Code bits: 6 bits (from left to right):
  - URG: Urgent Pointer field significant
  - ACK: Acknowledgment field significant
  - PSH: Push Function
  - RST: Reset the connection
  - SYN: Synchronize sequence numbers
  - FIN: No more data from sender
- Window: 16 bits
  - The number of data octets (beginning with the one indicated in the acknowledgment field) which the sender of this segment is willing to accept.



Transport Layer

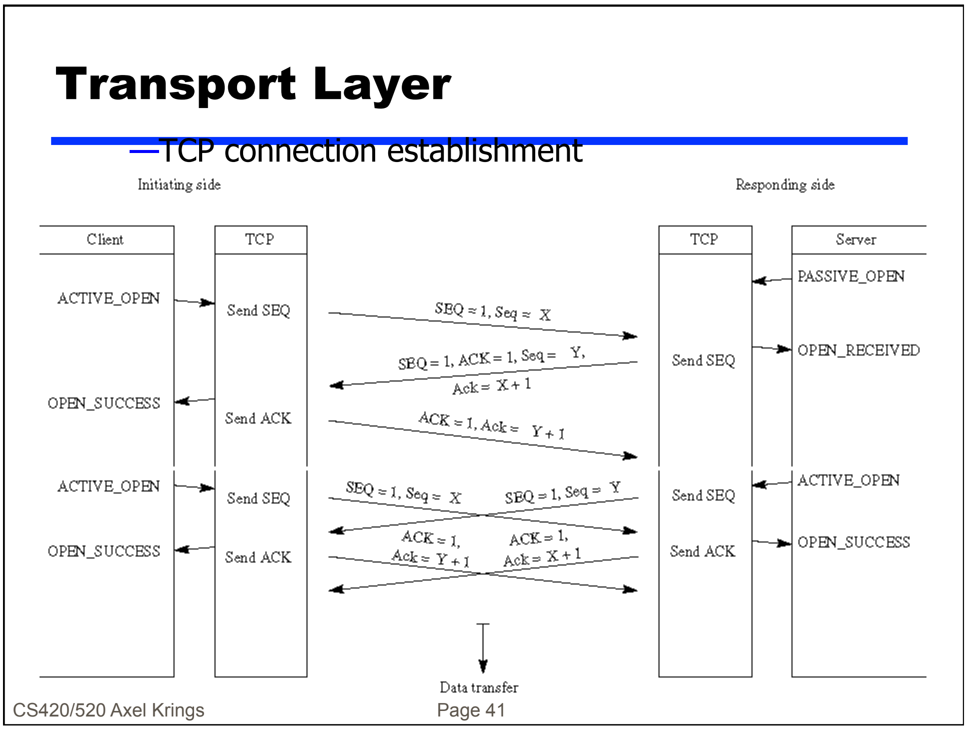
- Checksum: 16 bits
  - The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to form a 16 bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros. The checksum also covers a 96 bit pseudo header conceptually prefixed to the TCP header. This pseudo header contains the Source Address, the Destination Address, the Protocol, and TCP length. This gives the TCP protection against misrouted segments. This information is carried in the Internet Protocol and is transferred across the TCP/Network interface in the arguments or results of calls by the TCP on the IP.
  - The TCP Length is the TCP header length plus the data length in octets (this is not an explicitly transmitted quantity, but is computed), and it does not count the 12 octets of the pseudo header.

Transport Layer

- Urgent Pointer: 16 bits
  - This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The urgent pointer points to the sequence number of the octet following the urgent data. This field is only interpreted in segments with the URG bit set.
- Options: variable
  - Options may occupy space at the end of the TCP header and are a multiple of 8 bits in length. All options are included in the checksum. An option may begin on any octet boundary. There are two cases for the format of an option:
    - Case 1: A single octet of option-kind.
    - Case 2: An octet of option-kind, an octet of option-length, and the actual option-data octets.
  - The option-length counts the two octets of option-kind and option-length as well as the option-data octets. Note that the list of options may be shorter than the data offset field might imply. The content of the header beyond the End-of-Option option must be header padding.

Transport Layer

- TCP Operation:
  - Connection Establishment:
    - Uses a three-way handshake. Each side sends an initial sequence number (indicated by setting the SYN bit)
  - Data Transfer
    - Uses HDLC-like sliding window and GO-BACK-N protocols
    - Allows use of PUSH and URGENT flags
    - Can recover from errors due to crashes or old duplicate SYNs by use of RST command
  - Connection Termination
    - Graceful request for termination (Setting the FIN flag)
    - Response may contain data.



Transport Layer

—TCP connection establishment

Initiating side

Responding side

Client

TCP

TCP

Server

PASSIVE_OPEN

ACTIVE_OPEN

Send SEQ

SEQ = 1, Seq = X

Send SEQ

OPEN_RECEIVED

SEQ = 1, ACK = 1, Seq = Y,
Ack = X + 1

OPEN_SUCCESS

Send ACK

ACK = 1, Ack = Y + 1

ACTIVE_OPEN

Send SEQ

SEQ = 1, Seq = X

SEQ = 1, Seq = Y

Send SEQ

ACTIVE_OPEN

OPEN_SUCCESS

Send ACK

ACK = 1,
Ack = Y + 1

ACK = 1,
Ack = X + 1

Send ACK

OPEN_SUCCESS

Data transfer