Elements of Networks

CS342, Lecture 7

Tuesday, Sept. 26th, 2006
Wellesley College
Daniel Bilar
Today’s class goals

- Introduction to networks (structure, addressing, protocols)
- Familiarity with OSI reference model
- Understanding the responsibilities and functions of the layers of the TCP/IP Protocol Stack
- Parsing an HTTP session and ethereal (new name: wireshark)
Internetworking

- Multiple incompatible LANs can be physically connected by specialized hardware/software called routers.
- The connected networks are called an internetwork. The Internet is one (big & successful) example of an internetwork.

LAN 1 and LAN 2 might be completely different, totally incompatible LANs (e.g., Ethernet, Wi-Fi, ATM, Circuit-switched)
Internetworking Issues

- How do I designate a distant host?
  Addressing / naming

- How do I send information to a distant host?
  Underlying service model
    What gets sent?
    How fast will it go?
    What happens if it doesn’t get there?

Routing
  How to get information from A to B?

- Challenges
  Heterogeneity
    Assembly from variety of different networks
  Scalability
    Ensure ability to grow to worldwide scale
OSI Reference model

- **Open Systems Interconnection** - Released by International Organization for Standardization (ISO) in 1984
- Allows for a layered approach to implementation of networking
- 7 layers (for political reasons)
  - Application, Presentation, Session, Transport, Network, Data link, Physical

→ Model is used, protocols are not
Terminology

Service
Set of ‘primitives’ (operations) that a layer provides to the layer above it. Says what the layer does,
DESCRIPTION

Protocol
Set of rules governing the format and meaning of messages between peers within same layer
Specifies implementation of service,
IMPLEMENTATION

Interface
Set of methods, parameters and return values
Specifies how layer is accessed
INTERFACE

Think Object Oriented Programming
Why did OSI fail?

- **Bad implementations**
  - Huge and slow and inefficient in operation

- **Bad politics**
  - Perception of Euroweenies production

- **Bad timing**
  - “The apocalypse of the two elephants”

- **Bad technology**
  - Very complex

Q: What do you get when you cross a mobster with an international standard?
A: Someone who makes you an offer you can’t understand

---

David Clark’s (MIT) theory of standards
Introducing the Internet Network Layer (IP)

The IP layer provides interoperability between vastly different networks.

The IP layer is a ‘lowest common denominator’ for any participant on what we call the Internet.

The Internet IP layer is the linchpin that holds the whole Internet architecture together!
The Internet ‘Network’ layer

Host, router network layer functions:

- Routing protocols
  - path selection
  - RIP, OSPF, BGP
- IP protocol
  - addressing conventions
  - datagram format
  - packet handling conventions
- ICMP protocol
  - error reporting
  - router “signaling”

The Internet ‘Network’ layer’s job is to permit hosts to inject packets into *any* network and have them travel independently to the destination (potentially on a different networks)
The Internet Protocol (IP)

The Internet layer defines an official packet format and protocol called IP (Internet Protocol) -> An implementation of network layer

Designed for packet-switched network, each packet contains no more than 64K bytes

Connectionless: Each packet is routed independently with sender and receiver address (what is the advantage?)

Best-effort: Packets could be discarded during transmission because of the exhaustion of resources or a failure at the data link or physical layer

Unreliable: Reliability is ensured at higher layer, such as TCP
‘Reliable’: Circuit Switching

- Used by traditional telephone network
- A call has three phases:
  - Establish circuit from end-to-end (“dialing”),
  - Communicate,
  - Close circuit (“tear down”)
- Originally, a circuit was an end-to-end physical wire. Nowadays, a circuit is like a virtual private wire: each call has its own private, guaranteed data rate from end-to-end.
‘Unreliable’: Packet Switching

- Used by the ‘Internet’
- Each packet is individually routed packet-by-packet, using ‘routing tables’ of ‘routers’.
- Different packets may take different paths (see animation at http://computer.howstuffworks.com/internet-infrastructure.htm)
TCP/IP reference model

- From ARPANET (DoD)
- Specified in 1974 by Cerf and Kahn
- Bible is Stevens, [TCP/IP Illustrated Volume 1 – the Protocols](mailto:TCP/IP.Illustrated.Volume.1@Stevens.com)

Protocols are widely used, model is not
Addressing

a) Flat
   e.g., every host identified by its 48-bit MAC address
   Router would need entry for every host in the world
   Too big (although technology can help this)
   Too hard to maintain as hosts come and go

b) Hierarchy
   Address broken into segments of increasing specificity
   Example (Daniel’s office phone):
   781 (Wellesley Town) – 283 (College) – 3093 (office #)
   Route to general region and then work toward specific destination
   As people and organizations shift, only update affected routing tables
IP Addressing

- IPv4: 32-bit addresses

Typically, write in dotted decimal format

E.g., 149.130.12.213 (www.wellesley.edu)
Each number is decimal representation of one byte
(8 bits)

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hexadecimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>149</td>
<td>95</td>
<td>1001 0101</td>
</tr>
<tr>
<td>130</td>
<td>82</td>
<td>1000 0010</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>0000 1100</td>
</tr>
<tr>
<td>213</td>
<td>D5</td>
<td>1101 0101</td>
</tr>
</tbody>
</table>
IP Addressing and Forwarding

- **Routing table requirement**
  For every possible destination IP address, give next hop
  Nearly $2^{32}$ (4.3 x 10⁹) possibilities!

- **Hierarchical Addressing Scheme**

  Address split into network ID and host ID
  All packets to given network follow same route
  Until they reach destination network

  **Fields**
  - **pfx**
    Prefix to specify split between network & host IDs
  - **network**
    $2^x$ possibilities
  - **host**
    $2^y$ possibilities
IP Address Classes

- **Class A**
  - First octet: 1–126
  - Example: mit.edu: 18.7.22.69

- **Class B**
  - First octet: 128–191
  - Example: wellesley.edu: 149.130.12.213

- **Class C**
  - First octet: 192–223
  - Example: adsl-216-63-78-18.dsl.hstntx.swbell.net: 216.63.78.18

- **Classes D, E, F**
  - Not commonly used
Two levels: Basic IP addressing

- IP address Q
- Demultiplex with network id
  - pfx
  - network
  - host
- Demultiplex with host id
  - Host B

Size of subtree determined by Class of network id
IP Address Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Count</th>
<th>Hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$2^7-2 = 126$ (0 &amp; 127 reserved)</td>
<td>$2^{24}-2 = 16,777,214$ (all 0s, all 1s reserved)</td>
</tr>
<tr>
<td>B</td>
<td>$2^{14} = 16,398$</td>
<td>$2^{16}-2 = 65,534$ (all 0s, all 1s reserved)</td>
</tr>
<tr>
<td>C</td>
<td>$2^{21} = 2,097,512$</td>
<td>$2^8-2 = 254$ (all 0s, all 1s reserved)</td>
</tr>
<tr>
<td>Total</td>
<td>$2,114,036$</td>
<td></td>
</tr>
</tbody>
</table>

- Partitioning too coarse
  - No local organization needs 16.7 million hosts
  - Large organization likely to be geographically distributed
  - Many organizations must make do with multiple class C’s

- Too many different Network IDs
  - Routing tables must still have 2.1 million entries
Subnetting

- Add a third level
  - From the outside, appears as one monolithic network
    - Single entry in routing table
  - Within network, manage as multiple subnetworks
    - Internal routers must route according to subnet ID

- Subnet Mask
  - Way to specify break between subnet ID and host ID
    - Similar masks used in many contexts
Three levels: Subnetting

- Demultiplex with network id
- Demultiplex with subnet id
- Demultiplex with host id

Size of subtree determined by Length of subnet mask
Routing Table

<table>
<thead>
<tr>
<th>Address Pattern</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>149.130.12.213</td>
<td>255.255.255.0</td>
<td>R1</td>
</tr>
<tr>
<td>149.130.128.0</td>
<td>255.255.128.0</td>
<td>R2</td>
</tr>
<tr>
<td>149.0.0.0</td>
<td>255.0.0.0</td>
<td>R3</td>
</tr>
<tr>
<td>0.0.0.0</td>
<td>0.0.0.0</td>
<td>R4</td>
</tr>
<tr>
<td>149.130.0.0</td>
<td>255.255.0.0</td>
<td>R5</td>
</tr>
</tbody>
</table>

- **Address 149.130.12.213** matches 4 entries
- **Longest Prefix Match**
  - In English: Choose most specific case
  - Select entry with longest sequence of 1’s in mask
  - Most specific case
Transport Protocols Concern only End Hosts, not Routers

- Lowest level end-to-end protocol.
  Header generated by sender is interpreted only by the destination.
  Routers view transport header as part of the payload.
- Adds functionality to the best-effort packet delivery IP service.
  Make up for the shortcomings of the core network.
What do Transport Protocols do?

Multiplexing/demultiplexing for multiple applications
  Port abstraction
Connection establishment
  Logical end-to-end connection
  Connection state to optimize performance
Error control
  Hide unreliability of the network layer from applications
  Many types of errors: corruption, loss, duplication, reordering.
End-to-end flow control
  Avoid flooding the receiver
Congestion control
  Avoid flooding the network
Universal Datagram Protocol (UDP)

- An implementation of transport layer on top of IP
- Unreliable data transmission
  - No guaranteed on delivery
  - Packets could be received out of order
- Add port identification numbers and payload checksum to IP
  - Ports allow multiplexing of data streams
- Highly efficient because of low overhead
  - Suitable for delivering data that is small amount and needs to be sent frequently
  - Typically used for latency-sensitive or low-overhead applications (video, time, DNS, etc.)

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Dest. Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>Checksum</td>
</tr>
</tbody>
</table>
Transmission Control Protocol (TCP)

- An implementation of transport layer on top of IP
- Reliable data transmission that can be used to send a sequence of bytes
  - Provide guaranteed delivery and ordering of bytes, i.e., data are always received in their original order
- Port numbers, like UDP
- Checksums payload
- Flow control
  - Sensitive to packet loss and round-trip time
- Error recovery: retransmit lost/corrupted packets
TCP: Sequence Number Space

- Each byte in byte stream is numbered.
  - 32 bit value
  - Wraps around
  - Initial values selected at start up time
- TCP breaks up the byte stream in packets ("segments")
  - Packet size is limited to the Maximum Segment Size
  - Set to prevent packet fragmentation
- Each segment has a sequence number.
  - Indicates where it fits in the byte stream

![Diagram with sequence numbers and segments]
TCP: Important Flags

- SYN: Synchronize
  Used when setting up connection

- FIN: Finish
  Used when tearing down connection

- ACK
  Acknowledging received data
Establishing TCP Connection

“Three-Way Handshake”

Each side notifies other of starting sequence number it will use for sending
Each side acknowledges other’s sequence number
SYN-ACK: Acknowledge sequence number + 1
Can combine second SYN with first ACK
Protocol: Tearing Down Connection

- Either Side Can Initiate Tear Down
  - Send FIN signal
  - “I’m not going to send any more data”

- Other Side Can Continue Sending Data
  - Half open connection
  - Must continue to acknowledge

- Acknowledging FIN
  - Acknowledge last sequence number + 1
1. Establish a TCP Connection

The "three-way handshake"
2. Transmit data over TCP

Host A

seq 101 ACK 201
seq 201 ACK 121
seq 121 ACK 201

Host B
3a. Terminate TCP Connection gracefully

Host A → Host B

ACK FIN

ACK

ACK FIN

ACK

Seq & Ack numbers omitted for clarity
3b. Terminate TCP Connection abruptly

Unexpected ACK or SYN to a closed port

A Reset flag terminates communication w/o further ado
Tcpdump

- Tcpdump(8):
  Tcpdump outputs the headers of packets on a network interface that match the boolean expression.
  Output in ascii, hexadecimal or binary file
Tcpdump
TCP Connection establishment

03:08:39.91 : 192.168.1.4.3749 > 10.198.102.190.telnet: S
3588754520:3588754520(o) win 5840 <mss 1460,sackOK,nop> (DF)

03:08:39.94 : 10.198.102.190.telnet > 192.168.1.4.3749: S
1624898076:1624898076(o) ack 3588754521 win 1024 <mss 536>

3588754521:3588754521(o) ack 1624898077 win 5840 (DF)
Tcpdump
TCP Connection Termination

03:08:46.76: 192.168.1.4.3749 > 10.198.102.190.telnet: F
3588754563:3588754563(o) ack 1624898686 win 6804 (DF)

03:08:46.77: 10.198.102.190.telnet > 192.168.1.4.3749: F
1624898686:1624898686(o) ack 3588754564 win 982

03:08:46.83: 10.198.102.190.telnet > 192.168.1.4.3749: F
1624898686:1624898686(o) ack 3588754564 win 982

03:08:46.84: 192.168.1.4.3749 > 10.198.102.190.telnet: .
3588754564:3588754564(o) ack 1624898687 win 6804 (DF)
Tcpdump
Abrupt TCP Connection Termination

03:38:12.488670 > 192.168.1.4.3754 > 192.168.1.1.telnet: S
1166553013:1166553013(o) win 5840 <mss 1460,sackOK,timestamp
52171146 0,nop,wscale 0> (DF)

4500 003c 0000 4000 4006 b766 coa8 0104
coa8 0101 0eaa 0017 4588 2fb5 0000 0000
a002 16d0 147d 0000 0204 05b4 0402 080a
031c 118a 0000 0000 0103 0300

ip[9] = 0x06 (tcp)

tcp[13] = 0x14

03:38:12.489026 < 192.168.1.1.telnet > 192.168.1.4.3754: R 0:0(o) ack
1166553014 win 0

4500 0028 3447 0000 ff06 0433 coa8 0101
coa8 0104 0017 0eaa 0000 0000 4588 2fb6
5014 0000 a87b 0000 0000 0000 0000
Example Connection: HTTP

At puma, we are firing up a web browser and going to www.cnn.com
In the sending host, puma

1. **Application-Programming Interface (API)**
   Application (Firefox) forms requests TCP connection with the web server at www.cnn.com

2. **Transmission Control Protocol (TCP)**
   Creates TCP “Connection setup” packet
   TCP requests IP packet to be sent to “B”

TCP Segment

<table>
<thead>
<tr>
<th>TCP Data</th>
<th>TCP Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>Type = Connection Setup</td>
</tr>
</tbody>
</table>
In the sending host, puma

3. Internet Protocol (IP)
Creates IP packet with correct addresses. IP requests packet to be sent to router.

TCP Segment

| TCP Data | TCP Header |

Encapsulation

| IP Data | IP Header |

Destination Address: IP “B”
Source Address: IP “A”
Protocol = TCP
In the sending host, puma

4. **Link (“MAC” or Ethernet) Protocol**
   Creates MAC frame with Frame Check Sequence (FCS).
   Wait for Access to the line.
   MAC requests PHY to send each bit of the frame.

---

[Diagram]

- **IP Packet**
  - IP Data
  - IP Header
  - **Encapsulation**
  - Ethernet FCS
  - Ethernet Data
  - Ethernet Header

- **Destination Address**: MAC “R1”
- **Source Address**: MAC “A”
- **Protocol** = IP
In Router R1

5. Link ("MAC" or Ethernet) Protocol
Accept MAC frame, check address and Frame Check Sequence (FCS) to ensure no bit errors. Pass data to IP Protocol.

[Diagram showing IP Packet and Ethernet Frame with details of MAC and IP addresses]

Destination Address: MAC “R1”
Source Address: MAC “A”
Protocol = IP
In Router R1

6. Internet Protocol (IP)
   Use IP destination address to decide where to send packet next (“next-hop routing”). Request Link Protocol to transmit packet.

<table>
<thead>
<tr>
<th>IP Data</th>
<th>IP Header</th>
</tr>
</thead>
</table>

- **Destination Address**: IP “B”
- **Source Address**: IP “A”
- **Protocol**: TCP

---

**IP Packet**
Example Connection: HTTP

At puma we are firing up a web browser and going to www.cnn.com
In Router R1

7. **Link ("MAC" or Ethernet) Protocol**
   Creates MAC frame with Frame Check Sequence (FCS).
   Wait for Access to the line.
   MAC requests PHY to send each bit of the frame.

![Diagram of IP packet encapsulation in an Ethernet frame](image)

- **IP Packet**
  - IP Data
  - IP Header
  - Encapsulation

- **Ethernet Frame**
  - Ethernet FCS
  - Ethernet Data
  - Ethernet Header

  Destination Address: **MAC “R2”**
  Source Address: **MAC “R1”**
  Protocol = IP

Steps 8-15 are the same as before ...
In Router R5

16. Link (“MAC” or Ethernet) Protocol
Creates MAC frame with Frame Check Sequence (FCS)
Wait for access to the line.
MAC requests PHY to send each bit of the frame

Destination Address: MAC “B”
Source Address: MAC “R5”
Protocol = IP
Receiving host: Traverse the stack

17. Link (“MAC” or Ethernet) Protocol
Accept MAC frame, check address and Frame
Check Sequence (FCS)
Pass data to IP Protocol

--- IP Packet ---

Decapsulation

Ethernet FCS  Ethernet Data  Ethernet Header

Destination Address: MAC “B”
Source Address: MAC “R5”
Protocol = IP

--- Ethernet Frame ---
18. Internet Protocol (IP)
Verify IP address.
Extract/decapsulate TCP packet from IP packet.
Pass TCP packet to TCP Protocol

<table>
<thead>
<tr>
<th>TCP Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Data</td>
</tr>
</tbody>
</table>

Decapsulation

<table>
<thead>
<tr>
<th>IP Packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Data</td>
</tr>
</tbody>
</table>

Destination Address: IP “B”
Source Address: IP “A”
Protocol = TCP
Receiving host: Traverse the Stack

19. Transmission Control Protocol (TCP)
   Accepts TCP “Connection setup” packet
   Establishes connection by sending “Ack”.

20. Application-Programming Interface (API)
   Application receives request for TCP connection with puma.wellesley.edu

TCP Segment

<table>
<thead>
<tr>
<th>TCP Data</th>
<th>TCP Header</th>
</tr>
</thead>
</table>

Type = Connection Setup

Empty
Summary

- IP is the protocol that every participant on the Internet has to understand.
- Each layer of the TCP/IP protocol stack has its defined responsibilities:
  - Provides a service to the layer above
  - Communicates with same layer on peer host
For next time

- Read Tannenbaum’s intro to networks for review (on website)
- Try ethereal or wireshark to get a taste for traffic on a network, e.g.
  - Observe what you see when you don’t do anything – try to identify or guess what that traffic is about
- Go to a web page (HTTP, HTTPS)
- Check your mail with FirstClass
Additional material

- CIDR addressing
- TCP Flow control
- UDP, TCP, IP header layout
Classless Interdomain Routing

- CIDR, pronounced “cider”
- Arbitrary Split Between Network & Host IDs
  Specify either by mask or prefix length

- E.g., Wellesley can be specified as
  149.130.0.0 with netmask 255.255.0.0
  149.130.0.0/16
- Used to aggregate and split network entries as needed
Aggregation with CIDR

- **Original Use: Aggregate Class C Addresses**
- **One organization assigned contiguous range of class C’s**
  
e.g., Microsoft given all addresses 207.46.192.X -- 207.46.255.X

Specify as CIDR address 207.46.192.0/18

![CIDR Address Diagram]

- **Upper 18 bits frozen**
- **Lower 14 bits arbitrary**

Represents $2^6 = 64$ class C networks

- **Use single entry in routing table**
  Just as if were single network address
Important Concepts

Datagram service model
Hierarchical addressing critical for scalable system
  Don’t require everyone to know everyone else
  Reduces amount of updating when something changes
Non-uniform hierarchy useful for heterogeneous networks
  Class-based addressing too coarse
  CIDR helps
  Move to IPv6 due to limited number of 32-bit addresses

Implementation Challenge
  Longest prefix matching much more difficult than when no ambiguity
TCP Flow Control

- Sliding window protocol
  For window size $n$, can send up to $n$ bytes without receiving an acknowledgement
  When the data are acknowledged then the window slides forward

- Window size determines
  How much unacknowledged data can the sender sends
TCP Flow control: Complication

- TCP receiver can delete acknowledged data only after the data has been delivered to the application.

- So, depending on how fast the application is reading the data, the receiver’s window size may change!!!
TCP Flow control: Solution

- Receiver tells sender what is the current window size in every packet it transmits to the sender
- Sender uses this current window size instead of a fixed value

- Window size (also called Advertised window) is continuously changing
- Can go to zero!
  
  Sender not allowed to send anything!
IP Header

0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2 3 4 5 6 7 8 9 A B C D E F
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|   HL   |Type of Service|          Total Length      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Identification |Flags|      Fragment Offset |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Time to Live |   Protocol |   Header Checksum |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Source Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Destination Address                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options |   Padding |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
IP Header

- **Version: 4 bits**
  helps smooth the transition to future version of IP

- **Header length: 4 bits**
  limits the header to $15 \times 32\text{bits} = 60\text{ bytes}$

- **Type of Service: 4 bits**
  Specify a tradeoff between fast service and reliable service, not commonly used

- **Total length: 16 bits**
  Limits each packet to 64K bytes

- **Time-To-Live (TTL): 8 bits**
  limit the life of the packet on the network
  - Initialized to thirty
  - Decremented each time the packet arrives at a routing step
  - Discarded when it is equal to 0

- **Identification (16 bits), Flags (3 bits), and Fragment Offset (13 bits)**
  Partition a datagram into packet if it is too large
  - Each packet must be no larger than 64K
  - The maximum number of fragments per datagram is 8192
UDP Header

| 0 | 7 8 | 15 16 | 23 24 | 31 |
+---+-----+-------+-------+----+
| Source | Destination |
| Port | Port |
| +-----+-------+ |
| Length | Checksum |
+-------+-------+
TCP Header

```
0               1               2               3
+---------------+---------------+---------------+---------------+
| Source Port   | Destination Port |
+---------------+---------------+---------------+---------------+
| Sequence Number |                |
+---------------+---------------+---------------+---------------+
| Acknowledgment Number |
+-------------------+---------------+---------------+---------------+
| Data              | U|A|E|R|S|F| |
| Offset Reserved   | R|C|O|S|Y|I|   Window |
|                   | G|K|L|T|N|N| |
+---------------+---------------+---------------+---------------+
| Checksum        | Urgent Pointer |
+---------------+---------------+---------------+---------------+
| Options         | Padding       |
+---------------+---------------+---------------+---------------+
| data           |
+---------------+---------------+---------------+---------------+
```
Transport Layer (DOD)

- Provides End to End connections between two devices by performing sequencing, acknowledgements, checksums and flow control
- TCP
  Connection-oriented and reliable communications
- UDP
  Non-connection-oriented and unreliable
Transmission Control Protocol (TCP)

- **Segments Application-layer data stream**
  - Provides acknowledgment timers
  - Enables sequence number checking
  - Provides buffer management
  - Initiates connection with three-way handshake
  - Performs error and duplication checking
TCP Header Layout

| 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 |

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
<th>0 - 3</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Sequence Number</th>
<th>4 - 7</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Acknowledgment Number</th>
<th>8 - 11</th>
</tr>
</thead>
</table>

| Data | U|A|P|R|S|F|  | Offset| Reserved | R|C|S|S|Y|I| Window | 12 - 15 |

| G|K|H|T|N|N|  | Options |  | Urgent Pointer | 16 - 19 |

<table>
<thead>
<tr>
<th>Checksum</th>
<th></th>
<th>Urgent Pointer</th>
<th>16 - 19</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Options</th>
<th>Padding</th>
<th>20</th>
<th>---</th>
<th>---</th>
</tr>
</thead>
</table>

RFC 793