TCP/IP protocol
# TCP/IP Reference Model

<table>
<thead>
<tr>
<th>OSI</th>
<th>TCP/IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Application</td>
</tr>
<tr>
<td>6</td>
<td>Presentation</td>
</tr>
<tr>
<td>5</td>
<td>Session</td>
</tr>
<tr>
<td>4</td>
<td>Transport</td>
</tr>
<tr>
<td>3</td>
<td>Network</td>
</tr>
<tr>
<td>2</td>
<td>Data link</td>
</tr>
<tr>
<td>1</td>
<td>Physical</td>
</tr>
</tbody>
</table>

- Not present in the model: Session, Transport, Internet, Host-to-network
Protocols vs. TCP/IP Layering

- **Application Layer**
  - PING
  - TELNET
  - FTP
  - SMTP
  - DNS

- **Transport Layer**
  - TCP
  - UDP

- **Network Layer**
  - ICMP
  - IP
  - IGMP

- **Link Layer**
  - ARP
  - Hardware Interface
  - RARP

- **Media**
TCP/IP Network Interface Layer

- Ethernet
- Serial Line Internet Protocol (SLIP)
- Compress-SLIP
- Point-to-Point Protocol (PPP)
TCP/IP Network Layer
Network Layer

- Handle the movement of packets around the network.
  - IP (Internet Protocol)
  - ICMP (Internet Control Message Protocol)
  - IGMP (Internet Group Management Protocol)
    - Used for group member management in group communication environment.
    - More detail in chapter 14 multicasting.
- Identifies a particular machine

<table>
<thead>
<tr>
<th>Class</th>
<th>Netid</th>
<th>Hostid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>0</td>
<td>hostid</td>
</tr>
<tr>
<td>Class B</td>
<td>1 0</td>
<td>netid</td>
</tr>
<tr>
<td>Class C</td>
<td>1 1 0</td>
<td>netid</td>
</tr>
<tr>
<td>Class D</td>
<td>1 1 1 0</td>
<td>multicast Address</td>
</tr>
<tr>
<td>Class E</td>
<td>1 1 1 1 0</td>
<td>reserved for future use</td>
</tr>
</tbody>
</table>
The values 0 and –1 have some special meanings in IP address:

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>all 0s</td>
<td>This host</td>
</tr>
<tr>
<td>all 0s host</td>
<td>A host on this network</td>
</tr>
<tr>
<td>all 1s</td>
<td>Broadcast on the local network</td>
</tr>
<tr>
<td>net all 1s</td>
<td>Broadcast on the distant network</td>
</tr>
<tr>
<td>127 anything</td>
<td>Loopback</td>
</tr>
</tbody>
</table>
IP (Internet Protocol)

- Provides an unreliable, connectionless datagram delivery service

| Header | Data Area (upper layer datagram) |

IP datagram
The normal size of IP header is 20 bytes

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4</td>
</tr>
<tr>
<td>Header length</td>
<td>8</td>
</tr>
<tr>
<td>Type of service</td>
<td>8</td>
</tr>
<tr>
<td>Total length</td>
<td>16</td>
</tr>
<tr>
<td>Identification</td>
<td>16</td>
</tr>
<tr>
<td>Flags</td>
<td>3</td>
</tr>
<tr>
<td>Fragment offset</td>
<td>13</td>
</tr>
<tr>
<td>Time to live</td>
<td>8</td>
</tr>
<tr>
<td>Protocol</td>
<td>8</td>
</tr>
<tr>
<td>Header checksum</td>
<td>16</td>
</tr>
</tbody>
</table>

- Source IP address
- Destination IP address
- Option (optional)
- Data
**IP Header (1/3)**

- **TOS (Type of Service) field**
  - specifies how the datagram should be handled

<table>
<thead>
<tr>
<th>PRECEDENCE</th>
<th>Low delay</th>
<th>High throughput</th>
<th>High reliability</th>
<th>UNUSED</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **TTL (Time to Live) field**
  - sets a upper limit on the number of routers through which a datagram can pass

- **Protocol field**
  - specifies which high-level protocol (ICMP, UDP, TCP…) was used in the data area.
IP Header (3/3)

- Identification、Flag、Fragment offset fields
  - control the fragmentation and reassembly of dategrams

<table>
<thead>
<tr>
<th></th>
<th>Data1</th>
<th>Data2</th>
<th>Data3</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
<td>600 octets</td>
<td>600 octets</td>
<td>200 octets</td>
</tr>
</tbody>
</table>

- Fragment 1 (offset 0)
- Fragment 2 (offset 600)
- Fragment 3 (offset 1200)
Internet Control Message Protocol (ICMP)

- RFC 792
- used for reporting errors or providing control information

![Diagram showing the structure of an ICMP datagram, including IP header, ICMP header, ICMP message, and ICMP data.](http://dsns.csie.nctu.edu.tw)
ICMP message format

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(contents depend on type & code)

- **type** field: identify the particular ICMP message
- **code** field: specify the condition
- **checksum** field: algorithm used is the same as IP header checksum

http://dsns.csie.nctu.edu.tw
## ICMP Message Type

<table>
<thead>
<tr>
<th>Type field</th>
<th>ICMP Message Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 &amp; 8</td>
<td>echo reply &amp; request</td>
</tr>
<tr>
<td>3</td>
<td>Destination unreachable</td>
</tr>
<tr>
<td>4</td>
<td>Source quench</td>
</tr>
<tr>
<td>5</td>
<td>Redirect</td>
</tr>
<tr>
<td>9 &amp; 10</td>
<td>Router advertisement &amp; solicitation</td>
</tr>
<tr>
<td>11</td>
<td>Time exceed</td>
</tr>
<tr>
<td>12</td>
<td>Parameter problem</td>
</tr>
<tr>
<td>13 &amp; 14</td>
<td>Timestamp request &amp; reply</td>
</tr>
<tr>
<td>15</td>
<td>Information request &amp; reply</td>
</tr>
<tr>
<td>17</td>
<td>Address mask request &amp; reply</td>
</tr>
</tbody>
</table>

[http://dsns.csie.nctu.edu.tw](http://dsns.csie.nctu.edu.tw)
TCP/IP Transport Layer
Transport Layer

- provides a flow of data between two hosts, for the application above.
  - UDP (User Datagram Protocol)
    - unreliable
    - datagram-oriented
  - TCP (Transmission Control Protocol)
    - reliable
    - connection-oriented
User Datagram Protocol (UDP)

- RFC 768
- A simple, datagram-oriented, transport layer protocol. It provides an unreliable connectionless delivery service.
### UDP Header (1/2)

<table>
<thead>
<tr>
<th>Bit 0-15</th>
<th>Bit 16-31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source port number</td>
<td>Destination port number</td>
</tr>
<tr>
<td>UDP length</td>
<td>UDP checksum (optional)</td>
</tr>
</tbody>
</table>

- **source/destination port**: used to demultiplex datagrams among the processes waiting to receive.
- **length**: a count of octets in UDP header and data area
**checksum**: UDP prepends a pseudo-header for computing checksum.
- include some information in IP header
- verify that the UDP datagram has reached its correct destination.

<table>
<thead>
<tr>
<th>0</th>
<th>15</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zero</td>
<td>Protocol</td>
<td>UDP length</td>
<td></td>
</tr>
</tbody>
</table>
UDP datagram transmission problem (1/2)

- Fragmentation
  - when the UDP datagram size is bigger than path MTU, it needs to be fragmented.
- Problem:
  - if “don’t fragment” flag is turned on, the IP datagram will be discard.
  - If part of the fragmented datagrams lost, the whole data needs retransmit.
Packet lost
- occurs when
  - router receives a UDP datagram that required fragment, but “don’t fragmentation” flag is turned on.
  - router is too busy to handle all datagram.
- problem: the router report the source host with ICMP error message, but the original process is over.
Transmission Control Protocol (TCP)

- RFC 793
- Provides a connection-oriented, reliable, full-duplex, byte stream service.
  - Exactly two end point communication with each other.
TCP Reliability

- TCP provides reliability by doing the following
  - data is broken into what TCP considers the *best sized chunk* to send.
  - maintain a *timer*, and waiting for the end to acknowledge
  - when receive data, send back an *acknowledge*
  - maintain a *checksum* on its header and data
  - *Re-sequence* the received data if necessary
  - *discard duplicate data*
  - provide *flow control*
TCP Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header length</td>
<td></td>
</tr>
<tr>
<td>Source port number</td>
<td>0</td>
</tr>
<tr>
<td>Des. port number</td>
<td>15 16</td>
</tr>
<tr>
<td>Sequence number</td>
<td>31</td>
</tr>
<tr>
<td>Acknowledgement number</td>
<td></td>
</tr>
<tr>
<td>Header length</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>Urgent pointer</td>
<td></td>
</tr>
<tr>
<td>Window size</td>
<td></td>
</tr>
<tr>
<td>TCP checksum</td>
<td></td>
</tr>
<tr>
<td>Option(optional)</td>
<td></td>
</tr>
<tr>
<td>Data(optional)</td>
<td></td>
</tr>
</tbody>
</table>
TCP Connection Establishment

- Three-way Handshake

```
Step 1
Host A
ISN=100
SYN(100)

Step 2
Host B
ISN=200
SYN(200)
ACK(101)

Step 3
Host A
ACK(201)
```
TCP Connection Termination

- Full-duplex
  - each direction must be shut down independently
- half-close
  - active close
  - passive close

```
Host A            Host B
Step 1            FIN(500)
                 ACK(501)
                 FIN(400)
                 ACK(401)
Step 2
Step 3
Step 4
```
Flow control:
Sliding Window Protocol

- make stream transmission more efficient
- example: (Sliding window size = 3)
TCP/IP Application Layer
Application Layer

- handles the detail of the particular application
  - FTP (File Transfer Protocol)
  - SMTP (Simple Mail Transfer Protocol)
  - POP/IMAP (Post Office Protocol/Internet Message Access Protocol)
  - HTTP (Hypertext Transfer Protocol)
  - NNTP (Network News Transfer Protocol)
File Transfer Protocol (FTP)

- RFC 114
- The major TCP/IP file transfer protocol
- Features
  - Interactive Access: allows humans to easily interact with remote servers.
  - Format Specification: allows the client to specify the type and format of stored data.
  - Authentication Control: require client to authorize themselves by sending a name and password.
FTP connection

- use two TCP connection to transfer data
  - control connection: stay up for entire time and send command from client to server.
  - Data connection: created each time a file transferred between client and server.
Control Connection Establishment

1: Connect to port 21
2: Fork slave process
3: Build control connection
Data Connection Establishment

1: Send command "PORT x1,x2,x3,x4,x5,x6"

2: Fork process for data connection

3: Build data connection
4: Start data transmission

Client
At port x5*256+x6

Slave Server at port 21

process at port 20
Simple Mail Transfer Protocol (SMTP)

- use TCP connection for exchanging e-mail
- client request service through ASCII command
  - `HELO` : build connection with server
  - `MAIL FROM` : sender e-mail address
  - `RCPT TO` : receiver e-mail address
  - `DATA` : mail context
  - `QUIT` : close connection
- Special functionality
  - Mail forwarding
  - Alias expansion
Internet Electronic Mail

1. User at a terminal (sender)
2. User agent
3. Queue of mail to be sent
4. Message transfer agent
5. Client
   - Tcp connection
   - Tcp port 25
6. User agent
7. User mailboxes
8. Message transfer agent
9. User at a terminal (receiver)

http://dsns.csie.nctu.edu.tw
Mail forwarding & Alias expansion

Diagram:
- User interface
- Alias database
- Alias expansion
- Forwarding
- Outgoing mail spool area
- Server (to accept mail)
- Client (background transfer)
- TCP connection for outgoing mail
- TCP connection for incoming mail

User sends mail:
- User reads mail

URL: http://dsns.csie.nctu.edu.tw
Other Remote E-mail Access Protocol

- POP (Post Office Protocol)
- IMAP (Internet Message Access Protocol)
- DMSP (Distributed Mail System Protocol)
Hypertext Transfer Protocol (HTTP)

- RFC 2068
HTTP Message format

- HTTP request
  - Three requests are supported

<table>
<thead>
<tr>
<th>Request line</th>
<th>Headers (0 or more)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;blank line&gt;</td>
</tr>
<tr>
<td>Body</td>
<td>(only for POST request)</td>
</tr>
</tbody>
</table>

GET: get information
HEAD: test for validity, accessibility, and recent modification
POST: for posting email, news, or sending forms, a valid Content-Length header field is required

<table>
<thead>
<tr>
<th>status line</th>
<th>Headers (0 or more)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;blank line&gt;</td>
</tr>
<tr>
<td>Body</td>
<td>(HTML file, graph...)</td>
</tr>
</tbody>
</table>
HTTP Performance Problems

- Biggest factor: using TCP for HTTP
  - handshake message
  - slow start
  - TIME_WAIT delay on the server
- Proposed solutions
  - Persistent connection
  - GETALL, GETLIST request
Network News Transfer Protocol (NNTP)

- RFC 977, 1036