KTH ROYAL INSTITUTE
OF TECHNOLOGY

Networks and Interprocess Communication

Vladimir Vlassov and Johan Montelius

Requirements

- Performance
- Scalability
- Reliability
- Security
- Mobility
- Quality of Service
- Multicasting

Network. Internet

A ***network*** is a hardware and software data communication system that provides interconnection of computers and other devices.

Internet is a set of networks connected with routers.

The Internet is the largest internet that includes commercial, military, university, and other networks with different physical links and various protocols, including IP (Internet Protocol)

Types of networks

- WAN - Wide Area Networks
- MAN - Metropolitan Area Networks
- LAN - Local Area Networks
- PAN - Personal Area Networks

Latency

Transfer rate:

What is the rate at which we can send data?

Performance

- Latency - how long time does it take to send an empty message?
- Transfer rate - what is the rate at which we can send data?

Latency

Why does it take time to send a message?

- distance - speed of signal (light)
- access - granting of resource
- routing - processing in nodes

fast as ..

What is the speed of light?

300 000 km/s ... or 300 km/ms

Distance in ms:

Stockholm - Hamburg approx. 800 km or 3 ms

Stockholm - NYC approx. 6.600 km or 23 ms

Stockholm - Melbourne approx. 15.600 km or 52 ms

Routers. switches and fiber optics adds to this so Melbourne is approx. 300 ms away.

ping

vladv — bash — 80×24

```
pc65:~ vladv$ ping www.aflcommunityclub.com.au
PING www.aflcommunityclub.com.au (202.74.66.109): 56 data bytes
64 bytes from 202.74.66.109: icmp_seq=0 ttl=43 time=371.140 ms
Request timeout for icmp_seq 1
64 bytes from 202.74.66.109: icmp_seq=2 ttl=43 time=406.258 ms
64 bytes from 202.74.66.109: icmp_seq=3 ttl=43 time=626.502 ms
64 bytes from 202.74.66.109: icmp_seq=4 ttl=43 time=543.209 ms
64 bytes from 202.74.66.109: icmp_seq=5 ttl=43 time=461.641 ms
64 bytes from 202.74.66.109: icmp_seq=6 ttl=43 time=382.349 ms
64 bytes from 202.74.66.109: icmp_seq=7 ttl=43 time=611.176 ms
64 bytes from 202.74.66.109: icmp_seq=8 ttl=43 time=367.338 ms
64 bytes from 202.74.66.109: icmp_seq=9 ttl=43 time=367.141 ms
64 bytes from 202.74.66.109: icmp_seq=10 ttl=43 time=683.341 ms
64 bytes from 202.74.66.109: icmp_seq=11 ttl=43 time=605.175 ms
64 bytes from 202.74.66.109: icmp_seq=12 ttl=43 time=520.319 ms
^C
--- www.aflcommunityclub.com.au ping statistics ---
13 packets transmitted, 12 packets received, 7.7% packet loss
round-trip min/avg/max/stddev = 367.141/495.466/683.341/112.186 ms
pc65:~ vladv$
```

Using ICMP packages might give a better value, UDP might be slower.

Latency in different networks

• LAN/WLAN - local area networks (Ethernet/WiFi)	1 - 10 ms
• WAN - wide area networks (IP routed)	20 - 400 ms
• Mobile networks	40 - 800 ms
5G:	20-30 ms
• Satellite (geo-stationary)	> 250 ms

Message size

How does latency vary with the size of the messages?

- The packet delivery time or latency is the time from when the first bit leaves the transmitter until the last is received.
- In the case of a physical link, it can be expressed as:

  Packet delivery time = Transmission time + Propagation delay

  - where
    - Transmission time = Packet size / Bit rate
      - The transmission time should not be confused with the propagation delay, which is the time it takes for the first bit to travel from the sender to the receiver.
    - Propagation time = Distance / propagation speed

Transfer rate

The rate at which we can send data (does not mean that it has arrived).

What is the transfer rate of:

ADSL	1 - 20 Mb/s
Ethernet	100 Mb/s - 1 Gb/s
802.11	11 Mb/s, 54 Mb/s, 72 Mb/s ...
3G/4G	1 Mb/s, 2 Mb/s, ... 100 Mb/s
5G	over 1,000 Mb/s (1Gb/s).

Is this shared with others?

Overhead

medium access: 802.11 – RTS/CTS

error handling: detection, forward error correction, ARQ

header: MAC header, IP header, TCP ...

flow control: TCP window

What’s in it for me?

The application layer transfer rate is much lower than the physical layer bit rate.

How does the application layer latency differ from the network layer latency?

Latency and transfer rate

Stockholm to Gothenburg - 400 km, best possible data communication layer?

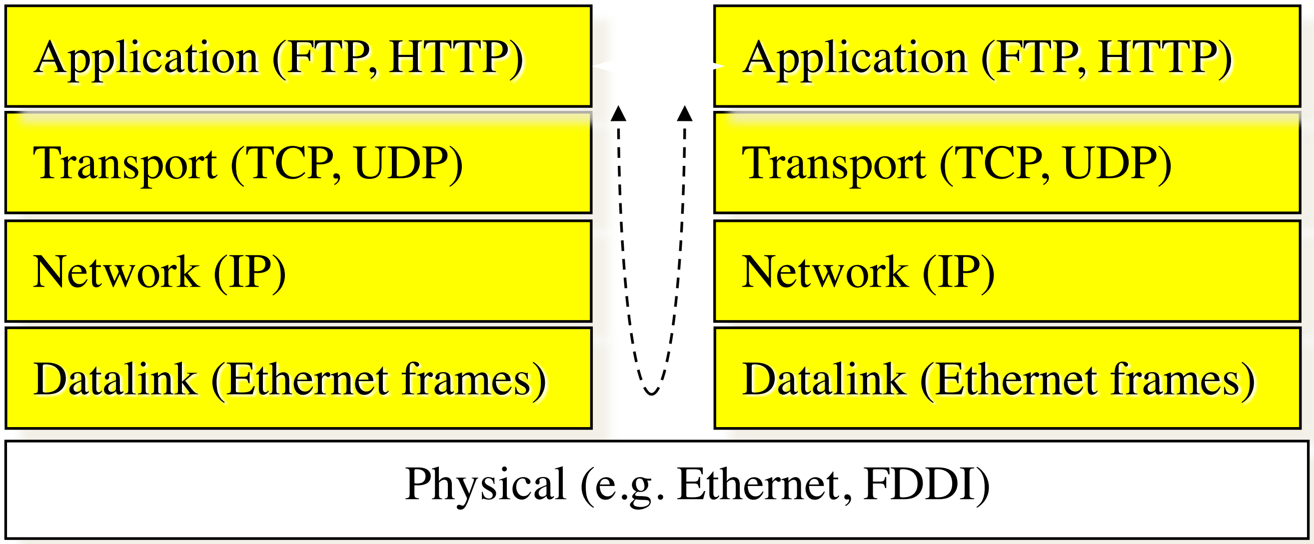
100 m^3 or five million BlueRay 50Gbyte disks, delivered in 6 h, two trucks every day

10 Gbit/s

Multi-Layered Network Architecture

The seven-layer OSI (Open System Interconnect) model

The IP networking stack includes 5 layers



Communication layers

Application	the end product
Presentation	encoding of information, serialization, marshaling
Session	security, authentication, initialization
Transport	messages, streams, reliability, flow control
Network	addressing of nodes in a network, routing, switching
Data link	point to point deliver of frames, medium access, link control
Physical layer	bits to analog signals, electrical, optical, radio ...

Internet stack

HTTP, FTP, SMTP

TCP, UDP, SCTP, ICMP

IP, ARP

Ethernet, WiFi, ..

What if

What would the world look like ...

.. if we only had Ethernet?

Routing

Two approaches:

- Distance vector: send routing table to neighbors, RIP, BGP
- Link state: tell everyone about your direct links, OSPF

Pros and cons?

IP addresses

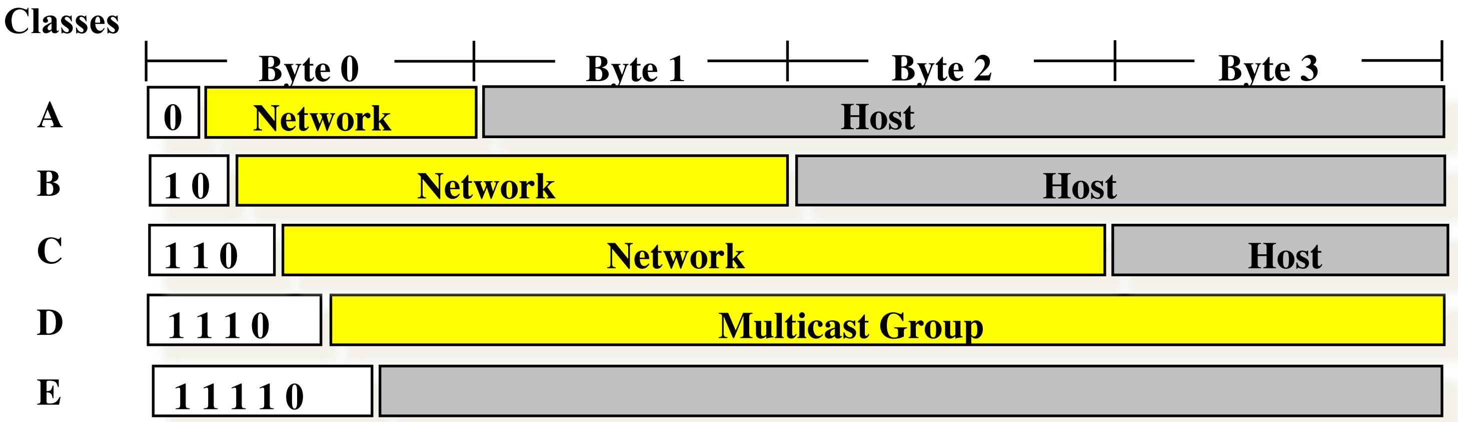
What is the structure of an IP address?

How would you allocate IP addresses to make routing easier?

What is happening?

IP Address Classes (Classful addressing)

A (1-126.x.x.x) – 126 address blocks, each of 16,000,000 addresses.
B (128-191.x.x.x) – one address block contains ~65,000 addresses.
C (192-223.x.x.x) – one address block contains 254 addresses.
D (224-239.x.x.x) – multicast addresses.
E (240-255.x.x.x) –reserved.



Classful Routing and Classless Routing

Recently, the classful addressing with five classes A-E in IPv4 has become obsolete and replaced with classless addressing.

- to tackle the problem of waste/lack of IP addresses

Classful routing: an address is divided into three parts: Network, Subnet, and Host

Classless routing: an address is divided into two parts: Subnet and Host

UDP and TCP

One word that describes the difference between UDP and TCP.

UDP and TCP

Introduces two communication abstractions:

- UDP: datagram
- TCP: stream

- Gives us port numbers to address processes on a node.
- About hundred other protocols defined using IP. (ICMP, IGMP, RSVP, SCTP...)
- More protocols defined on top of UDP and TCP.

UDP

- A datagram abstraction, independent messages, limited in size.
- Low cost, no set up or tear down phase.
- No acknowledgment.

TCP

- A duplex stream abstraction.
- Reliability, lost or erroneous packets are retransmitted.
- Flow control to prevent the sender from flooding the receiver.
- Congestion friendly, slows down if a router is choked.

UDP and TCP

- UDP: small size messages, build your streams
- TCP: large size messages, flow control of a stream of messages

Can you trust TCP delivery?

Sockets

A socket is the programmer's abstraction of the network layer

– an endpoint of a virtual network connection;

– identified by an IP address & port number, and a transport protocol (TCP, UDP, …)

- Datagram sockets for messages (UDP)
- Stream sockets for duplex byte streams (TCP)

Sockets, a.k.a. Berkeley sockets, were introduced in 1981 as the Unix BSD 4.2 generic API for inter-process communication.

- Earlier, a part of the kernel (BSD Unix)
- Now, a library (Solaris, MS-DOS, Windows, OS/2, MacOS)

Stream Socket

A TCP socket for stream-based communication

- Server
  - Creates a listening socket bound to a port (could be in several steps: create, bind, listen)
  - Accepts an incoming connection request and creates a communication socket for reading/writing a byte stream.
- Client
  - Creates a communication socket and connects it to a server identified by an IP address and a port.
  - Reads/writes from a socket.

A Server in Erlang

```
init(Port) ->
   case gen_tcp:listen(Port, [..]) of
      {ok, Listen} ->
         handler(Listen),
         gen_tcp:close(Listen);
      {error, Error} ->
         error
   end.
```

```
handler(Listen) ->
   case gen_tcp:accept(Listen) of
      {ok, Client} ->
         request(Client),
         handler(Listen);
      {error, Error} ->
         error
   end.
```

A Server in Erlang

```
request(Client) ->
   case gen_tcp:recv(Client, 0) of
      {ok, Request} ->
         Response = reply(Request),
         gen_tcp:send(Client, Response);
      {error, Error} ->
         error
      end,
   gen_tcp:close(Client).
```

```
reply(Request) ->
   :
   generate and return
   a byte sequence
```

Datagram socket

- Server
  - Create a message socket and bind it to a port.
  - Receive an incoming message (message contains a source IP address and port number).

- Client
  - Create a message socket bound to a source port.
  - Create a message and give it a destination address and port number.
  - Send the message.

Marshaling of data

How do we transform internal data structure into the sequencing of bytes?

- Language dependent: Java serialization, Erlang external term format
- Independent: XML, Google Protocol Buffer, ASN.1
  - message format defined by specification: XML Schema, .proto, ...
  - A compiler uses the specification to generate an encoder and decoder

Example

ASN.1 specification

```
FooProtocol DEFINITIONS ::= BEGIN
   FooQuestion ::= SEQUENCE {
      trackingNumber INTEGER,
      question IA5String}
   FooAnswer ::= SEQUENCE {
      questionNumber INTEGER,
      answer BOOLEAN}
END
```

C data structures

```
struct foo_question {
         int tracking_number;
         char question[128];
}

foo = {5, "Anybody there?"};
```

Summary

In a perfect world, the application layer should be independent of underlying layers.

The world is not perfect.

Understanding underlying network characteristics is essential when developing distributed applications.

ID2201 Distributed Systems

The lecture continues 16:15