Computer Networks - Summary

Patrick Pletscher

September 13, 2004

1 Introduction

1.1 Overview

Internet standards

RFC Request for comments

IETF Internet engineering Task force

What's a protocol?

Definition 1 (Protocol) *Protocols define format, order of msgs sent and received among network entities, and actions taken on msg transmisson, receipt.*

Circuit Switching

End-end resources reserved for "call". Bandwith link is divided into "pieces":

- Frequency division
- Time division

performance is guaranteed, "piece" can be idle and call setup is required.

Packet switching

Each end-end data stream divided into packets, this packets share network resources and each packet uses full link bandwith.

Good: resource sharing, no call setup
Bad: overhead, congestion control and protocols needed

2 different routing approaches:

- datagram network (destination address determines next hop)
- virtual circuit network (each packet carries tag, which determines next hop)

Delay in packet switched networks

- **Transmission delay**
  R: link bandwith (bps)
  L: packet length (bits)
  $$t_T = \frac{L}{R}$$
- **Propagation delay**
  d: length of physical link
  s: propagation speed ($\sim 2 \times 10^8\ m/s$)
  $$t_P = \frac{d}{s}$$
- **Queuing delay** a: average arrival rate (packets per second)
  Arrival rate $\lambda = L \cdot a$ (bps)
  Service rate $\mu = R$ (bps)
  Traffic intensity: $\rho = \lambda / \mu$
  if $\rho \geq 1$: average delay grows infinitely

WLAN

802.11b: 11 Mbps
802.11a: 54 Mbps

Internet protocol stack

- application (FTP, SMTP)
- transport (TCP, UDP)
- network (IP, routing)
- link (data transfer between neighboring network elements, ex: PPP, Ethernet)
- physical

ISO/OSI Reference Model

7 layers instead:
Applications, Presentation, Session, Transport, Network, Data Link, Physical

2 Applications

2.1 Internet transport protocols services

TCP

- connection-oriented: setup required between client, server
- reliable transport between sending an receiving process
- flow control: sender won't overwhelm receiver

- congestion control: throttle sender when network overloaded
- does not provide timing or minimum bandwith guarantees

UDP

- unreliable data transfer
- does not provide connection setup, reliability, flow control, congestion control, timing or bandwith guarantee

2.2 The Web

HTTP protocol

1. client initiates TCP connection (creates socket) to server, port 80
2. server accepts TCP connection from client
3. HTTP messages exchanged between browser and Web server
4. TCP connection closed

⇒ *HTTP is "stateless"* (server maintains no information about past client requests)

Non-persistent vs. persistent connections

Non-persistent:
Used in HTTP/1.0. Server parses request, responds, closes TCP connection.

Persistent:
Used in HTTP/1.1. On same TCP connection: server parses request, responds, parses new request,...

HTTP message format: Request

```
Request = Req_command {Header_lines} crlf
Req_command = "GET" | "POST" | "HEAD"
              sp Page Version crlf
Page = string
Version = "HTTP/1.0" | "HTTP/1.1"
Header_lines = string ":" sp string crlf
```

HTTP message format: Response

```
Response = Status_line {Header_lines}
           Data crlf crlf
Status_line = Version sp Statuscode
Statuscode = integer sp string
```

Examples of status codes:

Code	Meaning
1xx	Information
2xx	Success
3xx	Redirection
4xx	Client error
5xx	Server error

2.3 FTP

seperate control and data connection

- FTP client contacts FTP server at port 21, specifying TCP as transport protocol
- two parallel TCP connections opened (control, data)
- FTP server maintains "state": current directory, earlier authentication

2.4 Email

user agent sends email via SMTP to "his" server. This server sends the email to the server which is responsable for the To-Email address (also via SMTP).

SMTP

- uses TCP to reliably transfer email message from client to server, on port 25
- direct transfer: sending server to receiving server
- three phases of transfer (handshake, transfer of message)
- command/response interaction
- RFC 821
- SMTP uses persistent connections

Mail message format

A mail consists of the following parts:

- **header lines** (e.g. To:, From:, Subject:), but this are not SMTP commands, like the header of a letter, whereas SMTP commands are like the address on the envelope.
- blank line
- **body** the "message", ASCII chars only

MIME

MIME: multimedia mail extension, RFC 2045, 2056. Additional lines in message header are used to declare MIME content type.

```
[...]
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

ASCII-Message
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data ...
... base64 encoded data
--98766789
```

Mail access protocols

For the retrieval from server you can use: POP, IMAP or HTTP

2.5 DNS: Domain Name System

A map from names to IP addresses. It is implemented in hierarchy of many name servers (distributed database).

- local name servers
  - each ISP, company has local (default) name server
  - host DNS query first goes to local name server
- authoritative name server
  - for a host: stores that host's IP address, name
  - can perform name/address translation for that host's name

DNS Iterated Query

Contacted server replies with name of server to contact ("I don't know this name, but ask this server"). By contrast the *Recursive query* puts burden of name resolution on contacted name server.

DNS resource records

RR format: `(name, ttl, class, type, value)`

- Type = A
  - `name` is hostname
  - `value` is IP address
- Type = NS
  - `name` is a domain
  - `value` is IP address of authorative name server for this domain
- Type = CNAME
  - `name` is alias name for some "canonical" (the real) name
  - `value` is IP canonical name
- Type = MX
  - `value` is name of mail server associated with `name`

DNS protocol, messages

identification	flags
number of questions	number of answer RRs
number of authority RRs	number of additional RRs
questions (variable number of questions)	
answers (variable number of resource records)	
authority (variable number of resource records)	
additional information (variable number of resource records)	

In the 16 bit identification field is a number for the query stored, reply to sender uses same number. The flags field has the following flags: query or reply, recursion desired, recursion available, reply is authoritive.

2.6 Socket programming

Definition 2 (Socket) *a host-local, application-created/owned, OS-controlled interface (a "door") into which application process can both send and receive messages to/from another (remote or local) application process.*

TCP client

```
/*
   send line to server and receive
   modified line
*/
import java.io.*;
import java.net.*;

class TCPClient {
  public static void main(String argv[]) throws
   Exception
  {
    String sentence;
    String modifiedSentence;

    BufferedReader inFromUser =
     new BufferedReader(new InputStreamReader
     (System.in));

    Socket clientSocket =
     new Socket("hostname", 6789);

    DataOutputStream outToServer =
     new DataOutputStream(clientSocket.
     getOutputStream());

    BufferedReader inFromServer =
     new BufferedReader(new
     InputStreamReader(
     clientSocket.getInputStream());

    sentence = inFromUser.readLine();

    outToServer.writeBytes(sentence + '\n');

    modifiedSentence = inFromServer.readLine();

    System.out.println("FROM SERVER: "
     + modifiedSentence);

    clientSocket.close;
  }
}
```

TCP server

```
/*
```

```
   receive a line a send it back, but
   uppercase
*/

import java.io.*;
import java.net.*;

class TCPServer {
  public static void main(String argv[]) throws
   Exception
  {
    String clientSentence;
    String capitilizedSentence;

    ServerSocket welcomeSocket =
     new ServerSocket(6789);

    while(true) {
      Socket connectionSocket =
       welcomeSocket.accept();
      ServerThread thread =
       new ServerThread(connectionSocket);
      thread.run();
    }
  }

  public class ServerThread extends Thread {

    private Socket connectionSocket;

    public ServerThread(Socket connectionSocket)
    {
      this.connectionSocket = connectionSocket;
    }

    public void run() {
      BufferedReader inFromClient =
       new BufferedReader(new
       InputStreamReader(
       connectionSocket.getInputStream()));

      DataOutputStream outToClient =
       new DataOutputStream(
       connectionSocket.getOutputStream());

      clientSentence = inFromClient.readLine();

      capitalizedSentence =
       clientSentence.toUpperCase() + '\n';

      outToClient.writeBytes(
       capitalizedSentence);
    }
  }
}
```

UDP client

```
import java.io.*;
import java.net.*;

class UDPClient {
  public static void main(String args[]) throws
   Exception
  {
    BufferedReader inFromUser =
     new BufferedReader(new
     InputStreamReader(System.in));

    DatagramSocket clientSocket =
     new DatagramSocket();

    InetAddress IPAddress =
     InetAddress.getByName("hostname");

    byte[] sendData = new byte[1024];
    byte[] receiveData = new byte[1024];

    String sentence = inFromUser.readLine();

    sendData = sentence.getBytes();

    DatagramPacket sendPacket =
     new DatagramPacket(sendData, sendData.length,
     IPAddress, 9876);

    clientSocket.send(sendPacket);

    DatagramPacket receivePacket =
     new DatagramPacket(receiveData,
     receiveData.length);

    clientSocket.receive(receivePacket);

    String modifiedSentence =
     new String(receivePacket.getData());

    System.out.println("FROM SERVER:"
     + modifiedSentence);
    clientSocket.close();
  }
}
```

UDP server

```
import java.io.*;
import java.net.*;

class UDPServer {
  public static void main(String args[]) throws
   Exception
  {
    DatagramSocket serverSocket =
     new DatagramSocket(9876);

    byte[] receiveData = new byte[1024];
    byte[] sendData = new byte[1024];
```

```
    while(true)
    {
      DatgramPacket rececivePacket =
       new DatagramPacket(receiveData,
       receiveData.length);

      serverSocket.receive(receivePacket);

      String sentence =
       new String(receivePacket.getData());

      InetAddress IPAddress =
       receivePacket.getAddress();

      int port = receivePacket.getPort();

      String capitalizedSentence =
       sentence.toUpperCase();

      sendData =
       capitalizedSentence.getBytes();

      DatagramPacket sendPacket =
       new DatagramPacket(sendData,
       sendData.length, IPAddress,
       port);

      serverSocket.send(sendPacket);
    }
  }
}
```

3 Transport

3.1 Multiplexing/Demultiplexing

Definition 3 (Demultiplexing) *Delivering received segments to correct application layer processes.*

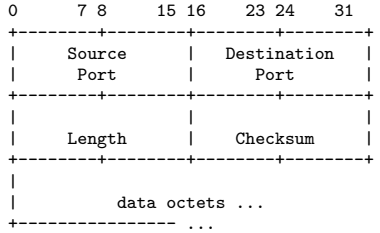
Definition 4 (Multiplexing) *Gathering data from multiple application processes, enveloping data with header (later used for demultiplexing).*

multiplexing/demultiplexing is based on sender, receiver port numbers, IP addresses.

3.2 UDP: User Datagram Protocol

UDP Segment Structure

Header fields are 16 bits.



length, in bytes of UDP segment, including header. The checksum is calculated by adding the segment contents, whereas treating segment contents as sequence of 16-bit integers. If there is a carry, forget the carry and add 1 to the result.

$$105C8 \rightarrow 05C9$$

After that the 1's complement of it (e.g. $110 \Rightarrow 001$) is calculated and the result is saved in the UDP checksum field. To detect errors the receiver can calculate the sum by adding all 16-bit integers (including the checksum field). If the result is $11\ldots1$ then no errors are detected, otherwise there occured an error.

3.3 Reliable data transfer

[TODO]

stop-and-wait operation

send one packet and wait until ACK'ed.

$$\text{Utilization } U = \frac{L/R}{RTT + L/R}$$

RTT: Round trip time (time to receiver and back)

Pipelined protocols

Pipeling: sender allows multiple, "in-flight", yet-to-be-ack'ed packets. So the range of seq numbers must be increased and buffering at sender and/or receiver is required.

There are two generic forms of pipelined protocols: Go-Back-N and Selective Repeat.

Go-Back-N

ACKs all packets up to and including sequence number n (= cumulative ACK).

- Sender
  - "Window" of up to N consecutive unack'ed packets allowed
  - timer for each in-flight packet
  - timeout(n): retransmit packet n and all higher seq# packets in window
- Receiver
  - ACK-only: always send ACK for correctly-received pkt with highest *in-order* sequence number, this may generate duplicate ACKs.
  - out-of-order packet are discard (no receiver buffering) and packet with highest in-order sequence number is re-ACK'ed.

Selective Repeat

Sender

- Get data from layer above: If next available sequence number in window, send packet.
- timeout(n): resend packet n, restart timer.
- ACK(n) in [sendbase, sendbase + N]: mark packet n as received. If n smallest unACKed pkt, advance window base to next unACKed sequence number.

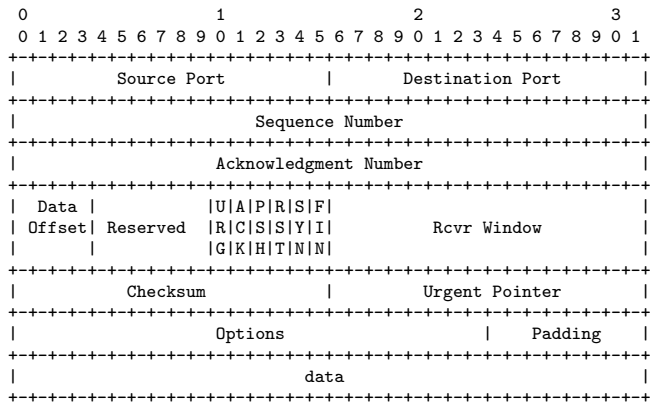
Receiver

- pkt n in [rcvbase, rcvbase + $N - 1$]: send ACK(n), if it is out of order: buffer, otherwise: deliver (also buffered in-order pkt) and advance window to next not-yet-received packet.
- pkt n in [rcvbase-N, rcvbase-1]: ACK(n)
- otherwise: ignore

3.4 TCP

Is pipelined (send & receive buffers).

TCP segment structure



The smallest fields in the table above are 16-bit. The counting (for the *seq number* and *ack number*) is done by bytes of data. The *Rcvr Window* holds the # of bytes the rcvr is willing to accept (flow control). The *checksum* is calculated like in UDP. In the FLAG field the following information is stored:

- *header length* tells how many 32-bit words are contained in the TCP header. (4 bits)
- *not used*: 6 bits
- *Urgent pointer bit* is used to indicate a byte offset from the current sequence number at which urgent data are to be found (generally not used)
- *ACK bit*: is set to 1 to indicate that ACK number is valid
- *Push bit*: The receiver is requested to deliver the data to the application upon arrival and not to buffer it.
- *Reset connection bit*
- *SYN bit*: used to establish connections. The connection request has SYN=1 and ACK=0.
- *FIN bit*: used to realease a connection

TCP sequence numbers and ACKs

Sequence numbers: byte stream "number" of first byte in segment's data.
ACKs: Sequence number of next byte expected from other side. Cumulative ACK.

TCP Round Trip Time and Timeout

$$\text{EstimatedRTT} = (1-\alpha)\cdot \text{Estimated RTT} + \alpha \cdot \text{SampleRTT}$$

This is a so called Exponential weighted moving average, a typical value: $\alpha = 0.125$.

Setting the timeout:

$$\begin{aligned} \text{Timeout} &= \text{EstimatedRTT} + 4\cdot \text{Deviation} \\ \text{Deviation} &= (1-\beta)\cdot \text{Deviation} \\ &\quad +\beta \cdot |\text{SampleRTT} - \text{EstimatedRTT}| \end{aligned}$$

TCP: Sender

```
NextSeqNum = InitialSeqNum
SendBase = InitialSeqNum

loop(forever){
  switch(event)

  event: data received from application above
    create TCP segment with sequence
     number NextSeqNum
    if (timer currently not running)
      start timer
    pass segment to IP
    NextSeqNum = NextSeqNum + length(data)

  event: timer timeout
    retransmit not-yet-ack'ed segment with
     smallest sequence number
    start timer

  event: ACK received, with ACK field value of y
    if (y > SendBase){
      SendBase = y
      if (there are currently not-yet-ack'ed
       segments)
        start timer
    }
}
```

TCP ACK generation

Event	TCP Receiver action
in-order segment arrival, no gaps, everything else already ACKed	delayed ACK. Wait up to 500ms for next segment. If no segment, send ACK
in-order segment arrival, no gaps, one delayed ACK pending	immediately send single cumulative ACK, ACKing both in-order segments
out-of-order segment arrival higher-than-expect seq #, gap detected	send duplicate ACK, indicating seq. # of next expected byte
arrival of segment that partially or completely fills gap	immediate ACK if segment starts at lower end of gap

Fast Retransmit

Hack: If sender receives 3 ACKs for the same data, it supposes that segment after ACKed data was lost.

TCP Flow Control

Definition 5 (Flow control) *sender won't overrun receiver's buffers by transmitting too much, too fast.*

`RcvBuffer`: size of TCP Receive Buffer
`RcvWindow`: amount of spare room in Buffer

Receiver: explicitly informs sender of (dynamically changing) amount of free buffer space, for that the `RcvWindow` field in TCP segment is used.

Sender: keeps the amount of transmitted, unACKed data less than most recently received `RcvWindow`

TCP Connection Management

- Opening connection
  1. client host sends TCP SYN segment to server. This specifies initial seq. number and contains no data.
  2. server host receives SYN, replies with SYNACK segment. The server will allocate buffers and specifies server initial seq. #.
  3. client receives SYNACK, replies with ACK segment, which may contain data.
- Closing connection
  1. client end system sends TCP FIN control segment to server
  2. server receives FIN, replies with ACK. Goes into CLOSEWAIT state, sends FIN.
  3. client receives FIN, replies with ACK. Enters "timed wait" - will respond with ACK to FINs
  4. server receives ACK. Connection closed.

3.5 Principles of congestion control and Queuing theory

Definition 6 (Congestion) *Too many sources sending too much data too fast for network to handle.*

Some terms

- Each costumer spends T seconds in the box.
- We assume that the system was empty at time $t = 0$.
- $A(t)$: Number of arrivals from time $t = 0$ to time t.
- $D(t)$: Number of departures
- $N(t)$: number of costumers in the system at time t. $N(t) = A(t) - D(t)$
- Throughput: average number of costumers/messages per second that pass through the system.
- Arrival Process: a_1: 1^{st} arrival in system. 2^{nd} comes a_2 times later. So the n^{th} customer comes at time $a_1 + a_2 + \ldots + a_n$.

Arrivals, Departures, Throughput

The long-term arrival rate λ is defined as:

$$\lambda = \lim_{t\to\infty} \frac{A(t)}{t} \quad \text{cust./sec}$$

And the throughput μ:

$$\mu = \lim_{t\to\infty} \frac{D(t)}{t} \quad \text{cust./sec}$$

the average service time is $1/\mu$.

Offered Load (or Traffic Intensity)

Offered load is defined as:

$$\rho = \frac{\lambda}{\mu}$$

The system is stable if $\rho < 1$.

Little's Law

$E[N]$: average number of customers
$E[T]$: average time spent in system

$$E[N] = \lambda \cdot E[T]$$

Binomial Random Variables & Poisson RV

Suppose we had n trials. Then for a series of trials, a binomial RV with parameter (n, p) is the probability of having exactly i arrivals out of n trials with independent arrival probability p:

$$P[X = i] = \binom{n}{i} p^i (1-p)^{n-i}$$

Binomial RV can be approximated with Poisson RV. With $\lambda_p = np$, the distribution of a Poisson RV is

$$P[X = i] = e^{-\lambda_p} \frac{\lambda_p^i}{i!}$$

The mean is λ_p.

The number of events occuring in any fixed interval of length t is

$$P[N(t) = k] = e^{-\lambda t} \frac{(\lambda t)^k}{k!}$$

Exponential RV / Memoryless Property

Modeling of the time between occurence of events, it satifies the "memoryless property".

The probability of having to wait at least h seconds is

$$P[X > h] = e^{-\lambda h}$$

Kendall Notation

Queuing systems are classified by a specific notation denoting:

1. The customer arrival pattern
2. The service time distribution
3. The number of servers
4. The maximum number of customers in the system (std. $= \infty$)
5. Calling population (std. $= \infty$)
6. Queuing discipline (FIFO, LIFO, etc.; std. $= \infty$)

1 and 2 can be either: M = Markov (Poisson or Exponential), D = Deterministic, E_k = Erlang with param. k, G = General.

M/M/1 Queue

The probability that an M/M/1 system is not idle is ρ.

$$E[A(t)] = \lambda t \text{ and } E[D(t)] = \mu t$$

Markovian Chains are used to describe such models. The equilibrium must balance:

$$(\lambda p_i)t = (\mu p_{i+1})t \rightarrow \rho p_i = p_{i+1}$$

In the equilibrium, the number of customers in the system is

$$E[N] = \frac{\rho}{1 - \rho}$$

and the mean time in the system

$$E[T] = \frac{1}{\mu(1 - \rho)}$$

Approaches towards congestion control

Two types of approaches usually used:

- **End-end congestion control**
  - no explicit feedback about congestion from network
  - congestion inferred from end-system observed loss, delay
  - approach taken by TCP
- **Network-assisted cong. control**
  routers provide feedback to end systems
  - single bit indicating congestion (used in SNA, DECbit, TCP/IP ECN, ATM).
  - explicit rate sender should send at.

Example for Network-Assisted Cong. Control: ATM ABR

ABR: available bit rate (so called "elastic service"). If sender's path "underloaded", the sender should use available bandwith and if sender's path congested, the sender is throttled to minimum guaranteed rate.

RM (resource management cells). This cells were sent by the sender interspersed with data cells. The following bits in RM cell were set by switches ("network assisted"): NI bit (no increase in rate = mild congestion) and CI bit (congestion indication). This RM cells were returned to sender by receiver.

There is also a two-byte ER (explicit rate) field in RM cell, a congested switch may lower this ER value in cell and the sender's rate is the minimum supportable rate on path.

EFCI bit in data cells: set to 1 in congested switch. If data cell preceding RM cell has EFCI set, sender sets CI bit in returned RM cell.

TCP Congestion Control

An end-end control (no network assistance) approach. The transmission rate is limited by congestion window size, `Congwin`, over segments.

w segments, each with MSS bytes sent in one RTT:

$$\text{throughput} = \frac{w \cdot MSS}{RTT} \quad \text{Bytes/sec}$$

This TCP congestion control works by "probing" for usable bandwith. Ideally it is transmitted as fast as possible (=`Congwin` as large as possible) without loss. The `Congwin` is increased until loss. If loss occurs the `Congwin` is decreased and then the probing restarts.

This method can be divided into two "phases": slow start and congestion avoidance. An important variable is the `Treshold` which defines where TCP switches from slow start to congestion avoidance.

TCP Slowstart

```
initialize: Congwin = 1
for (each segment ACKed)
  Congwin++
until (loss event OR Congwin > Threshold)
```

The `Congwin` is exponential increased (per RTT).

TCP Congestion Avoidance

```
/* slowstart is over */
/* Congwin > Threshold */
Repeat {
  w = Congwin
  every w segments ACKed:
    Congwin++
} until (loss event)
threshold = Congwin/2
Congwin = 1
Go back to slowstart
```

TCP Fairness

Definition 7 (Max-Min Fairness) *A set of flows is max-min fair if and only if no flow can be increased without decreasing a smaller or equal flow.*

How do we calculate a max-min fair distribution?

1. Find a bottleneck resource r (router or link), that is, find a resource where the resource capacity c_r divided by the number of flows that use the resource (k_r) is minimal.
2. Assign each flow using resource r the bandwith c_r/k_r.
3. Remove the k flows from the problem and reduce the capacity of the other resources they use accordingly.
4. If not finished, go back to step 1.

Is TCP Fair? Yes and no. TCP has an additive increase, multipliticative (AIMD) congestion control algorithm what's good.

TCP latency modeling

W: fixed congestion window with W segments
S: MSS (bits)
O: Object size (bits)

Here it is assumed that no loss and no retransmission occur.

- $WS/R > RTT + S/R$:

$$\text{latency} = 2 \cdot RTT + O/R$$

- $WS/R < RTT + S/R$:

$$\begin{aligned} \text{latency} &= 2 \cdot RTT + O/R \\ &\quad + (K-1)[S/R + RTT - WS/R], \\ &\quad \text{with } K = O/WS \end{aligned}$$

4 Network Layer

4.1 Virtual circuits and Datagram networks

Virtual circuits need a call setup on the network layer while Datagram networks don't need a such. In today's Internet Datagram networks are used.

4.2 Routing

Definition 8 (Routing protocol) *Goal: determine "good" path (sequence of routers) through network from source to destination.*

Routing Algorithm classification

Global or decentralized?

- Global
  - all routers have complete topology, link cost info.
  - "link state" algorithms
- Decentralized
  - router knows physically-connected neighbors, link costs to neighbors
  - iterative process of computation, exchange of info with neighbors
  - "distance vector" algorithms

Static or Dynamic?

- Static
  - routes change slowly over time
- Dynamic
  - routes change more quickly. Periodic link update in response to link cost changes.

Single Source Shortest Path: Algorithm idea

There are 3 groups of nodes in the network

- To the green nodes we know the shortest path
- The blue nodes are directly reachable from the green nodes
- All other nodes are black

Idea

- Start with source s as the only green node.
- Color the best blue node green, one after another, until all nodes are green.

Dijkstra's Algorithm (for source s and edge costs c)

```
s.visited := true; s.distance := 0;
s.pred := s; // init source s

for all nodes v in V\s do // init all other nodes
  v.visited := false; v.distance := infty;
  v.pred := undefined;

B := {} // B is the set of blue nodes
for all nodes v in V\s that are direct neighbors
 of s
  B := B + {v}; v.distance := c(s,v);
  v.pred := s;

while B not empty do // always choose best
 // blue node v
  v := node in B with minimum v.distance;
  B := B - {v};
  v.visited := true;
  for all neighbors w of v with
   w.visited = false; // update neighbors of v
    if w not in B then
      B := B + {w};
      w.distance := v+c(v,w); w.pred := v;
    if w in B then
      if (v.distance+c(v,w) < w.distance) then
        w.distance := v.distance+c(v,w);
w.pred := v;
endwhile
```

Dijkstra's algorithm complexity

n nodes, m directed edges.

With a Fibonacci-Heap, one can implement the whole algorithm in $O(m + n \log n)$.

4.3 Distance-Vector Routing

Each node communicates only with direct neighbors, and "asks" them how long it takes from them to a destination.

From Distance Vector to Routing table

A node x has for each neighbor z an entry in distance vector for each destination y; $D^x(y, z)$ denotes the distance from x to y through z. The best route for a given destination is marked by

$$\begin{aligned} D^x(y) &= \min D^x(y, z) \\ D^x(y, z) &= c(x, z) + D^x(y) \end{aligned}$$

Each node notifies neighbors *only* when its least cost path to any destination changes. But this can lead to the famous Count-to-infinity problem.

Hierarchical Routing

Aggregate routers into groups, "autonomous systems" (AS). Intra-AS and Inter-AS.

4.4 The Internet Network Layer

Different protocols: Routing protocols, IP protocol, ICMP protocol.

IP Addressing: CIDR

Classless InterDomain Routing. The network portion of address is of arbitary length. The address format is: $a.b.c.d/x$ where x denotes the number of bits in network portion of address.

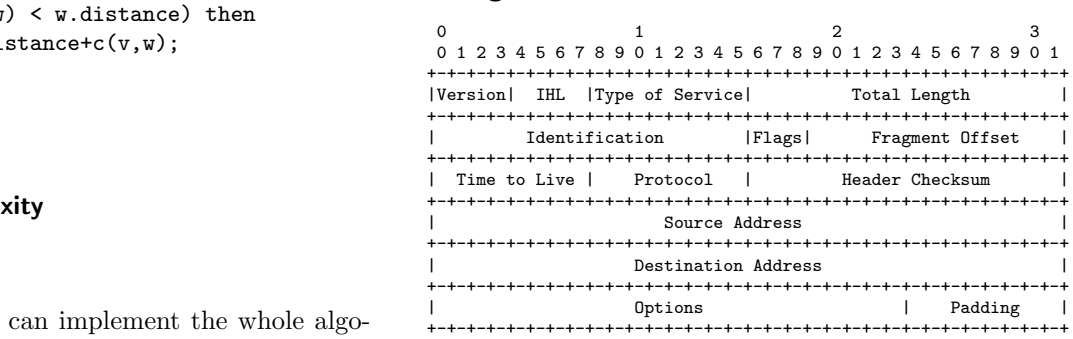
Example:

$$\underbrace{11001000\ 00010111\ 0001000}_{\text{network part}}\underbrace{000000000}_{\text{host part}} = 200.23.16.0/23$$

Getting a datagram from source to destination

If destination is in different network: look in routing table for next hop to destination, link layer sends it to this hop.

IP datagram format

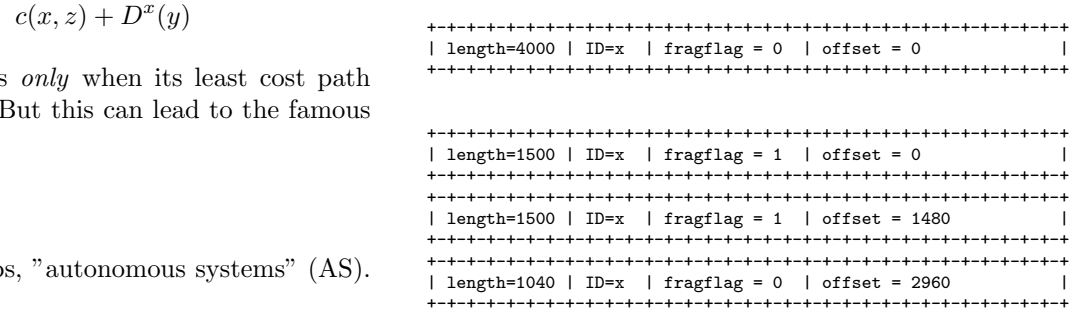


IHL: Header length in bytes.
Total Length: total datagram length (bytes).
the second line is used for fragmentation, reassembly
Protocol: is the protocol on the upper layer (e.g. TCP)
The checksum is calculated like in UDP and TCP.

IP Fragmentation and Reassembly

Network links have MTU (max. transmission unit), which denotes the largest possible link-level frame. Large IP datagrams are divided ("fragmented") within net if it is necessary. They were only "reassembled" at final destination.

Example:



4.5 ICMP: Internet Control Message Protocol

used by hosts, routers, gateways to communicate network-level information. It is an network-layer "above" IP, ICMP msgs carried in IP datagrams.

4.6 DHCP

Client sends "DHCP discover" to broadcast (255.255.255.255) (src-port 68, dest-port 67), "DHCP offer" back from server to broadcast (src-port 67, dest-port 68) here the IP address x is offered, "DHCP request" from host to broadcast (src-port 68, dest-port 67), "DHCP ack" from server to host with IP address x.

4.7 NAT: Network Address Translation

outgoing datagrams: replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #), the remote clients will respond using (NAT IP address, new port #) as destination addr.

remember: (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair.

incoming datagrams: replace (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table.

4.8 Intra-AS Routing

A.k.a Interior Gateway Protocols (IGP).

Most common IGPs: RIP (= Routing Information Protocol), OSPF (Open Shortest Path First), IGRP (Interior Gateway Routing Protocol).

RIP

Distance vector algorithm. Distance metric: number of hops. The distance vectors are exchanged every 30 sec via "advertisement". In each advertisement: route to up to 25 destination networks within AS.

If no advertisement heard after 180 sec → neighbor/link declared dead.

- routes via neighbor invalidated
- new advertisements sent to neighbors
- poison reverse used to prevent ping-pong loops. If Z routes through Y to get to X: Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z).

RIP Table processing

RIP routing tables managed by *application-level* process called route-d. Advertisements sent in UDP packets, periodically repeated.

4.9 OSPF (Open Shortest Path First)

Uses Link State algorithm, route computation using Dijkstra's algorithm, LS packet dissemination (=Ausstreuung).

Security: all OSPF messages authenticated, so no malicious intrusion. TCP connections used.

4.10 Inter-AS routing

BGP

A path vector protocol. Messages exchanged using TCP.

1. Receive BGP update (announce or withdrawal) from a neighbor.
2. Update routing table
3. Does update affect active route? (Loop detection with information about the whole path, which neighbor sends; policy). If yes, send update to all neighbors that are allowed by policy.

BGP convergence

If a link *comes up*, the convergence time is in the order of time to forward a message on the *shortest path*.

If a link *goes down*, the convergence time is in the order of time to forward a message on the *longest path*.

A "cause tag" attachted to the withdrawal message identifying the failed link/node reduces the convergence time to the shortest path.

4.11 Router Architecture Overview

Two key router functions:

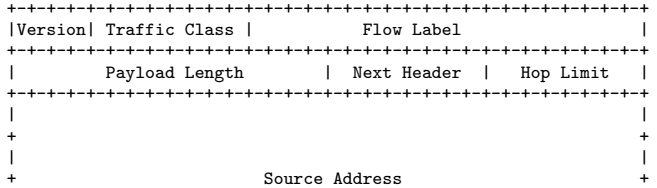
- run routing algorithms/protocols
- switch datagrams from incoming to outgoing link.

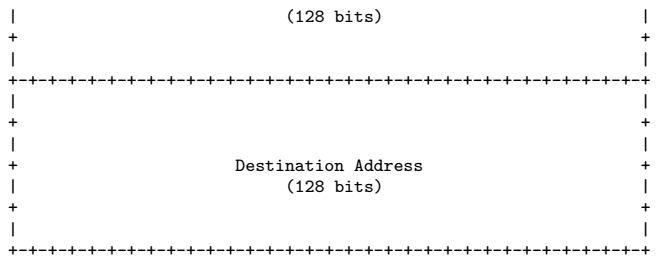
Input Port Functions

Decentralized switching

- given datagram dest., lookup output port using routing table in input port memory
- goal: complete input port processing at "line speed"
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

4.12 IPv6





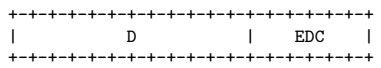
Next header: identify upper layer protocol for data.

5 Link layer

5.1 Implementation

Link layer implemented in "adapter" (a.k.a. NIC).

5.2 Error Detection



EDC : Error Detection and Correction bits (redundancy)
D : Data protected by error checking, may include header fields.

Parity Checking

Only one bit. To detect single bit errors. Is set to 1 if total sum of 1's in data is even, otherwise 0.

Cyclic Redundancy Code

Generator polynomial $G(x) = x^{16} + x^{12} + x^5 + 1$.
Let the whole frame (D+EDC) be polynomial $T(x)$.

Idea: fill EDC (CRC) field such that $T(x) \mod G(x) = 0$.

How to divide with polynomials? Example with $G(x) = x^2 + 1 (= 101)$

```
11101100 / 101 = 110110, Remainder 10
 100
  011
   111
    100
     010
```

The subtraction is an XOR, so there is no add carry. You can only divide if the divisor is of same length.

How to fill EDC: Calculate remainder of $D \parallel \underbrace{0 \ldots 0}_{|G|-1}$ divided by $G(x)$ and afterwards T becomes:

$$D \parallel D \mod G(x)$$

5.3 Multiple Access Links and Protocols

- point-to-point (single wire; e.g. PPP, SLIP)
- broadcast (shared wire or medium; e.g. Ethernet, WLAN)
- switched (e.g. switched Ethernet, ATM)

Channel Partioning: Frequency Division Multiplex (FDM)

Seperation of the whole spectrum into smaller frequency bands. A channel gets a certain band of spectrum for the whole time. Example: broadcast radio.

Channel Partioning: Time Division Multiplex (TDM)

A channel gets the whole spectrum for a certain amount of time. Example: Ethernet.

Channel Partitioning: Time/Frequency Division Multiplex

A channel gets a certain frequency band for some time. Example: GSM.

Channel Partioning: Code Division Multiplexing (CDM)

Each channel has a unique code. All channels use the same spectrum at the same time. Example: UMTS.

5.4 Multiple Access Control (MAC) Protocols

Distributed algorithm that determines how stations share channel, i.e. determine when station can transmit. The communcation about channel sharing must use channel itself.

"Taking Turns" MAC protocols

Polling

- master node "invites" slave nodes to transmit in turn
- Request to Send, Clear to Send messages
- concerns: polling overhead, latency, single point of failure (master)

Token passing (Token Ring)

- control token passed from one node to next sequentially
- token message
- concerns: token overhead, latency, single point of failure (token)

"Taking Turns" Protocols: Round Robin

- Round robin protocol: station k sends after station $k-1 (\mod n)$
- If a station does not need to transmit data, then it sends ε
- There is a maximum message size m that can be transmitted

5.5 Random Access protocols

- When node has packet to send: transmit at full channel data rate R, no a priori coordination among nodes
- Two or more transmtting nodes → "collision"
- Random access MAC protocol specifies how to detect collisions and how to recover from collisions.
- Examples: ALOHA and variants, Backoff protocols (CSMA)

Slotted Aloha

- Time is divided into equal size slots
- Node with new arriving packet: transmit at beginning of next slot
- If collision: retransmit packet in future slots with probability p, until successful

In slotted aloha a station can transmit successfully with probability at least $\frac{1}{e}$.

How quickly can an application send packets to the radio transmission unit?
The probability that a new packet is received by the buffer is λ; the probability that sending succeeds is $\mu = 1/e$, for any time slot. To keep the queue bounded we need $\rho = \lambda/\mu < 1$, thus $\lambda < 1/e$. This is for M/M/1.

Adaptive slotted aloha

Idea: change the access probability with the number of stations.

Idea: Try to estimate the number of stations

- If you see that nobody sends, increase p.
- If you see that more than one sends, decrease p.

Pure ALOHA

Packet needs transmission: send without awaiting for beginning of slot.

The rate is half of the slotted ALOHA.

Demand Assigned Multiple Access (DAMA)

Reservation:

- a sender reserves a future time-slot
- sending within this reserved time-slot is possible without collision
- reservation also causes higher delays
- typical scheme for satellite systems

Backoff Protocol

Backoff protocols rely on ack's only. Binary exponential backoff, for example, works as follows: If a packet has collided k times, we set $p = 2^{-k}$.

For infinitely many stations this protocol isn't stable for $\lambda > 0$, but if there are only finite stations, the protocol is only unstable for $\lambda > 0.568$.

CSMA: Carrier Sense Multiple Access

Idea: listen before transmit. If channel sensed idle: transmit entire packet; if sensed busy, defer transmission. There exist two variants for the defering: Persistant CSMA (retry immediately with probability p when channel becomes idle), Non-persistent CSMA (retry after random interval).

CSMA/CD (Collision Detection)

Collisions were detected within short time and colliding transmissions were aborted, reducing channel wastage. CSMA/CD used in Ethernet.

5.6 LAN Addresses and ARP

MAC address: used to get datagram from one interface to another physically-connected interface (same LAN). Therefore 48 bit MAC addresses burned in the adapter ROM are used.

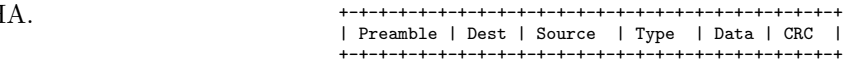
ARP: Address Resolution Protocol. IP-Address → MAC-Address.

Each IP node on LAN has ARP table in which the IP/MAC address mapping for some LAN nodes are stored together with the time after which the address mapping will be forgotten.

A knows B's IP address, wants to learn physical address of B:
A broadcasts ARP query packet, containing B's IP address.
B replies to A with its physical layer address.

5.7 Ethernet

Ethernet Frame Structure



Preamble:
7 bytes with pattern 10101010. Followed by 1 byte with pattern 10101011. They were used to synchronize receiver, sender clock rate.
Addresses:
6 bytes, frame is received by all adapters on a LAN and dropped if address does not match.
Type (2 bytes):
Indicates the higher layer protcol mostly IP (0x0800).
CRC (4 bytes):

checked at receiver, if error is detected, the frame is simply dropped.

Ethernet CSMA/CD algorithm

1. Adapter gets datagram from network layer and creates frame
2. If adapter senses channel idle, it starts to transmit frame. If it senses channel busy, waits until channel idle and then retransmit.
3. If adapter transmits entire frame without detecting another transmission, the adapter is done with frame.
4. If adapter detects another transmission while transmitting, aborts and sends jam signal (48 bit).
5. After aborting, adapter enters exponential backoff: after the m-th collision, adapter chooses a K at random from $0, 1, 2, \ldots, 2^m - 1$. Adapter waits $K \cdot 512$ bit times (e.g. 1/(100 Mbit)) and returns to Step 2.

CSMA/CD efficiency

$$\text{utilization} \approx \frac{1}{1 + 6.2 \cdot t_{prop}/t_{trans}}$$

t_{prop}: max. progagation time between any two nodes in LAN
t_{trans}: time to transmit max-size frame.
Utilization goes to 1 as $t_{prop} \to 0$ or $t_{trans} \to \infty$

Interconnecting with Bridges

A bridge is a link layer device. Stores and forwards Ethernet frames. Examines frame header and selectively forwards frame based on MAC destination address.

6 Peer-To-Peer Computing

6.1 Hashing

Distributed Hashing and Linear Hashing

$$\text{key} \longmapsto .10111010101110011\ldots \approx .73$$

Arrange all hosts on a line from 0 to 1. For all documents 0.101x peer x has stored the forward-pointer to the real file. Problem: if new machines join a lot of objects have to be moved. Linear hashing solves this problem by just moving a few objects to a new machine (about 1/n). It divides the files host y is responsible for to host y and the new host.

Consistent Hashing

Also the machines get hashed (IP + Port). Each machine is responsible for the files closest to it.

Problems

The problem with Linear and Consistent Hashing is, that every machine needs to know all the participants. Number one challange: Dynamics.

6.2 Search Tree

Peer x must only know subset of others (A host in the other subtree of every node you pass when searching after your hash). It sends his search query to this host which has the same prefix as the search hash. This host sends the query further and so on.

Time to search if tree is balanced: $O(\log n)$

Peer Join

Joiner must already know a peer in system (e.g. ping randomly, try some of those you met last time). After that you have to find your place in the P2P system.

Find your place

The random method: Choose a random bit string. Search for the bit string. Split with the current leave responsible for the bit string. Search for your neighbors.

Time to join: 1st part $O(\log n)$, 2nd part $O(\log^2 n)$

Since all peers chose their position randomly, the tree will more or less be balanced.

Leave

A Leave is detected by the neighbors in the P2P system (periodically ping).

If a peer that left was detected, it must be replaced. If peer had sibling leaf, the sibling might just do a "reverse split". If not search recursively:

1. Go down sibling tree, until you hit sibling leaves.
2. Make the left sibling the new common node.
3. Move the free right sibling to the empty spot.

6.3 Chord

Every peer has $\log n$ neighbors; one in about distance $2^{-k}, k = 1, 2, \ldots, \log n$. Imagine a ring.

Skip List

- (Doubly) linked list, with sorted items
- All items have additional pointers on levels $1, \ldots, k$, with probability 2^{-k}
- Search, insert, delete: Start with root, search for the right interval on highest level, then continue with lower levels.

Search, insert and delete: $O(\log n)$.

Skip Net

Use the skip list as a peer-to-peer architecture: Again each peer gets a random value between 0 and 1, and is then responsible for storing that interval. Instead of a root and a sentinel node, the list is short-wired as a ring.