The Transport Layer and Applications

Gralla chapters: 3-4, 17-18, 11-12

nixu

Transport Layer Protocols

- Provide services to applications
  - Network layer (IP) is host to host
  - Transport layer is data transport service from one application to another application
  - Additional addressing to the network layer host addresses
- Transport layer entities talk to each other in the transport layer protocols
  - A TCP software implementation (entity) in the kernel of an operating system talks TCP to another TCP entity
- Provides services to the applications protocols
  - TCP: reliable byte stream delivery
  - UDP: unreliable datagram delivery

nixu

UDP

- UDP = User Datagram Protocol
- Defined in RFC-768
- UDP packet syntax

Source port	Destination port
Length	UDP checksum
Data	

- Port is a 16-bit application identifier number.
- Checksum is calculated over both the header and the data. UDP checksum is optional.

…UDP

- UDP datagram is encapsulated into an IP datagram.

- Unreliable datagram-oriented transportation layer protocol
  - offers little extra functionality besides port numbers
  - simple, fast, light-weight, easy to implement
- Applications using UDP: DNS, Radius, NTP, SNMP, VoIP, streaming media

A UDP (DNS) Session Snoop

```
23 riku@mole $ dig a tapas.nixu.fi @194.197.118.20
;; got answer:
;; QUESTIONS:
;;      tapas.nixu.fi, type = A, class = IN
;; ANSWERS:
tapas.nixu.fi.  3600    A       194.197.118.24
;; AUTHORITY RECORDS:
nixu.fi.        3600    NS      ns2.tele.fi.
nixu.fi.        3600    NS      ns.nixu.fi.
nixu.fi.        3600    NS      ns.tele.fi.
;; ADDITIONAL RECORDS:
ns2.tele.fi.    35619   A       193.210.19.190
ns.nixu.fi.     3600    A       193.209.237.29
ns.tele.fi.     555991  A       193.210.19.19
ns.tele.fi.     555991  A       193.210.18.18
;; Total query time: 88 msec
;; FROM: mole.nixu.fi to SERVER: 194.197.118.20
;; MSG SIZE  sent: 31  rcvd: 175
24 riku@mole $
```

DNS Query, Ethernet Header

```
ETHER:  Packet 1 arrived at 11:19:24.80
ETHER:  Packet size = 73 bytes
ETHER:  Destination = 8:0:20:74:f1:2c
ETHER:  Source      = 0:0:3b:80:e:93
ETHER:  Ethertype = 0800 (IP)
```

DNS Query, IP Header

```
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 59 bytes
IP:   Identification = 35734
IP:   Flags = 0x4 (do not fragment)
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 7e65
IP:   Source address = 194.197.118.22
IP:   Destination address = 194.197.118.20
IP:   No options
```

DNS Query, UDP Header

```
UDP:  Source port = 38325
UDP:  Destination port = 53 (DNS)
UDP:  Length = 39
UDP:  Checksum = E34A
```

nixu

DNS Query, Headers and Data

```
0: 0800 2074 f12c 0000 3b80 0e93 0800 4500
.. t.,..;.....E.
16: 003b 8b96 4000 ff11 7e65 c2c5 7616 c2c5
.;..@...~e..v...
32: 7614 95b5 0035 0027 e34a 000a 0100 0001
v....5.'.J......
48: 0000 0000 0000 0574 6170 6173 046e 6978
.......tapas.nix
64: 7502 6669 0000 0100 0100
u.fi.....
```

- From now on only relevant portions of the headers will be displayed

DNS Reply Headers

```
ETHER:  Packet size = 217 bytes
ETHER:  Destination = 0:0:3b:80:e:93
ETHER:  Source      = 8:0:20:74:f1:2c
ETHER:  Ethertype = 0800 (IP)
IP:   Total length = 203 bytes
IP:   Flags = 0x4 (do not fragmnet)
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 8ed6
IP:   Source address = 194.197.118.20
IP:   Destination address = 194.197.118.22
UDP:  Source port = 53
UDP:  Destination port = 38325
UDP:  Length = 183
UDP:  Checksum = AD48
```

nixu

DNS Reply Headers and Data

```
0: 0000 3b80 0e93 0800 2074 f12c 0800 4500
..;..... t.,..E.
16: 00cb 7a95 4000 ff11 8ed6 c2c5 7614 c2c5
..z.@.......v...
32: 7616 0035 95b5 00b7 ad48 000a 8580 0001
v..5.....H......
48: 0001 0003 0004 0574 6170 6173 046e 6978
.......tapas.nix
64: 7502 6669 0000 0100 01c0 0c00 0100 0100
u.fi............
--- some reply data deleted ---
208: 087b db00 04c1 d212 124f
.{.......
```

nixu

TCP

- TCP = Transmission Control Protocol
- Defined in RFC-793
- Connection-oriented, reliable, byte-stream service
  – Provides one independent byte stream in each direction
- Application data is broken into segments, which are sent as IP datagrams.
- Features:
  – Checksums, timeouts and flow control
  – Segment reassembly in correct order, discarding duplicate packets
- Applications using TCP: SMTP, HTTP (WWW), NNTP (News),...

nixu

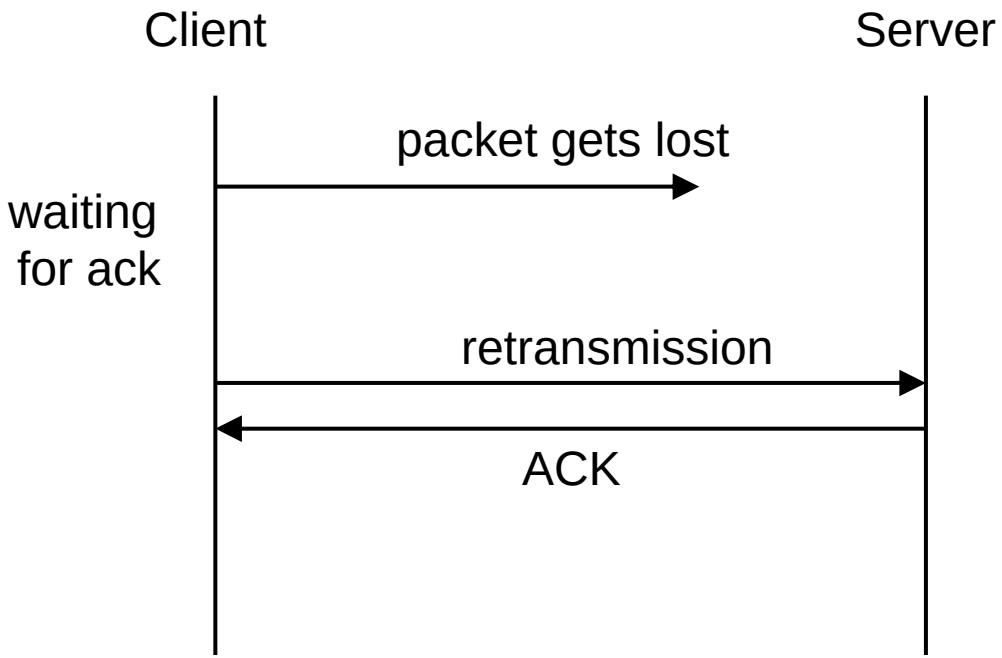
TCP Segment Format

<table>
<tr><td colspan="3">Source port number</td><td>Destination port number</td></tr>
<tr><td colspan="4">Sequence number</td></tr>
<tr><td colspan="4">Acknowledgment number</td></tr>
<tr><td>Hdrlen</td><td>Reserv.</td><td>Flags</td><td>Window size</td></tr>
<tr><td colspan="3">TCP checksum</td><td>Urgent pointer</td></tr>
<tr><td colspan="4">Options (if any)</td></tr>
<tr><td colspan="4">Data (if any)</td></tr>
</table>

- Ports identify source and destination applications.
- Sequence number identifies the first byte of the segment.
- Acknowledge number is the next expected sequence number for incoming data.

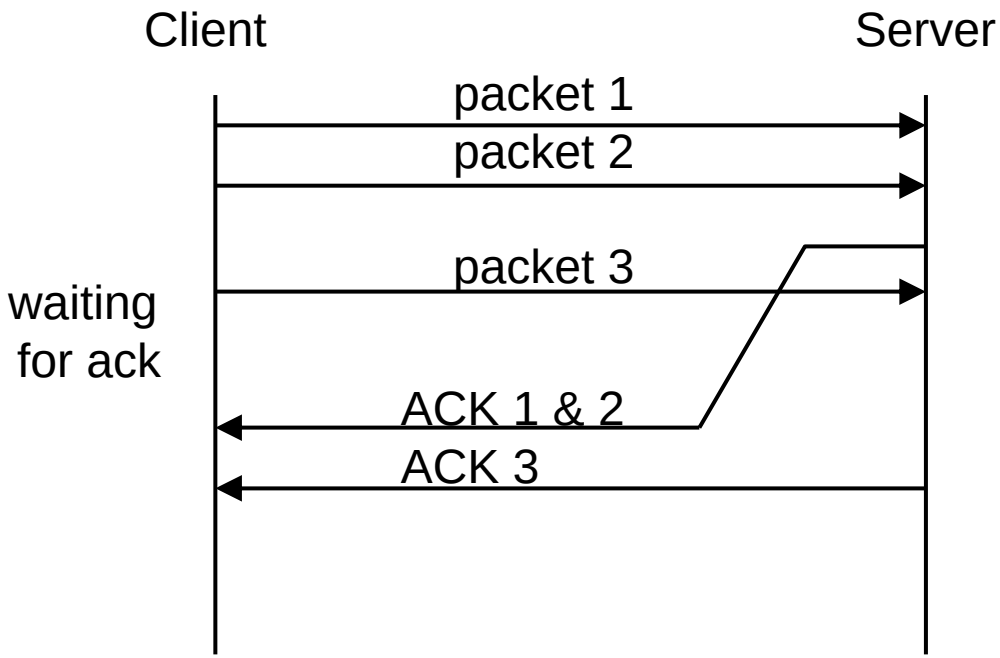
TCP Data Flow

- Receiver sends acknowledgment for each segment.
- If a packet gets lost, timeout will ensure it's retransmitted



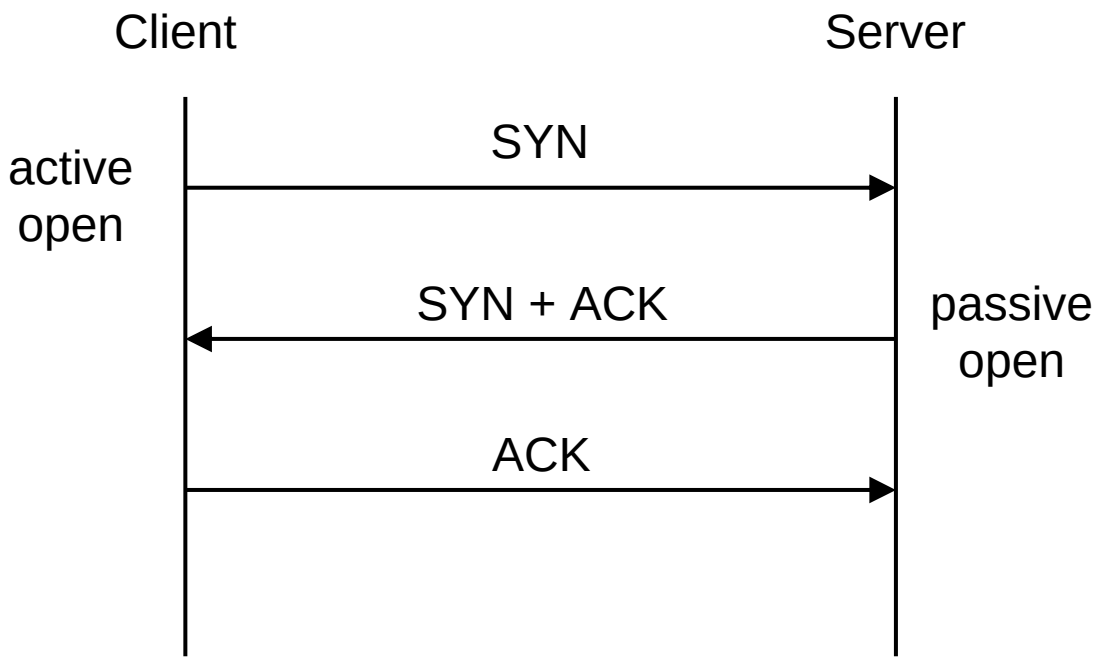
…TCP Data Flow

- Normally a sliding window technique
- The window size is changeable, default size is around couple dozen kilobytes depending on the implementation.

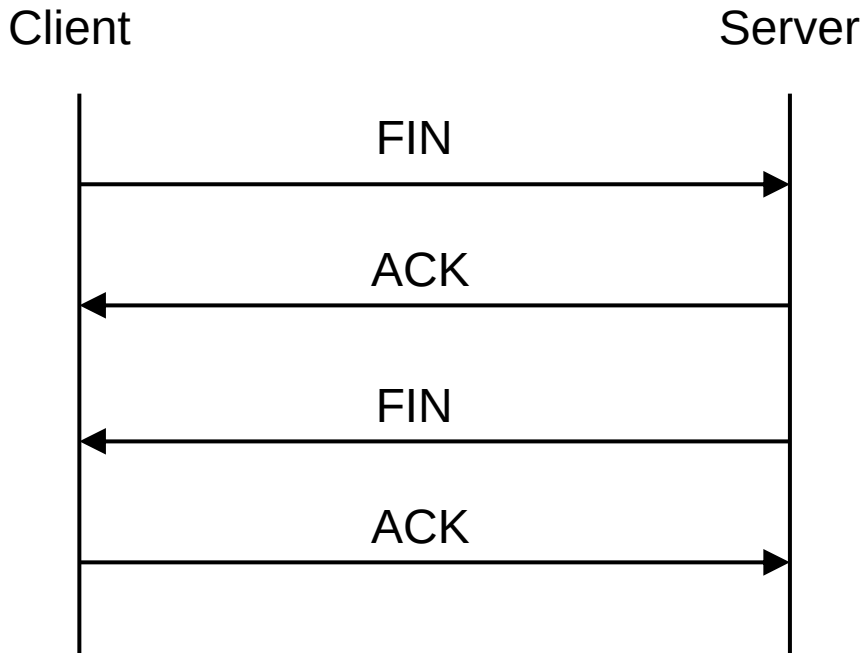


Establishing a TCP Connection

- The “three-way handshake”



Closing a TCP Connection



- Either participant may initiate closing the connection
  - Client and server are equal in this regard
- Often the application protocol session going over the TCP connection is closed first

nixu

Applications Layer

- Applications layer protocols are used by applications to talk to each other
  - Data is transported over the transport layer (TCP or UDP)
  - To an application these look almost like a file
    - At least in the C language
    - Different environments may define different interfaces for applications
- We introduce the protocols used by WWW and e-mail
  - WWW and e-mail are services to the user, they use several protocols to implement the service

nixu

HTTP

- Application-level protocol for distributed, collaborative, hypermedia information systems.
- Used by Web browsers to communicate with WWW servers.
- Generic, stateless, object-oriented

nixu

HTTP Communication (Client)

- Client (browser) opens a TCP connection to an HTTP server (e.g. Apache)
  - By default to port 80
- Client decodes the URL: http://www.nixu.fi:8080/
  - "http": use HTTP protocol
  - "//": absolute URL
  - "www.nixu.fi": the host name of the WWW server
  - ":8080": use port 8080
- Client translates the host name to an IP address by using DNS
- Client opens an TCP connection to the server
- Client sends a request line, some request headers and a blank line to server

nixu

HTTP Communication (Server)

- Server sends a response line, some response headers, a blank line and a document and closes the connection (on HTTP/1.1 connection is not closed)
- Every object on a page is requested separately.
  - HTML page with 3 pictures ->
    - with HTTP/1.0 four separate requests and connections.
    - HTTP/1.1 four requests over one connection
- Server response may be HTML, graphics, audio, VRML, Java...
  - Depends on what file formats the browser supports

Example

```
1  bash-2.03$ telnet www.nixu.fi 80
2  Trying...
3  Connected to jalopeno.nixu.fi.
4  Escape character is '^]'.
5  HEAD / HTTP/1.0
6
7  HTTP/1.1 200 OK
8  Date: Mon, 12 Apr 1999 10:26:06 GMT
9  Server: Apache/1.2.6
10 Last-Modified: Fri, 26 Feb 1999 15:28:20 GMT
11 Connection: close
12 Content-Type: text/html
13
14 Connection closed.
15 bash-2.03$
```

nixu

HTTP Response Status Line

```
HTTP/Version Status-Code Reason-Phrase
```

Status-Code categories

1xx: Informational - Not used, reserved for future use

2xx: Success - Action was successfully received, understood, and accepted.

3xx: Redirection - Further action must be taken in order to complete the request

4xx: Client Error - Request contains bad syntax or cannot be fulfilled

5xx: Server Error - Server failed to fulfill an apparently valid request

- These enable various needed features for communication from server to client

nixu

Predefined Status Codes (HTTP/1.1)

- "200" ; OK
- "201" ; Created
- "202" ; Accepted
- "203" ; Non-Authoritative Information
- "301" ; Moved Permanently
- "400" ; Bad Request
- "404" ; Not Found
- "500" ; Internal Server Error
- "505" ; HTTP Version not supported

nixu

What Is a Protocol?

- A protocol is an accepted method for two or more entities to talk to each other
  - In the case of HTTP there is a human readable but formal definition of certain codes and phrases
    - Extensible, allows introducing new versions and features
    - Relatively easy to debug for humans
  - TCP is a fixed binary format protocol
    - More efficient for software to process
    - Not human readable
- A protocol should be un-ambiguous and able to withstand all natural disturbances
  - Especially should not deadlock or get unsynchronized between the two endpoints

nixu

Internet E-Mail

- E-mail messages are transmitted over the Internet using the SMTP protocol
  – Simple Mail Transfer Protocol
- SMTP e-mail server receives a message and stores it to disk
  – After the message is stored, the server tries to contact next server and transmit the message forward to it
  – An SMTP server acts both as a server and as a client
- The message ends up in a file in the final server, where it is read by a e-mail program locally or over the network with some e-mail retrieval protocol

nixu

SMTP-protocol

- “Push protocol”, i.e. sender initiates
- Server is at TCP port 25
- Currently undeliverable messages can (and should) be queued
- RelatedStandards
  - RFC2821: Defines transfer-protocol
  - RFC2822: Defines message-form
    - These are updated by many other RFCs
  - RFC 1123: Internet Host Requirements
  - RFC 1870, 2821: SMTP Service Extensions
  - RFC 1891-1895: Even more extensions, now obsoleted by newer RFCs
  - RFCs 2045-2049: MIME

nixu

Mail Agents

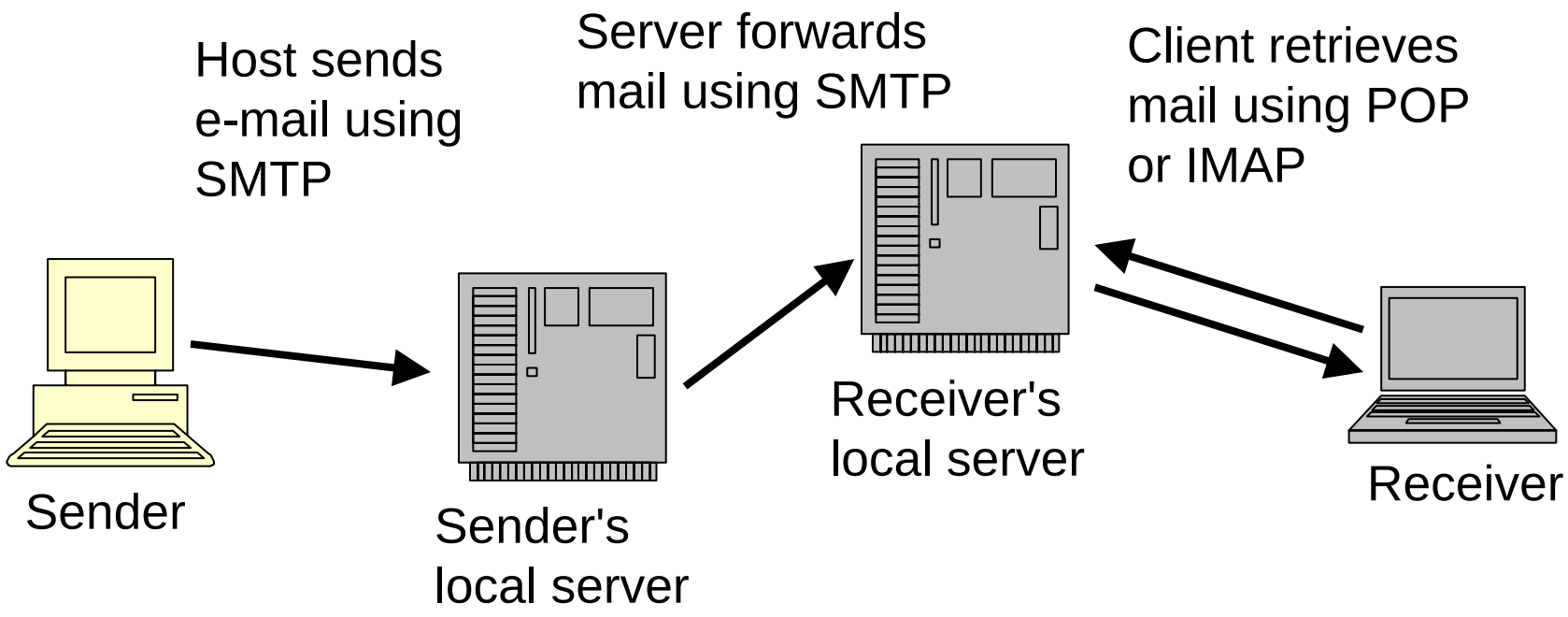
- Mail User Agents
  - MUAs are the source and destination of e-mail
  - Pine, Microsoft Outlook, MH, Mozilla, Elm, mail, Thunderbird, etc.
- Mail Transfer Agents
  - MTAs transport and route the messages from the sender's MUA to the recipient's MUA
    - This is applications level routing and similar to but not related to IP-routing
  - The decision is made based on the recipient's address
    - Spam blocking is an exception
  - The recipient's address may be changed
    - E.g. e-mail aliases, .forward

nixu

The e-Mail Message's Journey

- The message in the SMTP-standard consists of two parts
  - The envelope is information transmitted using SMTP protocol units
  - The contents includes the headers and body of the message
- The MUA receives the message from the end user and interprets the correct sender and receiver information
- The message is passed to the MTA for transportation over the network
  - Usually the message is first stored in a spool directory to wait until it can be transmitted to the next MTA
  - At the destination the message is placed into the recipient's mailbox
    - usually a file, can also be a directory or a database
- In practice the distinction between modern MTA and MUA software is not always clear

How the Mail Travels



Sample SMTP Session Initiation

```
18 riku@mole $ telnet nixu-gw.nixu.fi 25
Trying 194.197.118.1...
Connected to nixu-gw.nixu.fi.
220 nixu-gw.nixu.fi ESMTP Sendmail 8.9.3/8.9.3; Tue, 13 Apr 1999 13:40:05 +0300
HELP
214-This is Sendmail version 8.9.3
214-Topics:
214-    HELO    EHLO    MAIL    RCPT    DATA
214-    RSET    NOOP    QUIT    HELP    VRFY
214-    EXPN    VERB    ETRN    DSN
214-For more info use "HELP <topic>".
214-To report bugs in the implementation send email to
214-    sendmail-bugs@sendmail.org.
214-For local information send email to Postmaster.
214 End of HELP info
EHLO mole.nixu.fi
250-nixu-gw.nixu.fi Hello mole.nixu.fi [194.197.118.22], pleased to meet you
250-8BITMIME
250-SIZE
250-DSN
250-XUSR
250 HELP
```

nixu

Sending the Message in SMTP

```
MAIL From: <riku@mole.nixu.fi>
250 <riku@mole.nixu.fi>... Sender ok
RCPT To: <Timo.Kiravuo@nixu.fi>
250 <Timo.Kiravuo@nixu.fi>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
From: <riku@mole.nixu.fi>
To: <Timo.Kiravuo@nixu.fi>
Subject: foobar
Demo material for SMTP course
.
250 NAA12630 Message accepted for delivery
QUIT
221 nixu-gw.nixu.fi closing connection
Connection closed by foreign host.
19 riku@mole $
```

nixu

The Message Structure

- The envelope contains the MTA’s view of the sender and receiver
  - This is why you receive complaints about viruses and spam you have not sent
  - These are transported in the MAIL FROM and RCPT TO commands of the SMTP protocol
  - Notice the difference between the "From:" in the message headers and the "From" in the envelope
- Headers
  - From the beginning of the content until the first empty line
  - Format is "field-name: field body"
  - Some are mandatory, some not
- Body
  - After first empty line until the end of the message

SMTP and DNS

- MXs
  - Mail eXchanger - records in DNS
  - Enables mail forwarding in cases where access to customers mail-server is limited
  - Example: part of sral.fi MXs

```
sral.fi. IN MX 10 bar.foo.fi.
sral.fi. IN MX 20 smtp3.kolumbus.fi.
```

- Logic: Mail is transferred only closer to destination
  - Smaller MX value means that machine is closer to destination
  - Machine with the smallest MX value is tried first, then the machine with the next smallest and so on...
  - If no MX record, A record (IP address) is used

nixu

POP and IMAP Mail Read

- Post Office Protocol
- Internet Message Access Protocol
- An e-mail client program contacts a POP or IMAP server and asks for new e-mail for an user-ID

nixu

Multipurpose Internet Mail Extensions

- A standard to use in e-mail
  - Text other than US-ASCII
  - Non-textual data formats
  - Multipart messages
  - Textual header information using characters other than US-ASCII
- Several standards and extensions
  - 2045 MIME Part One: Format of Internet Message Bodies
  - 2046 MIME Part Two: Media Types
  - 2047 MIME Part Three: Message Header Extensions for Non-ASCII Text
  - 2048 MIME Part Four: Registration Procedures
  - 2049 MIME Part Five: Conformance Criteria and Examples
- MIME types are also used by other protocols and services
  - E.g. HTTP

MIME Message (Simplified)

```
FROM: "MS Security Center" <aytnddhiqp@support_msdn.net>
TO: "Partner" <partner@support_msdn.net>
SUBJECT: Current Net Security Patch
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary="dqrvzwnprd"

--dqrvzwnprd
Content-Type: multipart/related;
   boundary="jtdukndxczlsbnv";
         type="multipart/alternative"

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

Microsoft Partner

this is the latest version of security update, the...
```

MIME Message (Cont.)

```
--jtdukndxczlsbnv
Content-Type: text/html
Content-Transfer-Encoding: quoted-printable
   <HTML>...
--jtdukndxczlsbnv--
--dqrvzwnprd
Content-Type: image/gif
Content-Transfer-Encoding: base64

R0lGODlhaAA7APcAAP///+rp6puSp6GZrDUjUUc6Zn53mFJMdb...

--dqrvzwnprd
Content-Type: application/x-msdownload;
   name="Install65.exe"
Content-Transfer-Encoding: base64

TVqQAAMAAAAEAAAA//8AALgAAAAAAAAQAAAAAAAAAAAA...
--dqrvzwnprd--
```

MIME

- The body of the message can contain multiple data objects
- Some objects can be alternative to each other
  - E.g. text and HTML representation for the message text
  - The sender can not know the capabilities of the receiver's MUA
- Binary data is coded so that it can pass through the 7-bit e-mail system
  - Some SMTP protocol implementations can not handle 8-bit data
  - Base-64 is usually used for binary data
  - Quoted-printable is used to encode the individual special characters in text data
- Headers have their own coding

```
From: =?GB2312?B?za/R1cHB18s=?=
  <info@shanghaity.com>
Subject: =?GB2312?B?za/R1cHB18vT68T6ubK2ybn6x+w=?=
```

nixu

Spam

- Unsolicited advertising
  - A real problem because of huge volume (100-200 messages per day)
- Usually sent from an e-mail server that allows relaying
  - The server accepts a message, that is not from a domain served by the server and is not targeted towards such domain
  - Spam senders usually falsify the sender address
  - The server used receives one message with plenty or recipients and it has to bear the burden of delivery

nixu

Solutions to Combat Spam

- Basic checks
  - The e-mail server should verify that either the sender or receiver address of a message matches the server's domains
    - This prevents a lot of relaying
  - Sending host's IP address should have a reverse DNS record
- Server blacklists
  - Known servers that send or relay spam
- Bayesian (artificial intelligence) filtering
  - The system learns to recognize spam
  - Currently considered a promising approach
- Legal solutions
- For more information see http://spam.abuse.net/

nixu

Application Level Protocols

- Applications handle different kinds of content
  - e.g.. e-mail, web pages, voice
- Different types of content require different kinds of protocols
- Application level protocols
  - Transfer the application's content (application specific behavior)
  - Transfer information about the capabilities of the participants
  - Use lower layer protocols to avoid doing unnecessary work
- OSI model's session, presentation and application layers are combined to one layer in the TCP/IP model

nixu

Network Relations

- The network entities use different behavioral models on all protocol layers
  - Client-Server
  - Peer to peer
  - Middleware
  - Store and Forward
  - Connections
  - Connectionless communication

nixu

Client-Server Communications Model

- Examples:
  - A WWW client connects to a WWW server and requests a document
  - Xeyes program requests the X server for information about mouse cursor position
- Client is the active participant
- Sessions are initiated by the client
- Server is passive and waits for contact
- Client-server model is usually used to distribute data or CPU

nixu

Thin and Fat Clients

- These terms do not refer to the **communications logic**, but instead to the **software architecture**
- The client can be a simple user interface manager
  - E.g. WWW-client
  - The applications logic is in the server
- Or an application specific program capable of complex data processing operations
  - The applications logic is mostly in the client and the server is usually mostly a database server
- The difference in communications is between I/O (display) information and between raw data
- The current trend is towards thin clients and servers that provide the application logic and data

nixu

Peer to Peer Architectures

- P2P does not distinguish clients and servers, instead all entities can communicate with each other
- In practice many P2P implementations combine both client and server behavior in the same application
- The most interesting question in P2P is how to find the information/service sought after
  - Directory servers (breaking the P2P model)
  - Identifiers and search algorithms
    - Participants transfer information about what they can supply

Middleware

- Middleware is a term, which meaning depends on context
- In the client-server model middleware means usually software that implements the business logic
  - Middleware is connected to the user's thin client at one end and to a database at the other end
  - Typically different protocols are used
    - E.g. HTTP for the client and SQL*Net for the database
- Middleware can also mean a layer between the actual application and the communications layer (TCP & IP)
  - Provides e.g. AAA services, database access
- The interaction models between various parties are usually of client-server type
  - The user initiates actions from the web client, which the middleware translates to database queries and data processing
  - However actual business applications may break this pure model
    - E.g. the server sends a notification to the client

nixu

Store and Forward

- A message is stored until it can be forwarded
- Example:
  - An IP router stores an IP packet in its memory, until the next link is available for transmission
  - SMTP e-mail server receives a message and stores it to disk, after the message is stored, server tries to contact next server and transmit the message forward to it
    - An SMTP server acts both as a server and as a client
- Store and forward makes packet networks efficient and allow discarding the requirement for reserving bandwidth
  - Memory provides a buffer
  - If we run out of memory
    - An IP router discards packets
    - An SMTP server refuses to accept more data

nixu

Connection

- Examples:
  - An user connects to a Unix server from a PC using Telnet protocol
  - A WWW client program connects to a WWW server using HTTP protocol over TCP protocol and stays connected until all the elements of a WWW page are received
    - Two connections at different protocol levels, TCP and HTTP
- In a connection both ends share a state
  - The IP layer is not aware of a connection
- A connection can be broken by network fault

nixu

Connectionless Data Transfer

- Examples:
  - A DNS resolver sends a DNS server a UDP packet, containing a DNS query
  - A network management station queries routers using SNMP packets in UDP packets, if no reply is received after retries, a notification is generated
- In connectionless data transfer the entities transferring information are responsible of knowing the status of communication
  - A DNS server does not care
  - The DNS resolver must retry if the query or reply are lost (UDP is defined as unreliable) or if server is down
- Avoids the setup cost of a connection

nixu

What Do the Protocols Do?

- Protocols are the language different network entities use to talk to each other
  - Windows Netscape can send e-mail to a Sendmail program running on Unix operating system, because they talk same language
  - A method sufficiently understood by two entities so that they can communicate
  - Formal definition preferred
- Internet protocols provide layers of abstraction and higher level protocols rely on lower protocols to operate together