The OSI Reference Model and What the Protocols do
Contents

• OSI Reference model
• Protocols and services, the difference
• Protocol stacks and implementations
• What services protocols provide
Layered Protocols

- Protocols connect entities on same level, within the layer
- Higher layers use services provided by lower layers
- Layers are independent and defined by interface provided to the higher layer and required from the lower layer

Entities communicate by exchanging Protocol Data Units (PDUs) with entities on the same level
Why layers?

- Layers may be exchanged as long as interfaces stay unchanged
  - WWW was built on top of existing TCP/IP implementations
    - WWW does not have to care about the media (modem, Ethernet, radio link etc.)
    - ATM can be used to transport IP packets without any changes to applications
    - IPv6 will replace the entire IP part of the TCP/IP layer
  - Layering makes development easier and adds flexibility
Protocol standardization

- International Organization for Standardization (ISO)
  - http://www.iso.ch/
- Internet Society & Internet Engineering Task Force
  - http://www.isoc.org/
  - http://www.ietf.org/
- International Telecommunications Union (ITU)
  - http://www.itu.ch/
- The Institute of Electronics and Electrical Engineers (IEEE)
  - http://www.ieee.org/
- Industry consortia
  - WAP, 3GPP, ATM
  - World Wide Web Consortium
- De-facto standards
  - IBM PC
Protocol stacks

• Term protocol stack refers to all layers of a protocol family
  – E.g. having TCP/IP stack implemented in an operating system means that part of the OS software uses device drivers (Ethernet, PPP+serial driver etc.) to receive IP packets and provides socket services to applications programs
  – Protocol stack has nothing to do with push/pop stacks (data structures)
• TCP/IP by IETF is the most popular protocol stack in data communications
• Signaling System 7 is the most popular telecoms stack
• OSI protocol stack is hardly used in entirety except as a theoretical model (parts of it are in use)
• Implementations of protocol stacks often co-exist (OSI X.500 directory system over TCP/IP, TCP/IP communications over telephone network and SS7)
## OSI Stack

<table>
<thead>
<tr>
<th>OSI</th>
<th>IETF</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 Application</td>
<td>Sovellus</td>
</tr>
<tr>
<td>6 Presentation</td>
<td>Esitys</td>
</tr>
<tr>
<td>5 Session</td>
<td>Yhteys</td>
</tr>
<tr>
<td>4 Transport</td>
<td>Kuljetus</td>
</tr>
<tr>
<td>3 Network</td>
<td>Verkko</td>
</tr>
<tr>
<td>2 Data link</td>
<td>Siirto</td>
</tr>
<tr>
<td>1 Physical</td>
<td>Fyysinen</td>
</tr>
</tbody>
</table>
OSI Model

- Two (N)-level entities in different systems communicate using (N)-level protocol
- The services give by layer (N) to layer (N+1) are realized in (N)-level protocol by encapsulation and decapsulation
- Encapsulation means embedding each layer's Service Data Units (SDU) into the Protocol Data Units (PDU) of the layer immediately below it, decapsulation is the reverse process
- Entities at the same level in separate systems are called peer entities
- (N+1)-level entities are using (N)-level services through (N)-level Service Access Points ((N)-SAPs)
- One (N+1)-entity can be simultaneously connected to one or more (N)-SAPs
- One (N)-SAP is connected to one (N)-entity
OSI Layers

• Physical Layer
  – Defines the physical, electrical, mechanical, and functional
    procedures to use the physical characteristics of the
    communications path
    – Electrical, electromagnetic, optical, acoustic, printed
    – E.g. lower half of the Ethernet standard, RS-232

• Data Link Layer
  – Transfers frames (packets) over the physical layer
  – Hides the features of the physical implementation
  – E.g. Ethernet, PPP, HDLC

• Network Layer
  – Transfers packets over the entire communications network
    over various physical networks
  – Independent of physical implementations
  – E.g. IP, X.25
OSI Layers...

- **Transport Layer**
  - Provides end to end transport of session (application) data
  - E.g. TCP, UDP
- **Session Layer**
  - Provides dialogue (communications path) control
  - E.g. HTTP “Connection: keep-alive”; TCP urgent pointer
- **Presentation Layer**
  - Provides device independent presentation of data
  - E.g. HTTP “Accept: image/gif”; MIME headers
- **Application Layer**
  - How applications communicate
  - E.g. HTTP, SMTP...
A data packet

- Encapsulation and decapsulation over different types of links
- Application contents do not change
- IP protocol crosses over different link layer protocols

Host

PPP IP TCP SMTP FCS

Next router

Router

PPP IP TCP SMTP FCS
Application Level Protocol Sample

220 tcm.hut.fi SMTP/smap Ready.

helo jalopeno.nixu.fi
250 (jalopeno.nixu.fi) pleased to meet you.

mail from: kiravuo@jalopeno.nixu.fi
250 kiravuo@jalopeno.nixu.fi... Sender Ok

rcpt to: kiravuo@hut.fi
250 kiravuo@hut.fi OK

data
354 Enter mail, end with "." on a line by itself

From: kiravuo@jalopeno.nixu.fi
To: kiravuo@hut.fi
Subject: terveisia
Heipparallaa
.

250 Mail accepted

quit

221 Closing connection
Service Interface

- Protocols are usually internationally standardized
- Service interface implementations are usually operating system specific
  - Different implementations can communicate, with common protocol
  - Application programs need to be modified to use different interfaces
- Unix socket interface example:
  - 1. open socket
  - 2. parse address
  - 3. connect to server
  - 4. write request
  - 5. read response
  - 6. close socket
What Protocols do?

• A protocol shall be:
  – Completely and unambiguously defined
  – Free of dead-locks and live-locks
  – Able to recover from all error conditions

• Some possible functions of protocols
  – Addressing
  – Connections and confirmations
  – Error detection and correction
  – Flow control
  – Prioritization
  – Multiplexing, segmentation and blocking
Protocol Addressing

- **On the Internet:**
  - People usually use domain names (max 255 chars, e.g. www.nixu.fi)
  - IP-protocol uses numeric addresses (four octets, e.g. 194.197.118.20)
  - TCP and UDP use 16 bit port addresses (e.g. 80), separate address spaces
  - LANs and data link level protocols have their own addressing (e.g. Ethernet, six octets)
  - Application level programs must be able to handle both IP addresses and port addresses (Unix socket service), but not data link addresses
  - Application level protocols sometimes have their own addresses (e.g. Telnet protocol)
  - E.g. Telnet protocol does not do any addressing

- **Other**
  - Telephone numbers
  - Postal addresses
Connections

- Some protocols, like TCP provide a connection from end to end, some, like UDP are connectionless
- TCP must store the state and sub-states of the connection
  - No connection
  - Link set-up
  - Data transmission
  - Link disconnect
- Three packets to open a TCP connection and four to close
- Stateless and connectionless UDP is easier to implement and lighter on the network
  - Applications must worry about data loss, e.g. domain name re-requests
Confirmation

- Protocol can provide a receipt
- Example:
  - TCP header contains the sequence numbers of traffic
  - UDP itself provides only a single datagram transmission
  - SMTP server acknowledges when an e-mail message is received
  - SNMP traps are not acknowledged and may be lost
Error control

- Lowest level transmission channels are analog and somewhat unreliable
- Data may be changed, entire frames (packets) may be lost
- For reliable communications we need:
  - Error detection
  - Error correction
  - Data retransmission
- Having no error control is also option
  - E.g. old modems and terminals
Checksums

- Trivial checksum example
  - $1+2+3+4=10$, we transmit $1,2,3,4,10$
  - We receive: $1,3,3,4,10$ and check $1+3+3+4=11$ -> Error detected
  - Our algorithm is not very good, how about: $1,3,2,4,10$?
- Actual algorithms are more complex
  - CRC-CCITT catches certainly all burst errors of 16 bits or less
    - And is most likely to catch all other errors, too
- Parity bits
  - Odd parity, make data element always to have odd number of ones, e.g.: $11100010$ -> $111000101$
- Cryptographic checksums (often called also hashes) make it very hard or impossible to replace or change the data
- Checksum design depends on the type of error expected
  - Physical media errors often come in bursts
Error Detection

- Checksums can be done at different protocol levels and for different sized chunks of data
  - Parity bit for old terminals
  - IP has header checksum
  - TCP and UDP checksums check both header and payload
  - TCP/IP application level protocols do not usually do error detection, they trust TCP and UDP
  - Encryption protocols (e.g. SSH) usually have internal checksums to protect against tampering
Error Detected, then What?

- Ignore the faulty data
  - Used with redundant real time data, like voice
- Freeze totally (e.g. old IBM PC and memory parity check)
- Report error and let higher level protocol decide
- Correct error
  - The usual response
- And how about losing an entire frame (packet, cell) in transmission
  - Nothing received, nothing detected
  - Sender must wait and after a timeout retransmit
  - Or receiver polls for more data
  - Or it does not matter (real time voice and picture transmission)
Error correction

- **Forward Error Control**
  - Add enough redundancy to data to correct errors without delay (this costs)
  - E.g. Hamming code
  - Usually used for time critical or one way protocols
  - Several low level protocols also do this

- **Backward Error Control**
  - Catch error
  - Discard data
  - Ask for a retransmission
  - TCP does this
Flow Control

• What to do when the receiver’s memory is full?
• Special characters in data stream
  – E.g. XON/XOFF, ctrl-S/ctrl-Q
  – Used by terminals
• Limited receive window (e.g. TCP, Kermit)
  – Sliding window increases efficiency
  – The sender can not send more than a certain amount of data above what is acknowledged as received
• Receiver polls for individual data units
  – Not quite a flow control, really
  – UDP-based SNMP does this
• No flow control
  – E.g. IP
Prioritization and Quality of Service

• Currently not implemented in TCP/IP
• Will likely change in future
  – Operators offer QoS inside their own networks
• Several technologies
  – RSVP, reservation of bandwidth
  – Diffserv, higher service classes first
  – MPLS, Multi Protocol Label Switching
• QoS services compete with trivial solutions
  – Dedicated circuits to define the bandwidth to one customer
  – Massive bandwidth (e.g. Gigabit Ethernet)
• 3G (UMTS) telephone networks will use Voice over IP (VoIP) and require QoS
Segmentation and Concentration

- Underlying layers might have special limitations
  - Typically (Ethernet) packet maximum size is less than IP packet maximum size
- A protocol can perform segmentation and concentration operations to the higher level SDUs
  - IP packet fragmentation and reassembly is a sample of this
  - A large SDU is split to numbered smaller chunks
  - The chunks are sent separately
  - The receiver constructs the original SDU
Multiplexing and Blocking

• Multiplexing
  – The joining of several data streams in one connection
  – Usually done at low level of the network (at telephone system level)
  – Packet data networks do statistical multiplexing inherently
    – Everybody does not usually send at same time

• Blocking
  – Packing several SDUs into one PDU
Transparency

• How to transmit any data?
• How to do in-band signaling?
  – Out of band signaling = we have a separate channel for signaling
    – E.g. classical telephony
  – In-band signaling = control and data share same channel
    – E.g. Internet
• Encapsulate data in frames
  – Frames can have a start and stop marker
    – Markers in data have to be passed
  – Frame header can hold the count of objects in the frame
    – This is what TCP/IP and Ethernet do
  – Frames can be of fixed size (or time)
    – ATM cells
Transparency, Stuffing and Counting

- **Data Stuffing**
  - The frame is limited by markers
  - The marker starts a control sequence
  - One control sequence just passes data looking like the marker
    - E.g. send the marker twice to pass the marker
    - E.g. `&amp;` `&gt;` `&lt;` in HTML
    - Telnet has control codes in the data stream

- **Data counting**
  - The header of a frame holds count of the amount of data
    - Used by HTTP
Routing and switching

• How to get a packet to a socket over the net?
• Routing and forwarding:
  – Grab each packet, read its’ address and push it to right direction
• Switching (two operations):
  – 1. Reserve a path between the endpoints
  – 2. Transmit data along the path
Routing

- Router is a computer with two or more network interfaces
- Router receives a packet at its’ network interface
- Router reads the recipient address on the packet
- Routing table tells the router which interface to forward this packet to
  - Tables can be dynamic or static
- Each packet is an individual case
- IP routers can peek into TCP data, too, for security reasons
- A routing network can usually recover from link loss, sometimes without data loss
- Terms:
  - Routing: the decision about where to forward the packet
  - Forwarding: the actual task of moving the data
Circuit Switching

• A channel is allocated over the network for each host to host connection before any data is transferred
  – Channels are reserved, even if no data is transmitted
  – Used in telephone systems
• Other kinds of switching
  – IP packet switching is used on LANs for performance and security
    – Data paths are formed on the fly
  – ATM cell switching uses small, 53-byte cells
    – Easier to implement in hardware than variable length IP packets
    – ATM supports bandwidth reservation
• A switching network can usually recover from link loss, but connections over the missing link will be broken
Discovering the Features of the Transmission Path

- A protocol can optimize its behaviour
  - TCP discovers the Maximum Transmittable Unit for efficiency
- Is likely to become more important in future with the mobile devices, which move from low speed high latency 2G mobile networks to high speed low latency wireless LAN networks
Discovering the Features of the Protocol Entities

• Enables optimal use of features
  – E.g. a handheld device might not support colour
  – E.g. two implementations of an encryption protocol must agree on which algorithm to use
  – E.g. compression algorithm to be used

• “None” is often considered a feature and provides a lowest common level between implementations
  – Several encryption protocols have “none” as one of the standard encryption algorithms which must be implemented, while this is useful for debugging it is also potentially dangerous
Conclusion

If the label on the cable on the table at your house,
Says the network is connected to the button on your mouse,
But your packets want to tunnel on another protocol,
That's repeatedly rejected by the printer down the hall,
And your screen is all distorted by the side effects of gauss,
So your icons in the window are as wavy as a souse,
Then you may as well reboot and go out with a bang,
'Cause as sure as I'm a poet, the sucker's gonna hang!

http://www.netfunny.com/rhf/jokes/96q1/seuss.html