Topic 2 – Architecture and Philosophy

- Abstraction
- Layering
- Layers and Communications
- Entities and Peers
- What is a protocol?
- Protocol Standardization
- The architects process
  - How to break system into modules
  - Where modules are implemented
  - Where is state stored
- Internet Philosophy and Tensions
Abstraction Concept

A mechanism for breaking down a problem

*what* not *how*

- eg Specification *versus* implementation
- eg Modules in programs

Allows replacement of implementations without affecting system behavior

**Vertical versus Horizontal**

“Vertical” what happens in a box “How does it attach to the network?”

“Horizontal” the communications paths running through the system

**Hint:** paths are built (“layered”) on top of other paths
Partition system into modules & abstractions:

• Well-defined interfaces give flexibility
  – *Hides* implementation - can be freely changed
  – Extend functionality of system by adding new modules

• E.g., libraries encapsulating set of functionality

• E.g., programming language + compiler
  abstracts away how the particular CPU works …
Computer System Modularity (cnt’ d)

• Well-defined interfaces hide information
  – Isolate assumptions
  – Present high-level abstractions

• But can impair performance!

• Ease of implementation vs worse performance
Network System Modularity

Like software modularity, but:

• Implementation is distributed across many machines (routers and hosts)

• Must decide:
  – How to break system into modules
    • Layering
  – Where modules are implemented
    • End-to-End Principle
  – Where state is stored
    • Fate-sharing
Layering Concept

- A restricted form of abstraction: system functions are divided into layers, one built upon another
- Often called a *stack*; but **not** a data structure!

<table>
<thead>
<tr>
<th>speaking 1</th>
<th>thoughts</th>
</tr>
</thead>
<tbody>
<tr>
<td>speaking 2</td>
<td>words</td>
</tr>
<tr>
<td>speaking 3</td>
<td>phonemes</td>
</tr>
<tr>
<td>D/A, A/D</td>
<td>7 KHz analog voice</td>
</tr>
<tr>
<td>companding</td>
<td>8 K 12 bit samples per sec</td>
</tr>
<tr>
<td>multiplexing</td>
<td>8 KByte per sec stream</td>
</tr>
<tr>
<td>framing</td>
<td>Framed Byte Stream</td>
</tr>
<tr>
<td>modulation</td>
<td>Bitstream</td>
</tr>
<tr>
<td></td>
<td>Analog signal</td>
</tr>
</tbody>
</table>
Layers and Communications

• Interaction only between adjacent layers
• $layer\ n$ uses services provided by $layer\ n-1$
• $layer\ n$ provides service to $layer\ n+1$
• Bottom layer is physical media
• Top layer is application
Entities and Peers

*Entity – a thing* (an independent existence)

Entities *interact* with the layers above and below

Entities *communicate* with *peer* entities

– same level but different place (eg different person, different box, different host)

Communications between peers is supported by entities at the lower layers
Entities and Peers

Entities usually do something useful
  – Encryption – Error correction – Reliable Delivery
  – Nothing at all is also reasonable

Not all communications is end-to-end

Examples for things in the middle
  – IP Router – Mobile Phone Cell Tower
  – Person translating French to English
Layering and Embedding

In Computer Networks we often see higher-layer information embedded within lower-layer information:

- Such embedding can be considered a form of layering.
- Higher layer information is generated by stripping off headers and trailers of the current layer.
- E.g., an IP entity only looks at the IP headers.

\textit{BUT embedding is not the only form of layering.}

Layering is to help understand a communications system, \textbf{NOT} to determine implementation strategy.
Example Embedding
(also called Encapsulation)
Internet protocol stack *versus* OSI Reference Model

**OSI Reference Model**

- Application
- Presentation
- Session
- Transport
- Network
- Data Link
- Physical

**Internet Protocol stack**

- Application
- Transport
- Network
- Data Link
- Physical

**Ethernet payload** consists of individual octets

**TCP header**

**TCP payload**

**IP header**

**IP payload**

**Ethernet header**

**Ethernet payload**

**packet checksum**

**FRAMING:** Ethernet payload consists of individual octets

**CODING:** Each byte encoded into a 10 bit code-group using 8B/10B block coding scheme

**MODULATION:** Digital electrical signal converted to analogue optical signal and transmitted on fibre

...0010101011110010110100001110001010101001...

...110100100101010100110101110011...

...101 1 1 1...
ISO/OSI reference model

- **presentation**: allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- **session**: synchronization, checkpointing, recovery of data exchange
- Internet stack “missing” these layers!
  - these services, *if needed*, must be implemented in application

<table>
<thead>
<tr>
<th>Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
</tr>
<tr>
<td>presentation</td>
</tr>
<tr>
<td>session</td>
</tr>
<tr>
<td>transport</td>
</tr>
<tr>
<td>network</td>
</tr>
<tr>
<td>link</td>
</tr>
<tr>
<td>physical</td>
</tr>
</tbody>
</table>
What is a protocol?

**human protocols:**
- “what’s the time?”
- “I have a question”
- introductions

... specific msgs sent

... specific actions taken when msgs received, or other events

**network protocols:**
- machines rather than humans
- all communication activity in Internet governed by protocols

**protocols define format, order of msgs sent and received among network entities, and actions taken on msg transmission, receipt**
What is a protocol?

a human protocol and a computer network protocol:

Q: Other human protocols?
Protocol Standardization

• All hosts must follow same protocol
  – Very small modifications can make a big difference
  – Or prevent it from working altogether
• This is why we have standards
  – Can have multiple implementations of protocol
• Internet Engineering Task Force (IETF)
  – Based on working groups that focus on specific issues
  – Produces “Request For Comments” (RFCs)
  – IETF Web site is http://www.ietf.org
  – RFCs archived at http://www.rfc-editor.org
So many Standards Problem

- Many different packet-switching networks
- Each with its own Protocol
- Only nodes on the same network could communicate
INTERnet Solution
Internet Design Goals (Clark ‘88)

- Connect existing networks
- Robust in face of failures
- Support multiple types of delivery services
- Accommodate a variety of networks
- Allow distributed management
- Easy host attachment
- Cost effective
- Allow resource accountability
Real Goals

Internet Motto

*We reject kings, presidents, and voting. We believe in rough consensus and running code.* – David Clark

- **Build something that works!**
- Connect existing networks
- Robust in face of failures
- Support multiple types of delivery services
- Accommodate a variety of networks
- Allow distributed management
- Easy host attachment
- Cost effective
- **Allow resource accountability**
In the context of the Internet

Applications
...built on...

Reliable (or unreliable) transport
...built on...

Best-effort global packet delivery
...built on...

Best-effort local packet delivery
...built on...

Physical transfer of bits
Three Observations

• Each layer:
  – Depends on layer below
  – Supports layer above
  – Independent of others

• Multiple versions in layer
  – Interfaces differ somewhat
  – Components pick which lower-level protocol to use

• But only one IP layer
  – Unifying protocol
Layering Crucial to Internet’s Success

- Reuse
- Hides underlying detail
- Innovation at each level can proceed in parallel
- Pursued by very different communities
What are some of the drawbacks of protocols and layering?
Drawbacks of Layering

• Layer N may duplicate lower layer functionality
  – e.g., error recovery to retransmit lost data
• Information hiding may hurt performance
  – e.g., packet loss due to corruption vs. congestion
• Headers start to get really big
  – e.g., typical TCP+IP+Ethernet is 54 bytes
• Layer violations when the gains too great to resist
  – e.g., TCP-over-wireless
• Layer violations when network doesn’t trust ends
  – e.g., firewalls
Placing Network Functionality

• Hugely influential paper: “End-to-End Arguments in System Design” by Saltzer, Reed, and Clark (‘84)
  – articulated as the “End-to-End Principle” (E2E)

• Endless debate over what it means

• Everyone cites it as supporting their position
  (regardless of the position!)
Basic Observation

• Some application requirements can only be correctly implemented end-to-end
  – reliability, security, etc.

• Implementing these in the network is hard
  – every step along the way must be fail proof

• Hosts
  – Can satisfy the requirement without network’s help
  – Will/must do so, since they can’t rely on the network
Example: Reliable File Transfer

- Solution 1: make each step reliable, and string them together to make reliable end-to-end process
- Solution 2: end-to-end **check** and retry
Discussion

• Solution 1 is incomplete
  – What happens if any network element misbehaves?
  – Receiver has to do the check anyway!

• Solution 2 is complete
  – Full functionality can be entirely implemented at application layer with no need for reliability from lower layers

• Is there any need to implement reliability at lower layers?
Summary of End-to-End Principle

• Implementing functionality (e.g., reliability) in the network
  – Doesn’t reduce host implementation complexity
  – Does increase network complexity
  – Probably increases delay and overhead on all applications even if they don’t need the functionality (e.g. VoIP)

• However, implementing in the network can improve performance in some cases
  – e.g., consider a very lossy link
“Only-if-Sufficient” Interpretation

• Don’t implement a function at the lower levels of the system unless it can be completely implemented at this level

• Unless you can relieve the burden from hosts, don’t bother
“Only-if-Necessary” Interpretation

• Don’t implement anything in the network that can be implemented correctly by the hosts

• Make network layer absolutely minimal
  – This E2E interpretation trumps performance issues
  – Increases flexibility, since lower layers stay simple
“Only-if-Useful” Interpretation

• If hosts can implement functionality correctly, implement it in a lower layer only as a performance enhancement
• But do so only if it does not impose burden on applications that do not require that functionality
We have some tools:

- Abstraction
- Layering
- Layers and Communications
- Entities and Peers
- Protocol as motivation
- Examples of the architects process
- Internet Philosophy and Tensions
Distributing Layers Across Network

• Layers are simple if only on a single machine
  – Just stack of modules interacting with those above/below

• But we need to implement layers across machines
  – Hosts
  – Routers (switches)

• What gets implemented where?
What Gets Implemented on Host?

• Bits arrive on wire, must make it up to application

• Therefore, all layers must exist at the host
What Gets Implemented on a Router?

- Bits arrive on wire
  - Physical layer necessary

- Packets must be delivered to next-hop
  - Datalink layer necessary

- Routers participate in global delivery
  - Network layer necessary

- Routers don’t support reliable delivery
  - Transport layer (and above) *not* supported
What Gets Implemented on Switches?

• Switches do what routers do, except they don’t participate in global delivery, just local delivery

• They only need to support Physical and Datalink
  – Don’t need to support Network layer

• Won’t focus on the router/switch distinction
  – Almost all boxes support network layer these days
  – Routers have switches but switches do not have routers
There is just one network-layer protocol, IP. The “narrow waist” facilitates interoperability.
Alternative to Standardization?

• Have one implementation used by everyone

• Open-source projects
  – Which has had more impact, Linux or POSIX?

• Or just sole-sourced implementation
  – Skype, many P2P implementations, etc.
A Multitude of Apps Problem

• Re-implement every application for every technology?
• No! But how does the Internet design avoid this?
Solution: Intermediate Layers

- Introduce intermediate layers that provide set of abstractions for various network functionality and technologies
  - A new app/media implemented only once
  - Variation on “add another level of indirection”