Network Architecture

Stefan Saroiu
Computer Science
University of Toronto
Outline

- Overview
  - Protocols and layering
  - Brief Internet tour
- E2E paper
- Internet design philosophy
Protocols and Layering

- We need abstractions to handle all this system complexity

  A protocol is an agreement dictating the form and function of data exchanged between parties to effect communication

- Two parts:
  - Syntax: format -- where the bits go
  - Semantics: meaning -- what the words mean, what to do with them

- Examples:
  - Ordering food from a drive-through window
  - IP, the Internet protocol
  - TCP and HTTP, for the Web
Protocol Standards

- Different functions require different protocols
- Thus there are many protocol standards
  - E.g., IP, TCP, UDP, HTTP, DNS, FTP, SMTP, NNTP, ARP, Ethernet/802.3, 802.11, RIP, OPSF, 802.1D, NFS, ICMP, IGMP, DVMRP, IPSEC, PIM-SM, BGP
- Organizations: IETF, IEEE, ITU
- IETF (www.ietf.org) specifies Internet-related protocols
  - RFCs (Requests for Comments)
  - “We reject kings, presidents and voting. We believe in rough consensus and running code.”, Dave Clark.
Layering and Protocol Stacks

- Layering is how we combine protocols
- Higher level protocols build on services provided by lower levels
- Peer layers communicate with each other

Layer N+1 e.g., HTTP

Layer N e.g., TCP

Home PC

FireFox
Example – Layering at work
Layering Mechanics

- Encapsulation and de(en)capsulation

Messages passed between layers
A Packet on the Wire

- Starts looking like an onion!

- This isn’t entirely accurate
  - ignores segmentation and reassembly, Ethernet trailers, etc.
  - But you can see that layering adds overhead
More Layering Mechanics

- Multiplexing and demultiplexing in a protocol graph

- TCP port number
- IP protocol field
- 802.2 identifier
Outline

- Overview
  - Protocols and layering
  - Brief Internet tour
- E2E paper
- Internet design philosophy
A Brief Tour of the Internet

- What happens when you “click” on a web link?

- This is the view from 10,000 ft...
9,000 ft: Scalability

- Caching improves scalability

- We cut down on transfers:
  - Check cache (local or proxy) for a copy
  - Check with server for a new version
8,000 ft: Naming (DNS)

- Map domain names to IP network addresses
  - All messages are sent using IP addresses
    - So we have to translate names to addresses first
    - But we cache translations to avoid doing it next time (why?)

“What’s the IP address for www.google.com?”
“It’s 207.200.75.200”

Nameserver
128.95.2.1

128.95.2.106
7,000 ft: Sessions (HTTP)

- A single web page can be multiple “objects”
  
  GET index.html
  GET ad.gif
  GET logo.gif

- Fetch each “object”
  - either sequentially or in parallel
6,000 ft: Reliability (TCP)

- Messages can get lost
- We acknowledge successful receipt and detect and retransmit lost messages (e.g., timeouts)
5,000 ft: Congestion (TCP)

- Need to allocate bandwidth between users

- Senders balance available and required bandwidths by probing network path and observing the response

How fast can I send?
4,000 ft: Packets (TCP/IP)

- Long messages are broken into packets
  - Maximum Ethernet packet is 1.5 Kbytes
  - Typical web page is 10 Kbytes

- Number the segments for reassembly

```
GET
2. inde
3. x.ht
4. ml
```

GET index.html
3,000 ft: Routing (IP)

- Packets are directed through many routers

H: Hosts
R: Routers

Internet
2,000 ft: Multi-access (e.g., Cable)

- May need to share links with other senders

- Poll headend to receive a timeslot to send upstream
  - Headend controls all downstream transmissions
  - A lower level of addressing (than IP addresses) is used ... why?
1,000 ft: Framing/Modulation

- Protect, delimit and modulate payload as signal

| Sync / Unique | Header | Payload w/ error correcting code |

- E.g, for cable, take payload, add error protection (Reed-Solomon), header and framing, then turn into a signal

  - Modulate data to assigned channel and time (upstream)

  - Downstream, 6 MHz (~30 Mbps), Upstream ~2 MHz (~3 Mbps)
Outline

- Overview
  - Protocols and layering
  - Brief Internet tour
- E2E paper
- Internet design philosophy
End-to-End Principle

- Question: where should functionality be placed in the network?
- The argument:
  - 1. Functionality should be implemented at a lower level iff it can be implemented correctly and completely
  - 2. Incomplete functionality could be implemented at a lower level for performance
Example – File transfer

If not done E2E, checks must be done at every single step
Performance Optimizations

- Functionality at lower layer can enhance performance
  - Could come in very useful
- E.g. path with 15 hops
  - Prob of packet loss is 0.0001% per hop
    - Prob of e2e loss is 0.0015%
  - Prob of packet loss is 1% per hop
    - Prob of e2e loss is 14%
Optimization Trade-Off

- Higher layers have semantic information about the data
  - HTML text bits more important than IMG in the presence of massive loss
- Lower layers are “closer” to the data
  - They are the ones dealing with loss, bandwidth issues, corruption, etc…
- This trade-off is one of the main driving factors behind networking research
Additional Examples

- Networking
  - Reordering
  - Retransmission
  - Supressing duplicate messages

- Security
  - WEP
E2E Final Thoughts

- What’s the consensus?

- Do these violate E2E?
  - Proxy caches
  - NATs
Outline

- Overview
  - Protocols and layering
  - Brief Internet tour
- E2E paper
- Internet design philosophy
Design Philosophy of DARPA

Context:
- Multiple research and military networks
- How can we connect them?
- Before LANs existed
Main Internet Goal

- Multiplexed utilization of existing interconnected networks
  - i.e., functionality

- Minimal assumptions underlying networks
  - No support for broadcast, multicast, real-time
  - Packet switched, store-and-forward
  - “Gateways” interconnect networks
This is a hard problem

- Main reason: heterogeneity
  - Addressing: each network different addressing scheme
  - Bandwidth: dialups to dedicated links
  - Latency: seconds to nanoseconds
  - Packet size
  - Loss rates
  - Service guarantees
IP: Common Substrate

- Convert all formats to IP
  - IP over everything

- Alternative?
Secondary Goals

- Survivability in the face of failure
- Support for multiples types of service
- Support for a variety of networks
- Distributed management of resources
- Cost effectiveness
- Easy addition of new hosts
- Accounting of resource usage
Survivability Implications

- No hard-state in the middle
  - Routers are stateless
  - Endpoints responsible for failure recovery
    - “fate-sharing”
- Host machines are trusted
  - Must implement protocols correctly
  - Easy to be malicious
- Problems:
  - Hard to detect cause of failures
  - Hard to prevent hosts from becoming malicious
    - Much more complex functionality than routers
Types of Service

- Common denominator: IP
- Reliable delivery TCP
- Unreliable delivery UDP
- Real-time delivery: ???
- Multicast: ???
- Broadcast: ???
Network Variety

- Internet successful in part due to so little assumptions of the underlying network
  - Topology: point-to-point, bus, ring, radio
  - Characteristics: dial-up, fiber, etc…

- Does not mean IP is efficient:
  - ATM uses 53 byte cells, poor fragmentation for IP packets
Other Goals

- Distributed management
  - Different parts of network owned, controlled, and managed by different entities
  - Hard to get consensus
  - What does optimize mean in this context?

- Cost-effective
  - Routers are cheap compared to POTS
  - Efficient bandwidth use
  - But much more unreliable than POTS switches
Other Goals (2)

- Attachment costs
  - Success if you look at costs per user (millions of users)
  - Misbehaving hosts due to bugs
    - Think Wal-Mart vs. Harrods
- Accountability
  - Who pays for all of this
  - What is the economic model
  - Hot religious debate
Questions

- Did the Internet realized its vision?
- What is different about the Internet today?
- What are the limitations of Internet design?

- The Internet has evolved very much like a research project:
  - Many iterations, trial-and-error
Summary

- Layering: key technique for managing complexity in systems
- Key question:
  - Where to put functionality
- E2E: put functionality at the highest level where service is provided
  - Use lower-levels for optimizations
  - Trade-offs btw. complexity and performance
- Internet design points
  - Internetworking: minimal assumptions about networks
  - Fate-sharing: smart hosts, dumb core
  - All higher services built on top of IP
Next class

- Papers review

- Reviews due at 11am