6.896 Lecture 2: Architectural Principles *

Hari Balakrishnan

September 14, 1998

1 Outline

- **Readings: Zim80, Cla88, CT90**
- Protocol layering
- OSI architecture
  - Principles
  - Practice ("design-by-committee")
- Internet architecture
  - Principles
  - Features
  - IETF and the standards process; contrast with OSI practice
- Announcements and administrivia
- Disadvantages of layering; layering "violations"
- ALF and ILP

2 Protocol layering

Recap: What is a "protocol?"
Question: Suppose you were asked to design a network to provide cost-effective, robust, high-performance connectivity to a large number of users, evolving to meet changing user demands, changing hardware technologies, changing applications and changing amounts of traffic, how would you do it?
Problem: Want high price/performance, but large number of design choices exist, and often unknown, guaranteed-to-change target apps and underlying technologies.
Solution: Modularize! Very similar to software design problem of how to divide system into tractable, useful (i.e., reusable) components.
Layering: In protocol design, modularization going from hardware to apps.

* These notes are based on numerous sources, including Peterson and Davie's book (Chapter 1), Christian Huitema's book (Chapter 1), Zimmermann's OSI architecture paper (1980), Dave Clark's paper on Internet philosophy (1988), Clark and Tennenhouses's paper on ALF and ILP (1990), and Steve McCanne's CS 268 course notes.

- Not a linear sequence (e.g., multiple ways of doing reliability on top of same datagram interface).
- Interfaces: *Service interface* and *peer interface.* (Draw picture showing this.)
- *Protocols* implement these layers.
- Protocol "stack" (why is it a stack?)
- "Protocol" is an overloaded term! Refers both to specification and implementation. But there's a (big) difference between the two.
- Q: How do peers in the stack communicate?
  - Header and/or trailer *encapsulation.*
- Q: How does a lower layer know where to pass a message?
  - *Mux* and *demux.* Discuss demux key.

3 OSI Architecture

- Reference: Zimmerman paper
- ISO and ITU-T (formerly CCITT) specified *reference model* for architecture. Also called the "X dot" series (X.25 (internetworking), X.400 (mail), X.500 (user directory services), etc.)
- (Draw picture of different layers, incl. service and peer interfaces.) Bottom 4 layers pretty clear, the top set is fuzzy in practice!
- "Design by committee" based on some experience and numerous meetings. (More on this later).

4 Internet Architecture

- Unlike OSI's reference model, this is a lot harder to actually define! Best defined by saying "it's defined by its protocol suite."
- But there are a number of principles underlying its design.
  1. **IP over everything!** Embrace heterogeneity in this manner. Reject translation-based approaches. Contrast with X.25/X.75 approach of having separate internetworking protocol.
  2. **Accomodate multiple service types**
  3. **End-to-end argument:** (How many know what this is?) Put function in a layer where it can be done completely, or else if it leads to a significant performance improvement. Not under any other circumstance. Far-reaching impact, widespread invocation (sometimes wrongly too!). Engineering principle, not a "law."
  4. **Soft-state:** Essentially, only maintain routing tables in the network; these are periodically refreshed by routing protocols.
  5. **Fate sharing:** Share fate with other end-host, not with routers en route. Contrast with X.25 approach.
  6. **Corollary: Failures are not a special case!**

7. **Conservative-transmission/liberal reception:** Increases robustness.
8. And of course, **layering:** But with a twist—bypasses possible.

- Salient features
  - **Hourglass shape**: This is the secret to ubiquitous internetworking across numerous networks and technologies. (Draw picture). (Principle #1).
  - **Services:** Best-effort service model with multiple services built on top of IP. (Principle #2).
  - Communication continues despite the failure of networks or gateways. (Principles #5, 6).
  - **Simple internals:** Routers are simple with only "soft" state; all the intelligence and hard state is in the end-hosts. (Principles #3, 4, 5).
  - About 4 layers in practice. (Principle #8).
  - Decentralized administration.
- Some weaknesses/drawbacks
  - Greedy (non-cooperating users, buggy or malicious implementations). TCP does a good job of sharing bandwidth, but more and more traffic isn't TCP.
  - Weak accounting and pricing tools. One of the original goals, but hasn't developed much.
  - Administration and management tools for large-scale networks.
  - Incremental deployment: Not really a fundamental "weakness," but has to be recognized by every protocol designer.
- The standards process
  - The IETF: voluntary organization, meetings every 3 months, short-lived working groups (usually less than 2 years). Email discussions.
  - The dogma: "We reject kings, presidents, and voting; we believe in rough consensus and running code." (Dave Clark, 1992).
  - Usually need at least 2 independent, interoperating implementations.

5 Announcements

- Next class: SRC84 and Bol93. Pick up CK74, reading for next Monday (9/21).
- Papers I don't hand out are electronically available from reading list on web page.
- Homeworks: Two during semester, see Web page schedule for details
- Course project: Fairly large (8-10 week) hands-on project on a topic of your choice. List of suggestions on Web page in about a week to 10 days. Ideally, work in groups of 2.
- One 60 or 90 minute exam in the middle of the term.
- No final. 20-minute conference-style project presentations and a paper is due at the end of the term.
- Office hours: MW 2-3pm in NE43-510. Email: hari@lcs. This week W 2-3 moves to Th 2-3.

- If you're crediting, register on Web signup form. If you want to audit or sit in, make sure to talk to me or send me email.
- *ns:* Simulations using *ns*. Look at http://www-mash.cs.berkeley.edu/ns for more details and documentation. Also look at http://catarina.usc.edu/vint
- Quiz 0: If comments regarding background on your paper, please come and see me or send me email.

6 Problems with Layering

Three key problems:

1. Strict layering constrains architecture. Why? (Mandating functionality of layers leads to an inflexible architecture.)
2. Layering is a good way of handling complexity and thinking of protocol abstractions, but often leads to lousy implementation performance. Why? (Boundary crossings, context switches, repeated data manipulation and data handling).
3. Adverse interactions could occur between similar functionality implemented in different layers (e.g., TCP and link-layer protocols).

7 ALF and ILP

- Clark and Tennenhouse, 1990 SIGCOMM paper.
- ALF deals with problems of strictly layered architecture.
- ILP deals with inefficient implementations.

7.1 Manupulation and Control

- All protocols perform *data manipulation* and *transfer control* operations.
- Data manipulation
  - Move to/from network
  - Error detection
  - Buffering for retransmissions
  - Encryption
  - Move data to application
  - Presentation formatting (from byte stream to data structures)
- Transfer control
  - Flow/congestion control
  - Congestion detection, reordering detection
  - Acknowledgments
  - Mux and demux

- Timestamping
- Framing

- Performance of data manipulation *dominates* transfer control; presentation conversion, often done in app address space, could dominate everything.

7.2 ALF

- ALF allows the expression of application semantics on the design of protocol. One size does not fit all!
- If there aren't any losses or reordered packets, things usually go fine...
- But if not, mismatches between application and protocol destroy efficiency and performance. (Q: why does reordering happen? Why does performance degrade?)
- Example: Webcasting. TCP's in-order, reliable delivery not the best model. (But that's what's used today!). Can you use TCP to implement selective reliability? Another example is real-time multimedia transmission.
- ADU: Application Data Units. These are what application want, so don't force them to tailor to a particular type of network or technology or protocol semantics. ADU's can be processed in *any order*, so long as they are complete. Transport handles app complete ADU, but doesn't care about preserving inter-ADU ordering.
- Questions: How to name ADU? Can we always make ADU = 1 packet?
- Will look at ALF examples later in course.

7.3 ILP

- Layering fosters sequential processing of each layer (ease of implementation).
- Remember, layering is only a design choice, not a law.
- ILP motivated by widening memory/CPU bottleneck.
- Process all "layers" in one integrated loop, perform multiple manipulation (and control) ops in one loop.
- Have to minimize precedence/ordering constraints. (Compiler help).
- Practice
  - Holy grail: many papers in the 1990s. Approaches include automatic synthesis, compiler support, formal languages, etc. Difficult problem.
  - In many cases, application is the bottleneck, or network bandwidth is...