Network Layering
CS 118
Computer Network Fundamentals
Peter Reiher
Outline

• What is a layer?

• Goals of layering

• Internet and the One Ring
What is a layer?

- **A layer is:**
  - The largest set of parties (nodes) that can communicate
  - Using a single protocol
  - And a single name space
  - A kind of a “zone”

I.e., the transitive closure of a set of nodes via relayed communication
Limits of a single layer

- Homogeneity
- Complexity
Homogeneity 1

• No environment / context optimizations
  – Prevents customization, e.g., for wireless, satellite, low-power, lightweight devices, etc.

• Difficult to pick
  – Hard to pick one everyone will agree on
Homogeneity 2

• Locked-in once deployed
  – No incremental evolution

• Needs a “flag day” to change
  – First major change to Multics: June 14, 1966
  – Internet shift from NCP to TCP: Jan. 3, 1983
Complexity

- One large, flat layer
  - Difficult to manage
  - Difficult to route (determine paths)
  - Difficult to coordinate resources
- And its one, large name space
  - See above…

Groups of layers can be easier…
Reasons to have multiple layers

• Abstraction
• Emulation
• Containment
• Scale
Abstraction

• Model
  – Simplification
  – Extract key features
  – See: Picasso ->
Abstraction and Layering

• Present a simpler system
  – Topology
  – Protocol
  – Behavior

• Other simplifications
  – Node subset
  – Smaller distance
IEEE 802

• Base layer
  (802.2 LLC: Logical Link Layer)
  – Addressing
  – Frame format
  – Error detection
  – Upper-layer interface (API)

• Lower layer
  – Bridging (802.1), i.e., switch control and configuration

• Upper layers
  – Ethernet (802.3)
  – Token bus (802.4)
  – Token ring (802.5)
  – Wireless (802.11)
  – Personal (802.15)
  – Emergency (802.23)
Emulation

• Abstraction to copy behavior
  – Replicate capabilities of another system
  – By limiting, composing, or modifying an existing system

• Change existing system to another one by adding layers
Emulation and Layering

• PPP
  – Point-to-point protocol
  – Emulate a physical wire over a dialup modem

• PPPoE
  – Ethernet

• PPPoA
  – ATM
Containment

• Enforce boundaries
  – Protect interior
    • Ensure a stable locale
    • Enforce limited function
  – Protect exterior
    • Deploy new features
    • Support local capability
Containment and Layering

• Hide / protect a network from the world
• Hide / protect the world from an experiment
Scale

• Limit complexity
  – Hard to route everywhere, so maybe route via a tree?

• Overcome physical limits
  – Sharing can be very useful, but may require a very different protocol for relaying
Scale and Layering

- Tree routing
  - Know your descendants
  - Route down for known, up for unknown
  - Relies on varying detail at different layers
    - Label groups

- Cloud routing
  - Traverse the clouds without knowing what happens inside
  - Inside the cloud, it’s the cloud’s problem
Connections and layering

• **Dovetailed**
  - Edge connected
  - Peer translators
  - Both transit or terminus

• **Stacked**
  - Top-to-bottom connected
  - Treat lower layers as encapsulation transit
  - Never as terminus
Translation gateways

- Gateway converts
  - Bidirectional translation
  - Complex internal state

- Two views
  - Presents two views
  - Proxies in both directions
Benefits of adapters

• Allow local optimization

• Allow partial (subset) upgrades
Issues with adapters

- The telephone game
- Semantic gaps
- Gateway state
- Scale vs. number of “languages”
Lost in translation

• Protocols have idioms
  – Behaviors are “native” to a protocol

• Sequences of translations lose info
  – Telephone game
Semantic gaps

- No protocol is “complete”
  - Turing machine has “completeness”
  - But protocols built on finite state machines
  - FSM is a limited TM

- Some things don’t translate
  - “Ballpark”, “drop a dime”
State problems

• **Size**
  - Physical limits constrain state
  - State can explode
    - Combined FSMs = product of states

• **Reliability**
  - Gateway failure = lost state
  - Lost state affects protocols on both sides

• **Path correlation**
  - End-to-end exchanges need to go through the same gateway for their entire interaction
  - OR state needs to be shared among multiple gateways
Mutual modeling

• My world
  – Has a proxy for you
  – Ends at that proxy

• Your world
  – Has a proxy for me
  – Ends at that proxy

• Sort of independent but not really
  – We both think our worlds are contained
  – But my proxy and your proxy are coupled
Scale of translation

• More languages = more translators
  – Each new language has cost proportional to the number of languages currently in use
  – Gets worse as languages are added

• Who adds the translator?
  – Team effort – both new and old parties
  – Need someone “fluent” in both
Stacked layers

• The alternative to edge layers
• Put one layer on top of another
• Each layer has its own responsibilities
  – Need not worry about things higher layers handle
  – And higher layers assume lower layer has taken care of certain things
• Layers can be customized to circumstances
  – Wireless vs. wired links, e.g.
Stacked layers

- Each layer sits on a lower layer
- Each layer “depends” on the lower layer
  - But not directly on the layers below that
- We might have “icing” connecting the layers
  - To “hold them together”

- A network isn’t a cake, but it’s a little less different than you might think
But how do these different nets work together?

One layer to rule them all!
IP Layer

• The common layer for the Internet
• Packets
  – Variable length
  – Includes source and destination addresses
• Global addresses
  – Two-level hierarchy
    • High bits = assigned by central authority
    • Low bits = locally assigned
• Best-effort communication
  – Allows for loss, reordering
  – Enables simple forwarding, use smart ends to repair
Metcalfe’s Law

- **Network utility grows as** $N^2$
  - Number of possible pairs communicating

- **Other metrics**
  - Number of possible subsets: $2^N$

- If this is right, key to maximizing network utility is maximizing N
Internet principle

• Talking to everyone poorly is more important than talking to anyone well
  – Prefer capability over efficiency
  – Prefer capability over performance
  – Prefer capability over security
  – Prefer capability over anything else!

Capability is everything
Benefits of a common layer

• All the benefits of dovetailing
  – Abstraction
  – Emulation
  – Containment
  – Scale

• Common expectations
  – Encourages internetwork communication that avoids idioms, or uses them very carefully

• Stacking supports recursion
  – Can be applied in various places, repeatedly
  – AND multiple versions don’t interact
The Hourglass Principle
The Narrow Waist

HTTP/DNS/FTP/NFS/IM
TCP/UDP/SCTP/RTP
Ethernet/FDDI/SONET

λPPM, λCDMA, eNRZ, ePCM

IP
How may waists?

- **IP**
  - The one ring…
- **802.***
  - The lowest layer above hardware, for many
- **HTTP/HTTPS**
  - A user layer that works everywhere
  - Even where others are blocked
Issues with a common layer

• Lowered expectations
  – Avoiding idioms may be less efficient
  – Least common denominator may be weak

• Cost
  – Everyone translates, sometimes even when not necessary
  – Common language is native to no one
    • Like Esperanto
Why has stacked layering won?

• Benefits outweigh costs
  – There were critics as recently as 1990s, e.g., that IP was not fast enough for network disks
• Parallels lingua franca experience
  – Removes “home field advantage”
• Nobody “spoke” their “native languages” all that well, anyway
  – The lingua franca of networking developed as networking itself developed
How layers are used

• More about dovetailing (peer layers)

• More about stacked layers
Dovetailed (peer) layers
Translation
Peer – what is translated?

**Everything:**

- Message contents
- Addresses
• Remember, we’re thinking layers, not physical connections here
Stacked layers

Where does communication go?

• Between the subnets?
• Or through the common layer?
Subnet to subnet

- Common interchange
  - Translate to common
  - Relay common
  - Translate back
Subnet to subnet

- **Pros:**
  - Reduces number of translators
    - Compared to direct translation
  - Sub-layers can be tuned to native contexts

- **Cons:**
  - Twice translated
  - Like two-step dovetailing
Staying in the waist

• Common as primary
  – Use sub-layers as a transit
Staying in the waist

• Pros:
  – Sub-layers for local environments, local relaying
  – Translate just the name
  – Use sub-layer as a transit communications service

• Cons:
  – Enables communication only at the common layer
Conclusion: stacking as transit wins

• Avoids translation
  – Except for the names
  – And content, as necessary, at the endpoints

• Supports localization
  – Using subnets as transits
A layer vs. layering

• A layer (one layer)
  – A homogeneous network
  – Largest group that can communicate via transitive closure with one protocol and one name space
  – Largest network with one protocol and one namespace

• Layering (stacked transit using multiple layers)
  – Creating a homogeneous network from a set of heterogeneous networks
Stacking via transit

- Transiting other layers
  - A layer as unifying different networks
    - The basic definition

- Translating capabilities
  - A layer as an adapter between networks
    - Takes one capability and turns it into another
Services provided by a layer

• Communication
  – Shared state coordination

• Any computable function
  – High-level capabilities built from shared state
  – Typically that assist communication itself
Useful functions

• Emulation
  – Act like a single physical link

• Splitting
  – Large messages over smaller ones

• Joining
  – Pack many small messages in one larger one

• Correction
  – Repair the impact of loss, error, reordering
What defines a layer?

• WHO can communicate
  – Largest group using a single set of rules

• HOW they use communication
  – Services that users can apply to problems
  – 2-party? Multiparty?
  – Reliable? Ordered? Like a wire or like packages?

• HOW they interact with heterogeneous nets
  – The kinds of other networks that a layer can “group”
Layers as building blocks

• Capabilities it provides

• Expectations it assumes
Layer expectations

• A layer is a common language
  – But only for nets it can translate to

• Requirements:
  – Name translation
    • Identify party’s local name, given common one
  – Capability translation
    • Use the lower net as a communications link
Layer capabilities

• A layer provides a common language
  – A common interface
  – A common set of services
  – With which users can apply

• Expectations:
  – Name
    • Users must identify party’s common name
  – Capability
    • Users must be able to use the services provided
Things you can do with layers

- Unify heterogeneous networks
  - As we’ve seen
  - Bridge *communications* between other layers

- Create new services
  - Does not have to join heterogeneous nets
  - Can bridge *capabilities* between other layers
Glue layers

- Adapt capabilities
  - Lossy -> reliable
  - Reordered -> ordered
  - Jittery -> periodic
  - Stream -> message
  - Message -> stream
  - Multiple stream -> 1
  - 1 stream -> multiple
How layers interact

- **Common (upper) layer**
  - Translates its (common) names to that of the lower layer
  - Creates its own (common) services from those provided by the lower

- **Various (lower) layers**
  - Expect upper layer to use its names
  - Expect upper layer to use its services
  - Provides some set of communications capabilities within its network
Things that make you go hmm…

- **Upper layer also**
  - Expects its names
  - Provides its services

- **Lower layer also**
  - Translates its names
  - Creates its services
What’s the difference?

- Common layer
- Each heterogeneous layer
So what really defines a layer?

- Largest set that can communicate with:
  - Single name space
  - Single protocol

- A network with:
  - Common expectations
  - Common capabilities

Every layer is its own “one ring”
Examples of layers

• “Lowest” (touch the physical layer)
  – Ethernet, ATM
• Above Ethernet
  – IP, ARP, PPPoE
• Above IP
  – TCP, UDP, ICMP
• Above TCP
  – HTTP, SMTP, IMAP, DNS, NFS, NTP
Different expectations and capabilities

- **HTTP**
  - Provides request/response messaging
  - Expects reliable, ordered stream of bytes

- **TCP**
  - Provides reliable, ordered stream of bytes
  - Expects messaging
    - Delivered within 2 minutes if delivered
    - If lost, because of congestion (competition)
    - Mostly without errors

- **Ethernet**
  - Provides messaging without errors (can be lost)
  - Expects a shared wire (originally)
Messages inside messages

- Frame
  - Ethernet
- IP
  - TCP
- HTTP
Messages spread across messages

- IP datagram =
  - IP fragment 1
  - IP fragment 2
Summary

• Layering allows us to create powerful networks from components
• Stacking layers for transit
  – Transit is the win for heterogeneity
• Layers have requirements
  – What they expect
• Layers create capabilities
  – What they provide