Layer Optimization
CS 118
Computer Network Fundamentals
Peter Reiher

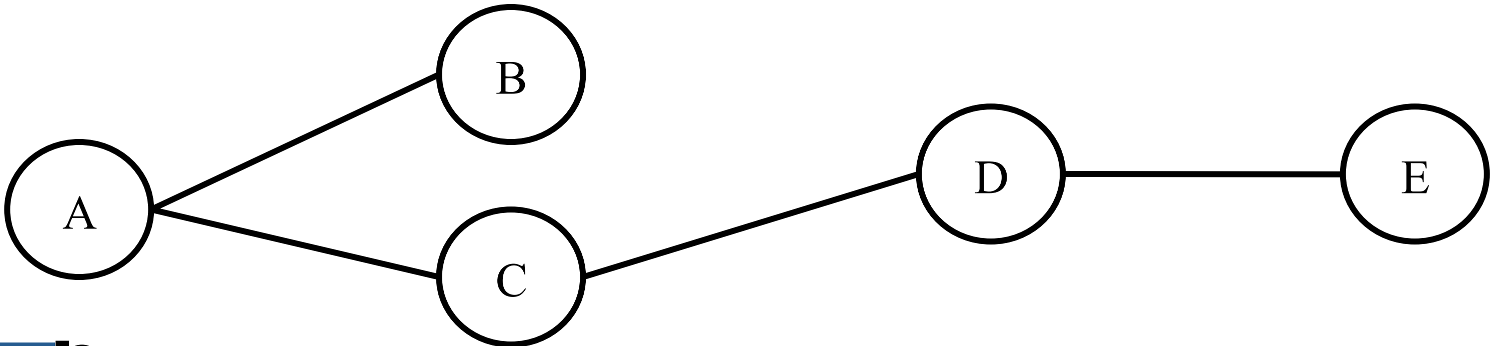
transitive closure - A - B and B - C, then A - C

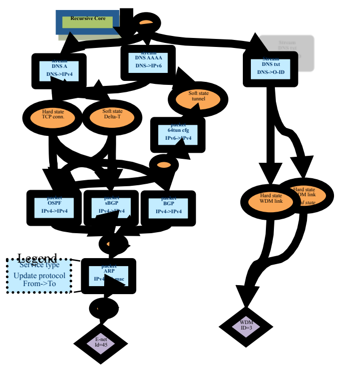
Where are we at?

- We understand communications over direct channels
- We understand building networking from layering and relaying
- We understand how a DAG explains layering
- We understand how to build routing for relaying purposes

movements through a DAG using recursion

For example,





A sends an HTTP request to E

`HTTP->TCP`

`TCP->IP`

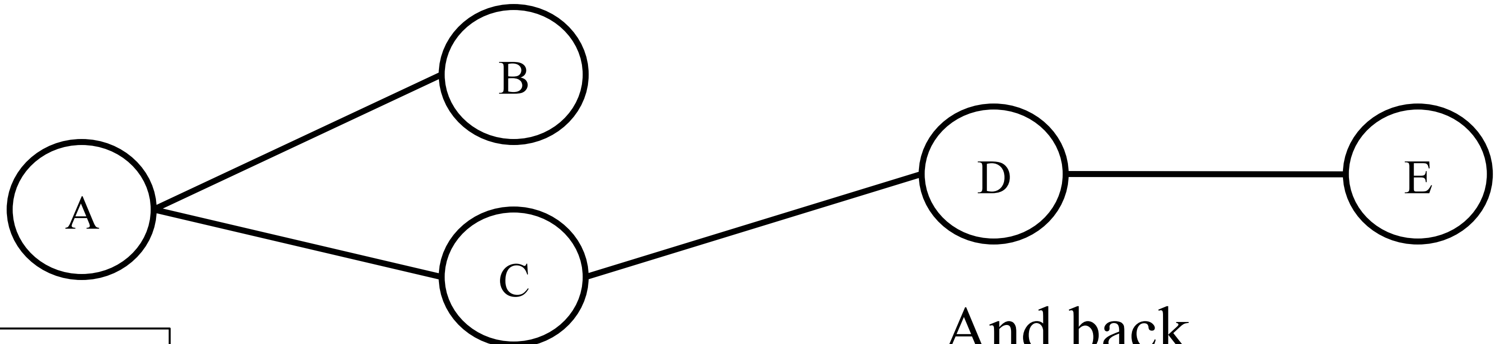
`IP->802.11`

`802.11->physical`

Now we need to relay through C

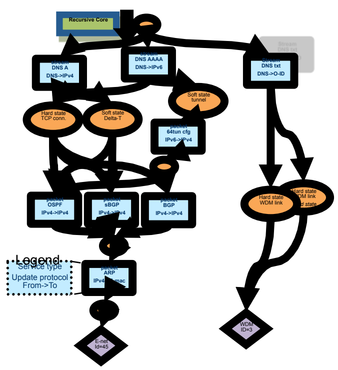
We recurse down the DAG

Relaying through C



And back down

C now uses its DAG



IP<-802.11

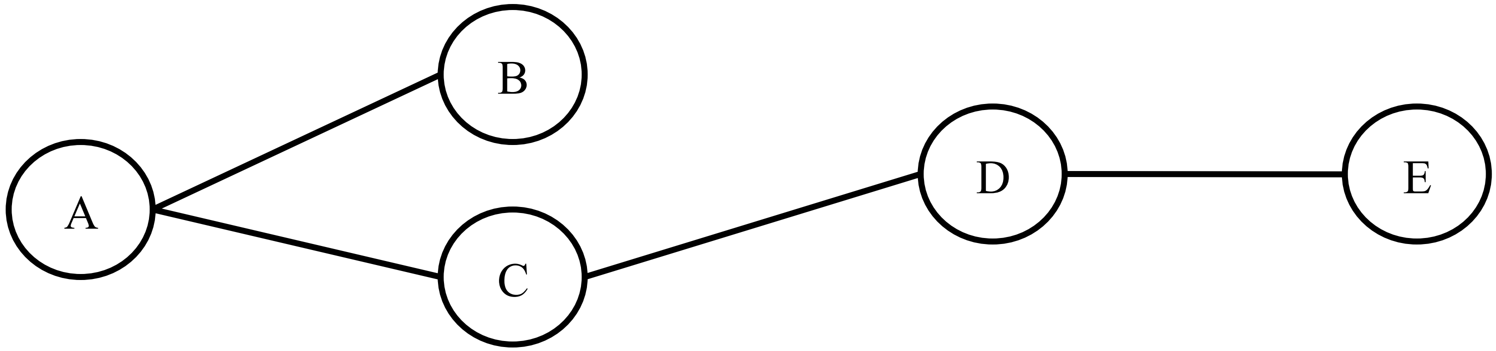
802.11<-physical

IP->ATM

ATM->physical

Now relay through D

Relaying through D

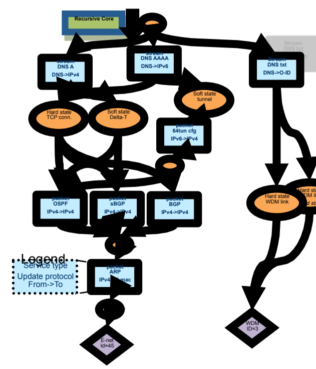


IP<-ATM

ATM<-physical

IP->ethernet

ethernet->physical

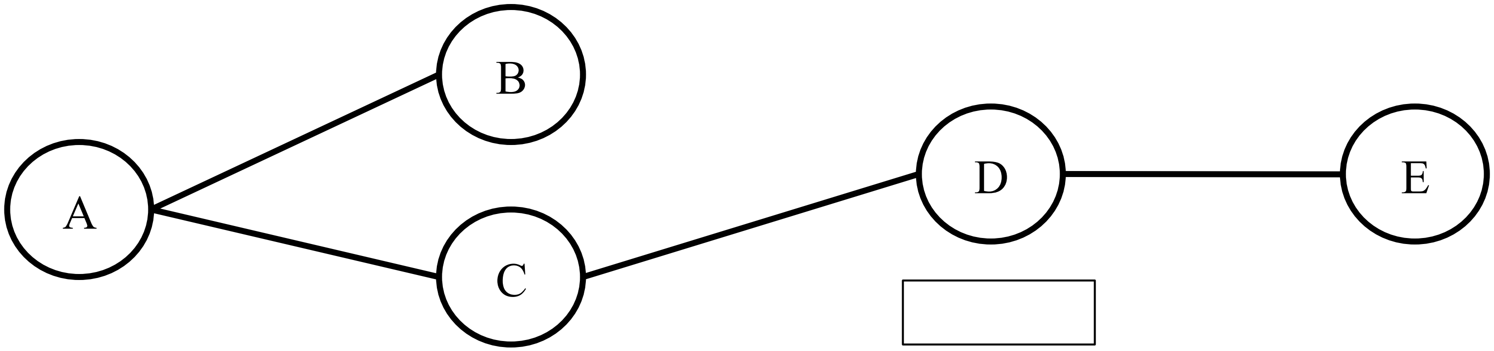


D now uses its DAG

And back down

Now relay to E

Delivering at E

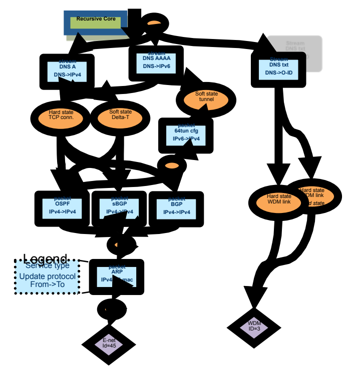


HTTP<-TCP

TCP<-IP

IP<-ethernet

ethernet<-physical



E now uses its DAG

And the message is delivered to E's web server

Where are we and where next?

- So now we know how to use networking between multiple network points
- Being able to communicate at all is more important than anything else
- But other things are important, too
- Like performance, reliability, security, and other properties
- How can we optimize the basic networking to achieve these goals?

Outline

- Background
- Deficiencies
- Performance
- Emulation
- Examples…

Networks and optimizations

- Using fully general mechanisms can be expensive
- Common special cases can be optimized to reduce costs
- Optimizations are possible at many semantic levels of networking

But where to optimize?

- Is optimization a layer or a communication issue?
- Where do we optimize?
  - At some layer?
  - Or for some connection?

Intra-layer vs. intra-communication?

- Optimizations involve shared context
  - Layers sharing common mechanisms
  - Connections managing shared state
- Either one can support optimization
  - Connections coordinate explicitly
  - Layer members coordinate implicitly

If layers, which ones?

- Do optimizations occur *only* at certain layers?
  - No
- Are optimizations *typical* at certain layers?
  - Yes, for several reasons
- Do optimizations interact across layers?
  - Absolutely

Living at a layer

- Optimizations *can* occur at *any* layer
  - They're increasingly used at many layers

Living at a layer

- Some occur *more often* at *certain* layers
  - Most information errors are at the physical layer
  - Once corrected at the next layer up,
    they tend not to occur again
  - So optimizations based on these errors often at low levels

Living at a layer

- Some optimizations occur *because of* a layer
  - I.e., TCP provides an ordered data stream but IP does not
  - So TCP corrects ordering, but IP does not
  - Therefore, any ordering optimization occurs above IP

What connection?

- Optimizations often share state over a connection
- State can be
  - hard (maintained)
- or
  - soft (recoverable)

- State can be for one connection or a group

- State can be explicit or inferred

What connection?

- Not everything is associated with an explicit, stateful connection
  - I.e., there's more than TCP, web, and e-mail

caching?

Now let's explore how to optimize:

- Deficiencies
- Performance
- Emulation

end to end principle
- put application specific network functionalities at the endpoints, not in the middle
- functionality in the middle should be generally usable by all applications
- not a hard and fast rule
 - a principle that is influential

Deficiencies

- Optimizations sometimes overcome deficiencies in the communication
- For example, deficiencies in:
  - Integrity
  - Authentication
  - Privacy

behave in a way that has a property that it does not have

be sure that the receiver does this

receive the content of the message

Why do you care about deficiencies?

- Deficiency impedes communication
  - You can't share state, which means you can't share information
- Deficiency impedes relaying
  - If you can't relay for others, they can't communicate
- Deficiency impedes networking
  - The two above also mean you can't automatically manage your network configuration, routing, DAGs, etc.

Integrity

- Definition
- Types:
  - Corruption
  - Loss
  - Tampering

Integrity

the quality of being whole

- Definition: the quality of being whole
  - WYGIWWS: what you *got* is what was *sent*

- For a message:
  - Not split up
  - Not missing pieces
  - Not altered (accidentally or deliberately)

Integrity: Corruption

- Accidental alteration
- For a message:
  - Symbol changed (noise)
  - Symbol is ambiguous (equivocation)
  - A portion is deleted

Integrity: Loss

related to corruption

- Missing everything!
  - Degenerate case of corruption
  - Receiver doesn't know a message arrived

- Why?
  - Not sent (by origin or relay)
  - Not received (by destination or relay)
  - Corrupt beyond recognition

It’s 2:20pm, do you know where you are?

- Time is fleeting…
  - But it keeps coming up
  - How do you detect loss?

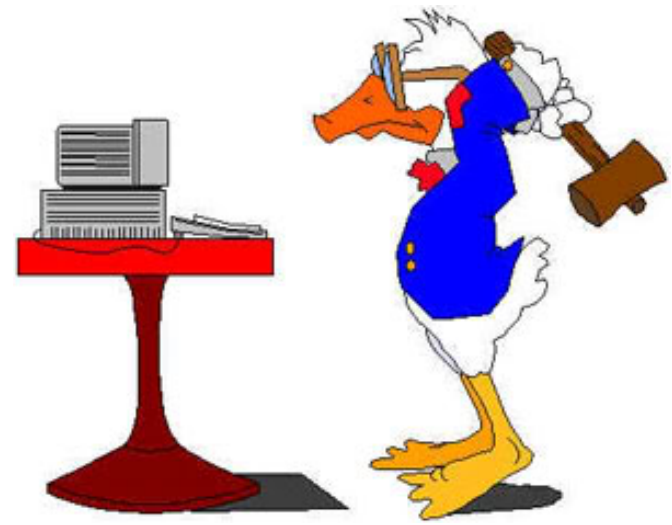
- Useful to know about time
  - Longest time until delivery

Integrity: Corruption vs. loss

- Corruption
  - A message arrives
  - Unrecoverable error
- Loss
  - No message arrives
  - Timer implies loss
- Indistinguishable when:
  - Destination name is corrupted or missing
  - Source name is corrupted or missing
  - Key portions of message are corrupt or missing
- Difference
  - How much of the message is “gone”

Integrity: Tampering

- Deliberate alteration
  - To corrupt
  - To alter to different content (thought to be non-corrupt)



- How?
  - Intercept and retransmit (e.g., during relay)
  - Overlap physical signals

Integrity: Corruption vs. tampering

- Corruption
  - During origination, receipt, or relay
  - Detect via error checks
- Tampering
  - During relay or receipt
  - Detect via integrity checks
    - That a relay can't 'fake'
- Difference
  - Intent
  - Probability of generating different but valid message
  - Tampering is similar to the worst case for corruption

What about order?

- Order is not a property of a message
  - It is a typical property of a channel
  - Only a deficiency if you need to assume it
  - We'll come to that when we talk about channel emulation

misordered delivery when i thought it was all delivered

How much integrity protection?

- Easy to encapsulate
  - You don’t want to modify contents anyway
- Relay might become more difficult
  - May need to change portions of the message
  - E.g., hopcount, route path record, etc.
  - Does the integrity cover those?
- So you might not protect the entire message

Authentication

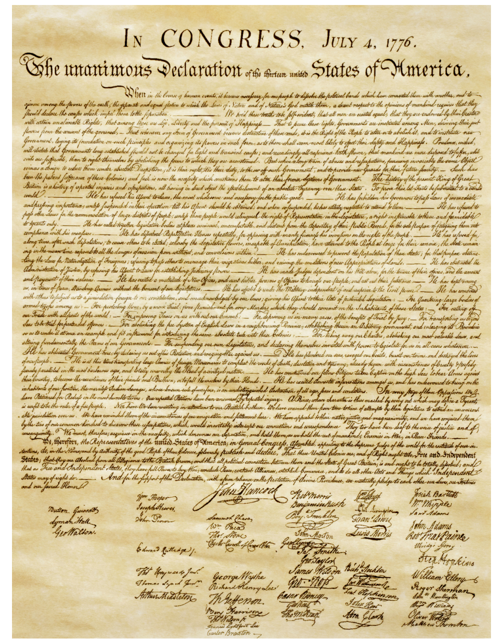
- Definition

- Types
  – Origin and/or destination
  – Control
  – Content

Authentication

you can identify who created the information

- Definition: ensuring that particular information was created by a particular party



- For a message, ensuring:
  – All important elements of the message
  – Were created by the sender

Authentication: Name

- The source and/or destination noted in the message comes from the source that generated it
  - The message says it came from John, and it actually came from John
  - The message says it goes to Ben and that name came from John too

- Why?
  - Protects the entire message’s path
  - Protects the endpoint machines

Authentication: Control

- The message includes control signals that come from the source that generated it
  - Such as what layers it uses
  - And parameters to those layers
  - Ensuring that sender used those layers and those parameters

- Why?
  - Layers use state machines
  - Protects operation of the state machines

Authentication: Content

- The content of the message comes from the source that generated it
  - I.e., the data that is shared with John actually came from John

protect integrity without providing authenticity

if you want to know who sent it, then you need athentitiy

- Why?
  - Protects information that the machines share

Why bother with all three?

- Why do we separate:
  - Identity
  - Control
  - Content

three of these may have different parties sign the followig
we go up and down layers at the relays
we may want to authenticate what happens at the relays

- They could be signed by different parties
  - Different endpoints, different layers, etc.

Why?

- For example, content comes from the "top" of the stack
  – So the top layer should authenticate content
- The identity might be associated with a proxy at a lower layer
  – So that layer should authenticate identity
- The control might be for a very low layer
  – So control must be authenticated there

How much to authenticate?

the message will change from place to place

- Same problem as integrity
  - Easy to attest to the source of the entire message when encapsulated
  - Hard to make that guarantee if portions change
  - Again, might want to authenticate only part

Privacy

- Definition

- Types
  - The origin and/or destination
  - Control
  - Content

Privacy

- Definition: hiding the information in a message from all parties except the receiver

- For a message:
  – Hide the origin
  – Hide the message
  – From everyone but the receiver

Privacy: Identifier

- Hide the endpoints
  - Who sent it
  - Who receives it
- From whom?
  - Source: everyone except receiver
  - Destination: everyone except relay and receiver
    - Perhaps even from relay
- Why?
  - "who talks to whom" exposes information!

metadata: who communicate to whom

clever data mining: find out a lot about who talked to who, when, and perhaps for how long?

Privacy: Control

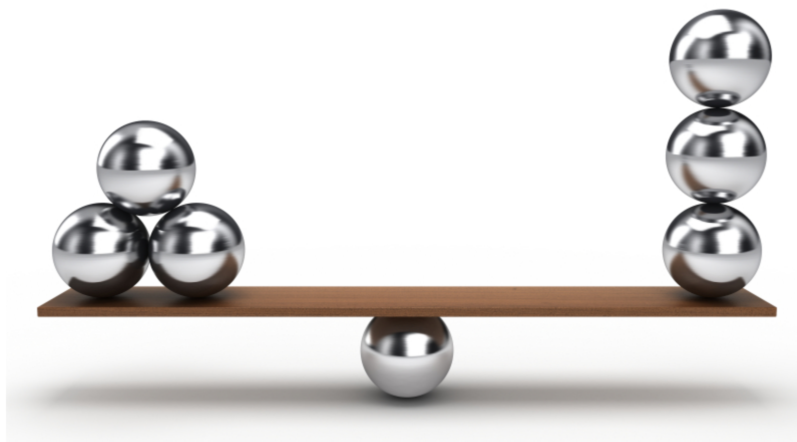
- Hide the state machine control signals
- From whom?
  - Everyone except receiver
- Why?
  - Exposes what the state machine is doing
  - That information can be used to attack the machine
  - Or deduce things about the communications

Privacy: Content

- Hide the information shared between the parties
- From whom?
  - Everyone except receiver
- Why?
  - (should be obvious)

Impact of deficiencies

- Need to balance
  - Relay, source, receiver perspective
  - Various preferences, requirements, and limits
  - Various costs (time, space, CPU effort)



Performance deficiencies

- Time
- Space
- Energy

Time

a deficiency

- Rate
  - Messages per time

    how much info from A to B in a given point of time

- Latency/jitter
  - Time per message (between send and receive)

  it takes some time from getting one bit here to there

  jitter - derivative of the latency; is it constant?
  better to have constant latency rather than changing latency

Rate

- How many messages can you send?
  – Messages per unit time
  – 1/(time to send a msg) * (# msgs sent concurrently)
- How to improve?
  – Less time for each message (higher BW)
  – More messages sent concurrently (parallelism)
  – Messages at the rate the receiver supports
  – Messages at the rate the network allows

Flow control

- Messages arrive at the receiver's rate
  - Avoid overwhelming the receiver
  - Avoid using excess storage resources
- How?
  - Control pacing (inter-message timing)
  - Control number of unanswered messages
  - Use feedback from the receiver

Congestion control

- Messages arrive at the relay's rate
  - Avoid overwhelming the network
  - An aggregate, network variant of flow control
- How?
  - Similar mechanism as flow control
  - Different source of feedback (net, not receiver)

Latency

- How long for a message to arrive?
  - Time per message between send and receive
- How to improve?
  - Decrease distance between sender/receiver
  - Increase BW
  - ...

Space

- How much space to represent a message?
  - Bits per message

- How to improve?
  - Compress (remove predictable patterns)
    - Within a message, across messages, etc.
  - Encode efficiently

Energy

- CPU capacity
- Actual energy

CPU

- How much work to process a message?
  - How many opcodes?
- How to improve?
  - Save reusable results / avoid duplicate effort
  - Alternate algorithm
    - Same result a different, “cheaper” way
    - Different, “cheaper” result with similar properties

Actual energy

- How much energy to process a message?
  - Electrical power, heat to dissipate, etc.
  - Not just “green”; saves $, heat, and space

- How to improve?
  - Reuse rather than recompute
  - Lower clock rates
  - Avoid conversions

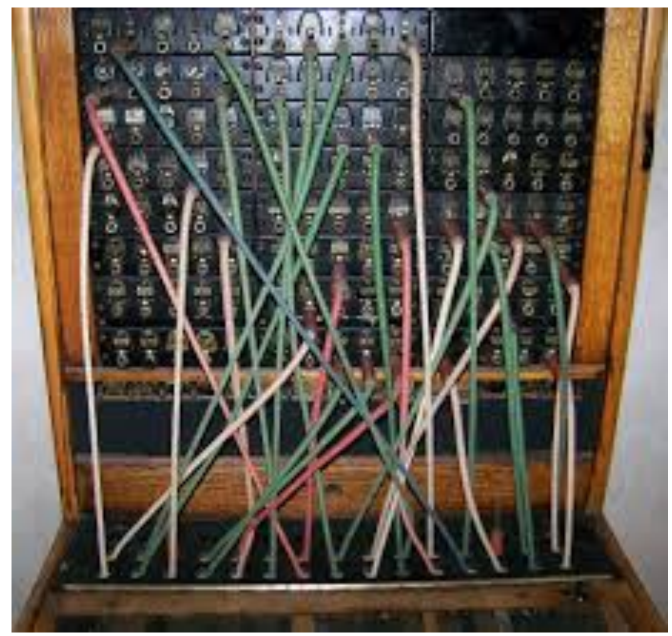
Emulation

- Wires
- Boxes and bundles
- Transactions and beyond

Wires

- Making a circuit from packets

- Pseudowire



Circuits from packets

- Reliable info stream from messages
  - Ordered and reliable
  - Typically relies on endpoint state
  - Not necessarily guaranteeing performance
  - More *like* a wire than a message; not equivalent
- Examples
  - TCP from IP (Internet)
  - AAL 1-4 from ATM (ATM)
  - TP4 from CLNP (OSI)

Pseudowires

- A channel from messages
  - Ordered, reliable, static capacity and delay
    - I.e., performance emulation, too
  - As close to a channel as possible
- Examples
  - SONET
  - TDMoIP (TDM emulation over IP)
  - PWE (pseudowire emulation)

Order

- Circuits and pseudowires emulate channels
  - Most channels assume ordered signal transfer
  - Need to detect and correct misordering
- Examples
  - TCP over IP (Internet)
  - TP4 from CLNP
  - Not ATM! (ATM is never misordered!)

Boxes and bundles

- Boundaries
- Flows

Boundaries

- Marking edges between items
  - Multiple items in one message
  - An item that spans multiple messages
- Examples
  - IP message vs. its fragments
  - DCCP, SCTP
  - HTTP over TCP (stream)

Flows

- Grouping separate connections to act together
  - Striping (increased capacity)
  - Coordinated management (shared control)
  - Alternate/backup (fault tolerance)

- Examples
  - ISDN channel bonding
  - TCP control block sharing
  - Multipath TCP, SCTP

Transactions and beyond

- Transactions
- Translation
- Other services

Transactions and beyond

- Extend service beyond information sharing
  - Support specific structured interactions
- Why?
  - Mostly software reuse
    (any of these can be built on any communication service)

Transactions

- Conditional information flow
  - Serial: send B only if A is complete
  - Conjunction (AND): send C only after A and B
  - Disjunction (OR): send C only after A or B
- Many variations:
  - N of M: send Z only if at least 3 of 6 alternates
  - Send Z only if exactly 3 of 6 alternates

Translation

- Convert one message to another
  - Occurs within the recursive block
  - Also occurs for “gateway” relays
- Examples
  - Language translation (content)
  - Format conversion (HTML to ASCII)
  - Display conversion (desktop to mobile web)

Other services

- If you can dream it, you can do it!
  - Any capability another user/system wants

- Some dreams are nightmares, though…

Overview of issues - sources

- Deficiencies
  - Integrity
    - Error
    - Loss
    - **Reordering?**
    - **Tampering**
  - **Authentication**
  - **Privacy**
- Performance
  - Time
    - Rate (flow, congestion)
    - Latency
  - Space
    - Compression
    - Caching
  - Energy
    - CPU, actual energy
- Emulation
  - Circuit/wire
    - **Reordering?**
    - Boundaries
    - Flows
  - Transactions
  - Translation

Summary

- Deficiencies need to be fixed first
  – Can’t communicate if we can’t communicate
- Then performance should be addressed
  – Go fast, go cheap
- Then emulation
  – Make it look like the user wants it to look