Lecture 2: Internet Design Principles

Design principles for Internet and its protocols.

- A network: End-to-End
- The Internet: State sharing, Packet switching, state sharing, datagram & connection-based services, minimal service model,
- Revealing end-to-end: variants and critique.
- Modern technology and research and the design principles

End-to-End principle: Simplified view of a n/w.

Successful data transfer requires several things to happen:
- Routing
- Naming/Location
- QoS
- Congestion/flow control
- Security/Reliability

Right boundary between hosts and the n/w.

Who should do what? N/w, End-host/application, Combination?

The principle: 1 Function: → Successful implementation eventually depends on whether it satisfies application/user requirements or not.

- For completely and correctly implementing the function, we need end-application knowledge and help.
- Therefore, providing it solely as a feature of the n/w is not possible.

- Sometimes, an incomplete version could be provided by the n/w as a perf. enhancement.
Performance aspects | Identifying the ends.

Reliable file xfer: End-to-end reliability provided by the m/w:
  - Data delivered reliably to NIC
  - Not enough → software, disk subsystem at the sending

Hardware

Applications: App ends data, then integrity checksum, sent by receiver.

Bad if file is large and likelihood of errors is high
  - Chance of a bit flip too high
    - Will take multiple tries.
  - Check and send → but costly as each app must now invent
    - Its own reliability mechanism.

Thus it may be useful for the lower levels to play some role

L → engineering trade-off.

1. How much to put
2. How easy is it to put in the m/w. To employ
3. Impact on apps that don't need it: real-time apps.

Identifying the ends: depends on the application in question...

Consequence: if applied dogmatically → there is no reason to put any complex

Candidates for end: Transport, application, user

The ultimate end used for all practical purposes.
So why the over-reliance on transport?

1. Cost
2. Difficulty of app design
3. Error rate within end system is negligible.

leads to  

2. Trust issues: if the app did not do it, who should it be someone in the local system as the app

has a better understanding of local system's reliability than

other parts of a mw.

Consequences: [transport protocol] [route design] - common case fast

Should the mw not do anything at all?

Should it be largely dumb?

Some arguments for implementing functions within the mw:

1. Efficiency: multicasting < duplication
   
   -> improve efficiency by aggregating info about
   
   multiple users sharing in common.

   -> Routing alternatives

2. Disruption: email -> SMTP servers

3. Economics of scale: forwarding as opposed to broadcast.
   
   -> cost: source routing overhead.

Some arguments for moving stuff out of the mw and

into end-hosts:

1. Speed and simplicity of mw infrastructure

2. Debugging and troubleshooting of problems is easy

3. Easier to model and understand mw

4. mw emp: too egalitarian

   emp: only those who want it, pay for it.
Connecting up multiple nlws.

Some other key principles in Internet design and protocols:

Original: effective techniques for multiplexed utilization of existing nlws

1. Packet switching: existing nlw were packet switches!
2. What does effective mean?
   - Survivable
   - Support for multiple types of services
   - Accommodate a variety of nlws.

Survivable: comm. must go on despite nlw failure
   - fail only if nlw is partitioned
   - Implication: state in the nlw → more node proliferation

Failure → node

Comm fails only if one of the end-hosts that participate fails: fail sharing.

Acceptable to lose state info about an entity if at the same

loss of state

cause

application

fail

Again, where in the end-host.

Somewhere in the comm. subsystem but not in the nlw.

Consequences:
   - Stabilizes into channels
   - Large amount of trust in end-hosts

Types of service: TcP/Ip was one
   - not suitable for many apps / debugging
   - real-time

TCP
UDP
Ip
Incorporate and use
Support for variety of protocols.

1. Minimum set of assumptions about the flow: the flow will either offer "Transport a datagram" that is needed.
   A number of services are explicitly not required.

End-to-end principle and fault sharing today.

1. Transport / congestion control
2. Routing and forwarding
3. QoS
4.