

## Lecture 2: Internet design principles

~~Design principles for Internet and its protocols.~~

Design principles for Internet and its protocols.

- A network: End-to-End
- The Internet: ~~File sharing~~, Packet switching, ~~file sharing~~, datagram & connection-based services, minimal service model,
- Revisiting 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 xfer requires several things to happen.  
A given n/w provides several functions —

routing	}	*
naming/location		
DOS		
congestion/flow control		
security, reliability		

~~Rigid boundary between hosts and the n/w.~~

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

The principle: • Function  $f$  <sup>required by an app.</sup> → Successful implementation eventually depends on whether it satisfies application/users 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 n/w.

- Data delivered reliably to NIC

App  
Export:  
N/w  
- Not enough → software, disk subsystem. at the sending or receiving ends.

Applications: App sends 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.

chunk 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

↳ engineering trade-off.

① How much to put

② How easy is it to perturb the n/w. to empl.

③ Impact on apps that don't need it: real-time apps.

for.  
- choose to use vs. not

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

Consequence: if applied dogmatically → there is no reason to put any complex/ ~~use~~ for in the n/w.  
↳ the ultimate end

Candidates for end: Transport, application, user

used for →  
all practical purposes.

So why the overt reliance on transport

↳ ① cost

↳ difficulty of app designers

② error rate within end system is negligible.

leads to (3) ~~app clients useful for an unreliable net.~~  
burst issues: if the app did not do it, the who should? → someone in the local system as the app has a better understanding of local system's reliability than other parts of a n/w.

Consequences: transport protocol | router design - common case fast  
Should the n/w not do anything at all?  
Should it be largely dumb?

Some arguments for implementing functions within the n/w:

① Efficiency: multicasting  $\leftarrow$  duplication  
caching → improve efficiency by aggregating info. about multiple users sharing in common.  
→ Routing alternatives

② Disruption: email → SMTP servers...

③ Economies of scale: forwarding as opposed to broadcast.  
→ cost: source routing overhead.

Some arguments for moving stuff out of the n/w and into end-hosts:

① speed and simplicity of n/w infrastructure

② Debugging and troubleshooting of problems is easy

③ Easier to model and understand n/w

④ n/w impl: too egalitarian

end-impl: only those who want it, pay for it.

Connecting up multiple n/w.

Some other key principles in Internet design and protocols.

Original: effective techniques for multiplexed utilization of existing n/w

① Packet switching: existing n/w were packet switches!

② What does effective mean?

- Survivable
- Support for multiple types of services
- Accommodate a variety of n/w.

Survivable: - comm. must go on despite n/w failure

- fail only if n/w is partitioned

Implication: state in the n/w → ~~more~~ needs protection.  
move it all to end-host

failure →  
indicate to hosts  
but too complex

comm-fails only if one of the end-hosts that participates

fails: fail sharing.

Acceptable to lose state info about an entity if at the same time the entity itself fails.

↳ again where is the end-host.

↳ somewhere in the comm. subsystem but not in the n/w.

loss of state  
causes  
applications  
to fail

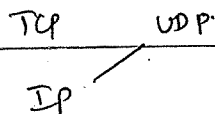
Consequences: - Stateless n/w elements

- large amount of trust in end-hosts

Types of service: TCP/IP was one

- not suitable for many apps

↳ debugging  
↳ real time



Incorporate and use

Support for variety of n/w.

- ① minimum set of assumptions about the fns. the n/w will offer → "transport a datagram" that all that is needed

A number of services are explicitly not required

End-to-end principle and fate sharing today

- ① Transport / congestion control
- ② Routing and forwarding
- ③ DoS
- ④