CSE 588: Network Systems

Discussion Notes: Introduction

Overview

Here's the challenge: design your own Internet.

Design challenge: build an Internet (network of networks) capable of scaling to:

How many hosts?

A trillion hosts, spread over the world (solar system?)

How many organizations?

Millions

How many device manufacturers/software companies?

Thousands

Which applications?

Every conceivable application that uses communication

Which kinds of networks?

Every conceivable communication medium

Anything else?

Efficiency, cost of ownership, reliability, availability, scalability, evolvability

OK, where do we start? How about priorities. Engineering is all about tradeoffs, so need to say what we're willing to sacrifice.

We can revisit, but I'd like to make a first cut at these today. We can refer back to them as we proceed, to see if we agree they make sense. Not what Clark said – what are our priorities TODAY.

Suggested priorities (in no particular order):

Availability (99% or 99.9999%)

What would it take for us to trust the red phone to the Internet?

Scalability

100+ million hosts now; doubling each year. So by 2010, 20B; 2015, T

Reliability (don't communicate wrong stuff)

Security (authentication and DoS)

Supports variety of applications: best-effort, real-time, multicast, …

Evolvability
    How do we change our minds?

Interoperability with existing networks; support future networks

Flexibility
    Ability to support variety of

Many administrative domains

Performance efficiency

Cost-effectiveness for ISPs

Cost of ownership for end hosts

Accountability (related to security)

Accommodate heterogeneity in workload (highly bursty – e.g., when Lady Di's car crashed)

Accommodate heterogeneity in provisioning (from 1Kb/s wireless modem to 10Gb/s long haul link)

Responsiveness/low latency

Performance predictability/stability
    Example: if add hardware, performance should improve
    Example: if subtract workload, performance should improve
    Example: if add workload/subtract hardware, performance should gracefully degrade

What architecture should we use to meet those priorities?

Network of networks – really the only way to have interoperability. How do we connect two networks of different types? Need a box that intermediates. Called a gateway, or more typically today, a router.

Digression: what are the physical properties of networks?

Deliver packets between two points

Vast heterogeneity, from 1Kb/s to 10Gb/s

Wire or wireless

Simplex or duplex

Bus or switched

Some are circuit switched, and some are packet switched (circuit switched means reserve bandwidth for duration of connection)

Packets can be corrupted (Shannon/Nyquist, clock synch)

Packets have some limited size (typically)

Packets can be reordered, duplicated, lost (why?)

Some provide support for real-time guarantees (e.g., ATM, FDDI)

Some guarantee delivery if there are no bit errors (e.g., single segment Ethernet, AN2), some don't (e.g., ATM, switched Ethernet)

Some provide support for multicast/broadcast (e.g., Ethernet), some don't

Architecture: how do we assign responsibility?

What should a host do?

What should a network be able to do?

What should the Internet be able to do?

What does a router need to do?

Tasks:

Routing

Corrupt packet detection

Lost/corrupt packet recovery

Ordering

Duplicate suppression

Connection establishment

Address assignment

Naming

Congestion control/resource allocation

Resource reservation

Multicast delivery

Security

Mobile hosts/mobile routers

Deadlock avoidance

Monitoring and repair of failed hardware

Performance monitoring

Deployment of new protocols

Computation/transformations

Internet choice: minimize assumptions made of underlying network. Is that still valid? Design to lowest common denominator?

Digression: ATM interaction with IP packet size and tail-dropping

Other alternatives? Telephone network supports dumb phones and smart switches.

What about boundary between organizations? Border router.

What should each organization do?

What should network of organizations do?

Other architectural components:

Client application

Transport

Name servers

Caches

Middle boxes (NATs, load balancers, firewalls)

Server application

Trusted third parties (authentication servers, PKI servers)

Human operators

What should each of these components be responsible for?

What are the design principles we should use in developing protocols? (Some of these are principles, some are techniques)

Put functionality at lowest layer where it can be completely implemented
Agree/disagree?

What does Salzer say?

Why does every network implement its own error checking? (e.g., Ethernet)

End hosts should be responsible for error recovery
Lower layers can do it as an optimization

Use soft state wherever possible: robustness via self-healing
Assume failures are the common case

Keep local decisions local; use hierarchy wherever possible

Keep receivers simple where possible (even if senders become more complex)

Let receivers decide where possible (e.g., multicast, real-time)

Let applications decide where possible (ALF, fragmentation)

Globally usable names and addresses

Exploit shared learning

Bad news should propagate quickly; good news should propagate slowly

Be liberal in what you accept, conservative in what you send

Protocol specifications should be explicit and backed by working code

Every network element should enable/allow deployment of new functionality

Keep interfaces simple
    Ex: TCP byte vs. packet confusion

Reduce dependencies between components
    Minimize trust

Explicitly verify information provided by peers
    Be skeptical

Protect resources
    SYN cookies

Contain faults/Defense in depth
    BGP route damping, TTL to catch loops
    Ex: route updates should not lead to loops, but use TTL anyway

Expose errors
    TCP checksum, silent problems

Use incentives
    If no incentive, no one will fix problem