Architectural Principles of the Internet

Mark Handley
ACIRI
mjh@aciri.org

Aim of this Talk

- **To provide a high level overview of:**
  - How the Internet actually works
  - How it got to be this way
  - The design principles that are important to the growth and evolution of the Internet, and why they're different from traditional telephony networks.

- **This is intended to set the scene for discussions over the next couple of days.**

Computers and Humans are Different.

- **Communication between machines is very bursty.**
  - Want high speed and low latency.
  - Most end-systems communicate rapidly for short periods and are then idle for long periods.
  - Most end-systems talk to a large number of other end-systems, sometimes simultaneously.
  - These trends are increasing as technology improves.
- **Human voice communication is much less bursty.**
  - Talk slowly
  - Context-switch poorly.
  - Need continual feedback for acknowledgment of understanding.
  - Can’t cope with simultaneous conversations.

Different Systems, Different Networks?

If you design a network for human voice communication, it looks a lot like the telephone network:

- **Switched, low bandwidth circuits.**
  - No simultaneous conversation requirement
  - Continuous transmission and feedback requirement
- **Very predictable service -**
  - Should sound like communication through air

But it's very poor for communication between machines.
It's also pretty poor for non-voice communication between humans.

Different Systems, Different Networks?

If you design a network for machine communication, you have different priorities:

- **Extremely low cost of communication setup.**
- **Support short high bandwidth communications.**
- **High reliability**
  - because machines are dumb
- **No strong predictability requirement**
- **Need to be able to communicate with many end-systems more or less simultaneously.**

If you only satisfy these requirements, this is pretty poor for synchronous voice communication between humans (not predictable enough). However, it's just fine for asynchronous communication between humans.

The Internet is a Datagram Network

- **Datagrams carry their own addressing information**
  - Assumes switches have enough knowledge to route the datagram to the next switch
  - No connection setup is required for communication to take place.
  - The network state can change after sending, and a datagram should still be able to arrive.
- **Datagrams are independent of each other**
  - No two datagrams need to take the same route.
- **There's no guarantee of when a datagram will arrive at the destination.**
  - The network doesn't know in advance when communication will take place.
- **There's no guarantee that a datagram will arrive.**
  - This allows the network to locally shed load.

Key Principles

- **Robustness**
- **Heterogeneity**
- **Ability to Evolve**
- **Administrative Autonomy**
- **Ease of Interconnection**
- **Distributed Architecture**
- **Cost Effective**

Robustness

- **Preventing Failure vs Designing for Failure**
  - Both of these can work.
- **However preventing failure only works if:**
  - Evolution is slow.
  - The network is fairly homogeneous.
  - There are minimal feature interactions.
  - Standards are centralized.
  - Administration is fairly centralized.
  - Interconnection is very carefully engineered.
  - Network growth is carefully planned.
- **If any of these is untrue, careful engineering of everything will not work.**

Robustness

How do you make a communications network robust?

Historically, there are two examples:

- **The telephone way:**
  - switched (virtual) circuits.
  - very careful engineering to keep circuits up.
- **The Internet way:**
  - assume failures will happen.
  - minimal fate sharing.
  - use datagrams.
  - adaptive routing copes with failures.
  - provide end-to-end reliability at the end-systems.
  - graceful degradation

End-to-end Principle

- *Functionality that can be provided end-to-end (ie by the end-systems) should be provided end-to-end. The network should only provide functionality to get datagrams to the required end-system(s).*

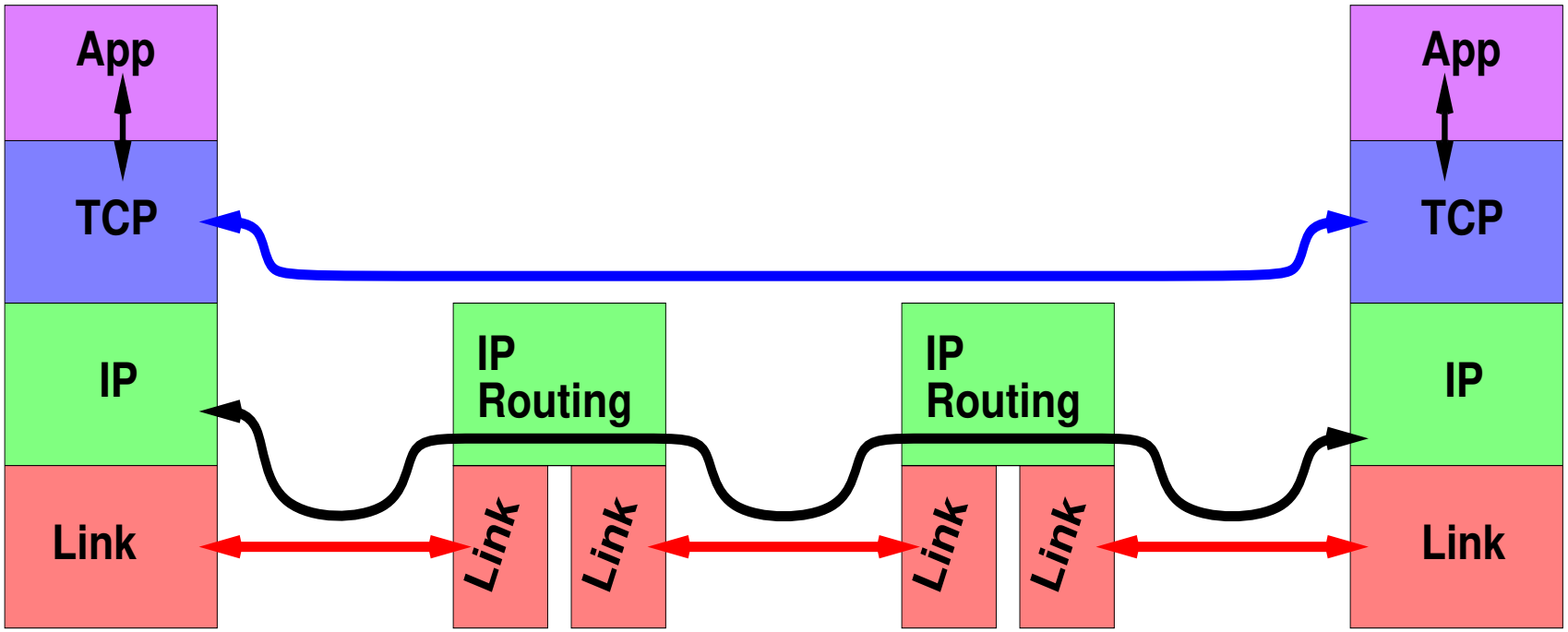
- **The argument is one of fate-sharing:**
  - If a failure happens in the network, end-to-end communication should be able to continue so long as an alternative path is available.
  - This means that we don't trust the network to always deliver datagrams, and don't assume they always get there uncorrupted.
- **The end-systems monitor reliability, and resend when data is missing or corrupted.**

Layering

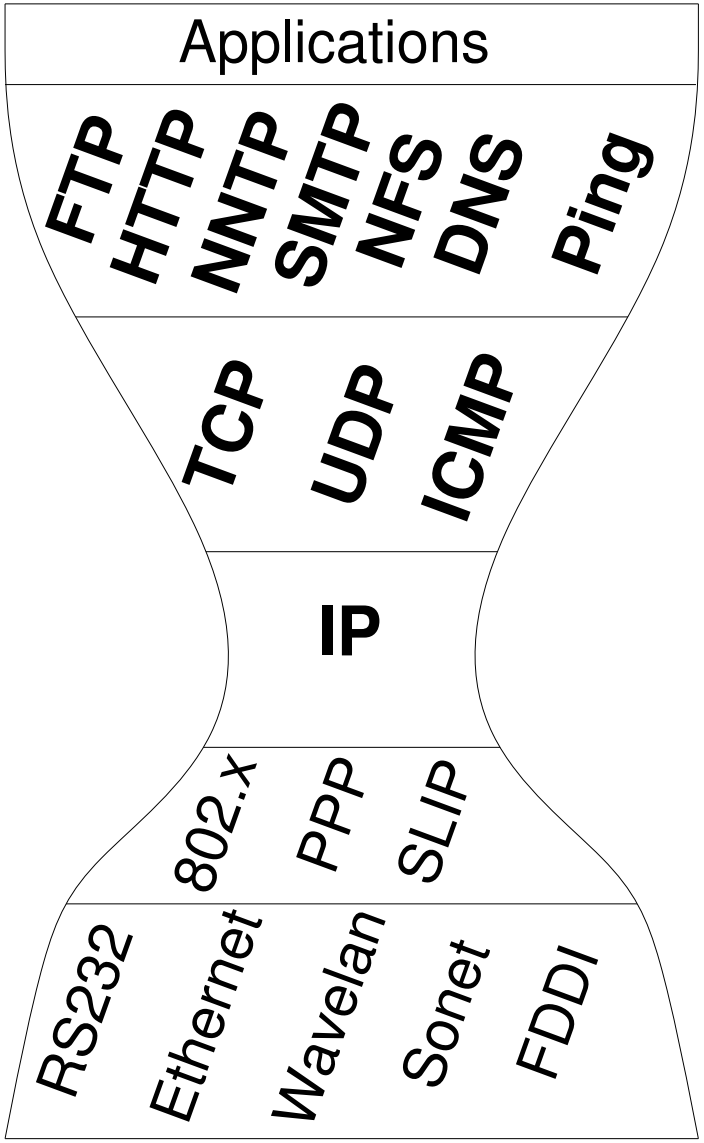
The end-to-end principle gives guidance on which functionality should occur in which network elements. This in turn influences protocol design.

- **IP is used to get packets from one end-system to another across many different heterogeneous physical networks.**
- **TCP is used to provide end-to-end reliability (and congestion control)**
  - Routers shouldn't need to know about TCP.
- **UDP is used for unreliable applications**
  - Or ones that require a different concept of reliability than TCP.
- **Routing protocols are used between routers to convey reachability information.**

Layering


App
TCP
IP
Link
IP
Routing
Link
Link
IP
Routing
Link
Link
App
TCP
IP
Link

Layering (cont)



Layering (cont)

- **Protocol layering is an important abstraction because it allows a degree of independent evolution of layers.**
  - Eg, TCP doesn't care whether IP runs over Wavelan or FDDI.

- **However, this does require that the higher layers are *adaptive***
  - TCP adapts the transmission rate to cope with whatever bandwidth is available.
  - TCP adapts its retransmission timers based on measured RTT.
  - DNS uses performance measurements to choose a root name server.

IP

- **IP is the only protocol that all nodes in an internet need to understand.**
- **It provides minimal functionality:**
  - Addressing
  - Best-effort datagram transmission
  - Header checksum
  - Fragmentation
  - Loop protection
- **Minimal assumptions about the link layers.**
  - No assumptions of delay, reliability, bandwidth, MTU, bit-errors.
- **Does assume:**
  - Globally unique addressing
  - Routing has been set up

Best-Effort Service

- **IP provides *best effort* service:**
  - No hop-by-hop reliability
  - No a-priori scheduling of network resources
  - Statistical multiplexing of link capacity

- **Thus at some times, more traffic will arrive at a router than it can forward down a link.**
  - This traffic is queued.
  - Queues are bounded.
  - Sometimes they overflow.

- **Packet loss due to congestion is a normal part of Internet behaviour.**

IP and the Key Principles

- **Robustness**
  - IP does not assume or provide reliability.
  - Transport layers do end-to-end reliability.
- **Heterogeneity**
  - IP works across virtually all link layers.
  - Transport layers adapt to observed network.
- **Ability to Evolve**
  - Layering helps evolution.
  - Evolving IP itself is somewhat more difficult.
- **Administrative Autonomy**
- **Ease of Interconnection**
  - Cooperation about addressing and routing
- **Distributed Architecture**
  - Only IP address allocation is centralized.

Transport Layer Adaptation

The transport layer must:

- **Measure Round Trip Time (RTT) to know when to expect responses from the other end.**
- **Adapt to use available bandwidth.**
  - Efficiently
  - Fairly
- **Adapt packet sizes to avoid unnecessary fragmentation of packets.**
  - MTU discovery

The most important of these is adapting to available bandwidth:

- **Congestion Control**

Congestion Control

- **Because resources aren't reserved before an attempt to use them, congestion can and does occur.**
  - Primary symptom of congestion is packet loss.
  - Also use of explicit congestion notification.

- **End-to-end transport protocols take packet loss as an indication of congestion and reduce their rate accordingly.**
  - The use of explicit congestion notification bits does not change the basic statistical multiplexed FIFO model.
  - We could use alternative queuing models such as Fair Queuing in routers at the expense of per-flow state. In practice this isn't done much.

Congestion Control

- **Without explicit rate feedback, the only way to get good network utilization is to aim to drive the network into congestion but aim to not drive the network unnecessarily far into congestion.**
  - Explicit rate feedback is hard because the traffic is not very predictable - network traffic exhibits fractal properties.
  - Traffic aggregates are not more predictable as the degree of aggregation increases.
- **To get fairness, we're looking for a dynamic equilibrium between attempting to increase the load, and backing off.**
  - TCP uses linear increase with no loss, exponential decrease when loss is detected.
  - Sally Floyd will talk more about this later.

Naming, Addressing and Routing

Internet Engineering rule of thumb #7:

"Flat until it stops scaling, then add hierarchy"

This applies to routing, addressing, naming, ...

- **The Arpanet address space was flat.**
- **The Internet address space used to consist of flat space of network addresses and within each, and a flat space of host addresses on each network.**
  - Only needed routing information for networks not hosts.
- **Now there is hierarchy with topological significant in the network address parts.**
  - Routing information can be aggregated to reduce forwarding tables

Internet Routing

The general idea of Internet routing is to propagate to all routers sufficient information that every router knows which direction to route a packet to reach any destination.

- Must adapt rapidly to link or router failures.

- **Originally Internet routing consisted of propagating distance vectors to all destinations.**
  - Flat distance vector scales poorly - takes too long for routing to converge after changes, and in a large network there are always changes.
- **Newer (circa 1980) Internet routing consists of propagating link-state information to all routers.**
  - Responds rapidly to changes
  - Too much state to scale to big networks

Internet Routing Today

Internet routing consists of a two-layer hierarchy of routing and a multi-layer hierarchy of address allocation.

- **BGP (Border Gateway Protocol) does inter-domain routing.**
  - This is a path-vector mechanism.
  - It relies on reasonable address allocation so that routing tables can contain aggregate routes.
- **Many different protocols function within domains.**
  - Link state algorithms are frequently used.

Biggest problem:

- **Insufficient capability for traffic engineering (efficiently balancing traffic across the parallel paths in a well-connected mesh network)**

Naming vs Addressing

The *Ability to Evolve* principle applies to networks as well as protocols. For example it must be possible to:

- **Move functionality between hosts**
  - Eg when the web server gets too busy for the old hardware
- **Change provider**
  - This requires renumbering if the routing is to aggregate properly.

Thus we need to separate the name of a service (what you ask for) from the address of that service (where you go to get it).

- **The Domain Name System (DNS) does this name-to-address mapping.**

Domain Name System

DNS is a good example of Internet design principles. It's a distributed directory for mapping names to addresses and addresses to names.

- **The hierarchy of names is reflected in a hierarchy of name servers.**
  - Eg. **`www.ucl.ac.uk`**
    - root nameserver gives nameserver address for **`.uk`**
    - **`.uk`** nameserver gives nameserver address for **`.ac.uk`**
    - **`.ac.uk`** nameserver gives nameserver for **`.ucl.ac.uk`**
    - **`.ucl.ac.uk`** nameserver gives the actual host address for **`www.ucl.ac.uk`**
- **Distributed administration of servers and namespace.**
- **Replication of functionality at every level for robustness.**
- **Caching of responses makes performance acceptable.**

Evolving the Internet

The Internet is currently evolving in a number of different ways:

- **Multicast Service**
  - Many-to-many communication
- **Quality of Service Enhancements**
- **Securing the network**

Multicast

IP Multicast provides many-to-many data distribution.

A sender sends a packet once, and routers replicate the data so that it reaches all interested receivers. The way this works follows the general design principles of the Internet:

- **Senders send datagrams to a "virtual" destination address that represents the group of receivers.**
- **Receivers simply "express an interest" in receiving traffic destined for that group address.**
- **Routers conspire to deliver datagrams from senders to receivers.**

Senders don’t need to inform the network before sending, know where receivers are, or know how the routing actually takes place.

Multicast (cont)

It's hard to make inter-domain multicast routing scale. Originally multicast addresses were a flat space.

- **Current proposals dynamically allocate multicast addresses in a hierarchy that reflects the underlying network topology and provider peering relationships.**
- **Routing then uses the implicit structure in the address space to build a data distribution tree from senders to receivers.**
- **Scalable and robust multicast address allocation is critical to this process.**

Multicast is not currently a ubiquitous service. Even when it is, it requires per-group forwarding state in network routers - something that unicast IP normally tries to avoid.

Multicast and Robustness

Although multicast itself has scaling issues if too many multicast groups are used, multicast is a very valuable mechanism for helping other protocols be robust, distributed and adaptive.

- **This is because senders and receivers do not need to know each other's locations to bootstrap communication.**
- **Multicast groups provide a level of indirection that permits unicast mechanisms to be self-configuring or self-reconfiguring.**

Multicast and Large-scale Data Distribution

- **IP multicast is receiver driven (receivers explicitly join a group), and so changes to the multicast distribution tree can be made locally to the receivers that join or leave.**
  - This means that IP multicast groups can easily scale to very large numbers of receivers.
  - However IP is unreliable, and so IP multicast is also unreliable in that packets may be lost.
- **How to do scalable *reliable* multicast and scalable multicast *congestion control* are the two big challenges for using IP multicast.**

Quality of Service Enhancements

The basic internet design principles result in a network that is very unlike the telephone network in both architecture and service. What happens when we want to carry telephony-like traffic?

- **Packet loss is a problem.**
  - Caused by congestion, and a basic property if we want good utilization.
  - Caused by re-routing.
- **Jitter (delay variance) is a problem.**
  - Caused by variations in queue size.
  - Caused by re-routing.

Quality of Service Enhancements (cont)

We don't want to abandon the basic design principles, but we can provide enhanced service if the end-systems are prepared to signal their requirements, and if we allow fall-back to best effort when the enhancements fail.

- **QoS enhanced service is more fragile than regular service because there's more fate-sharing and the network cannot easily cope with failure and seamlessly maintain enhanced service.**
- **QoS enhancements will typically mean that the enhanced flows do better at the expense of unenhanced flows (it's a zero-sum game).**
  - This normally requires some form of negative feedback, usually in the form of pricing.

Quality of Service Enhancements (cont)

Different forms of QoS enhancement affect the basic properties differently.

- **Signalling hop-by-hop the precise requirements of a flow and wiring that state up in the routers takes us towards a virtual-circuit network.**
  - Lots of state in the network.
  - Fragile.
  - Close control over the service quality.
- **Using a few bits in the packets, and policing the flows at a few points in the network is much closer to the basic internet model.**
  - Less state in the network
  - More robust because the network has more room to adapt.
  - Less tight control over the service quality.
  -

Quality of Service Enhancements (cont)

Is loose control of QoS acceptable?

- **This is a network of computers.**
- **Smart end-systems can adapt to the state of the network as observed by the data flow.**
- **Jitter can be removed by adaptive buffering at the receiving end-systems at the expense of adding delay which is proportional to the jitter.**

Almost all applications don't need to have precise control of jitter - they just need it to remain within reasonable bounds almost all the time. The remaining jitter can be removed at the receiver.

Summary

The Key Design Principles of the Internet are:

- **Robustness**
- **Heterogeneity**
- **Ability to Evolve**
- **Administrative Autonomy**
- **Ease of Interconnection**
- **Distributed Architecture**
- **Cost Effective**

These principles tend to result in a network that can grow to a large scale and can also grow rapidly without centralized planning or understanding of what the network will look like in the future.

- **They are perhaps necessary, but not sufficient.**