Naming & Identification in the Internet

R. Atkinson & S. Bhatti
UCL Computer Science
July 2004

Existing Name Types

- IP Address (*1.2.3.4*)
- IP Subnet (*1.2.3.0/26*)
- Domain Name (*host.cs.ucl.ac.uk*)
- Communication End-Point or "Socket" (*TCP port 23 @ host.cs.ucl.ac.uk*)
- Mailbox (*a.lastname@cs.ucl.ac.uk*)
- URL (*http://www.cs.ucl.ac.uk/index.html*)

Quick History

- In the beginning were *Addresses*
- Then *hosts.txt* appeared, creating a flat namespace
  - (example hostname: *ucl-cs-host*)
- Needed heirarchy as network grew in size
- Which led to the *Domain Name System (DNS)*
  - *(*example hostname: *host.cs.ucl.ac.uk)*

Domain Name System

- Originally a simple mapping service between Fully-Qualified Domain-Name and Address (examples: A, PTR)
- Modern DNS usage examples:
  - explicitly to provide service location information (example: MX, KX, SRV)
  - implicitly to create service names via the CNAME record (e.g. *ftp.cs.ucl.ac.uk*)
- DNS overloading keeps increasing with many other directory services being added over time
- Mutation more than Evolution

Addresses

- Early application designers did not have DNS
  - hence BSD's Sockets API used raw IP addresses,
  - hence IP addresses were embedded in many applications and application protocols
- Class-full nature and ad-hoc allocation practices created excessive routing table growth rate, so switched to class-less addressing (CIDR) to minimise routing table growth rate and increase utilisation efficiency
- Network Address Translation (NAT) came into common use for various reasons

So what's wrong ?

- Example: DNS overloaded to implicitly name a service, rather than a host (e.g. www.cnn.com)
- Most networking APIs lack appropriate object types in their interfaces
- Community failed to use the right abstractions -- mostly for historical reasons
- Example: Modifying the Address because the device moved ought not have any impact on applications

What to do ?

- Revisit the naming architecture of the Internet, applying all we know today that was not known originally
- Consider adding additional namespaces
  - Service Names
  - Network-Layer host identifiers (not used for routing)
  - Others also, perhaps

Architectural Implications

- Addresses resume their original limited role -- basically used for routing only.
- Transport protocols and Application protocols substitute more appropriate identifiers for addresses
  - Might need to use raw addresses in a special instances (e.g. control messages)
- Networking APIs need to change to use proper abstractions

Strawman Approach

- Add a new ID resource-record to DNS
  - One-way mapping from FQDN to ID
  - Can use PTR lookup to get from Address to ID
  - Use DNSsec to authenticate (FQDN->ID) mapping
  - Secure Dynamic DNS Update to modify A records
- Add ICMP extensions
  - to obtain a FQDN hint from any remote system, etc.
- Modify other protocols to use ID, not address

Deployment

- Strawman described above is obviously partly-baked, not fully sorted out.
- Numerous obstacles exist to deploying a clean architecture
- Nonetheless worthwhile to devise a clean architecture
- Useful to think about which architectural approaches might be easier/harder to deploy

Benefits: Routing

- Improved Internet routing system
  - Mobility is easy because transport-protocol state and application state bind to host's identity, not address
  - Multi-homing is easy because transport-protocol state and application state bind to host's identity, not address
  - DFZ not impacted by multi-homed sites
- Changes to addressing/routing system do not impact host identity or user applications
  - In-transit address modifications (e.g. NAT) do not have any impact outside the routing system

Benefits: Security

- Eliminates need to use unauthenticated addresses as host identifiers
  - Instead, use new authenticatible host identity
- Facilitates deployment of cryptographic security (e.g. IPsec, Routing Authentication)
  - IPsec would work through a NAT trivially, without needing special consideration
  - IPsec would naturally work with truly mobile hosts or even mobile networks
- Facilitates improved firewalls

Benefits: Other

- By adding Service Names explicitly, service location should be much easier
- By reducing overloading of semantics, the architecture becomes much cleaner
- By having a cleaner architecture, the programming APIs can use better interface objects
- In turn, this should make application development easier and faster

Questions ?