Name-Based Networking

Dave Thaler
dthaler@microsoft.com

We already have an ID/locator split in hosts…

- From “Architectural Principles of the Internet” [RFC1958], section 4.1:
  - “In general, user applications should use names rather than addresses.”

- ***What if that were true!***

- How far can we get with the notion that most applications shouldn’t ever see addresses?
  - Applications deal with IDs
    - ID == name
  - Routing deals with locators
    - Locator == IP address

What about OS support?

- Trend: Most new apps now use higher layer APIs/frameworks, NOT classic sockets
    - Java, P2P frameworks, SOAP, RESTful web services, JavaScript, websockets, etc.
- Even new versions of many existing apps are moving
- These generally use names NOT addresses (e.g. connect-by-name semantics)
- This means we can already do a lot of things **without changing apps** and **without new APIs** beyond those already emerging

- Question:
    – Can we just concentrate on fixing the name/address split?

Host Mobility 1/2: Accept new connections right after a move

Q: So what’s the problem?

A: Mainly design limitations of current solutions:

- Many hosts have no name in the DNS today
  - Can be solved with DNS dynamic update and a relationship with a DNS provider
- Inability of name resolution (DNS) to deal with rapid changes
  - Some DNS servers don’t respect small TTLs
  - But there’s already a push to update them for DNSsec
- Addresses are cached by applications and services
  - Applications don’t respect TTLs either
  - But remember app trend

Host Mobility 2/2: Preserve established connections

- Locators change over time
- There can also be periods of complete disconnectivity
  - Travel between work and home (long)
  - Ride in an elevator (medium)
  - Just walk past a cement pillar (short)
- **To deal with disconnectivity, some layer must do a reconnect transparent to the user**
- There are often user experience benefits to applications handling disconnectivity themselves

So if apps or some layer below does reconnects, is this sufficient?

- For non real-time interactive (email/web/IM/…), probably!
- For real-time interactive (e.g. VoIP), arguments for no seem to be current design limitations, not inherent
  - Name often not available below the app (but see app trend)
  - Long reconnect time for DNS + TCP
  - Inability of name resolution (DNS) to deal with rapid changes
  - Inability to communicate predicted name-to-address changes
- Claim: All of the above can be addressed without any new ID/loc split inside the host
  - Questions then are whether it's less *problematic*, *easier* to deploy, and has incentives better aligned

Adding *another* ID concept still has the same problems (again)

- How secure binding from ID to locator?
- How deal with dynamically changing locators?
- How deal with multiple locators?
- How deal with period of disconnectivity?
- How provide high availability & DoS resistance?

- If we need change hosts (or even apps), can’t we just ride the existing trend and fix the name/addr split?

To complete a name-based solution, hosts want

- Relationship with “dynamic DNS” provider
- Apps (& protocols) that use names not IP addrs
- App- or (preferably) session-layer reconnects
- Optimized reconnect time for DNS & TCP
- DNS servers and API frameworks that respect small TTLs
- Ability to communicate predicted name-to-address changes

- Some, but not all, of the above may be IETF items