Why do we really want an ID/locator split anyway?

Dave Thaler
dthaler@microsoft.com

Starting from basics

- Users deal with *names*, not addresses (esp. in IPv6)
  - Humans need “friendly” identifiers that can be remembered and typed
  - Name = who (informally) you are
- Security deals with *identities* that can be used as principals
  - Identity = who you are (really!)
- Routing deals with *locators* (IP addresses)
  - Locator = where you are
- Applications deal with *identifiers*
  - May or may not be the same as one of the other terms above

Problematic Trends

- Trend #1: Mobility = locators change over time
  - Laptops and PDAs roam to WiFi hotspots
  - Sites change ISPs
  - Even entire networks can move around
- Trend #2: Multihoming = multiple locators at same time
  - Laptops have both wired and wireless interfaces
  - Phones with WiFi + GPRS/etc
  - Even if device has only one interface, the network may be multihomed to two providers for failover/redundancy

Mobility Basics

- If routing had no scaling or convergence time limitations, mobility could be handled by routing
  - Just use dynamically updated host routes
- If name resolution had no scaling or convergence time limitations, mobility could be handled by name resolution
  - Just use dynamically updated name records

Common idea: ID/locator split

- Separate identifiers used by apps from addresses (locators) used by routing
- Examples:
  - Separate IP address seen by apps: Mobile IP, SHIM6, HIP, etc
  - Separate IP address seen by edge vs backbone: NAT, LISP, etc

Actually, we already have an ID/locator split…

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

- Applications deal with IDs
  - ID == names (but not all apps)
- Routing deals with locators
  - Locator == IP address

Another trend

- Trend #3: App/protocol frameworks
  - Most new apps now use higher layer APIs/frameworks, NOT sockets
    - Web services, Java, P2P frameworks, etc.
  - Even new versions of many existing apps are moving
  - These generally use names not addresses (e.g. connect-by-name semantics)
  - This means you can do a lot of things without changing apps

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

- Let's look back at some goals and see...

Host Mobility 1: Accept new connections right after a move

Q: So what's the problem?

A: Mainly design limitations of current solutions:

- 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 trend #3

Host Mobility 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 usually user experience benefits to applications handling disconnectivity themselves

So if apps or layers below do 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
  - 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
  - Questions then are whether it's less *problematic*, *easier* to deploy, and have incentives better aligned

Site Mobility: Ease Renumbering

- Motivates provider-independent addressing, impacting routing table growth
- Renumbering pains depend on how many places addresses are configured:
  - Routers
  - Hosts
  - DNS servers
  - DHCP servers
  - Firewall
  - Remote monitoring systems
  - Intrusion detection systems
  - Load balancers
  - Management tools/databases
  - Etc.
- Whether renumbering is any easier or not with an ID/loc split depends how many of above have to change, and whether the change is just config or code
  - Existing name/addr split still requires most of them to change to renumber (trend #3 does not help!)

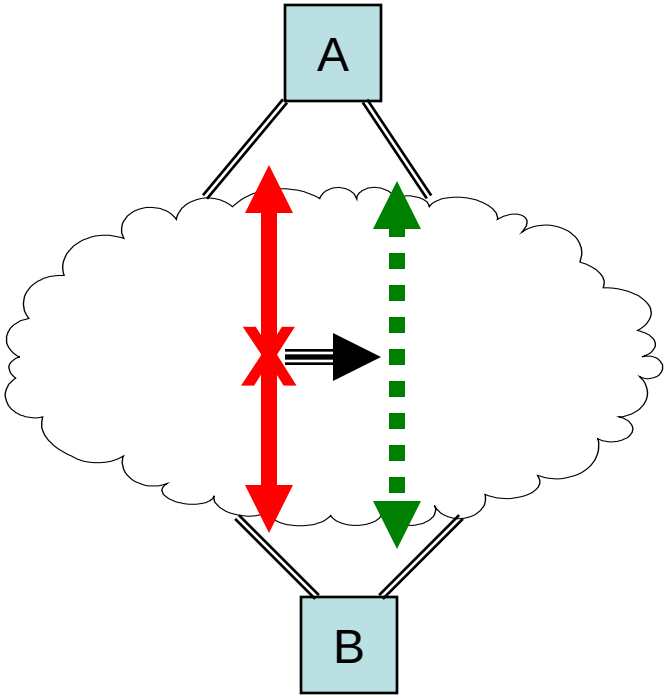
A note about management & security systems…

- These are often the last/hardest to change code
- Most of them assume upper-layer identifier == locator
  - Separation makes it harder for intermediate system to peek in and look at the identifier
- Unlike apps, you have to work with **all** of them before you can deploy in a real network
- Implies either blocked on changing them, or else must have identifier == locator within a real network

Multihoming: Support redundancy, load sharing, etc

- Named entities exist on machines with a set of locators
- Efficient load sharing & redundancy needs a locator set to be communicated somehow
  - One end chooses which locators are communicated
  - Other end chooses among locators communicated
- Problems:
  - Various applications and protocols (TCP, SIP, etc.) today only communicate one address
  - They also don’t re-bind during connections
- Again fixable without ID/loc split either by a higher layer or by directly changing the protocols

Multihoming: Span outages



When a path breaks for a given pair of locators, can continue with another pair

Problems:

- Protocols and apps today don't do this
- How do you discover which pair works? (e.g. SHIM6 logic)

This doesn't require a new ID/loc split either, just a common ID (e.g. name), and reconnect logic

Ok so where are we?

- Claim host mobility/multihoming can be solved without a new architectural ID/loc split
  - But is it less *problematic*?
  - *Easier* to deploy?
  - Have incentives better aligned?
- Site mobility/multihoming can be solved either with
  - PI addressing, at expense of routing state/churn
  - Renumbering, which requires *many* things to change
  - ID/loc (actually loc/loc split) at border
  - Same questions apply…

Working with existing apps

- ID/loc split schemes typically motivated by either
  - Making existing apps work better
  - Optimizing for something else (e.g. route scalability) without breaking existing apps
- So let's look at some things existing apps do…
  - Note: "apps" here really means anything above IP
- Many apps have embedded assumptions (or myths, increasingly…)
- Making them less true can break apps
- Making them more true can "fix" apps
- Let's look at a few that are relevant to MobiArch

Addresses are stable over long periods of time

- Examples of behavior:
  - Apps resolve names to addresses and cache them without any notion of lifetime
  - Name resolution APIs don't even provide the lifetime
- Status:
  - Much less true with DHCP, roaming, etc.
  - PMIP trying to restore within a local network
  - MIP, HIP, etc trying to restore to some extent by adding an additional address that is stable
  - Over time, fewer applications directly assuming this (trend #3)

A host has only one address and one interface

- Examples of behavior:
  - Apps resolve name to address and just use the first one returned
  - Some apps use address to identify users/machines
  - Some DHCP options are defined as machine-wide
- Status:
  - Much less true with multihoming, dual-stack nodes, VPNs, etc.
  - MIP, HIP, etc trying to restore to some extent
  - Over time, fewer applications directly assuming this (trend #3)

An "address" used by an application is the same as the "address" used for routing

- A.k.a. “ID == Locator”
- Examples of behavior:
  - Apps make assumptions about locality (e.g., same subnet) by comparing addresses
  - Server-selection apps/protocols make assumptions about locality by comparing source address against configured ranges
  - Apps use raw sockets to read/write packet headers
  - IP address policies in security devices like firewalls

- Status:
  - Not true with tunneling, most ID-locator split schemes, etc.
  - Some ID-locator split schemes (LISP, etc) only break it in the core of the Internet so only affects apps running there
  - Trend #3 only partly helps (doesn’t help firewalls etc.)

E2E delay of first packet to a destination is typical

- Examples of behavior:
  - Applications “ping” candidate servers and use the first one to respond
  - May also apply to some P2P apps choosing “local” peers

- Status:
  - PIM-SM, MSDP, MIPv6, etc allow deterministic path switching during initial data burst
  - “Choice” of server can hence be highly non-optimal, resulting in longer paths, lower throughput, and higher load on the Internet
  - **ID/loc split schemes can cause problems here if introduce loss and/or delay in routers**

Identifiers work with referrals

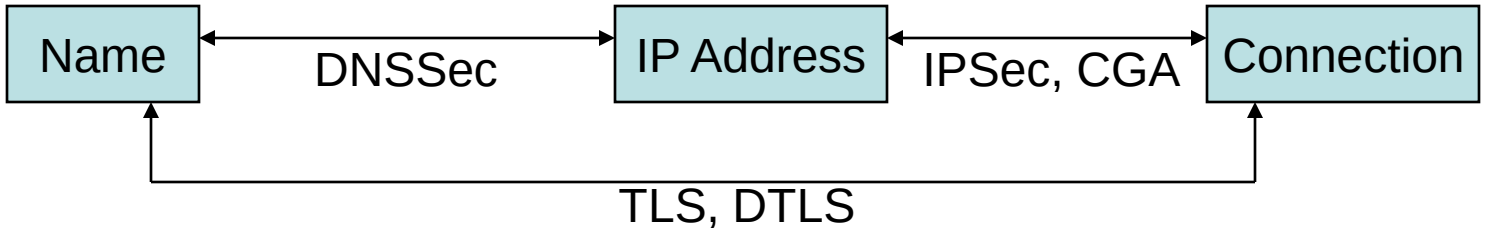
- One application/user/service wants to refer/redirect you to another one (or itself)
  - Why not just use a name? (example: HTTP redirect URL contains hostname)
  - Inefficiency of subsequent name-to-locator mapping step
    - But refer/redirect could provide a locator hint
  - Other current design/deployment limitations:
    - Many protocols are defined to refer/redirect to IP address
    - Some apps might only cache addresses
    - Not all applications/users/services have a name today

Security Basics

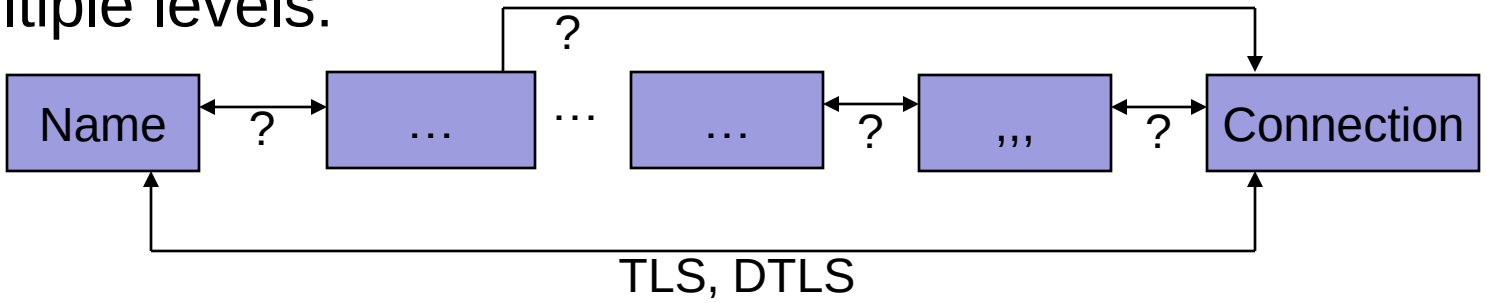
- Need a chain of trust from a user-friendly name to a connection
  - DNSsec alone is not sufficient if the locator can be spoofed
  - IPsec or CGAs alone are not sufficient if the name-to-locator mapping can be spoofed
  - If names are authenticated directly (e.g., TLS/DTLS) then any spoofing attacks are reduced to DoS
- Need a chain of trust from whatever an application starts from, to a connection
  - Not all applications act on behalf of humans (e.g., server apps)
  - Either application always needs to start from a name, or also need chains of trust from whatever other type of identifier is in use

Securing Mappings

Currently defined (examples):



Multiple levels:



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?
- Another point of failure / DoS opportunity

- If change hosts, is it really any better than just fixing the name/addr split?

Just changing boxes in the middle of the network

Key question: How & When do you learn the ID->locator mapping?

- A priori:
  - How much data has to be learned a priori?
    Is this any better than the original routing scalability problem?
    (If so, great!)
- Name resolution time:
  - But not all apps resolve names (server apps, referrals, etc.)
- At time of first packet:
  - **Forced to buffer/drop packets -> more apps break!** ☹
  - Control plane load caused by data plane may also cause problems
- Hybrid: Forward on alternate topology until resolve
  - Delay+Reordering -> some apps still break! ☹

Incentive Issues

- There must be positive net value at *each* organization where change is required
- Best if *only* requires changes by entities actually feeling pain, e.g.
  - Service Provider (Routers): routing scalability
  - End-user (Hosts): mobility, host multihoming
- Often **only one entity** experiences the pain, and so is incented to change
  - Best if provides actual benefits when **only** that entity is changed

Summary (1/2)

- Site mobility/multihoming:
  - Focus on mapping issue (loss? delay? state?)
  - Danger in new ID/loc split is causing harm to existing apps and/or networks
  - Any changes to hosts/apps has incentive issues
  - The only mapping distribution that doesn't have incentive or app issues is a priori
    - Opportunity for research on state reduction
  - Other possible research areas
    - Any ways to ease renumbering?
    - How bad is app impact today?

Summary (2/2)

- Host mobility/multihoming:
  - Non-real-time apps have to change to deal with disconnectivity
  - Apps are changing anyway for higher layer APIs
  - DNS changing anyway for security
  - **Can we just concentrate on fixing the name/address split per Internet principles?**
    - Maybe even use mobility & multihoming as another reason to move to name-based APIs that provide security etc
- Danger in new ID/loc split is causing harm to existing apps and/or networks

Thank you