Lecture 12 Overview

- Naming: end points vs. data/services
- DONA
Design Paradigm

● Internet was designed for host-to-host communication
  - “contact this host...”

● Internet is mainly used for data access
  - “get me this data.....”

● Mismatch between usage and design:
  - data migration and replication unnecessarily hard
  - requires Akamai- and BitTorrent-like designs to scale

● Question: what would the Internet look like if we designed it around data access?
What Do Users Care About?

- **Persistence** of names:
  - Follow data migration
  - Today: HTTP redirects, email forwarding

- **Availability** of data: (both latency and reliability)
  - Take advantage of replicated data
  - Today: Akamai/BitTorrent

- **Authenticity** of data:
  - Know that data came from intended source
  - Today: securing the channel (IPsec, TLS), or PKI
Current Barriers

- **Rigid and Weak Naming**: hostname/path
  - Ties data to host, making migration/replication hard
  - Doesn’t help much with authentication

- **Protocol Mess**: e.g., DNS, TCP, HTTP
  - Data isn’t named until the application is invoked
  - Caching (and other data handling) is application-specific
  - No low-level support for anycast-like service discovery
#1: Flat, Self-certifying Names

- Self-certifying (SFS, HIP)
  - Data associated with principal \( P \) with public key \( K_p \)
  - Names are of the form: \( \text{Hash}(K_p): \text{label} \)
  - Data requests return: \( <K_p, \text{label}, \text{data}, \text{signature}> \)
  - Can verify name related to \( K_p \), and \( K_p \) signed data

- Name not tied to location, so naming isn’t rigid
  - resolution mechanism described later
Better Separation of Concerns

- Names take care of persistence, authenticity
- Protocols can then focus on availability
  - latency
  - reliability
#2: Better Protocol Structure

- Implement replication and caching at lower level:
  - find closest replica without application-level tricks
  - caching infrastructure not application-specific

- DONA does this through two major changes:
  - insert data-handling shim layer right above IP
  - resolve name by routing to data (TRIAD)
    » better fate sharing
    » less complicated management

- No DNS, no lookup, just routing on names
New Network Entities

- **Data Handlers (DHs):** operate at data-handling layer
  - Called “Resolution Handlers (RH)” in paper
  - DHs do name-based routing and caching
  - A logical DH per administrative unit
    » think of AS hierarchy (and finer grain below)

- **Authoritative Resolvers (ARs):**
  - AR(P) can point to authoritative copy of P’s data
  - Think of as P’s DNS resolver
  - Solves the Tier-1 DH unknown issue…
New Network Primitives

- **Fetch**(name): data request
  - data name
  - transport header
  - application header

- **Register**(name): offer to serve data (authenticated)
  - Register individual data items: Hash(Kp):label
  - Register authoritative resolver: Hash(Kp):*
Overview

- Clients configured with local DH (replaces DNS) to which they send their fetch requests

- DHs respond to fetch if data is in cache

- Otherwise, DH routes fetch towards nearest copy of data by sending to a next-hop DH
  - can insert own address in fetch packet so that it receives data on way back

- If name isn’t in routing table, fetch routed towards AR
Routing Fetches

- Need to implement anycast routing at DONA-layer
- Use DH hierarchy to guide routing
DHs forward register commands to parents and peers
...Establish Routing State

- Arrows represent next-hop entries for registered data
- Scaling: DHs only hold state for items below
  - core: few TBs
  - edge: typically far less than a GB
Anycast Routing of Fetches

- If there’s an entry for a data item, follow next-hop
- Otherwise, send to parent
- Standard routing behavior, but at DONA-layer
DONA

- Naming makes it easy to authenticate data

- DONA-layer provides easy access to data:
  - name-based “resolution through routing”
  - caching and replication infrastructure

- DONA makes it easier to build transport, applications
Extensions

- Cache both data and fetches:
  - RSS-like behavior, cache invalidation

- Policy (tbd):
  - based on first packet, domains can decide to
    - deny
    - route through proxies
    - ask for more authentication
    - hand back capabilities
    - set up state
  - think of this as “off path” signaling (PF)
Open Questions

- Does this scale?
  - Boils down to money: edges cheap, core not-so-cheap
  - Big unknown: (compounded) cache hit rates

- Is caching and replica-selection too app-specific?
  - Authors’ guess is no, but the burden of proof is on them

- Is there an incrementally deployable version of DONA?
For Next Class…

- Read and review Entact paper
- Keep moving on term projects!