Announcements

- Presentations are not on the assigned reading
  - We all read it; no need to see a detailed talk on it
  - Pick from the optional reading, or email me to suggest another paper

- Paper reviews are not paper summaries
  - Keep your review to two short criticisms / comments which demonstrate that you read the paper
“All problems in computer science can be solved by another level of indirection.”

– Butler Lampson

so...

What two problems is this paper solving?
One: Services tied to hosts

• Today: application trying to access a service binds to an IP address
  • Ties service to particular end-host
  • When is this inconvenient?

• This coupling is unnecessary: application cares only about identifying the service it wants
  • Solution: Name maps to service identifier (SID), independent of machine hosting service
Two: Hosts tied to location

• Today: transport layer trying to reach a host binds to an IP address

  • Ties connection to physical location of end-host, and even to one interface to access that host
  • When is this inconvenient

• This coupling is unnecessary: transport layer cares only about identifying the host to send to

• Solution: SID maps to endpoint identifier (EID), independent of host’s location in network
2.4 Sequences of Destinations

In addition to sending to a single location, applications may need to send to a sequence of identifiers (either IP addresses or EIDs). In this way, both senders and receivers could loosely dictate the paths of packets sent from them. Although the IP layer cannot do this, the application architecture at the end-system might do so. In fact, many of the details here arose during an implementation effort, described in [60].

The two mechanisms are logically distinct and need not be coupled. Section 3, not every application will require both SIDs and EIDs. The discussion of intermediaries in Section 3.2 will complicate matters, but we note how it yields the three benefits stated in Section 1, namely making services and data first-class objects, support for mobility, and graceful incorporation of intermediaries.

In traditional IP routing, the routing protocol is responsible for forwarding packets but may act on them in non-trivial ways. Destinations, as specified by sources and receivers, could loosely dictate the paths of packets sent from them. The application architecture at the end-system might do so. Although the IP layer cannot do this, the application architecture at the end-system might do so. In fact, many of the details here arose during an implementation effort, described in [60].

Consider an SID-aware application. We now give more details on how these resolution layers are used, starting with the application. The four general principles led us to claim that (1) two additional sets of names (SIDs and EIDs) should exist, (2) these names should be flat, (3) the architecture should support delegation as a basic primitive, and (4) destinations, whether specified by the source or transport layer, should view this architecture merely as an existence proof that the general principles can be realized, not as their definitive embodiment. In fact, many of the details here arose during an implementation effort, described in [60].

We should first note that our belief in the general principles was a consequence of how to achieve the benefits that flow from these principles, but one can be critical of the details that follow. The description below is intended to illustrate the general principles.
What’s in a name?

- Goal: persistence
- So, name shouldn’t encode unnecessary information
  - e.g., organizational affiliation might change

~ Shakespeare Approves ~
“What's in a name? That which we call a rose
By any other name would smell as sweet.”
- Juliet

- Solution: flat names: just a string of bits
Self-certifiable flat names

• If name is a public key or secure hash of a public key, source can verify identity of remote party

• Alice gives Bob a link to SID x
• Bob contacts x, begins conversing with party P
• Bob challenges P: “prove you have private key associated with public key x”
• If successful, Bob knows P is party Alice meant to contact (assuming private key not compromised)

• DNS solution: P sends public key x; trusted 3rd party certifies binding between x and high-level name
Resolving flat names

• Challenge: scalability
  • flat means not hierarchical
  • hierarchy is what helps DNS scale

• One solution: distributed hash tables
  • Distributes hash table functionality (e.g., SID to EID mapping) across arbitrarily large
  • We’ll see how DHTs work on Nov 2
Implications of decoupling

- Human-readable user-level descriptor (ULD) decoupled from service ID (SID)

- One implication: can accommodate many ULD resolution services

- A possible extreme case: everyone is their own ULD resolution service
  - i.e., ULDs have local meaning only
• Each person uses names that have meaning to them
• One name may map to different objects depending on who is resolving the name
• One object may have many personal names associated with it
Creating personal names

- **Personal group**: consistent names across devices
  - **Challenge**: must keep view of group consistent across sporadically-connected devices
  - **Challenge**: what happens if you lose your phone?
Connecting personal devices

- Want to refer to and connect to my phone, laptop, camera, from anywhere
  - Why is this not possible today?

- Solution:
  - Name bound to persistent self-certifying flat EID
  - Overlay network connects devices, preferring social links as neighbors in overlay

- Challenge:
  - Overlay network has no regular structure
  - How do we locate EIDs in the overlay?
• One solution: token-based flooding

• Basic problem: routing on flat names. See October 5 discussion!
Take-away: 2 primitives

- **Separation of identity and location**
  - Conflated in current Internet
  - e.g., apps need identity but must bind to location

- **Self-certifying names**
  - One use of flat names that is enabled by locator/identifier separation