Identifying Communication Partners

- **Names**
  - Human readable identifiers that can be remembered!
    (e.g., DNS name, URI, URN)

- **Identifiers and addresses**
  - Machine-processable identifier (e.g., Host Identity, HI)
  - Protocol-level identifier (e.g., IP address)

- **Locators**
  - Information about the location of a partner in the network topology

- **Different levels:** interfaces vs. machines vs. applications vs. users

- **Need to be managed (unique assignment)**
  - Or chosen randomly (and defended) in ad-hoc environments (*birthday paradox*)

- **One needs to resolved into the other**
  - Address books, (distributed) data bases (e.g., DNS, DHTs), protocol exchanges, caching, (manual) configuration, …
Some Examples

- 130.233.238.133
- fe80::20f:eaff:fe57:efe3
- 00-20-E0-74-22-53
- Port 80
- mail.ieee.org
- tel:+358-9-451-1234
- jo@netlab.tkk.fi
- http://www.acm.org/
- sip:alice@example.com

Typical Usage Example (1)

http://www.acm.org/index.html

Port: 51111 @ 130.233.238.138
Port 80 @ 63.118.7.35 : 80
00-16-41-52-DE-EF
00-0D-56-2A-AC-92

DNS server
Typical Usage Example (2)

- **Application layer: URI**
  - Access protocol identifier
  - DNS name of the server
  - Resource name

- **Transport layer: Type and port number**
  - Obtained from access protocol identifier by static convention
  - Obtained dynamically via DNS service or NAPTR lookup
  - Local identifier typically chosen dynamically

- **Network layer: IP addresses**
  - Obtained from the DNS name via DNS A/AAAA record lookups (or /etc/hosts)
  - Local identifier obtained via DHCP or zeroconf or statically configured

- **Link layer: MAC addresses**
  - Obtained via broadcast using ARP (cached)
  - Local identifier from the network interface card

Typical Usage Example (3): Functions

- **URI**
  - Modestly user readable abstraction of lower layer identifiers

- **DNS name**
  - Indirection mechanism
  - Independent of IPv4 or IPv6 address
  - Support for load balancing, redundancy, …

- **Port number**
  - Transport layer demultiplexing

- **IP address**
  - Locates the node (host part) in a specific network (network part): routing
  - Identifies the endpoint for the transport layer (e.g., TCP)

- **MAC address**
  - Local relevance only
Name Spaces

- Needed for all kinds of things
  - Host names
  - IP address
  - The Web
  - Protocol identifiers
  - Protocol field names and possibly values

- Structure
  - Structured: DNS names, URIs, URNs
  - Semi-structured: IP addresses
  - Unstructured: port numbers, cryptographic host identifiers
  - Tuple spaces: collections of attributes

- Available addresses
  - Finite: IP addresses (v4 & v6), port numbers, cryptographic host identifiers
  - Infinite: DNS names, URIs, URNs

- Scope
  - Local scope: link local addresses, private address spaces, source routes
  - Global scope: public IP address, most DNS names, etc.

- Validity: “permanent” vs. ephemeral

Semantics (*casting)

- Purpose of an address
  - “Addressing” / referring to one or more entities

- For nodes: to identify
  - A single entity (unicasting)
  - All entities in a group (multicasting)
  - All entities (broadcasting)
  - Any (e.g., the closest) entities serving a certain purpose (anycasting)
    - Closely related to service location

- May be encoded into the address structure
  - IP and 802 layer multicast addresses

- May become visible only when resolving the address
  - Mail or SIP URI, tuple spaces
Name and Address Assignment

- **Static allocation**
  - Obtain an address from an organization (IEEE, IANA, ...)
  - Past: your static IP subnet or address assignment
  - Protocol registries (e.g., IANA)

- **Hierarchical assignment delegation**
  - Allocate base addresses and delegate sub-address allocation
  - DNS names, IEEE 802 MAC address, IP subnet addresses

- **Dynamic assignment**
  - Obtaining an address upon request (e.g., DHCP, SIP GRUUs)
  - Administering entity needed (DHCP server, kernel for dynamic port numbers)

- **Self-assignment**
  - Derive from other address and/or properties: UUIDs, IPv6 addresses
  - Generate and defend addresses (zeroconf)
  - Choose based upon unlikely collisions: cryptographically generated identifiers

Resolution or Mapping

- Names and addresses need to be converted into (other) names and addresses

- **Mechanisms**
  - Built-in resolution (mapping)
    - By convention ("well known"): you “know” that port 80 is HTTP, IPv4 all routers is 224.0.0.2
    - By algorithm: how to construct an 802 multicast address from an IPv4 multicast address
  - “Centralized” resolution (possibly multiple “central nodes”)
    - Need one or more rendezvous points (centralized/locatable per domain)
    - Examples: SIP, Mobile IP
  - Hierarchical resolution
    - DNS
  - Broadcast-/multicast-based (distributed) resolution
    - ARP, service location protocols
  - Distributed resolution
    - Overlays (e.g., DHTs)

- **Responsibility for mapping/resolving**
  - Single entity: message originator, proxy (deferred resolution)
  - Some (or multiple) entities “on the way”: late binding

- Helpful: if responsibilities for administration and resolution of addresses match
Location and Forwarding

- Need to find the way towards an addressed entity
- From an address to the locator: another resolution step
  - One-stop: given the address, obtain the locator
    - IP address = locator (exception: mobile IP)
    - DNS name to IP address conversion
  - Incremental: step-by-step resolution along with forwarding
    - Routing: routing tables in each router show the next hop towards the destination

- Locators and forwarding
  - (Hierarchical) locator structure enables routing aggregation
    - Downside: locators change with point of network attachment
    - Example: IP address structure of (network, host)
  - Special case: source-routing

- Location-free addresses (no locators)
  - Downside: lots of routing/forwarding information data to store

Mobility and Multicasting

- Name to identifier/address to locator binding
  - Mobility changes the identifier to locator binding
  - Multicasting impacts the name to identifier/address binding
    - and leads to multiple (many) locations
  - Anycasting impacts the name to id/addr or the id/addr to locator bindings

- Changes need to be reflected in resolution/mapping and/or location/forwarding
  - In a single node: e.g., mobile IP Home Agent, SIP registrar, current peer(s)
  - In the network: e.g., multicast state in routers, anycast nodes
    - Global network mobility example: Connexion by Boeing (BGP routing tables)

- Issues with update frequency, overhead, consistency, …
Tradeoffs

- Name to id/address to locator bindings require mappings
  - Convenience (user)
  - Flexibility, redundancy, efficiency (system)
- Finding the way to an entity requires locating/forwarding

- Naming and addressing conventions (structure, etc.) define where you push the effort to
  - Examples
  - Indirections increase flexibility but add infrastructure and latency
  - Structure helps with routing but creates (e.g., topological) dependencies
  - Flat name spaces can help mobility but may increase cost

Example: IP Address Functions

- Node location for routing
  - Structure: (network, host) pair
  - Locates the node (host part) in a specific network (network part)
- Node identification
  - Identifies the endpoint for the transport layer (e.g., TCP)
  - Identifier the node for a security association (e.g., security context, certificate)
- Communication type identification
  - Unicast vs. broadcast vs. multicast addresses
  - Anycasting support in cooperation with routers
- May limit the propagation
  - Administratively scoped multicast addresses
Issues with IP Addresses

- **Dual nature: Locator and Identifier**
  - An IP address refers to an interface (not a node!)

- **Some issues**
  - **Mobility**
    - A node with a change in the point of attachment, changes its IP address
    - (one suboptimal remedy: mobile IP)
  - **Multi-attachment**
    - Failover between different interface does not work transparently to the transport protocol
  - **Network address translators (NATs)**
    - Identifiers do not refer to the endpoint
    - Identifiers may change (e.g., for NATs with multiple external IP addresses)
  - Identifiers depend on the IP version used

**Case Study:**

*Host Identity Protocol (HIP)*
Starting Point

- Current naming in the Internet world
  - Domain names
    - Used to name a limited number of hosts, typically well-known hosts
    - Many hosts do not have names associated with them
  - URLs
    - Application-specific extensions to DNS
  - IP addresses: two functions for interfaces
    - Topological locators for network attachment points (used in routing)
    - Naming of interfaces (used by higher layer transport protocols)
    - Issues with address changes impact transport and application layer protocols
  - A naming scheme supporting all hosts does not exist today

- HIP: Add a new name space for identifying computing platforms
decouple network aspects from transport and applications

Requirements for a New Namespace

- Applied to the “IP kernel” – across network interfaces
- Decouple higher layers from internetworking
- Do not mandate administrative infrastructure
  - (enable pairwise deployment)
- Names should have a fixed length representation
- Acceptable packet size for use in other protocols
- Names should be statistically globally unique
- Names should have a localized abstraction for use in APIs and existing protocols
- Possibility to create names locally (anonymity)
- Names should be long-lived but still replaceable at any time
Host Identity Namespace

- Provides identifiers for computing platforms across interfaces
- **Host Identifiers (HI)**
  - The Public Key of a Public-Private key pair
  - Allows for decoupling + provides authentication
  - Self-asserted identities + third party authentication (e.g. X.509 certificates)
  - May be stored in DNS, other PKI
- **Host Identity Tag (HIT)**
  - 128 bit representation of HI
    - Regular hosts: prefix (01) + lower 126 bits of SHA-1 digest of normalized HI
    - Well known hosts: prefix (10) + authority assigned value + lower 64 bits of SHA-1 digest
- **Local Scope Identifier (LSI)**
  - 32 bit locally generated (and mutually agreed upon) identifier
  - Looks like drawn from the IPv4 1.0.0.0/8 address space
  - Used in local APIs

Internet Protocol Stack Positioning

- **Application**
  - DNS -> IP address
  - TCB: A+X, Src+Dst Port
  - A
  - Link A

- **TCP**
  - TCB: A+X, Src+Dst Port
  - A
  - IP
  - B
  - Link B

- **TCP**
  - TCB: H+X, Src+Dst Port
  - A
  - IP
  - B
  - Link B

- **HIP H**

© 2008 Jörg Ott & Carsten Bormann
Host Identity Protocol (HIP)

- Specific protocol exchange defined for association setup
  - 4-way handshake
    - Authenticates peers
    - Establishes IPsec security association + Diffie-Hellman based keys
    - Protects against DoS attacks
  - Subsequent data exchange uses IPsec ESP for tunneling packets
- Dynamic rekeying during the exchange
  - Update exchange for keying material
- Support for multi-homing and mobility
  - Update and validate peer addresses
  - Dynamics supported by rendezvous server
- Initial contact via DNS
  - Resolve to IP address of the target system or its rendezvous server

Basic HIP Operation

Initiator I

\[ I1: \text{Trigger Message: HIP (I), HIP (R)} \]

- Check signature
- Solve puzzle

Responder R

\[ R1: \text{puzzle, D-H, key, signature} \]

- Select pre-computed R1

\[ I2: \text{solution, D-H, \{key\}, signature} \]

- Check cookie
- Check puzzle
- Check signature

\[ R2: \text{signature} \]
Updating Peer Addresses

- IP addresses are no longer needed for identifying endpoints
- Their routing function still is
- IP addresses may need updating
  - as interfaces come up and go down
  - as an interface address changes due to mobility
- Send REA parameter (remote address) to peer
- Wait for new security parameter index (SPI) from peer
- Then transmit data using new SPI
- Second and third step used for target address validation
  - Protection against e.g. DoS

HI Resolution

- Initial use of DNS
  - Map DNS name to IP address
  - Map DNS name to HI
    - No mapping from HI to IP address provided (DNS hierarchy unsuitable)
  - Send IP packet (I1) to target, negotiate bindings
  - Provide remote address updates during operation as necessary
- Issues
  - Dynamic changes of IP address
    - Difficult to update timely with DNS (overhead, authentication, caching, ...)
  - Not all hosts have visible IP addresses
- Indirection mechanism: Rendezvous Server
- (other mechanisms such as Distributed Hash Tables conceivable)
Sample Rendezvous Server Operation

1. Update IP(R)
2. Register RVS for FQDN(R)
3. Query FQDN (R)
4. HI (R), IP(RVS)
5. HI Message
6. HI Forwarding
7. – 9. Remainder of HIP Base Exchange

Concluding Remarks on HIP

- Rendezvous server may also help interworking with non-HIP systems
  - Provide fixed point of contact (despite sub-optimal routing)
  - Perform packet forwarding
  - May provide protocol / address translation as necessary

- HIP provides third namespace in addition to IP address and DNS
- Allows IP address independent naming of computation platforms
  - Supports multi-homing, mobility
  - Identifiers works across NATs and other middleboxes
  - Provides security for all exchanges

- Issue: quite some effort towards deployment
Some Discussion

- The Purpose of HIP
  - “Lowest layer name that does not have a location property.”

- What are the short-term motivator for HIP deployment?
  - Why should Microsoft, Sun, Apple, etc. put this into their OSes?

- Prospective uses
  - HIP to allow for anonymity (self-generated HIs and HITs)
  - HIP to support security (enabler for secure communications, IPsec)
    - Secure storage of permanent HIs?
    - Enable secure communication without PKI after initial contact
  - HIP to enable mobility (instead of mobile IP?)
  - HIP as enabler for middlebox traversal?
    - But at what cost?

- How user friendly is HIP / must HIP be?
  - Configuration and management of HIs
  - Transparent re-use of existing application?
    - With / without API modifications