Science of Communication Networks
The University of Kansas EECS 784
Identifiers, Names, and Addressing

James P.G. Sterbenz

Department of Electrical Engineering & Computer Science
Information Technology & Telecommunications Research Center
The University of Kansas

jgps@eeecs.ku.edu

http://www.ittc.ku.edu/~jgps/courses/scinets
Science of Computer Networks
Identifiers, Names, and Addressing

ID.1 Identifiers
ID.2 Names vs. Addresses

Primary references: [D2008]
Identifiers, Names, and Addressing

Identifiers

ID.1 Identifiers
ID.2 Names vs. Addresses
Identifiers

Introduction

• **Identifier** is a label to identify a particular entity
  – vertex (node, program, person)
  – edge (link, association, relationship)
  – we generally identify vertices rather than edges
    • edges are then represented by vertex pair identifiers

• **Representation within an identifier space**
  – generally binary
    • may have human-friendly representation
  – typically finite range in real systems
    • 32-bit IPv4 address
    • 48-bit IEEE 802 MAC address
    • 128-bit IPv6 address
Identifiers

Uniqueness

- Identifier
  - may be *globally unique*
    - within a global identifier space
  - but may only have *local significance*
    - within one of many local identifier spaces
      - e.g. IP private addresses
  - may only be unique with some probability
    - secondary mechanism used for uniqueness, if necessary
      - e.g. people named “John Smith”
  - may share name with many other entities
    - e.g. gender-type identifier of “female”
Identifiers

Types

- Identifier types
  - name: globally unique and persistent
  - address: logical identifier
    - without global uniqueness and persistence requirements
  - location: topological or geographic identifier
  - type: identifies characteristics of entity

who?
where?
what?
Identifiers
Identifier Space Types

• Flat: random assignment of identifiers to entities
  – no coördination of assignment needed
  – examples
    • IEEE 802 MAC addresses
    • human names

• Aggregateable
• Hierarchical
• Geographic
• Metadata
Identifiers
Identifier Space Types

• Flat
• Aggregateable
  – coordination of assignment beneficial
  – examples
    • Internet IP addresses aggregated by service provider
• Hierarchical
• Geographic
• Metadata
Identifiers
Identifier Space Types

- Flat
- Aggregateable
- Hierarchical
  - coordination of assignment necessary
  - examples
    - PSTN E.164 addresses (phone numbers)
    - Internet DNS names
- Geographic
- Metadata
Identifiers

Identifier Space Types

- Flat
- Aggregateable
- Hierarchical
- Geographic
  - identifier bound to location in graph
    - changes if node moves
  - examples
    - GPS coördinate of node
- Metadata
Identifiers
Identifier Space Types

- Flat
- Aggregateable
- Hierarchical
- Geographic
- Metadata
  - identifier is value of a type space
  - examples
    - cyan in RGB colour space
    - female in gender space
    - USA in nation space
Identifiers, Names, and Addressing

Names vs. Addresses

ID.1  Identifiers
ID.2  Names vs. Addresses
Names vs. Addresses

Names

- **Name** is a *global persistent* identifier of an entity
  - globally unique within (global) namespace
    - in practice for reasonable definition of global
  - persistent
    - name remains bound to entity for all time
Names vs. Addresses

Example Namespaces: Individual People

• Individual people
  – James Philip Guenther Sterbenz
    • note that in this case the conventional name may be sufficient
    • James Philip Guenther Sterbenz 司徒傑莫 송재윤
      almost certainly unique
  – John Smith
    sufficient?
Names vs. Addresses

Example Namespaces: Individual People

- Individual people
  - James Philip Guenther Sterbenz
    - note that in this case the conventional name may be sufficient
    - James Philip Guenther Sterbenz 司徒傑莫 宋재윤
      almost certainly unique
  - John Smith 314159265
    - note that in this case a unique identifier needs to be added
Names vs. Addresses

Example Namespaces: Authored Works

  
  www.isbn-international.org  [ISO 2108:2005]
  
  - unique identifier for a book (but *not* for an instance)
  - `<GS1-prefix>-<registration-group-element>-<registrant-element>-<publication-element>-<check-digit>`

- **Example**
  
Names vs. Addresses

Example Namespaces: Authored Works

- **DOI**: digital object identifier
  - [www.doi.org](http://www.doi.org) [ISO 26324:2012]
    - unique identifier for a creative work
      (but *not* for an instance)
    - documents, video productions
    - doi:<registry>.<registrant>/<item-id>
      - http://dx.doi.org/>

- **Examples:**
  - doi:10.1016/j.comnet.2010.03.005
  - doi:10.1145/570681.570685
  - doi:10.1109/WCNC.2011.5779182
Names vs. Addresses

Example Namespaces: URIs

- **URI**: uniform resource identifier [RFC 3986]
  - **URL**: uniform resource locator
    - address of a resource, not a name
  - **URN**: uniform resource name
    - formerly, any URI of urn: scheme [RFC 2141]
    - now any URI that is a name
Names vs. Addresses

Addresses

- **Address** is an identifier of a node
  - need not be globally unique nor persistent
  - may only be machine readable (binary address)
    - e.g. `10000001 11101101 01010111 00010010`
  - may be represented by human readable number
    - e.g. `129.237.87.18` or `148.88.3.47`
  - may be indirection by human friendly form (e.g. DNS name)
    - e.g. `www.eecs.ku.edu` or `www.comp.lancs.ac.uk`
    - unfortunately “name” is commonly used in this context
Names vs. Addresses
Directories and Binding

- **Binding** is mapping of a name to an entity
  - used for DNS “name” to IP address
- **Directory** holds binding
  - e.g. DNS server
- **Resolution** is the process of binding
  - e.g. DNS resolution
- **Lookup** is the process of binding address to interface
  - e.g. IP address lookup
Names vs. Addresses

Example: IP Addresses

- All interfaces that use IP have an address
  - host–network interfaces
    - many hosts have more than one
  - router ports
- 32-bit addresses
  - e.g. www.eecs.ku.edu to 129.237.87.18
  - example: www.eecs.ku.edu (resolved via DNS to) 10000001 111101101 01010111 00010010
- Dotted decimal notation:
  - \( b_7 b_6. b_5 b_4. b_3 b_2. b_1 b_0 \) converted to decimal in 4×8-bit chunks
  - example: 129.237.87.18
IP Addresses
IPv4 Address Assignment

- IP addresses not randomly assigned to hosts
  - *every* table would have to contain *every* Internet host
  - *billions* of entries
IP Addresses
IPv4 Address Assignment

• IP addresses not randomly assigned to hosts

*why?*
IP Addresses
IPv4 Address Hierarchy

- IP addresses assigned *hierarchically*
  - address aggregation dramatically improves scalability
  - forwarding table only needs to contain *network address*
  - routing advertisements only contain network address prefix

### ISP_A

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>199.31.0</td>
<td>X</td>
</tr>
<tr>
<td>200.23.16.4</td>
<td>0</td>
</tr>
<tr>
<td>200.23.16.12</td>
<td>2</td>
</tr>
<tr>
<td>200.23.16.45</td>
<td>1</td>
</tr>
</tbody>
</table>

### ISP_B

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>38.66.126.2</td>
<td>1</td>
</tr>
<tr>
<td>200.23.16</td>
<td>X</td>
</tr>
</tbody>
</table>

### Tier1_x

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>200.23.16</td>
<td>A</td>
</tr>
<tr>
<td>199.31.0</td>
<td>B</td>
</tr>
</tbody>
</table>
IP Addresses
Class-Based Addressing Hierarchy

- Divide IP address into 3 level hierarchy
  - class, network address, host address
  - byte aligned
  - simple IP address lookup (3 major cases)
  - class D for multicast addresses

<table>
<thead>
<tr>
<th>Networks</th>
<th>Hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>16M</td>
</tr>
<tr>
<td>16K</td>
<td>64K</td>
</tr>
<tr>
<td>2M</td>
<td>256</td>
</tr>
<tr>
<td>128 networks</td>
<td>×16M hosts</td>
</tr>
<tr>
<td>16K networks</td>
<td>×64K hosts</td>
</tr>
<tr>
<td>2M networks</td>
<td>×256 hosts</td>
</tr>
</tbody>
</table>

- class A: 0
- class B: 10
- class C: 110
- class D: 1110
- class E: 1111

Lecture NR
IP Addresses
Class-Based Addressing Problems

- Principle behind division
  - A: very large network providers
  - B: large organisations
  - C: LANs

- Reality: rigid structure
  - doesn’t match all organisations perfectly
  - doesn’t match many organisations well
    - especially class B: “three bears problem”

- Inefficient partitioning of address space
  - large fraction of unusable addresses
  - imminent exhaustion of IP address space led to...
IP Addresses

Subnets

16K networks $\times$ 64 subnets $\times$ 1024 hosts

- **Subnets** [RFC 0950 / STD 0005]
  - originally way to divide address class within organisation
  - example: 6b subnet to class B
  - subnet mask

- **Hosts in subnet** share upper IP address bits
  - natural to cluster similar IP addresses
  - efficient IP routing to subnet
  - switched layer 2 LAN with no layer 3 routing  *Lecture LL*
IP Addresses
Classless Addressing (CIDR)

- **CIDR**: classless interdomain routing [RFC 1519]
  - eliminate assignment of IP address blocks by class
  - $b_7b_6 . b_5b_4 . b_3b_2 . b_1b_0 /x$
  - x-bit prefix = arbitrary number of network bits
  - example: 11001000 00010111 00010000 00000000
    200.23.16.0/23

- Service providers get variable IP block
  - based on need from RIR (or NIR)

- Significant improvement in IP address use
  - at the cost of significant increase in complexity of IP lookup
IP Addresses
Strict Hierarchy with CIDR

- Forwarding table entries unique to networks

ISSUE?
IP Addresses
Strict Hierarchy

- Forwarding table entries unique to networks
  - all organisations *must* change IP address with ISP change
IP Addresses

Loose Hierarchy

- Forwarding table entries not unique to networks
  - longest prefix is used for forwarding (most specific)
IP Addresses
Software IP Lookup

- Longest prefix match
- Critical parameters
  - worst case lookup time
    - brute force: $O(\log_2 n)$
    - $n$ hundred thousands
  - memory required
  - forwarding table update time
IP Addresses
Fast Software IP Lookup Example: Trie

- Many algorithms
- Example: trie
  - sparse binary tree
  - valid prefixes are root
  - lookup time $O(a)$
    - $a =$ number of address bits
IP Addresses

Fast Hardware IP Lookup

- Ternary CAM
  - 1, 0, X (don’t care)
  - expensive and complex
    - relative to RAM
- Simultaneous match
  - lookup time constant
    - $O(1)$

<table>
<thead>
<tr>
<th>prefix</th>
<th>$p_{out}$</th>
<th>$s_{state}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00xxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001xxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001xx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101xx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>101xxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10100x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11xxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>111xxx</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

payload

$101\ 011\ 01$

$p_{out}$

$101\ 011\ 01$

payload

- hop count
- checksum fix

priority mux
Identifiers, Names, and Addressing

References and Further Reading

End of Foils