• Naming in systems
• Case study: DNS
Last Time: Enforced Modularity via Client/Server Model

Today: Naming
allows modules to interact
### Naming

<table>
<thead>
<tr>
<th>csail.mit.edu</th>
<th>hostname</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:katrina@csail.mit.edu">katrina@csail.mit.edu</a></td>
<td>email</td>
</tr>
<tr>
<td>katrina</td>
<td>username</td>
</tr>
<tr>
<td>R0</td>
<td>x86 register name</td>
</tr>
<tr>
<td>main</td>
<td>function name</td>
</tr>
<tr>
<td>WebBrowser</td>
<td>class name</td>
</tr>
<tr>
<td>/mit/6.033/www/schedule.shtml</td>
<td>path name</td>
</tr>
<tr>
<td><a href="http://web.mit.edu/about">http://web.mit.edu/about</a></td>
<td>URL</td>
</tr>
<tr>
<td>617-253-7341</td>
<td>phone number</td>
</tr>
<tr>
<td>128.30.2.121</td>
<td>IP Address</td>
</tr>
</tbody>
</table>
why use names?
Naming Schemes

1. Set of all possible *names*

2. Set of all possible *values*

3. **Look-up algorithm** to translate a name into a value (or set of values, or “none”)
Domain Name System

1. **names**: hostnames (*web.mit.edu*)

2. **values**: IP addresses (*18.9.22.69*)

   IP addresses are imbued with location information: routers can send packets to an IP address, but not to a hostname

3. **look-up algorithm**: resolves a hostname to an IP address so that your machine knows where to send packets
DNS Hierarchy
(a partial view)
DNS Look-up

query to: com edu berkeley mit eecs web www

result:
DNS Look-up

query to: 198.41.0.4

result: edu. 192.41.162.30
DNS Look-up

query to: 192.41.162.30  
result: mit.edu. 18.72.0.3
DNS Look-up

query to: 18.72.0.3  
result: web.mit.edu. 18.9.2.69
query to: 18.72.0.3  
result: web.mit.edu. 18.9.2.69
• **Modularity** *(previous lecture)*
  Modularity (and abstraction) limit complexity. One way to enforce modularity is to use a client/server design

• **Naming**
  Naming is what allows modules — for example, a client and a server — to communicate; it is pervasive across systems

• **DNS**
  DNS maps hostnames to IP addresses; its design is scalable and fault tolerant
what if we don’t want our modules to be on entirely separate machines? how can we enforce modularity on a single machine?