6.033 Spring 2019

Lecture #2

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

Client \[\rightarrow\text{internet} \rightarrow\] Server

load(kaws.com/buy.html?hands)

state on server
client | UID | reply

replay results from table
Last Time: Enforced Modularity via Client/Server Model

Client ↔ internet ↔ Server

load(kaws.com/buy.html?hands)

state on server
client | UID | reply

replay results from table

Today: Naming
allows modules to interact
<table>
<thead>
<tr>
<th>Examples of Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>mit.edu</td>
</tr>
<tr>
<td><a href="mailto:lacurts@mit.edu">lacurts@mit.edu</a></td>
</tr>
<tr>
<td>lacurts</td>
</tr>
<tr>
<td>R0</td>
</tr>
<tr>
<td>main</td>
</tr>
<tr>
<td>WebBrowser</td>
</tr>
<tr>
<td>/mit/6.033/www/schedule.shtml</td>
</tr>
<tr>
<td><a href="http://web.mit.edu/about">http://web.mit.edu/about</a></td>
</tr>
<tr>
<td>617-253-7341</td>
</tr>
<tr>
<td>128.30.2.121</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>hostname</th>
<th>email</th>
<th>username</th>
<th>x86 register name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>main</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WebBrowser</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>/mit/6.033/www/schedule.shtml</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><a href="http://web.mit.edu/about">http://web.mit.edu/about</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>617-253-7341</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128.30.2.121</td>
</tr>
</tbody>
</table>
why use names?
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”)
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 data
DNS Hierarchy
(a partial view)
DNS Look-up for web.mit.edu

query to:  result:

```
root
  com
  berkeley
  edu
    mit
      eecs
      web
      www
```
DNS Look-up for web.mit.edu

query to:  

result: 
DNS Look-up for web.mit.edu

query to: 198.41.0.4  result:
DNS Look-up for web.mit.edu

query to: 198.41.0.4

result:
DNS Look-up for web.mit.edu

query to: 198.41.0.4

result:

198.41.0.4

root

root table

com. 192.5.6.30

edu. 192.41.162.30

com

edu

berkeley

mit

eecs

web

www

6.033 | spring 2019 | lacurts@mit.edu
query to: 198.41.0.4  
result: edu. 192.41.162.30
DNS Look-up for web.mit.edu

query to: 198.41.0.4

result: edu. 192.41.162.30
DNS Look-up for web.mit.edu

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

query to: 192.41.162.30

result:
DNS Look-up for web.mit.edu

query to: 192.41.162.30

result:

198.41.0.4

root

com. 192.5.6.30
edu. 192.41.162.30

root table

mit.edu. 18.72.0.3
berkeley.edu. 128.32.136.14

edu table
DNS Look-up for web.mit.edu

Query to: 192.41.162.30
Result: mit.edu. 18.72.0.3
DNS Look-up for web.mit.edu

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

query to: 18.72.0.3

result:
DNS Look-up for web.mit.edu

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

query to: 18.72.0.3  
result: 

198.41.0.4 root

com
192.41.162.30

edu

root table

18.72.0.3 mit

mit table

eecs
web
www

com. 192.5.6.30
edu. 192.41.162.30

mit.edu. 18.72.0.3
berkeley.edu. 128.32.136.14

eecs.mit.edu. 18.62.1.6
web.mit.edu. 18.9.2.69
www.mit.edu. 18.9.22.169

6.033 | spring 2019 | lacurts@mit.edu
DNS Look-up for web.mit.edu

query to: 18.72.0.3  
result: web.mit.edu. 18.9.2.69
DNS Look-up for web.mit.edu

query to: 18.72.0.3

result: web.mit.edu. 18.9.2.69
DNS Hierarchy
(a partial view)
• **Modularity** (and abstraction) limit complexity. One way to enforce modularity is to use a client/server design.

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

• **DNS** maps hostnames to IP addresses. It is also a good example of **hierarchy**.
Lingering Problem

what if we don’t want our modules to be on entirely separate machines? how can we enforce modularity on a single machine?