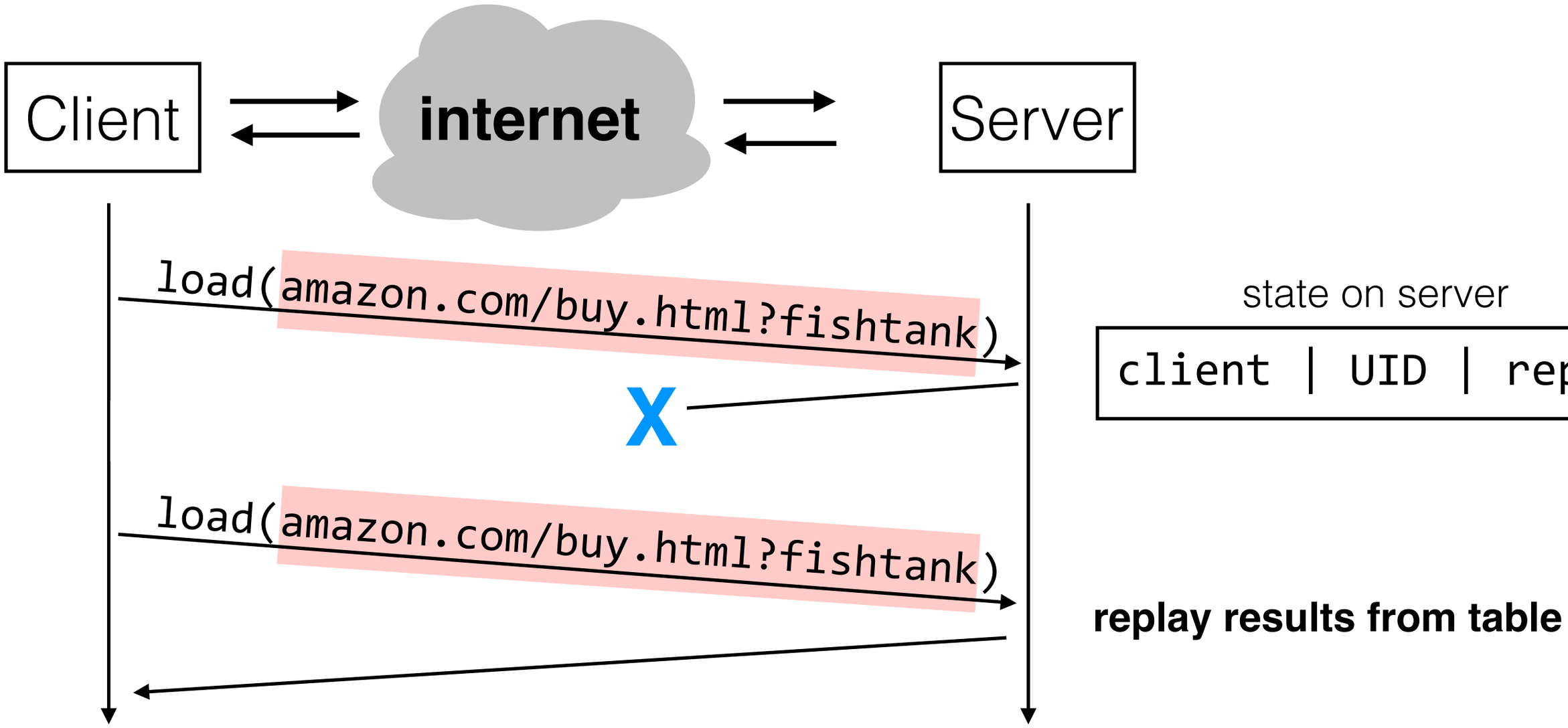
6.033 Spring 2015

Lecture #3

- Naming in systems
- Case study: DNS

Katrina LaCurts | lacurts@mit | 6.033 2015

Last Time: Enforced Modularity via Client/Server Model



Today: Naming

allows modules to interact

Katrina LaCurts | lacurts@mit | 6.033 2015

Naming

csail.mit.edu	hostname
katrina@csail.mit.edu	email
katrina	username
R0	x86 register name
main	function name
WebBrowser	class name
/mit/6.033/www/schedule.shtml	path name
http://web.mit.edu/about	URL
617-253-7341	phone number
128.30.2.121	IP Address

why use names?

Katrina LaCurts | lacurts@mit | 6.033 2015

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`)

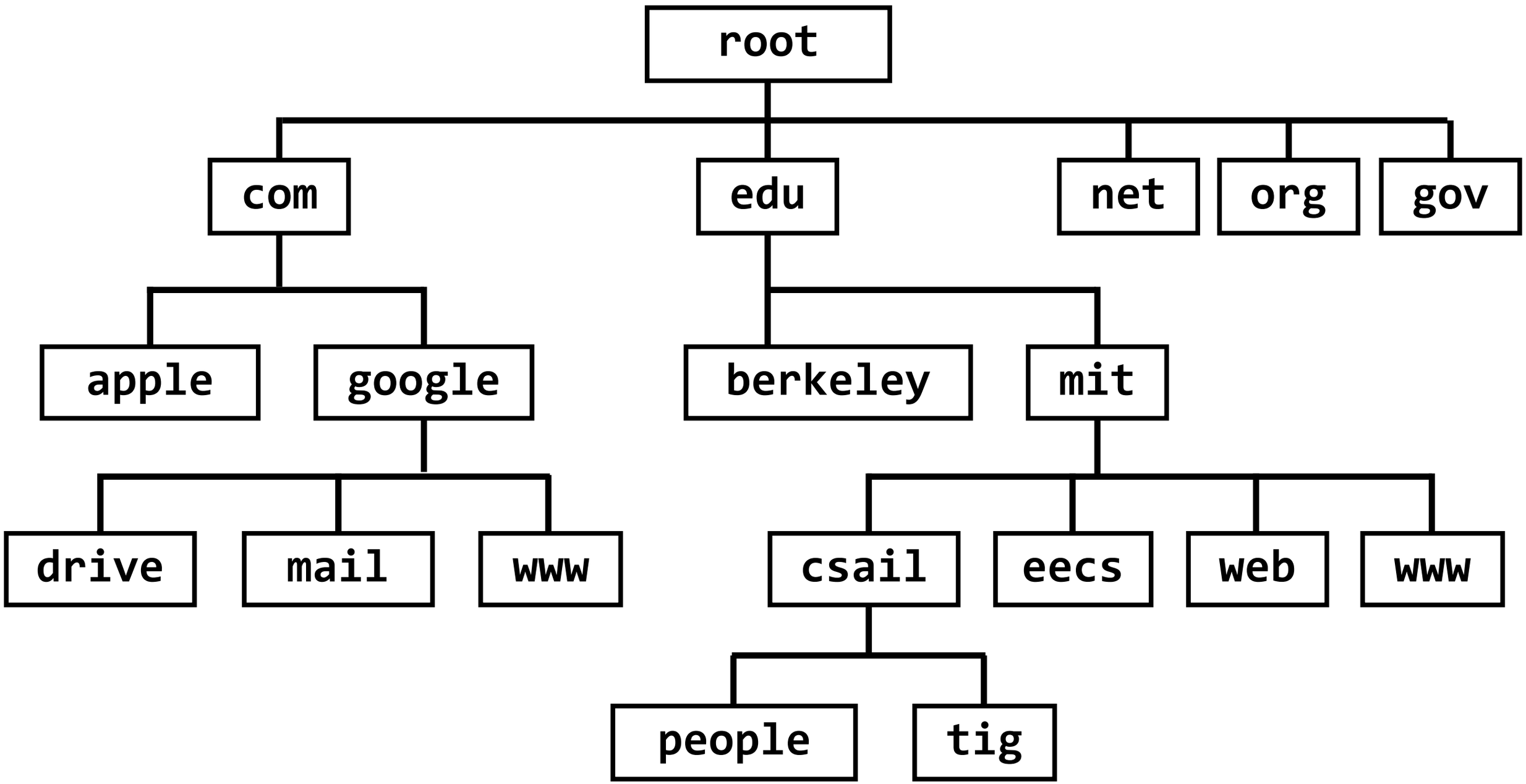
2. **values:** IP addresses (`18.9.22.69`)

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

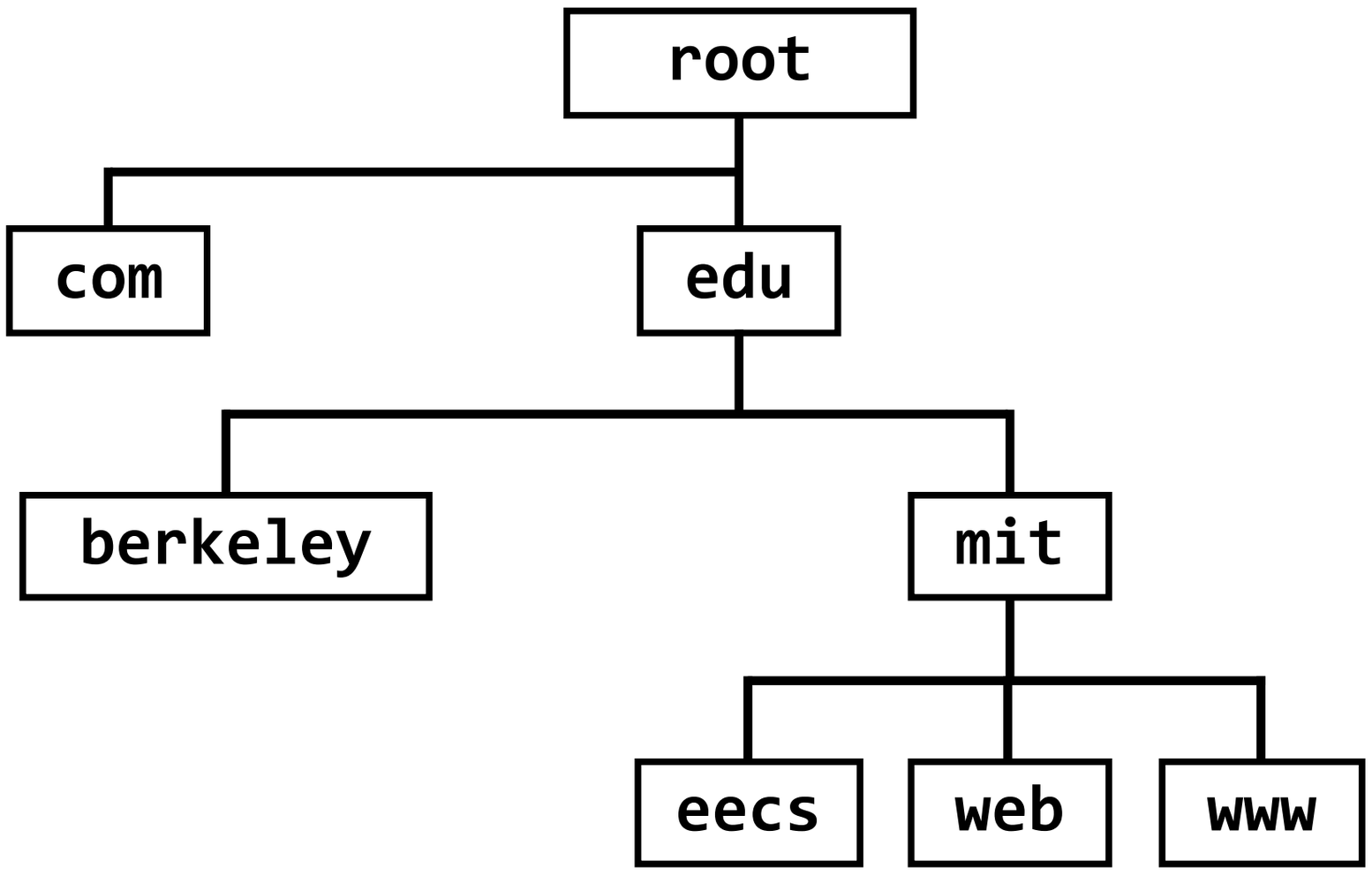
3. **look-up algorithm:** resolves a hostname to an IP address so that your machine knows where to send packets

DNS Hierarchy

(a partial view)



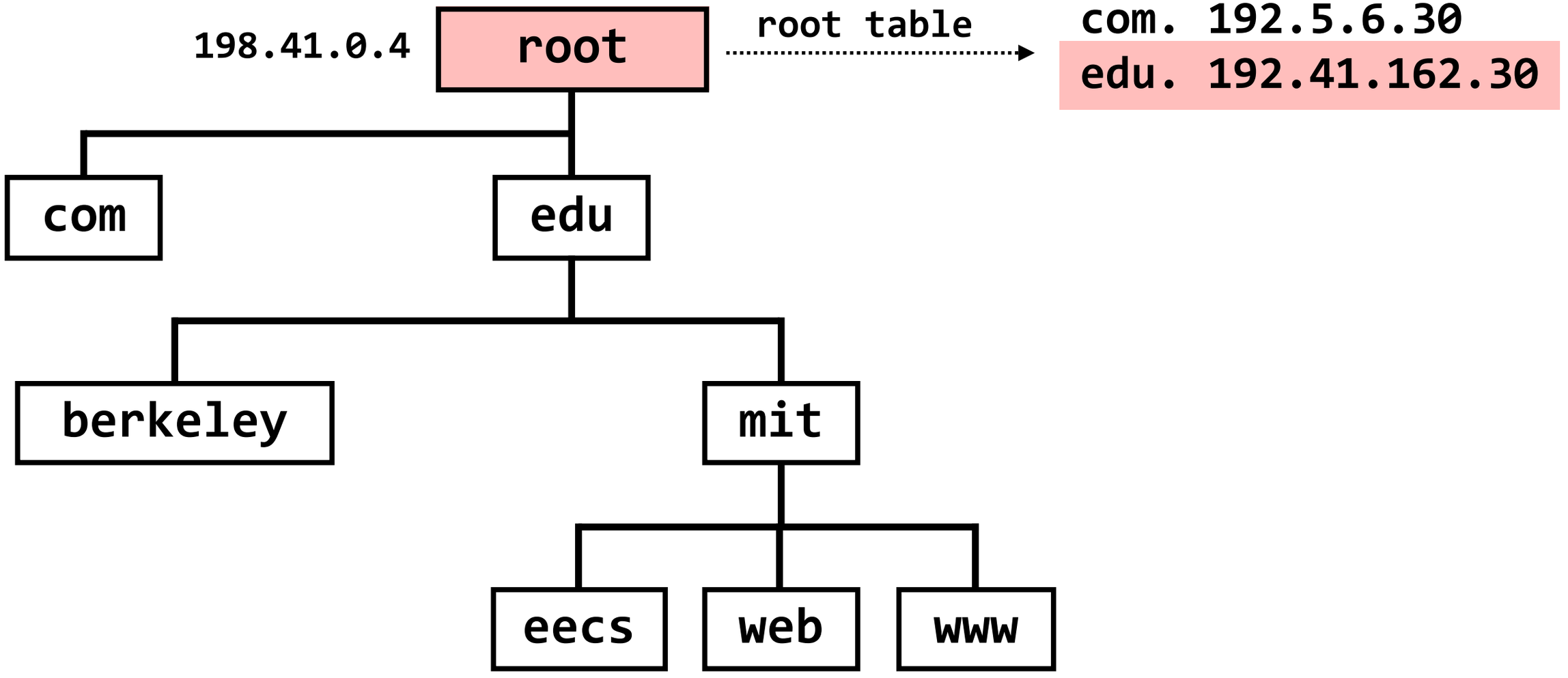
DNS Look-up



query to:

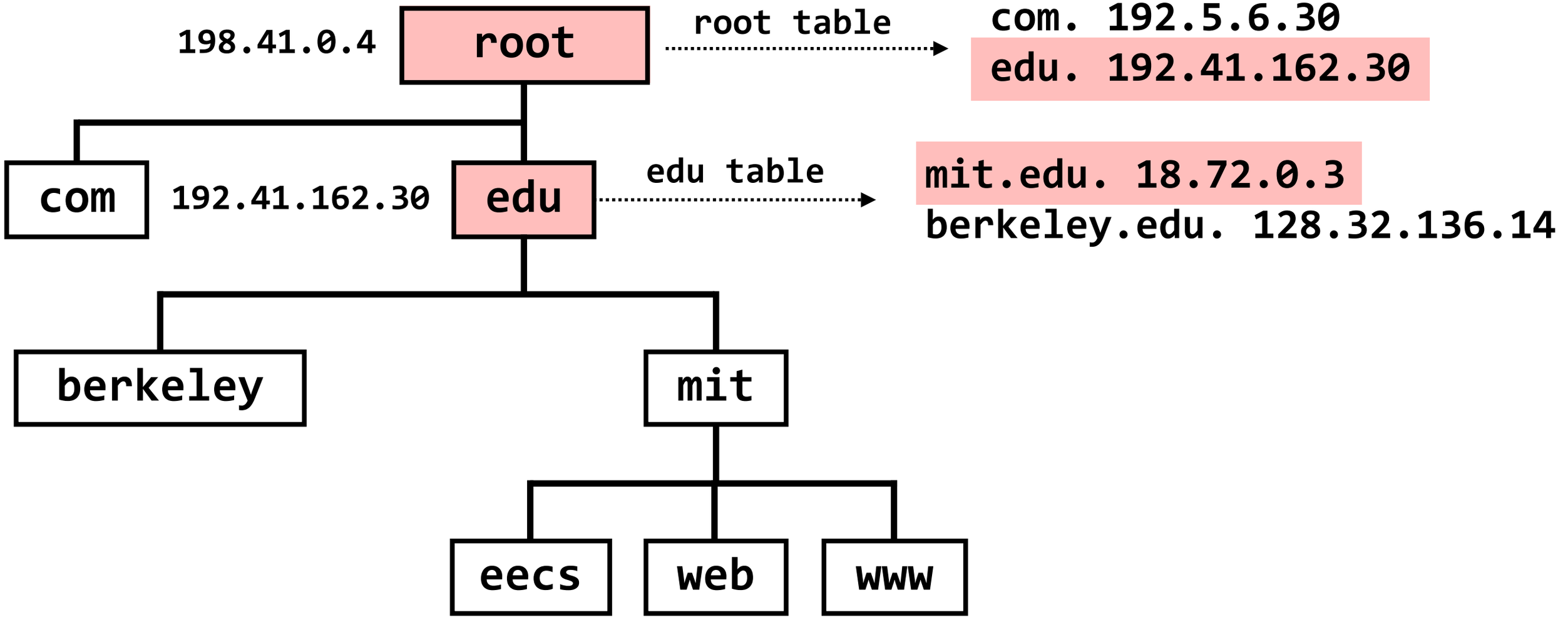
result:

DNS Look-up



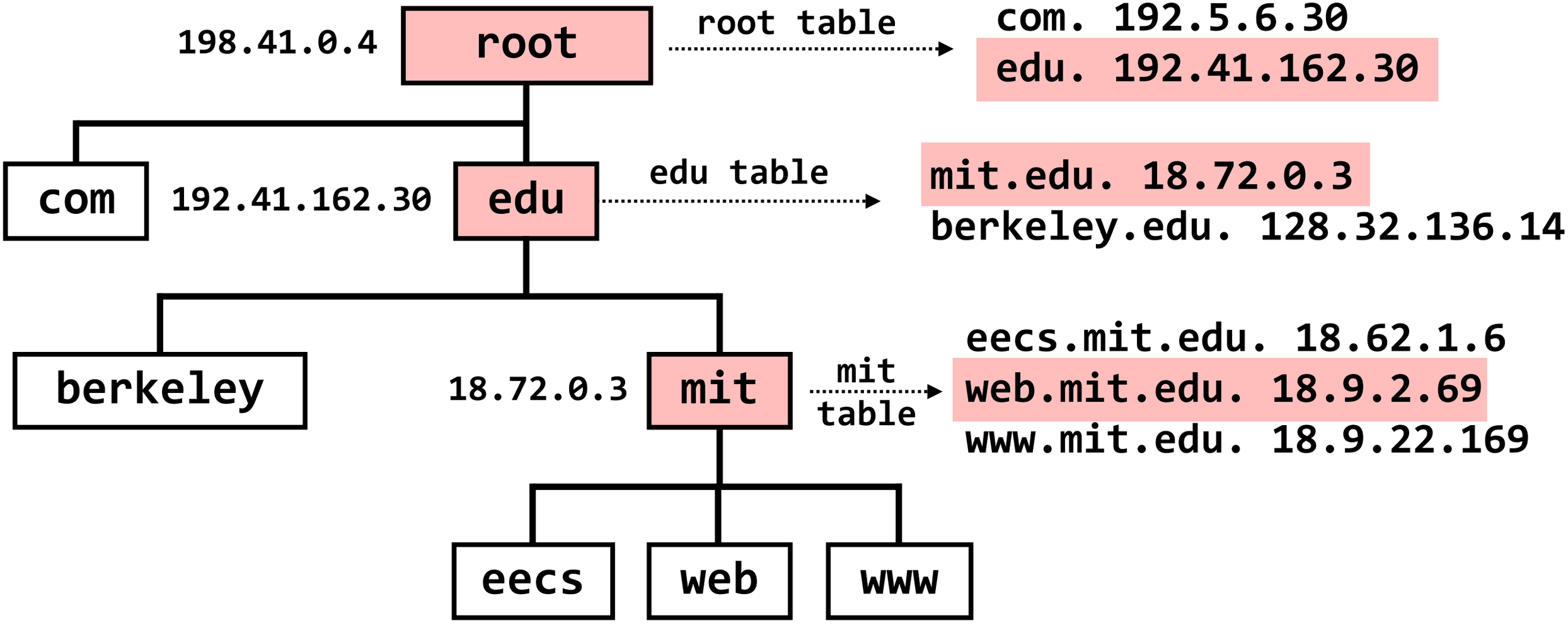
query to: `198.41.0.4` **result:** `edu. 192.41.162.30`

DNS Look-up



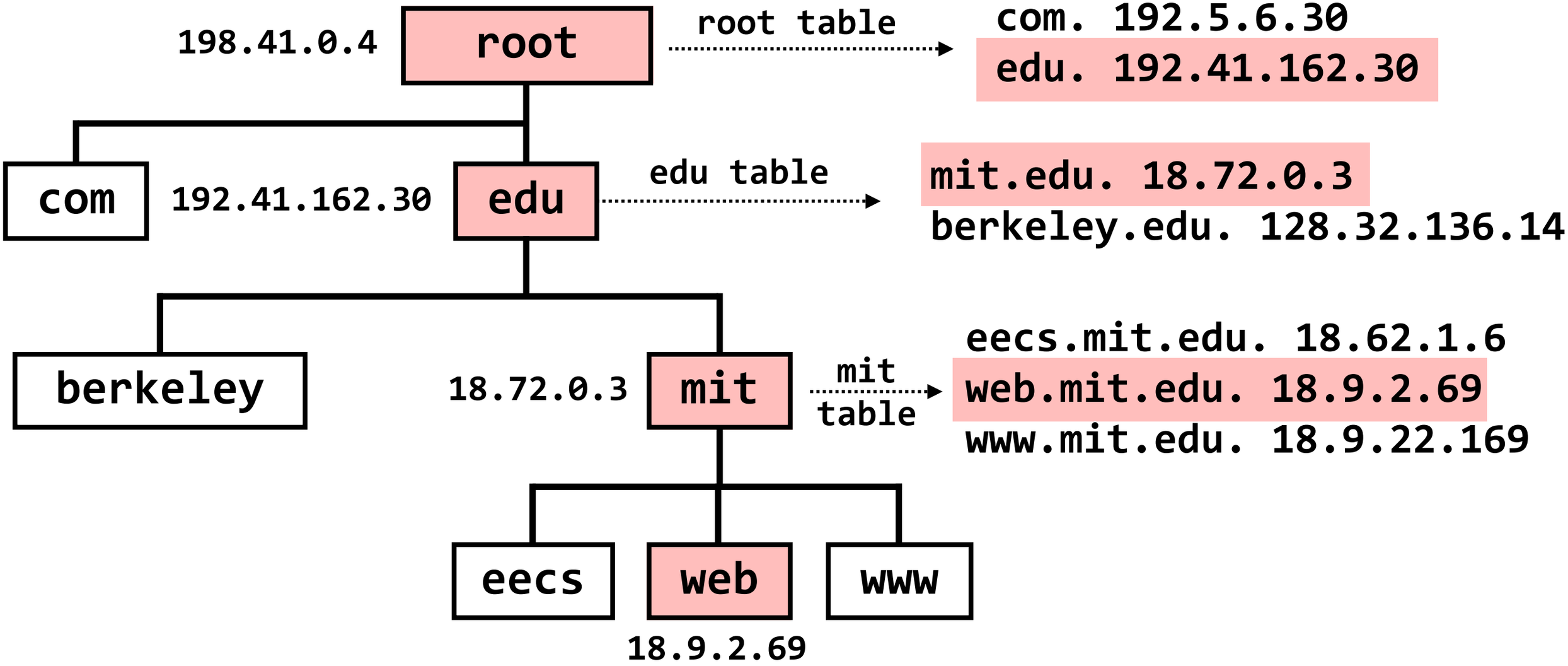
query to: `192.41.162.30` **result:** `mit.edu. 18.72.0.3`

DNS Look-up



query to: `18.72.0.3` **result:** `web.mit.edu. 18.9.2.69`

DNS Look-up



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

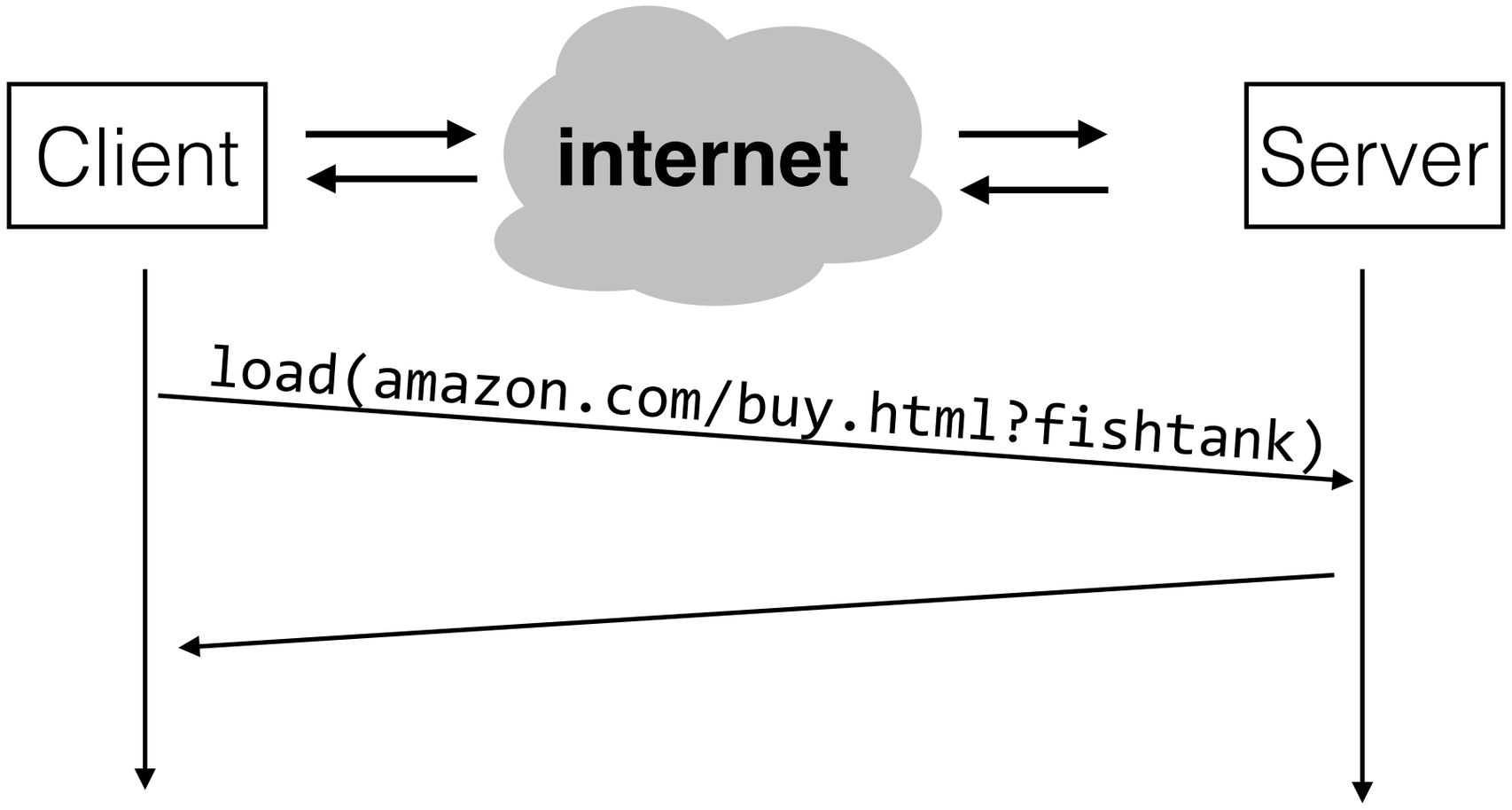
- **Naming**

  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

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**?