Lecture #2: Naming
plus a case-study on DNS
A widespread internet outage caused several major websites to shut down Thursday afternoon, including Amazon, Delta, Capital One and Costco.

Akamai, a content distribution network that helps with the spread of data around the internet, posted on Twitter that a software configuration update caused a bug in its DNS system.

A DNS, or domain name service, helps match a website’s name to its IP address. If the DNS fails, it becomes impossible to search and connect to a website by name.
6.1800 in the news

Bad Ubuntu update crashes global Azure Kubernetes services

A flawed Ubuntu systemd update appears to have taken Azure virtual machines running on Ubuntu offline by breaking DNS - causing a significant Azure Kubernetes outage for Ubuntu users.

source: https://thestack.technology/azure-kubernetes-outage-ubuntu-dns/
6.1800 in the news

1/25  Post Incident Review (PIR) – Azure Networking – Global WAN issues (Tracking ID VSG1-B90)

What happened?

Between 07:08 UTC and 12:43 UTC on 25 January 2023, customers experienced issues with network connectivity, manifesting as long network latency and/or timeouts when attempting to connect to resources hosted in Azure regions, as well as other Microsoft services including Microsoft 365 and Power Platform. This incident also impacted Azure Government cloud services that were dependent on Azure public cloud. While most regions and services had recovered by 09:05 UTC, intermittent packet loss issues caused some customers to continue seeing connectivity issues due to two routers not being able to recover automatically. All issues were fully mitigated by 12:43 UTC.

How did we respond?

Our monitoring detected DNS and WAN issues starting at 07:11 UTC. We began investigating by reviewing all recent changes. By 08:20 UTC, as the automatic recovery was happening, we identified the problematic command that triggered the issue. Networking telemetry shows that nearly all network devices had recovered by 09:05 UTC, by which point most regions and services had recovered. Final networking equipment recovered by 09:25 UTC.

source: https://status.azure.com/en-us/status/history/ (from 1/25/2023)
Russian top-level internet domain suffers massive outage

Russian citizens couldn't access the majority of websites on the country’s .ru domain for several hours on Tuesday, including the Yandex search engine, the VKontakte social media platform, the major state-owned bank Sberbank and news outlets.

The outage was reportedly caused by a technical problem with the .ru domain’s global Domain Name System Security Extensions, or DNSSEC. It appeared to be unintentional, unlike other recent blackouts of Russian internet services, which observers have tied to government intervention.
def main():
    html = browser_load_url(URL)
    ...

def browser_load_url(url):
    msg = url
    # could reformat
    send request
    wait for reply
    html = reply
    # could reformat
    return html

def server_load_url():
    ...
    return html

def handle_server_load_url(url):
    wait for request
    url = request
    html = server_load_url(URL)
    reply = html
    send reply

last time: enforced modularity via client/server
last time: enforced modularity via client/server

```
def main():
    html = browser_load_url(URL)
    ...

def browser_load_url(url):
    msg = url  # could reformat
    send request
    wait for reply
    html = reply  # could reformat
    return html
```

```
def server_load_url():
    ...
    return html

def handle_server_load_url(url):
    wait for request
    url = request
    html = server_load_url(URL)
    reply = html
    send reply
```

client

Class Browser
(on machine 1)

server

Class Server
(on machine 2)

network

load("kaws.com/buy.html?item=guac")

stub
last time: enforced modularity via client/server

---

Class Browser
(on machine 1)

```python
def main():
    html = browser_load_url(URL)
...

def browser_load_url(url):
    msg = url  # could reformat
    send request
    wait for reply
    html = reply  # could reformat
    return html
```

---

Class Server
(on machine 2)

```python
def server_load_url():
    ...
    return html

def handle_server_load_url(url):
    wait for request
    url = request
    html = server_load_url(URL)
    reply = html
    send reply
```

---

Network
**last time:** enforced modularity via client/server

**today:** naming, which allows modules to interact
why use names?

**Class Browser** (on machine 1)

```python
def main():
    html = browser_load_url(URL)
    ...

def browser_load_url(url):
    msg = url  # could reformat
    send request
    wait for reply
    html = reply  # could reformat
    return html
```

**Class Server** (on machine 2)

```python
def server_load_url():
    ...
    return html

def handle_server_load_url(url):
    wait for request
    url = request
    html = server_load_url(URL)
    reply = html
    send reply
```

load("kaws.com/buy.html?item=guac")
why use names? they let us achieve modularity by providing communication and organization, as well as a number of other properties

---

client

**Class Browser** (on machine 1)

```python
def main():
    html = browser_load_url(URL)
    ...

def browser_load_url(url):
    msg = url # could reformat
    send request
    wait for reply
    html = reply # could reformat
    return html
```

server

**Class Server** (on machine 2)

```python
def server_load_url():
    ...
    return html

def handle_server_load_url(url):
    wait for request
    url = request
    html = server_load_url(URL)
    reply = html
    send reply
```
why use names? they let us achieve modularity by providing communication and organization, as well as a number of other properties

Class Browser (on machine 1)

```python
def main():
    html = browser_load_url(URL)
    ...
```

def browser_load_url(url):
    msg = url # could reformat
    send request
    wait for reply
    html = reply # could reformat
    return html

stub

retrieval
the client can retrieve the guacamole page because it can name it

Class Server (on machine 2)

```python
def server_load_url():
    ...
    return html
```

def handle_server_load_url(url):
    wait for request
    url = request
    html = server_load_url(URL)
    reply = html
    send reply

stub
why use names?

they let us achieve modularity by providing communication and organization, as well as a number of other properties

client

```
Class Browser
(on machine 1)
```

```
def main():
    html = browser_load_url(URL)
...
```

```
def browser_load_url(url):
    msg = url  # could reformat
    send request
    wait for reply
    html = reply  # could reformat
    return html
```

server

```
Class Server
(on machine 2)
```

```
def server_load_url():
    ...
    return html
```

```
def handle_server_load_url(url):
    wait for request
    url = request
    html = server_load_url(URL)
    reply = html
    send reply
```

retrieval

the client can retrieve the guacamole page because it can name it

sharing

the server can share the guacamole page with multiple clients (i.e., multiple clients can view this page)

```
**why use names?**

they let us achieve modularity by providing communication and organization, as well as a number of other properties

---

**client**

```python
Class Browser
(on machine 1)

def main():
    html = browser_load_url(URL)
    ...

def browser_load_url(url):
    msg = url # could reformat
    send request
    wait for reply
    html = reply # could reformat
    return html
```

---

**server**

```python
Class Server
(on machine 2)

def server_load_url():
    ...
    return html

def handle_server_load_url(url):
    wait for request
    url = request
    html = server_load_url(url)
    reply = html
    send reply
```

---

**retrieval**

the client can retrieve the guacamole page because it can name it

**sharing**

the server can share the guacamole page with multiple clients (i.e., multiple clients can view this page)

---

**user-friendly IDs**

kaws.com is easier to remember than (say) 18.25.4.171; the variable name “html” is easier to remember than a particular location in memory
why use names? they let us achieve modularity by providing communication and organization, as well as a number of other properties

Class Browser (on machine 1)

```python
def main():
    html = browser_load_url(URL)
    ...
```  
def browser_load_url(url):
    msg = url # could reformat
    send request
    wait for reply
    html = reply # could reformat
    return html

stub

Class Server (on machine 2)

```python
def server_load_url():
    ...
    return html
```  
def handle_server_load_url(url):
    wait for request
    url = request
    html = server_load_url(URL)
    reply = html
    send reply

stub

retrieval
the client can retrieve the guacamole page because it can name it

sharing
the server can share the guacamole page with multiple clients (i.e., multiple clients can view this page)

user-friendly IDs
kaws.com is easier to remember than (say) 18.25.4.171; the variable name “html” is easier to remember than a particular location in memory

addressing
some names also specify location information

Katrina LaCurts | lacurts@mit.edu | 6.1800 2024
why use names? they let us achieve modularity by providing communication and organization, as well as a number of other properties

```python
def server_load_url():
    ...
    return html

def handle_server_load_url(url):
    wait for request
    url = request
    html = server_load_url(URL)
    reply = html
    send reply
```
why use names? they let us achieve modularity by providing communication and organization, as well as a number of other properties
why use names? they let us achieve modularity by providing communication and organization, as well as a number of other properties.

hiding

code on the server can access guac_data.txt without having to worry about how the file is laid out in memory.
why use names?

they let us achieve modularity by providing communication and organization, as well as a number of other properties

hiding

code on the server can access guac_data.txt without having to worry about how the file is laid out in memory

indirection

the server can change the memory layout of guac_data.txt without notifying the user
why use names? they let us achieve modularity by providing communication and organization, as well as a number of other properties
**why use names?**

They let us achieve modularity by providing communication and organization, as well as a number of other properties.

The design of a system’s **naming scheme(s)** helps it achieve these properties.
why use names? they let us achieve modularity by providing communication and organization, as well as a number of other properties

the design of a system’s naming scheme(s) helps it achieve these properties

a naming scheme includes
why use names? they let us achieve modularity by providing communication and organization, as well as a number of other properties

the design of a system’s naming scheme(s) helps it achieve these properties

a naming scheme includes

1. the set of all possible names
why use names? they let us achieve modularity by providing communication and organization, as well as a number of other properties.

the design of a system’s naming scheme(s) helps it achieve these properties:

a naming scheme includes:

1. the set of all possible names
2. the set of all possible values
why use names? they let us achieve modularity by providing communication and organization, as well as a number of other properties

the design of a system’s naming scheme(s) helps it achieve these properties

a naming scheme includes

1. the set of all possible names
2. the set of all possible values
3. a look-up algorithm to translate a name into a value (or a set of values, or “none”)

Katrina LaCurts | lacurts@mit.edu | 6.1800 2024
Mosquito Capital @MosquitoCapital · Nov 18, 2022
52) DNS! If you f up DNS settings somehow, your entire *everything* can go dark. And I mean everything. Internal tools, the entire website, the literal DOORS TO YOUR OFFICE (that's not hypothetical). And if your DNS registration lapses, someone can steal your site! Forever!

source: https://twitter.com/MosquitoCapital/status/1593541217136316416
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)
naming case study: the **domain name system (DNS)**, which maps hostnames (eecs.mit.edu) to **IP addresses** (18.25.0.23)

the **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation
naming case study: the **domain name system (DNS)**, which maps hostnames (eecs.mit.edu) to **IP addresses** (18.25.0.23)

the **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation
naming case study: the **domain name system (DNS)**, which maps hostnames (**eecs.mit.edu**) to **IP addresses** (**18.25.0.23**) the **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

```
com
  apple
  google

net
  berkeley
  mit

  drive
  mail
  www

  web
  www
  eecs
```
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

![A partial view of the DNS hierarchy. Each box represents a zone. Name servers within a zone keep track of that zone’s mappings.](image-url)
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a zone. name servers within a zone keep track of that zone’s mappings
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a zone. name servers within a zone keep track of that zone’s mappings
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

A partial view of the DNS hierarchy. Each box represents a zone. Name servers within a zone keep track of that zone’s mappings.
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a zone. name servers within a zone keep track of that zone’s mappings
naming case study: the **domain name system (DNS)**, which maps hostnames (eecs.mit.edu) to **IP addresses** (18.25.0.23)

the **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation.

A partial view of the DNS hierarchy. Each box represents a **zone**. Name servers within a zone keep track of that zone’s mappings.
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a zone. name servers within a zone keep track of that zone’s mappings
naming case study: the **domain name system (DNS)**, which maps hostnames (eecs.mit.edu) to **IP addresses** (18.25.0.23)

the **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a **zone**. name servers within a zone keep track of that zone’s mappings
naming case study: the **domain name system (DNS)**, which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

A partial view of the DNS hierarchy. Each box represents a **zone**. Name servers within a zone keep track of that zone’s mappings.
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a zone. name servers within a zone keep track of that zone’s mappings
naming case study: the **domain name system (DNS)**, which maps hostnames (eecs.mit.edu) to **IP addresses** (18.25.0.23)

the **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a **zone**. name servers within a zone keep track of that zone’s mappings

Katrina LaCurts | lacurts@mit.edu | 6.1800 2024
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a zone. name servers within a zone keep track of that zone’s mappings

DNS client

e.g., your laptop
naming case study: the **domain name system (DNS)**, which maps hostnames (eecs.mit.edu) to **IP addresses** (18.25.0.23)

the **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a **zone**. name servers within a zone keep track of that zone’s mappings

**DNS client**

**query sent to:**

**response:**
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a zone. name servers within a zone keep track of that zone’s mappings
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a zone. name servers within a zone keep track of that zone’s mappings

DNS client e.g., your laptop

query sent to: 198.41.0.4
response: try 192.14.171.191
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a zone. name servers within a zone keep track of that zone’s mappings
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation.

A partial view of the DNS hierarchy. Each box represents a zone. Name servers within a zone keep track of that zone’s mappings.
naming case study: the **domain name system (DNS)**, which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

![A partial view of the DNS hierarchy. Each box represents a zone. Name servers within a zone keep track of that zone’s mappings.](image-url)
naming case study: the **domain name system (DNS)**, which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23).

The **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation.

A partial view of the DNS hierarchy. Each box represents a **zone**. Name servers within a zone keep track of that zone’s mappings.

DNS client

<table>
<thead>
<tr>
<th>Query sent to</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.72.0.3</td>
<td>18.25.0.23</td>
</tr>
</tbody>
</table>

First, the DNS client sends a query to 198.41.0.4, which is the root nameserver.

Then, the DNS client sends a query to 192.14.171.191, which is the nameserver for edu.

Finally, the DNS client sends a query to 18.72.0.3, which is the nameserver for mit.edu, and receives the IP address 18.25.0.23.
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a zone. name servers within a zone keep track of that zone’s mappings
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

performance issue: this is a lot of queries, especially to the root server

a partial view of the DNS hierarchy. each box represents a zone. name servers within a zone keep track of that zone’s mappings
naming case study: the **domain name system (DNS)**, which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

**performance issue:** this is a *lot* of queries, especially to the root server

**reliability issue:** what happens when a nameserver fails or (security issue) is attacked?

a partial view of the DNS hierarchy. each box represents a **zone**. name servers within a zone keep track of that zone’s mappings

Katrina LaCurts | lacurts@mit.edu | 6.1800 2024
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.25.0.23)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a zone. name servers within a zone keep track of that zone’s mappings

performance issue: this is a lot of queries, especially to the root server

reliability issue: what happens when a nameserver fails or (security issue) is attacked?

control issue: who should manage the root server?
bonus case study:  **Course 6 subject numbers**, all of which changed recently (to many people’s annoyance, but to my delight)

6.031: Elements of Software Construction
6.032: *didn’t exist*
**6.033: Computer Systems Engineering**
6.034: Artificial Intelligence
6.035: Computer Language Engineering
6.036: Introduction to Machine Learning
6.037: Structure and Interpretation of Computer Programs
6.038: Representation & Inference in AI
6.039: Operating Systems Engineering
bonus case study: Course 6 subject numbers, all of which changed recently (to many people’s annoyance, but to my delight)

6.031: Elements of Software Construction
6.032: didn’t exist
**6.033: Computer Systems Engineering**
6.034: Artificial Intelligence
6.035: Computer Language Engineering
6.036: Introduction to Machine Learning
6.037: Structure and Interpretation of Computer Programs
6.038: Representation & Inference in AI
6.039: Operating Systems Engineering

**previous scheme:** subject number alone gave you very little information about the topic of the class
bonus case study:  Course 6 subject numbers, all of which changed recently (to many people’s annoyance, but to my delight)

6.1800: Computer Systems Engineering
6.1810: Operating Systems Engineering
6.1820: Mobile and Sensor Computing
6.1830: doesn’t exist
6.1840: doesn’t exist
6.1850: Computer Systems and Society
...
6.5810: Operating System Engineering (G)
6.5820: Computer Networks (G)

current scheme: subject number tells you something about the topic (nearby numbers are in similar areas of EECS)
bonus case study: **Course 6 subject numbers**, all of which changed recently (to many people’s annoyance, but to my delight)

6.1800: Computer Systems Engineering  
6.1810: Operating Systems Engineering  
6.1820: Mobile and Sensor Computing  
6.1830: *doesn’t exist*  
6.1840: *doesn’t exist*  
6.1850: Computer Systems and Society  
...
6.5810: Operating System Engineering (G)  
6.5820: Computer Networks (G)

**current scheme:** subject number tells you something about the topic (nearby numbers are in similar areas of EECS)

why four digits after the 6? the registrar doesn’t allow a department to reuse a permanent subject number for a new class within five years. solution: create a new three-digit based scheme, append 0’s to make then four digits. after five years, revert back to three digits.
modularity and abstraction mitigate complexity. A client/server model allow us to enforce modularity by putting modules on physically separate machines.
modularity and abstraction mitigate complexity. a client/server model allow us to enforce modularity by putting modules on physically separate machines.

naming is what allows modules to interact, and can help us achieve other goals through properties such as indirection, user-friendliness, etc.
Modularity and abstraction mitigate complexity. A client/server model allow us to enforce modularity by putting modules on physically separate machines.

Naming is what allows modules to interact, and can help us achieve other goals through properties such as indirection, user-friendliness, etc.

The domain name system is a great case-study in naming, and also illustrates principles such as hierarchy, scalability, delegation, and decentralization.
modularity and abstraction mitigate complexity. A client/server model allow us to enforce modularity by putting modules on physically separate machines.

naming is what allows modules to interact, and can help us achieve other goals through properties such as indirection, user-friendliness, etc.

the domain name system is a great case-study in naming, and also illustrates principles such as hierarchy, scalability, delegation, and decentralization.

the example you saw in lecture was a fairly basic one; you will talk more about DNS’s performance enhancements in recitation tomorrow, which change how some (many) DNS queries are resolved.
modularity and abstraction mitigate complexity. A client/server model allow us to enforce modularity by putting modules on physically separate machines.

naming is what allows modules to interact, and can help us achieve other goals through properties such as indirection, user-friendliness, etc.

the domain name system is a great case-study in naming, and also illustrates principles such as hierarchy, scalability, delegation, and decentralization

and client/server models, and (tomorrow) caching, and (in May) security…