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

```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**

```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**

**client**

```
load("kaws.com/buy.html?dogfood")
```

**stub**

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
```

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

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

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

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

client

load("kaws.com/buy.html?dogfood")

server
**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)
...
```

```python
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
```

```python
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

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
```

stub

 Retrieval
the client can retrieve the dog-food page because it can name it

client

```python
load("kaws.com/buy.html?dogfood")
```

server

```python
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

**retrieval**
the client can retrieve the dog-food page because it can name it

**sharing**
the server can share the dog-food 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**

```
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)

load("kaws.com/buy.html?dogfood")

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 dog-food page because it can name it.

**sharing**

The server can share the dog-food 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.

Katrina LaCurts | lacurts@mit.edu | 6.033 2022
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)

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)

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 dog-food page because it can name it

sharing
the server can share the dog-food 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
why use names? they let us achieve modularity by providing communication and organization, as well as a number of other properties.

```
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

stub
```
why use names?

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

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

dogfood_data.txt

price
weight
brand
...

Class Server
(on machine 2)

server
kaws.com / 18.25.4.171

stub
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 dogfood_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 dogfood_data.txt without having to worry about how the file is laid out in memory

indirection

the server can change the memory layout of dogfood_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”)
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
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

---

**Diagram:**

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

[Diagram of DNS hierarchy]

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

```plaintext
  root
 /    \
|      |
com    net
  \
apple google
  \
|   |   |
drive mail www
  \
|   |   |
web www eecs
  \
|   |
mit
```

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

The server `nameserver` has an IP address of `192.14.171.191` and is authoritative for the domain `edu`. It has references to other name servers for `com` and `net`. The server `nameserver` for `mit` has an IP address of `192.14.171.192` and is authoritative for the domain `edu`. It has references to other name servers for other subdomains.
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.

A 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:

- **root nameserver**
  - edu. 192.14.171.191 NS
  - com. 192.14.171.192 NS
  - net. 192.14.171.193 NS

- **edu. nameserver**
  - mit.edu. 18.72.0.3 NS
  - berkeley.edu. 128.32.136.14 NS

- **mit. nameserver**
  - eecs.mit.edu. 18.25.0.23 A
  - web.mit.edu. 18.9.2.69 A
  - www.mit.edu. 18.9.22.169 A

**DNS client**
- query sent to:
- response:

Example entries:
- drive.mit.edu. 192.14.171.191 NS
- www.mit.edu. 18.9.22.169 A
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:

Katrina LaCurts | lacurts@mit.edu | 6.033 2022
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
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.

DNS client
query sent to: 192.14.171.191
response: try 18.72.0.3
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: 18.72.0.3

response: 18.72.0.3
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: **18.72.0.3**
response: **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

[Diagram of DNS hierarchy]

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.
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?

control issue: who should manage 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
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.

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