6.033 Spring 2018

Lecture #1

- Complexity
- Modularity and abstraction
- Enforced modularity via client/server models

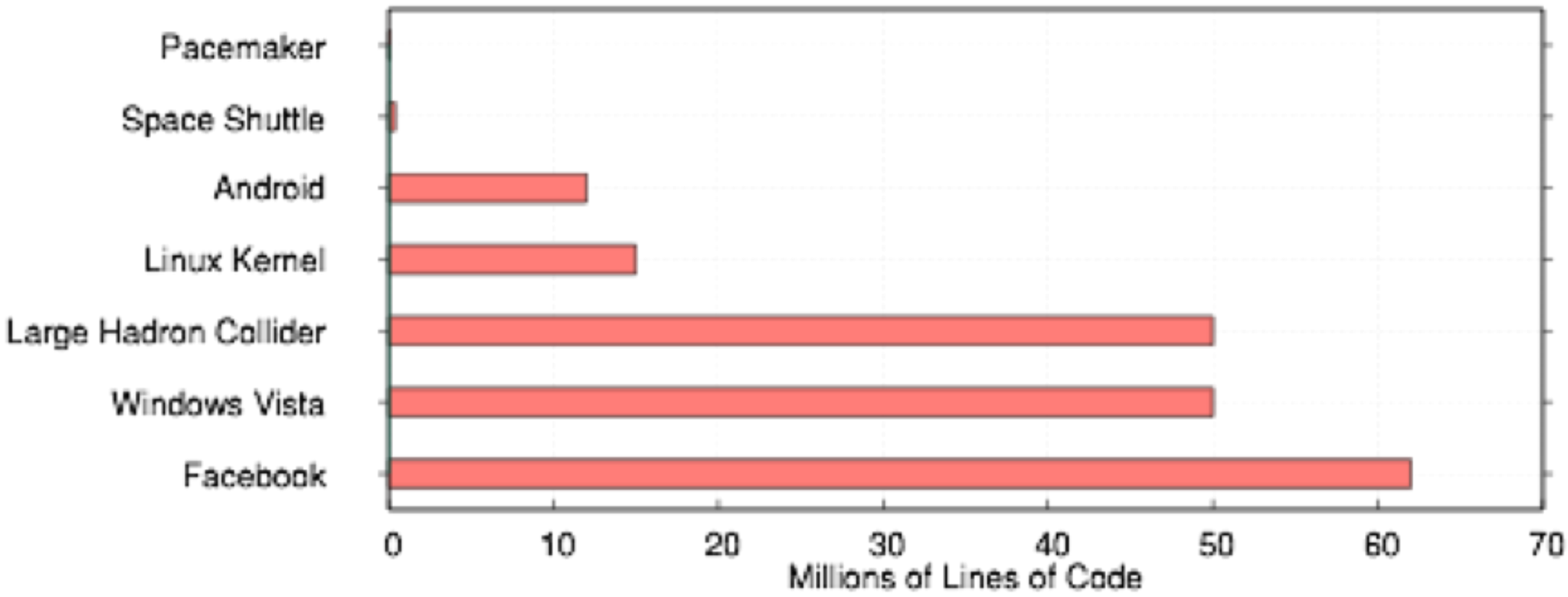
what is a system?

a set of interconnected components that has an expected behavior observed at the interface with its environment

what makes building systems difficult?

complexity

Today's Systems are Incredibly Complex



source: http://www.informationisbeautiful.net/visualizations/million-lines-of-code/

complexity limits what we can build and causes a number of unforeseen issues

how do we mitigate complexity?

with design principles such as **modularity** and **abstraction**

how do we enforce modularity?

one way is to use the **client/server model**

Class Browser
(on machine 1)

Class Server
(on machine 2)

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

request

reply

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

Stub Clients and RPCs

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
```

stub

request

reply

Class Server
(on machine 2)

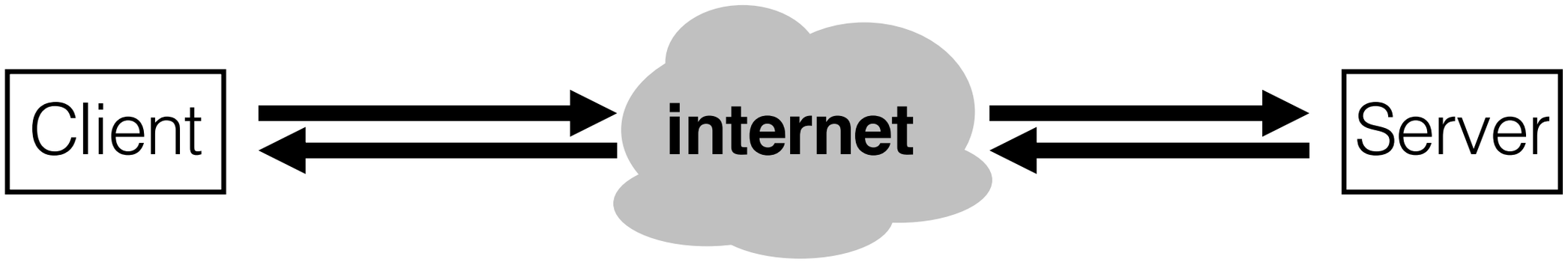
```
def server_load_url():
  ...
  return html
```

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

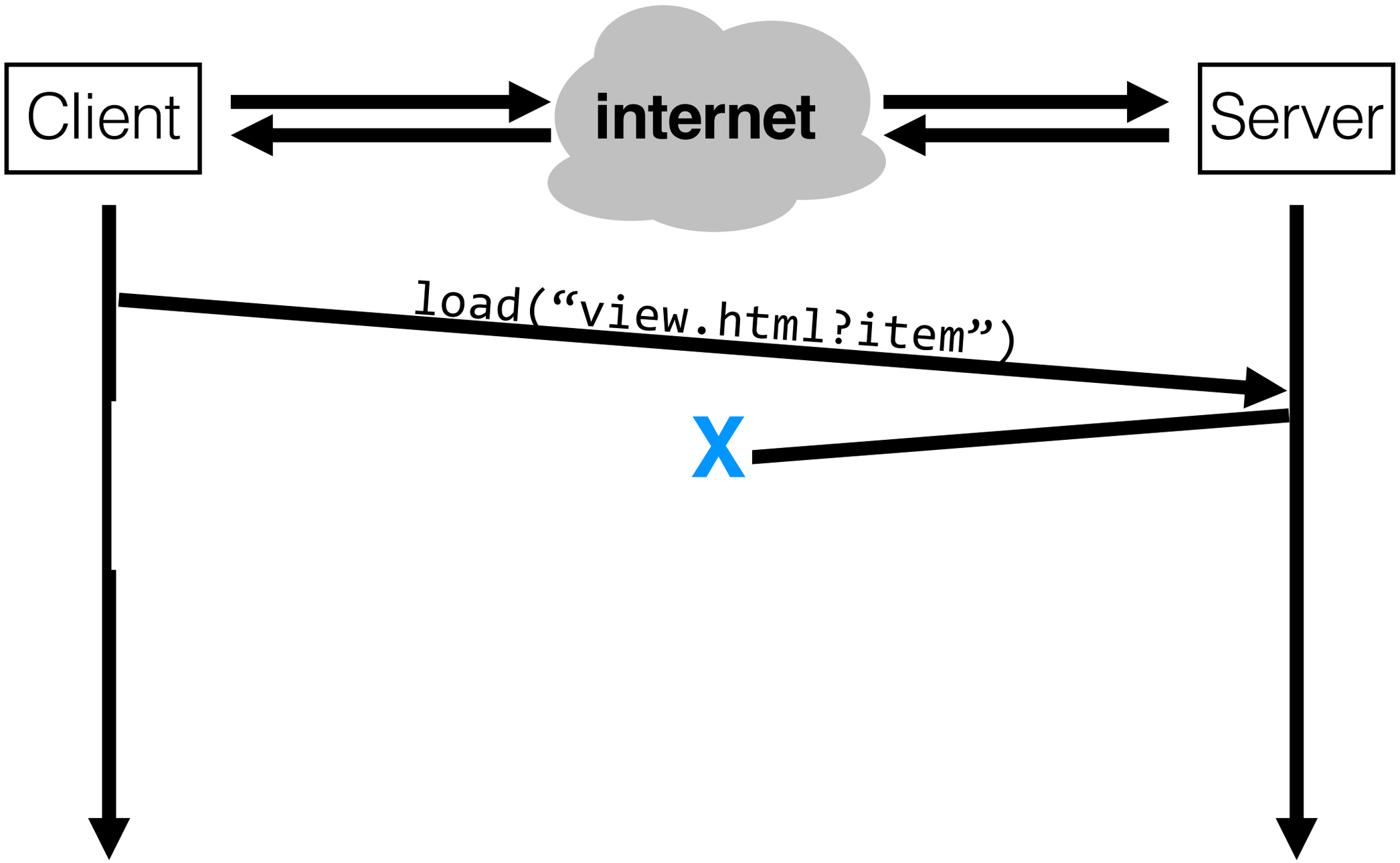
stub

Challenges with RPCs

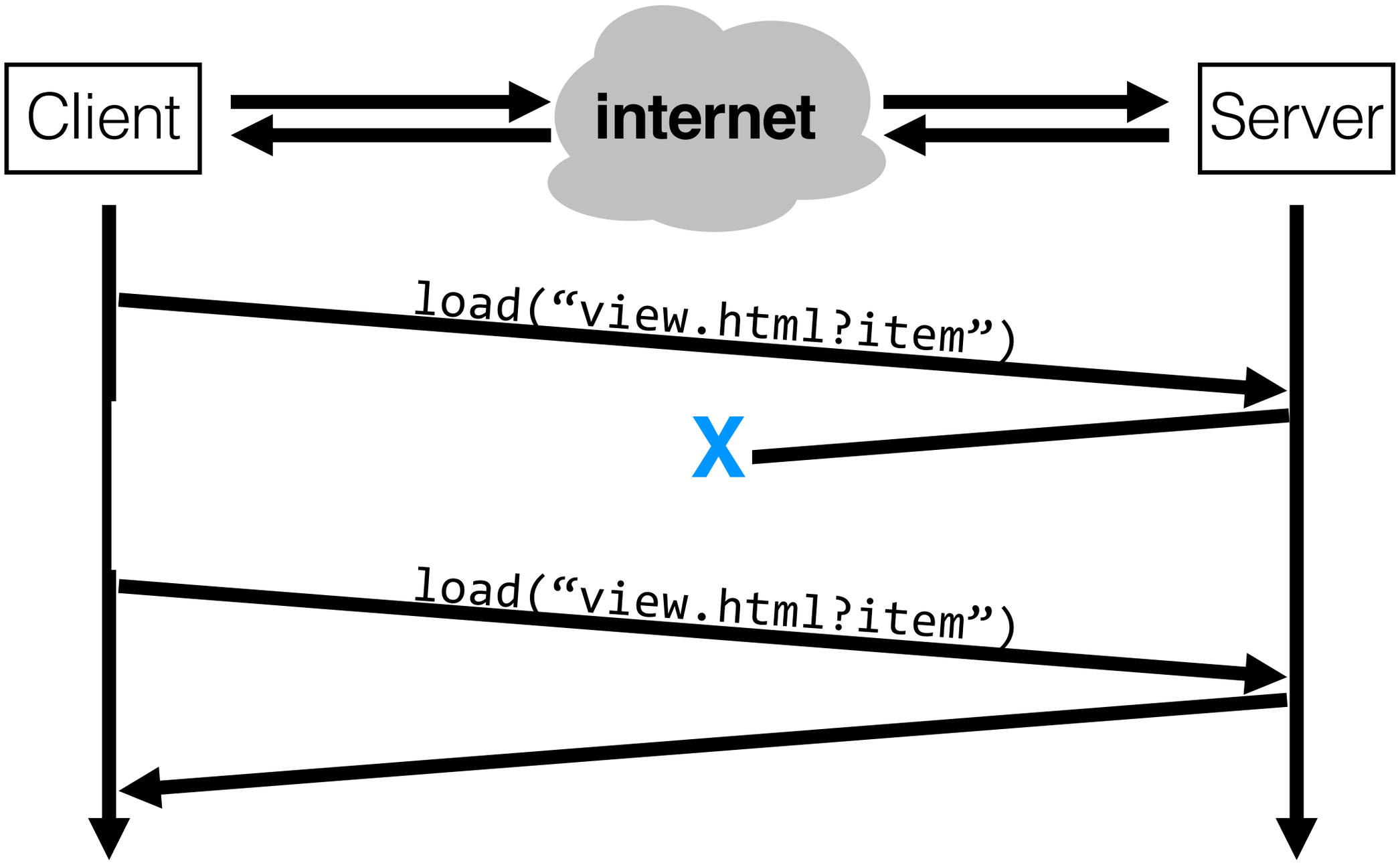
Challenges with RPCs



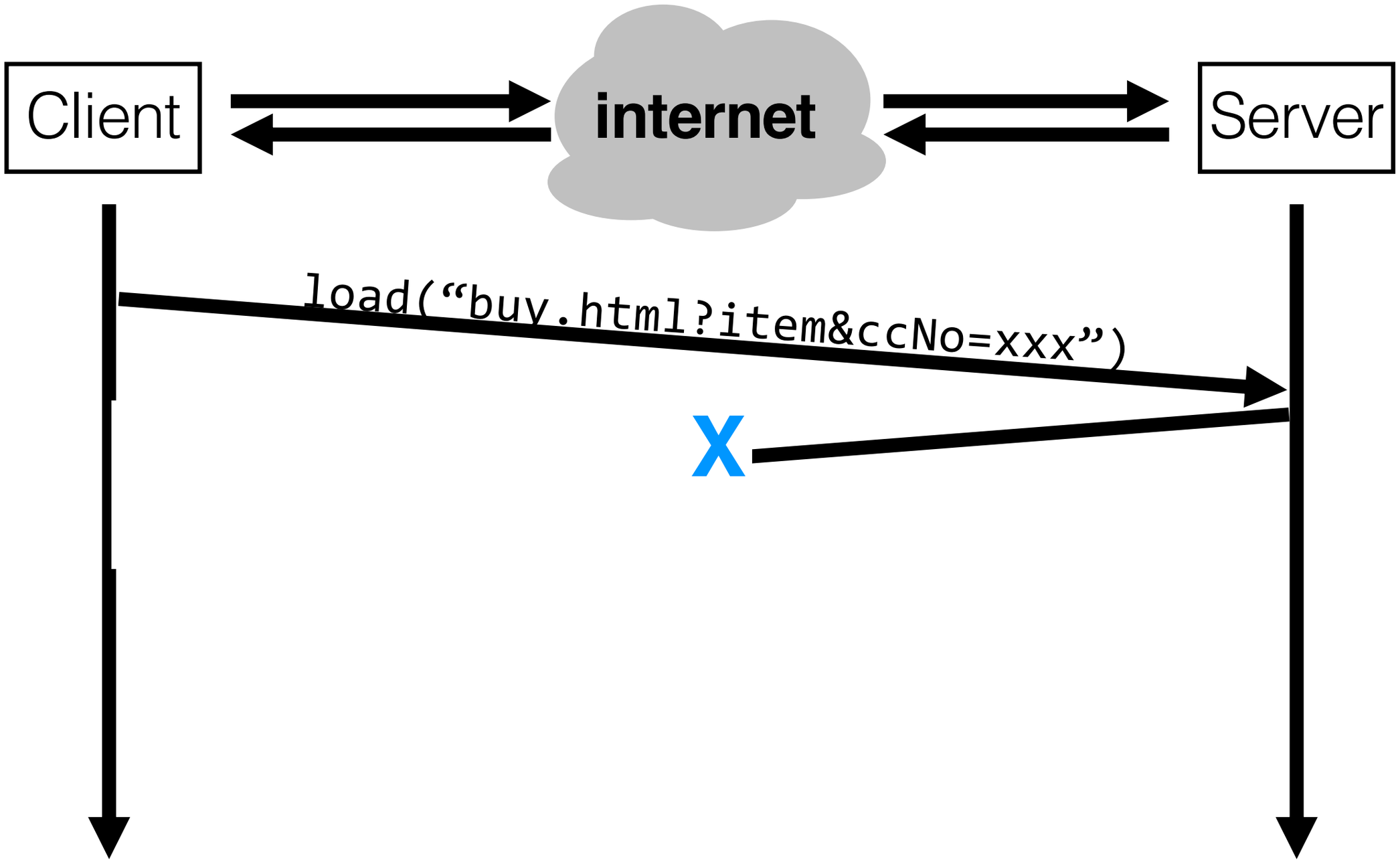
Challenges with RPCs



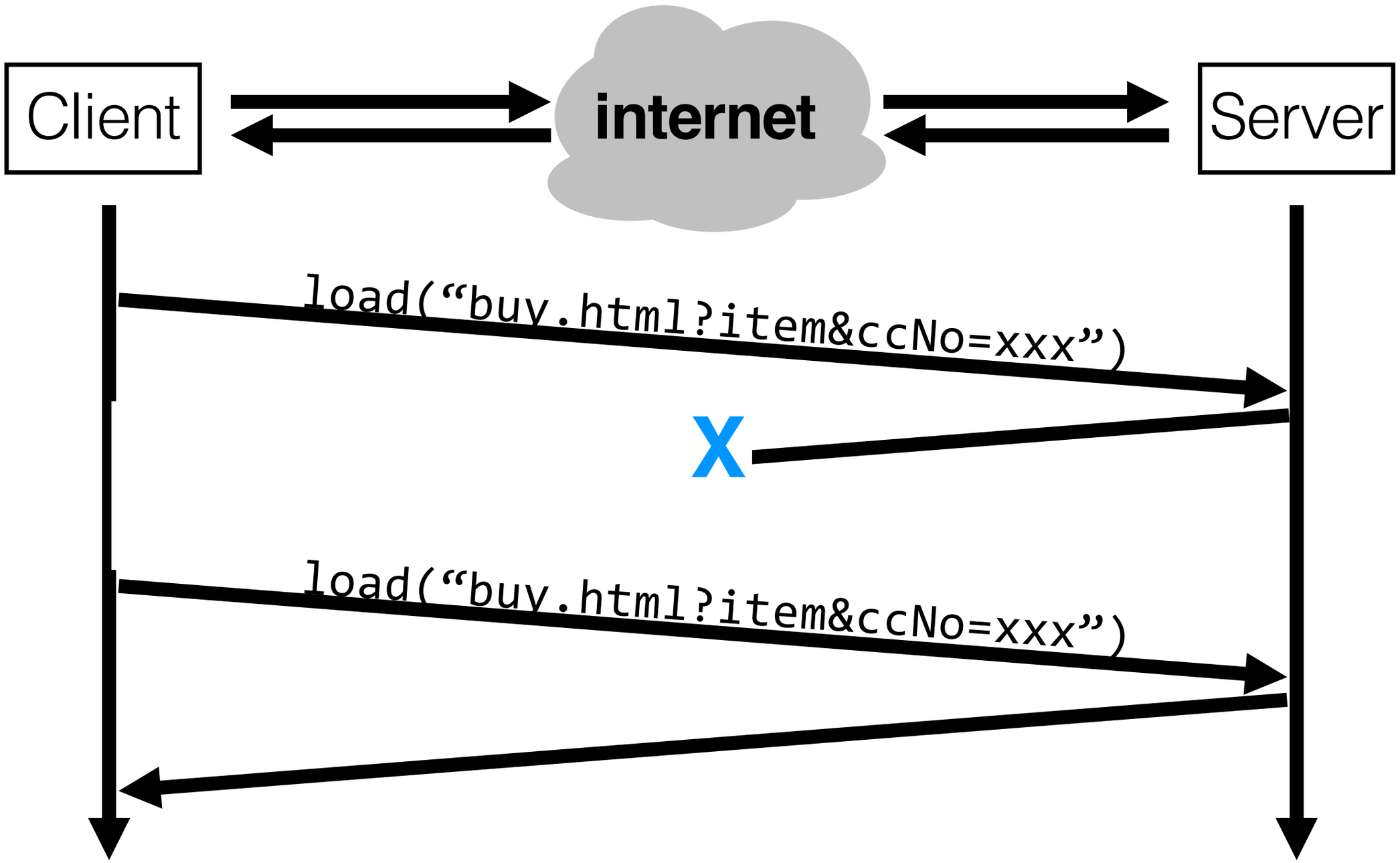
Challenges with RPCs



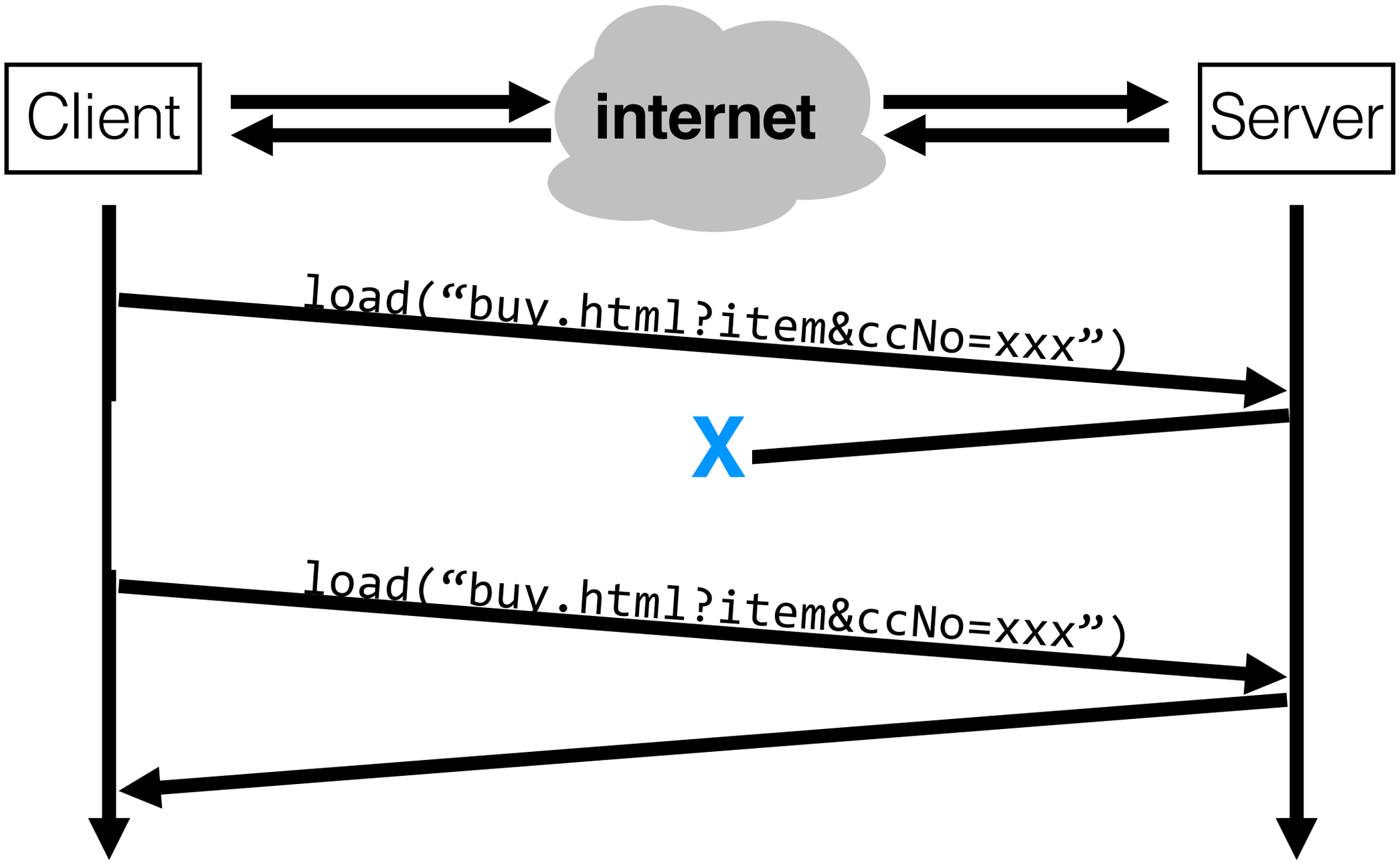
Challenges with RPCs



Challenges with RPCs

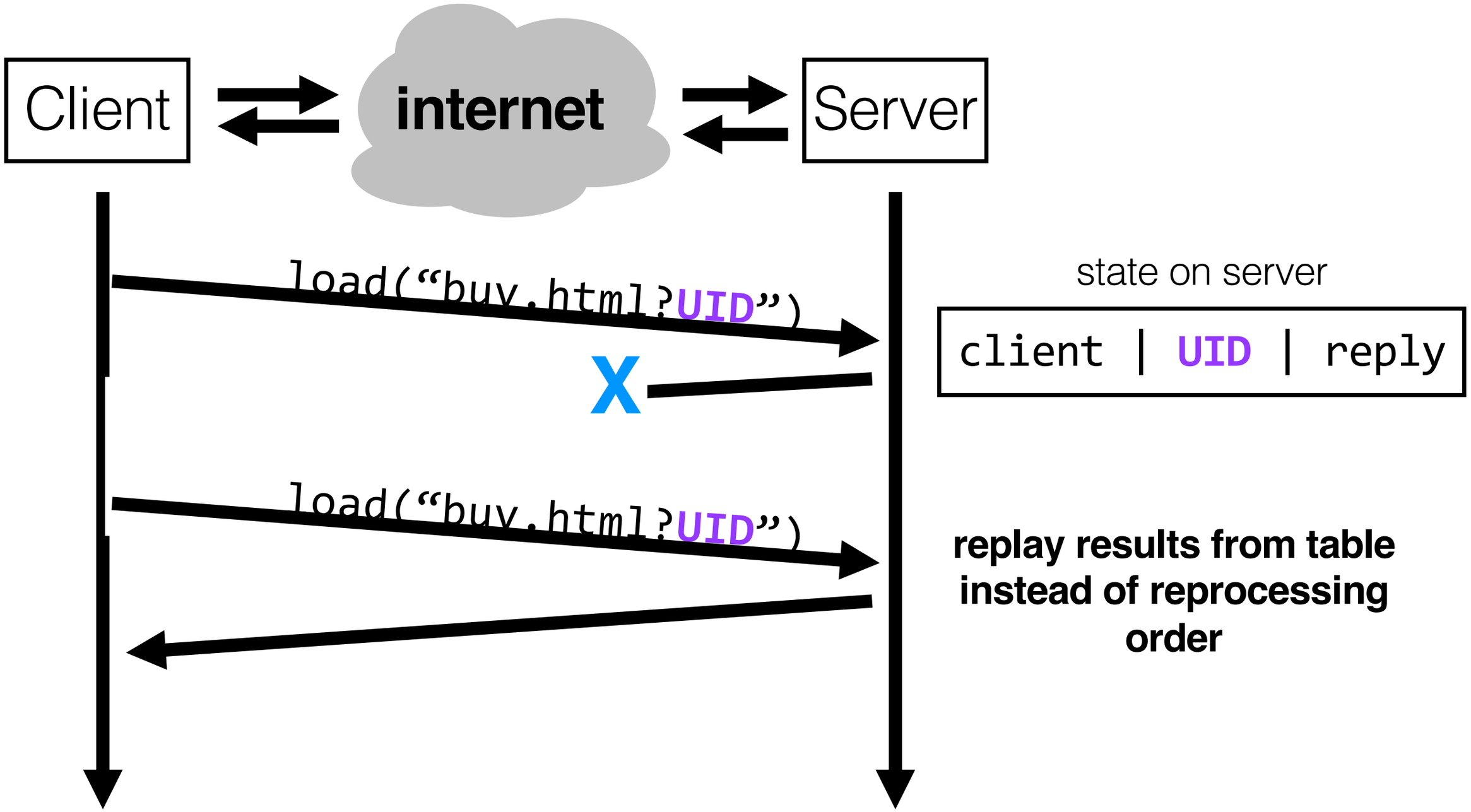


Challenges with RPCs



problem: just bought the same thing twice

Challenges with RPCs

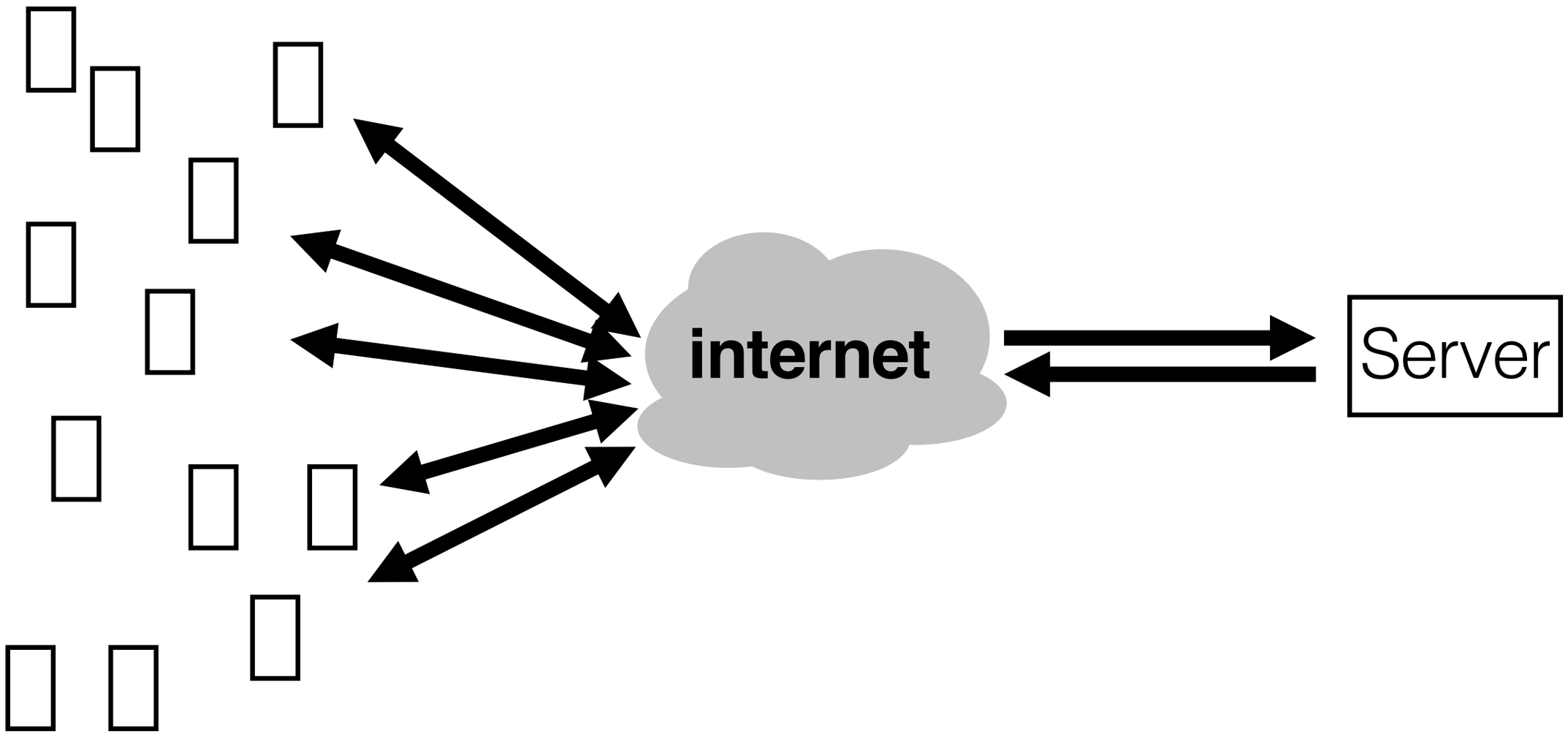


problem: server can still fail

What else might we want?

What else might we want?

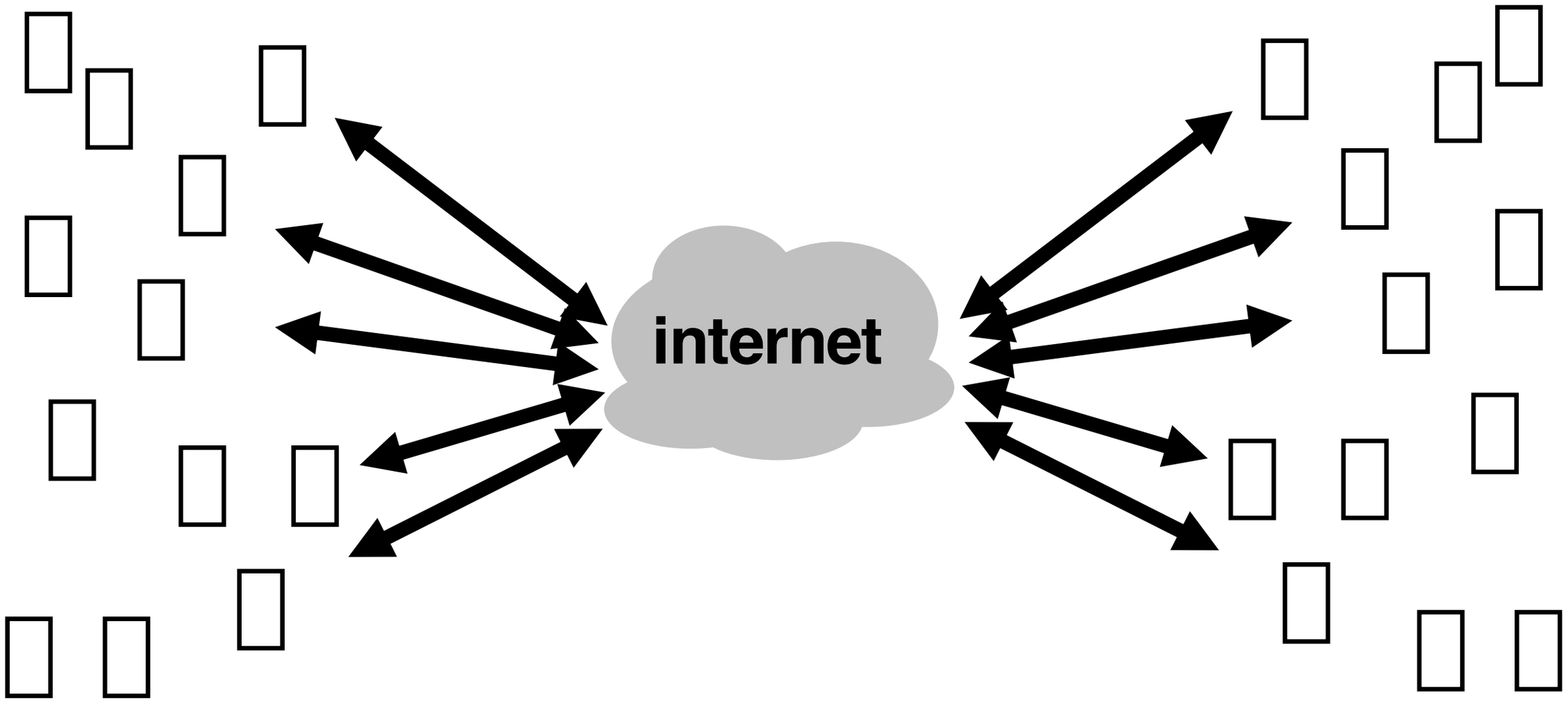
scalability



What else might we want?

scalability

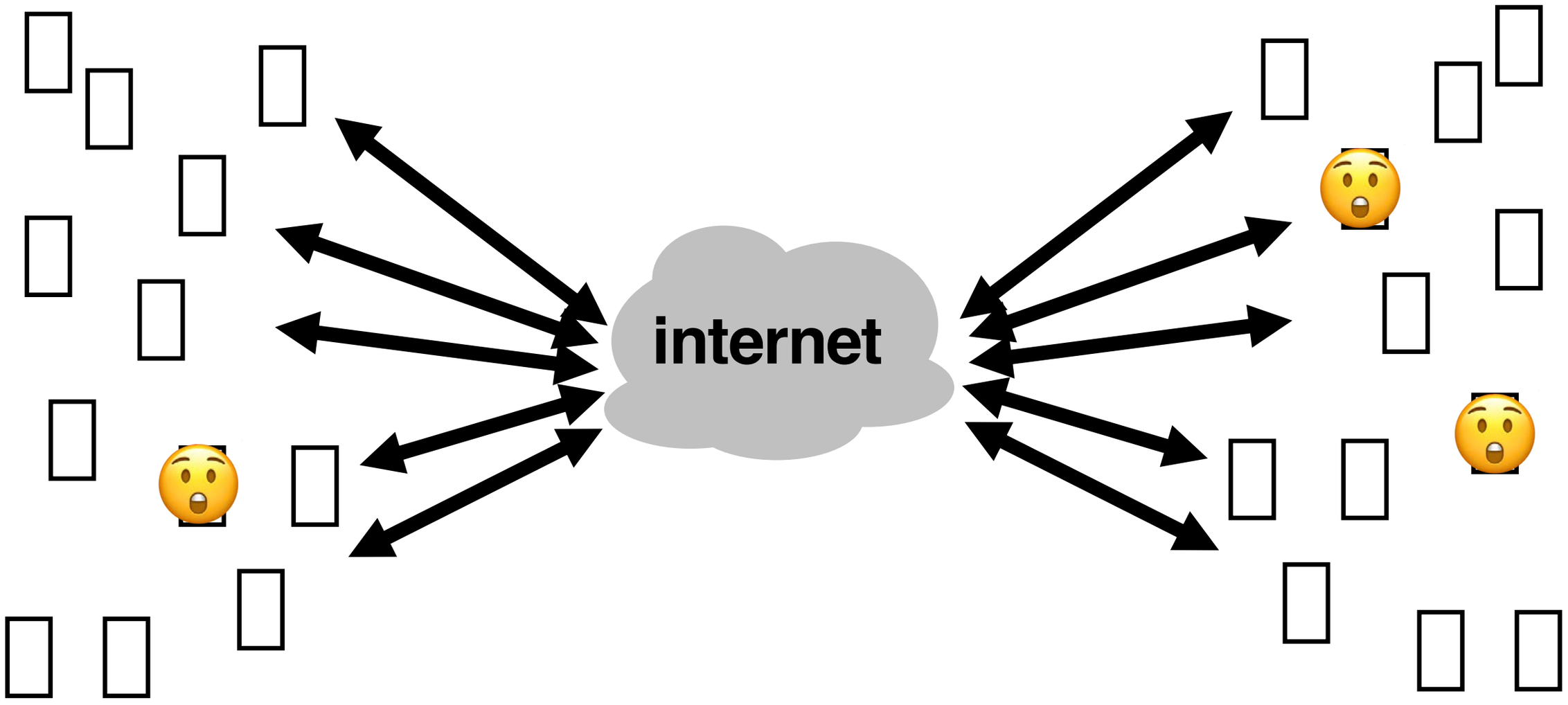
fault-tolerance/reliability



What else might we want?

scalability

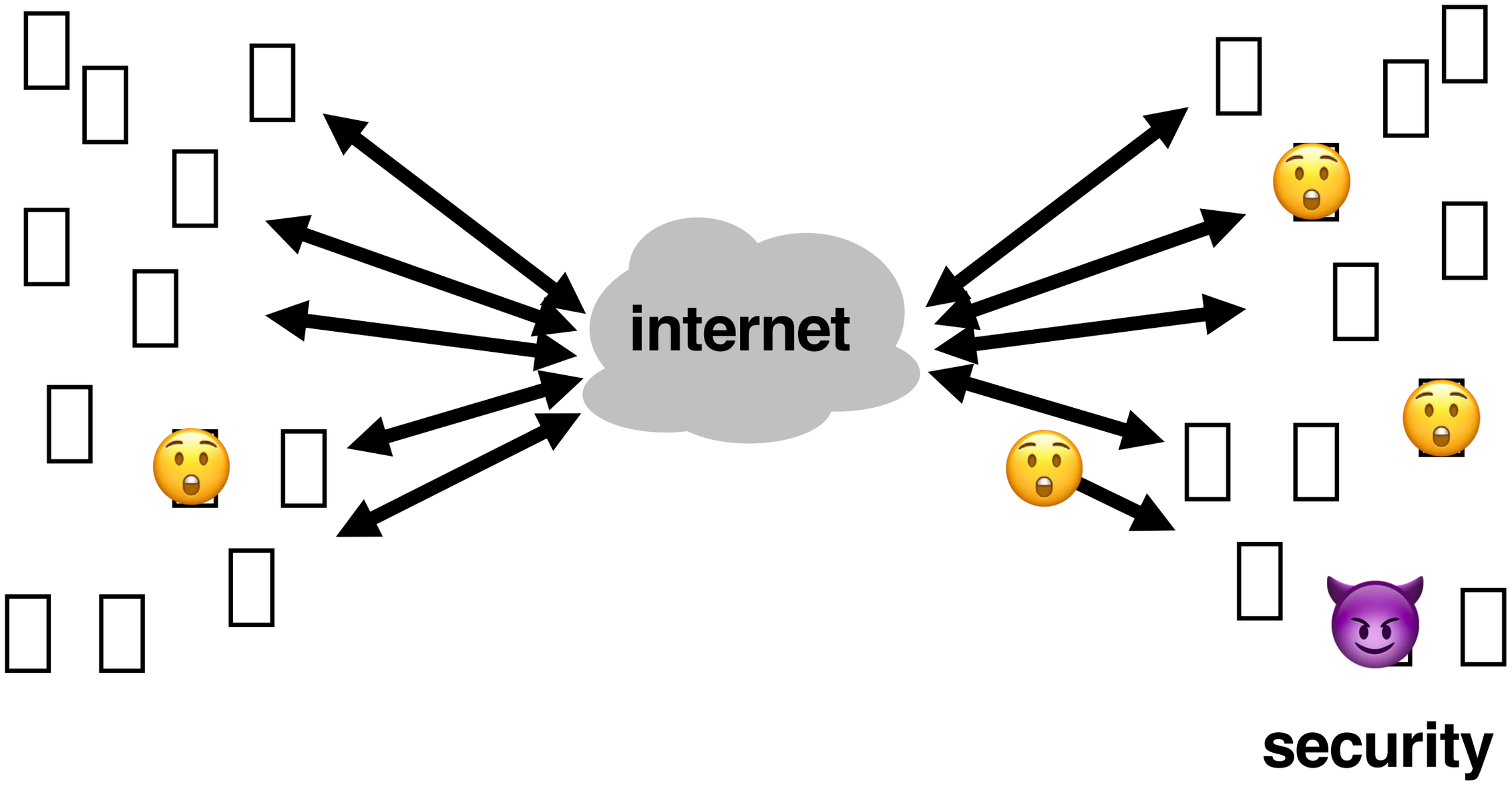
fault-tolerance/reliability



What else might we want?

scalability

fault-tolerance/reliability



security

- **Complexity** limits what we can build, but can be mitigated with **modularity** and **abstraction**

- One way to **enforce modularity** is with a **client/server model**, where the two modules reside on different machines and communicate with RPCs; network/server failures are still an issue

next lecture: naming, which allows modules to communicate

coming up: operating systems, which enforce modularity on a single machine

6.033 Computer System Engineering
Spring 2018