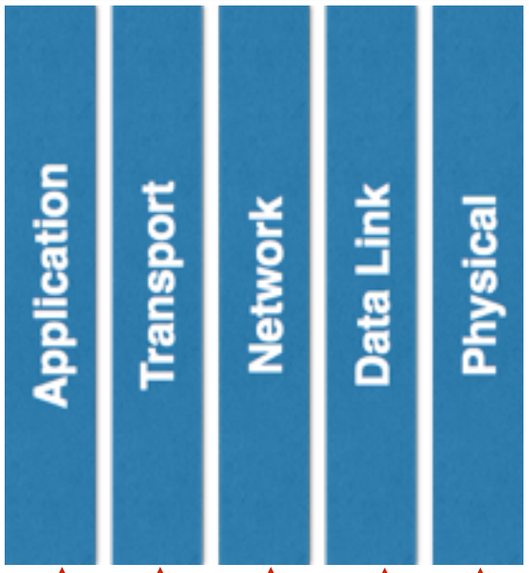
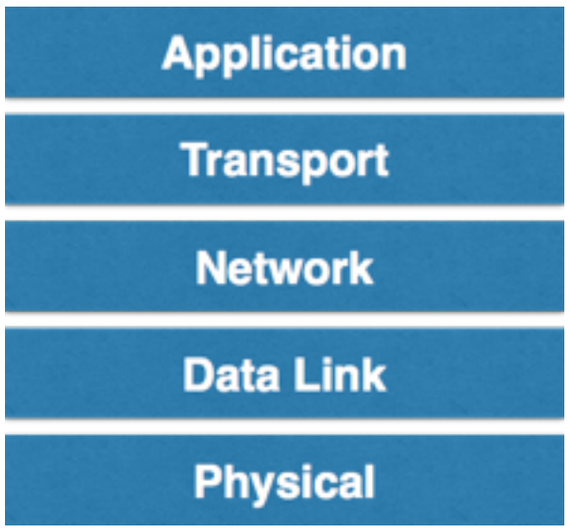
FIT1047 Week 8

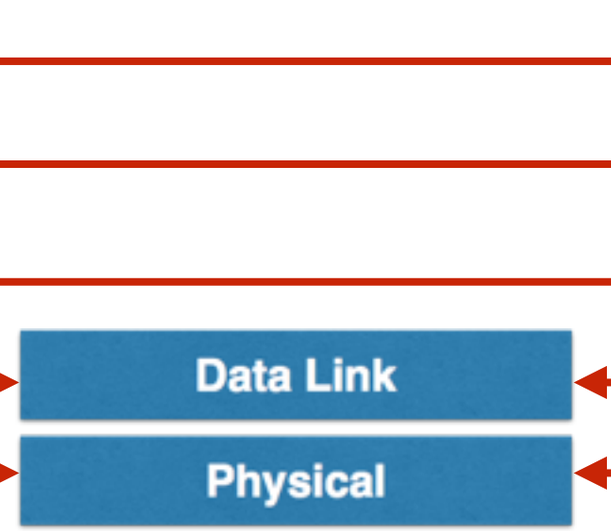
Networks: Application Layer

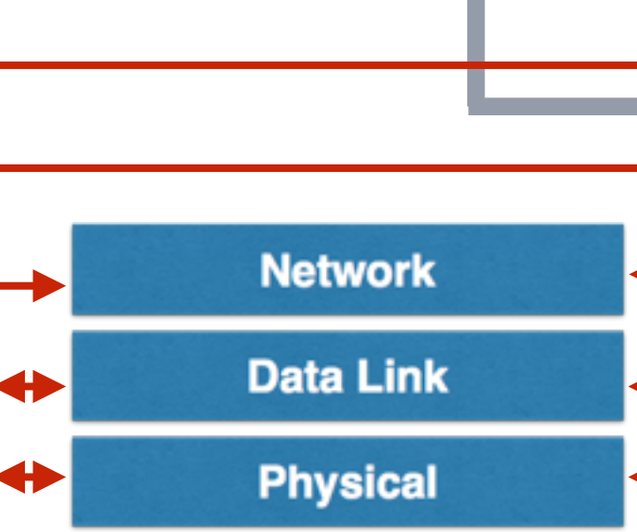
MONASH University

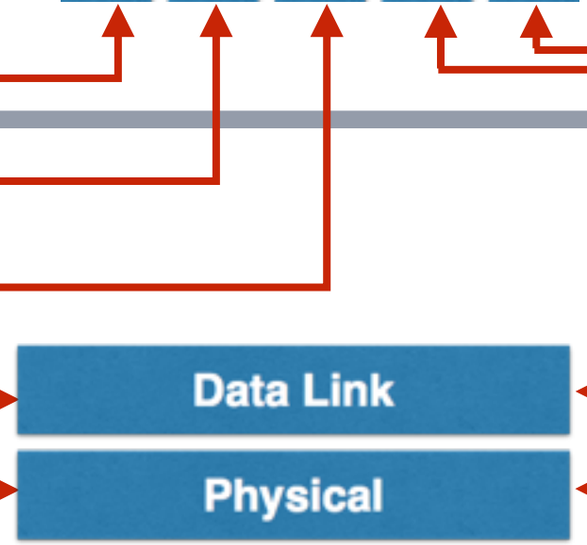
Layers of Abstraction



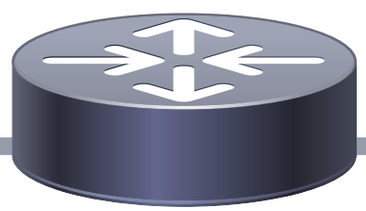








Switch



Router

Switch

Architectures

Application Architectures

Application Architectures

Presentation logic

The user interface. Controls the application.

Application Architectures

Presentation logic

The user interface. Controls the application.

Application / business logic

Defines what the application does.

Application Architectures

Presentation logic

The user interface. Controls the application.

Application / business logic

Defines what the application does.

Data access logic

Defines how the application manages its data.

Application Architectures

Presentation logic

The user interface. Controls the application.

Application / business logic

Defines what the application does.

Data access logic

Defines how the application manages its data.

Data storage

Where the data is kept, e.g. files or data base.

Application Architectures

Presentation logic

The user interface. Controls the application.

Application / business logic

Defines what the application does.

Data access logic

Defines how the application manages its data.

Data storage

Where the data is kept, e.g. files or data base.

Who does what?

Application Architectures

Presentation logic

Application / business logic

Data access logic

Data storage

Who does what?

Server-based Architecture

"dumb" terminal

Presentation logic
Application / business logic
Data access logic
Data storage

Server-based Architecture

"dumb" terminal

Client sends keystrokes to the server, displays text according to server's instructions.

Problems:
Server can become a bottleneck.
Upgrade expensive and "lumpy".

Presentation logic
Application / business logic
Data access logic
Data storage

Client-based Architecture

Presentation logic
Application / business logic
Data access logic

Data storage

Client-based Architecture

Presentation logic
Application / business logic
Data access logic

All the logic is performed by the client.
The server stores the data.

Problems:
All data must travel back and forth between server and client.

Data storage

Client-Server Architecture

Presentation logic
Application / business logic

Data access logic
Data storage

Client-Server Architecture

Presentation logic
Application / business logic

Balance the processing load between client and server.

Data access logic
Data storage

Thin-Client Architecture

Presentation logic

Application / business logic
Data access logic
Data storage

Thin-Client Architecture

Presentation logic

Server handles most of the application logic.

Advantage:
Only one server needs updating.

Application / business logic
Data access logic
Data storage

Multi-Tier Architecture

Presentation logic

Application / business logic

Data access logic
Data storage

Multi-Tier Architecture

Presentation logic

Application / business logic

Data access logic
Data storage

Multi-tier architecture

Multi-Tier Architecture

Presentation logic

Application / business logic

Data access logic
Data storage

Multi-tier architecture


client


server


server

Peer-To-Peer Architecture

Presentation logic
Application / business logic
Data access logic
Data storage

Presentation logic
Application / business logic
Data access logic
Data storage

Peer-To-Peer Architecture

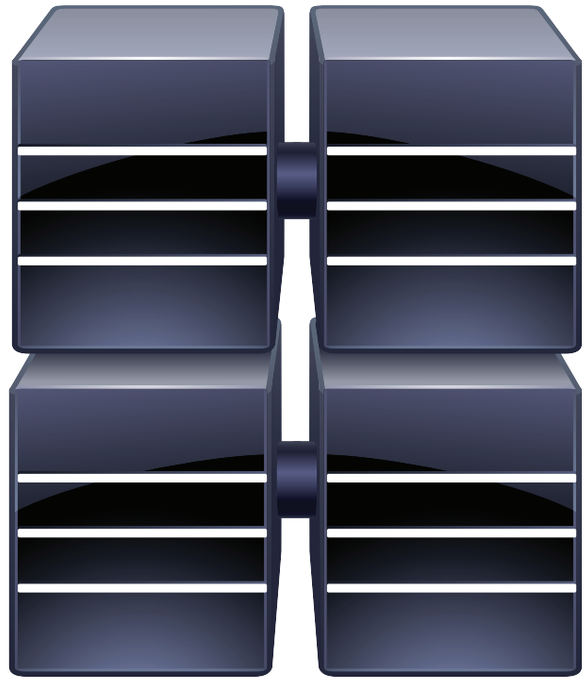
Presentation logic
Application / business logic
Data access logic
Data storage

All computers act as both clients and servers.

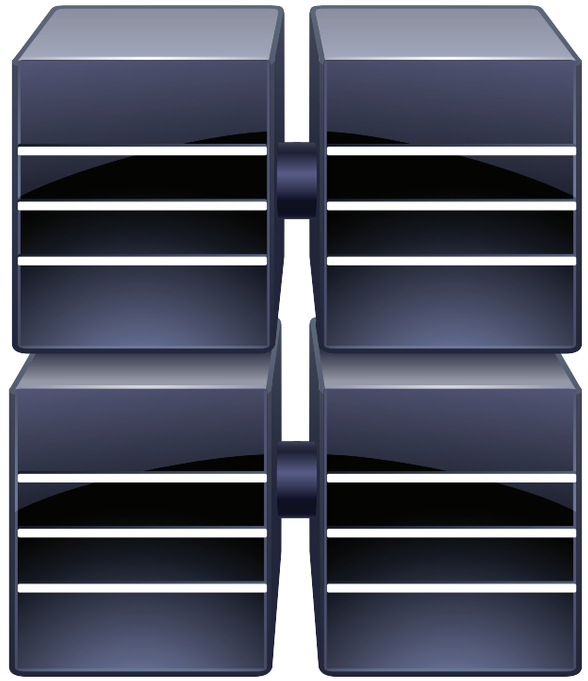
Use local logic to access data stored on another computer.

Presentation logic
Application / business logic
Data access logic
Data storage

World Wide Web



Internet

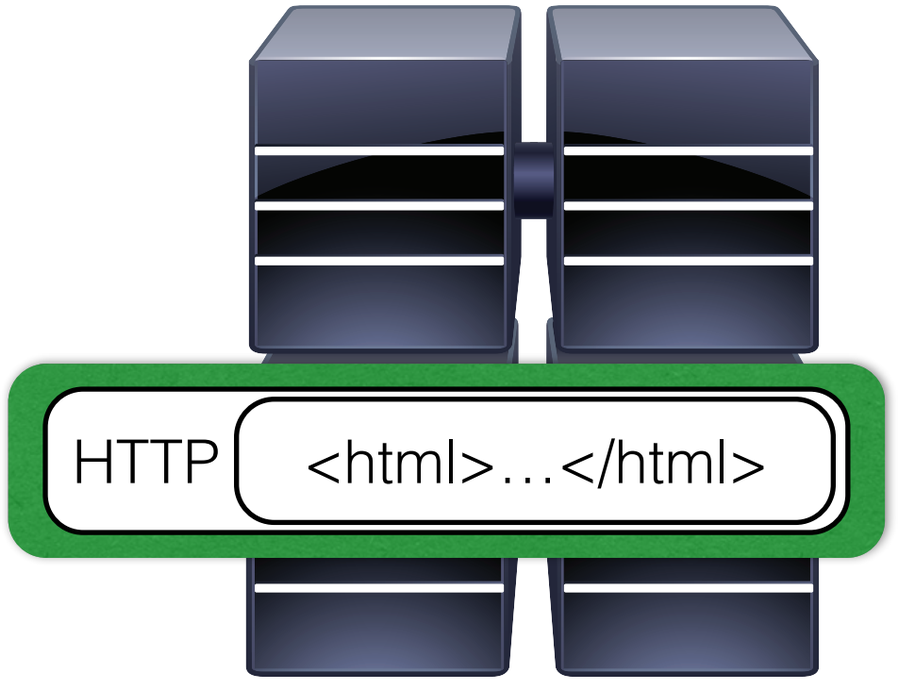


Internet


HTTP
www.youtube.com


Internet


The IT Crowd - Series 3 - Episode 4: The Internet


HTTP
<html>…</html>


Internet


The IT Crowd - Series 3 - Episode 4: The Internet

Internet

What is HTTP?
What is HTML?

Hyper**T**ext **T**ransfer **P**rotocol (HTTP)

- Defines how web **browsers** talk to **web servers**

HyperText Transfer Protocol (HTTP)

- Defines how web **browsers** talk to **web servers**
- Invented in 1989 by Tim Berners-Lee at CERN:
  “How will we ever keep track of such a large project?”

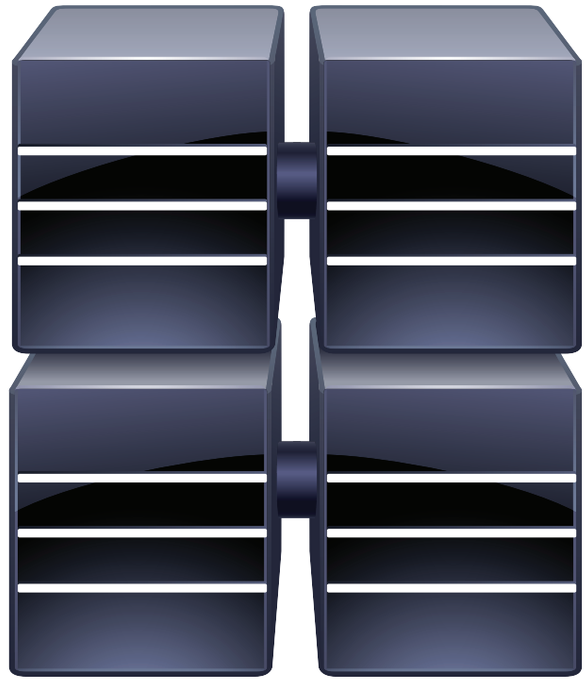
HyperText Transfer Protocol (HTTP)

- Defines how web **browsers** talk to **web servers**
- Invented in 1989 by Tim Berners-Lee at CERN:
  "How will we ever keep track of such a large project?"
- Based on two innovative ideas:
- **Hypertext:**
  A document containing links to other documents
- **Uniform Resource Locators (URLs):**
  A standard for identifying links to other documents

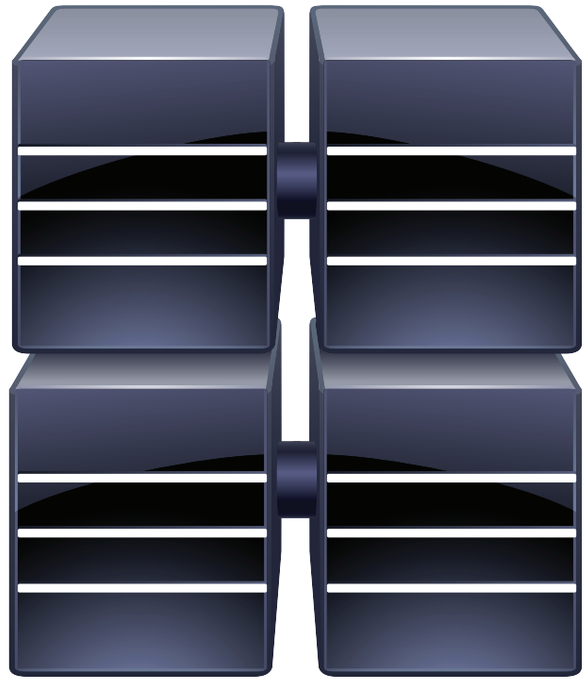
HyperText Transfer Protocol (HTTP)

- Defines how web **browsers** talk to **web servers**
- Invented in 1989 by Tim Berners-Lee at CERN:
  "How will we ever keep track of such a large project?"
- Based on two innovative ideas:
  - **Hypertext:**
    A document containing links to other documents
  - **Uniform Resource Locators (URLs):**
    A standard for identifying links to other documents
- HTTP defines **how** documents are requested and transferred

Request - Response-Cycle



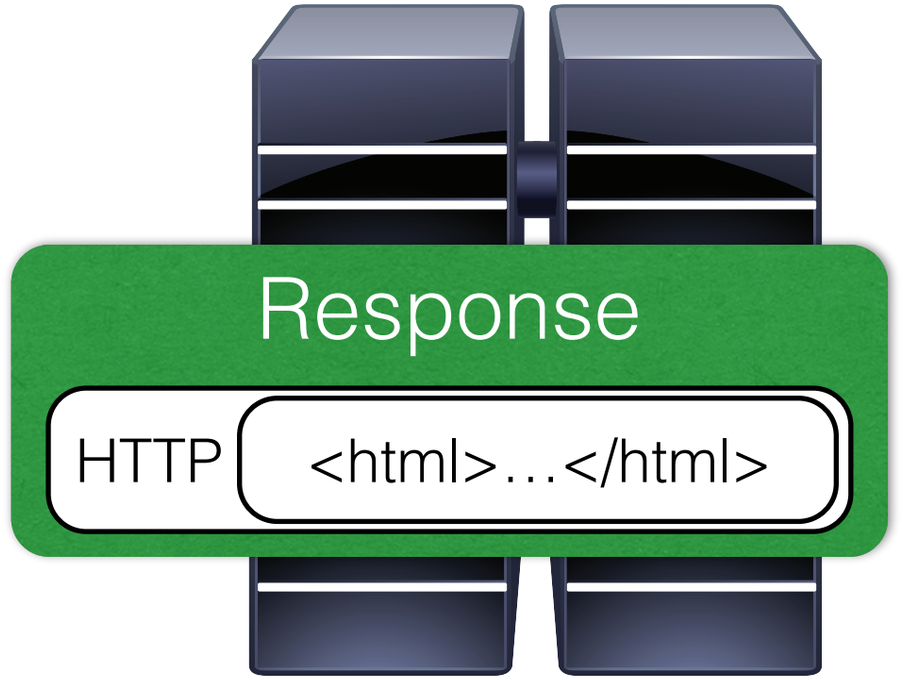
Request - Response-Cycle



Internet

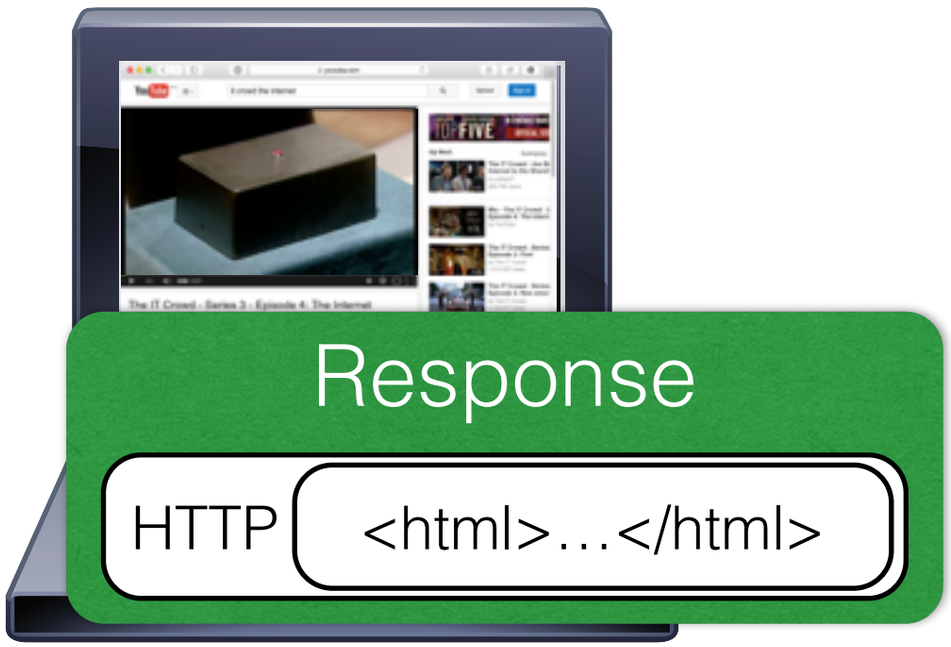
Request - Response-Cycle

Request - Response-Cycle



Request - Response-Cycle

Internet



Basic HTTP session

Basic HTTP session

client:

```
GET /~guidot/test.html HTTP/1.1
Host: www.csse.monash.edu
```

Basic HTTP session

client:

```
GET /~guidot/test.html HTTP/1.1
Host: www.csse.monash.edu
```

Request line

Basic HTTP session

client:

```
GET /~guidot/test.html HTTP/1.1
Host: www.csse.monash.edu
```

Request line

Req. header

Basic HTTP session

client:

```
GET /~guidot/test.html HTTP/1.1
Host: www.csse.monash.edu
```

Request line

Req. header

server:

```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:30:48 GMT
Server: Apache/1.3.26 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html

<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Basic HTTP session

client:

```
GET /~guidot/test.html HTTP/1.1
Host: www.csse.monash.edu
```

server:

```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:30:48 GMT
Server: Apache/1.3.26 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html

<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Basic HTTP session

client:

```
GET /~guidot/test.html HTTP/1.1
Host: www.csse.monash.edu
```

server:

```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:30:48 GMT
Server: Apache/1.3.26 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html

<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Response status

Basic HTTP session

client:

```
GET /~guidot/test.html HTTP/1.1
Host: www.csse.monash.edu
```

server:

```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:30:48 GMT
Server: Apache/1.3.26 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html

<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Basic HTTP session

client:
```
GET /~guidot/test.html HTTP/1.1
Host: www.csse.monash.edu
```

server:
```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:30:48 GMT
Server: Apache/1.3.26 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html

<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Response header

Basic HTTP session

client:

```
GET /~guidot/test.html HTTP/1.1
Host: www.csse.monash.edu
```

server:

```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:30:48 GMT
Server: Apache/1.3.26 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html

<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Basic HTTP session

client:
```
GET /~guidot/test.html HTTP/1.1
Host: www.csse.monash.edu
```

server:
```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:30:48 GMT
Server: Apache/1.3.26 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html
```

```
<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Response body

Basic HTTP session

client:

```
GET /~guidot/test.html HTTP/1.1
Host: www.csse.monash.edu
```

server:

```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:30:48 GMT
Server: Apache/1.3.26 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html

<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Basic HTTP session

```
<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Basic HTTP session

```
<html>
<body>
   <h1>Guido Tack</h1>
   <img src="images/guido3.jpg">
</body>
</html>
```

Basic HTTP session

```
<html>
<body>
   <h1>Guido Tack</h1>
   <img src="images/guido3.jpg">
</body>
</html>
```

client:
```
GET /~guidot/images/guido3.jpg HTTP/1.1
Host: www.csse.monash.edu
```

Basic HTTP session

client:

```
GET /~guidot/images/guido3.jpg HTTP/1.1
Host: www.csse.monash.edu
```

Basic HTTP session

client:

```
GET /~guidot/images/guido3.jpg HTTP/1.1
Host: www.csse.monash.edu
```

server:

```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:31:23 GMT
Server: Apache/1.3.26 (Unix)
Last-Modified: Tue, 20 Nov 2012 03:29:22 GMT
Accept-Ranges: bytes
Content-Length: 15681
Content-Type: image/jpeg

JFIFHH@ICC_PROFILE0appl mntrRGB
…
```

HTTP Methods

- GET:
  Retrieve specified URL from server

HTTP Methods

- `GET`:
  Retrieve specified URL from server

- `HEAD`:
  Retrieve **only header** for specified URL

HTTP Methods

- `GET`:
  Retrieve specified URL from server

- `HEAD`:
  Retrieve **only header** for specified URL

- `POST`:
  Add data specified in request body to specified URL

  E.g. add a message to a web forum, or an item to a shopping cart. Also retrieves document.

HTTP Methods

- `GET`:
  Retrieve specified URL from server

- `HEAD`:
  Retrieve **only header** for specified URL

- `POST`:
  Add data specified in request body to specified URL

  E.g. add a message to a web forum, or an item to a shopping cart. Also retrieves document.

- Other methods (`PUT`, `DELETE`, `OPTIONS`…) less common

Full HTTP Request

```
POST /~guidot/test_form.php HTTP/1.1
Host: www.csse.monash.edu
Connection: keep-alive
Accept: text/html,application/xhtml+xml
User-Agent: Mozilla/5.0 (Macintosh)
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US;en
Content-Type: application/x-www-form-urlencoded
Content-Length: 26

search=data+communications
```

Full HTTP Request

```
POST /~guidot/test_form.php HTTP/1.1
Host: www.csse.monash.edu
Connection: keep-alive
Accept: text/html,application/xhtml+xml
User-Agent: Mozilla/5.0 (Macintosh)
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US;en
Content-Type: application/x-www-form-urlencoded
Content-Length: 26

search=data+communications
```

Request line

Full HTTP Request

```
POST /~guidot/test_form.php HTTP/1.1
Host: www.csse.monash.edu
Connection: keep-alive
Accept: text/html,application/xhtml+xml
User-Agent: Mozilla/5.0 (Macintosh)
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US;en
Content-Type: application/x-www-form-urlencoded
Content-Length: 26

search=data+communications
```

Request line

Request header

Full HTTP Request

```
POST /~guidot/test_form.php HTTP/1.1
Host: www.csse.monash.edu
Connection: keep-alive
Accept: text/html,application/xhtml+xml
User-Agent: Mozilla/5.0 (Macintosh)
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US;en
Content-Type: application/x-www-form-urlencoded
Content-Length: 26

search=data+communications
```

Request line

Request header

Request body

Full HTTP Request

```
POST /~guidot/test_form.php HTTP/1.1
Host: www.csse.monash.edu
Connection: keep-alive
Accept: text/html,application/xhtml+xml
User-Agent: Mozilla/5.0 (Macintosh)
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US;en
Content-Type: application/x-www-form-urlencoded
Content-Length: 26

search=data+communications
```

Request line

Request header

Request body

Request header and body are optional.
HTTP 1.1 requires at least the `Host` header.

Full HTTP Response

```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:30:48 GMT
Server: Apache/1.3.26 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html

<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Full HTTP Response

```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:30:48 GMT
Server: Apache/1.3.26 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html

<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Response status

Full HTTP Response

```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:30:48 GMT
Server: Apache/1.3.26 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html
```

Response status

Response header

```
<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Full HTTP Response

```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:30:48 GMT
Server: Apache/1.3.26 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html
```

Response status

Response header

```
<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Response body

Full HTTP Response

```
HTTP/1.1 200 OK
Date: Thu, 05 Mar 2015 08:30:48 GMT
Server: Apache/1.3.26 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html
```

Response status

Response header

```
<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
</body>
</html>
```

Response body

Response header and body are optional.

HTTP is stateless

- Each request is an **independent transaction**
- Example: shopping cart

HTTP is stateless

- Each request is an **independent transaction**
- Example: shopping cart
  - *Client* browses online store

HTTP is stateless

- Each request is an **independent transaction**
- Example: shopping cart
  - *Client* browses online store
  - *Server* responds with web pages

HTTP is stateless

- Each request is an **independent transaction**
- Example: shopping cart
  - *Client* browses online store
  - *Server* responds with web pages
  - *Client* puts item into shopping cart

HTTP is stateless

- Each request is an **independent transaction**
- Example: shopping cart
  - *Client* browses online store
  - *Server* responds with web pages
  - *Client* puts item into shopping cart
  - *Client* continues browsing

HTTP is stateless

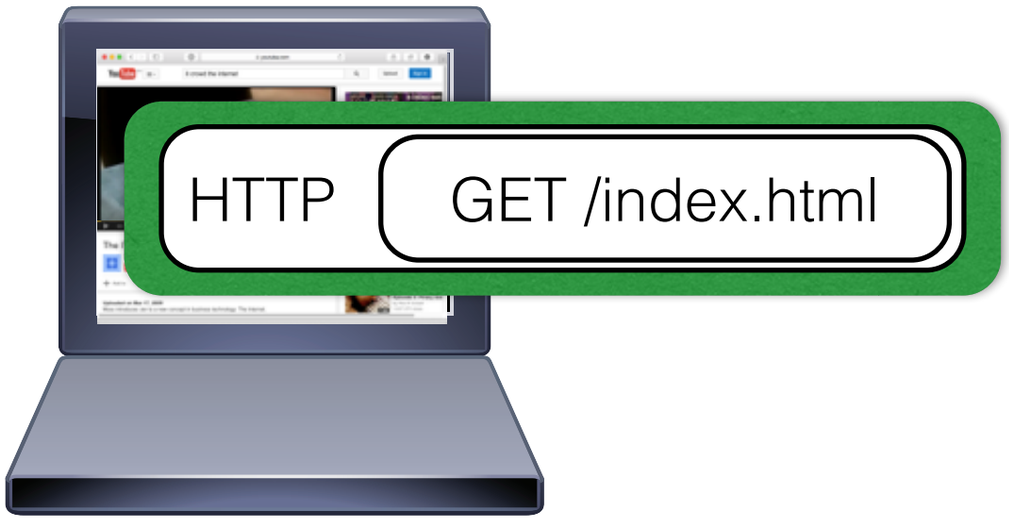
- Each request is an **independent transaction**
- Example: shopping cart
  - *Client* browses online store
  - *Server* responds with web pages
  - *Client* puts item into shopping cart
  - *Client* continues browsing
  - How does the server keep track of **which client** put **which items** into the shopping cart?
    (This is called the **state** of the **session**)

Adding state to HTTP

- Two approaches:
  - Client sends **session identifier** with every request, e.g. encoded in URL or as data in POST
  - Server sets a **cookie,** client transmits the cookie with **every future request**

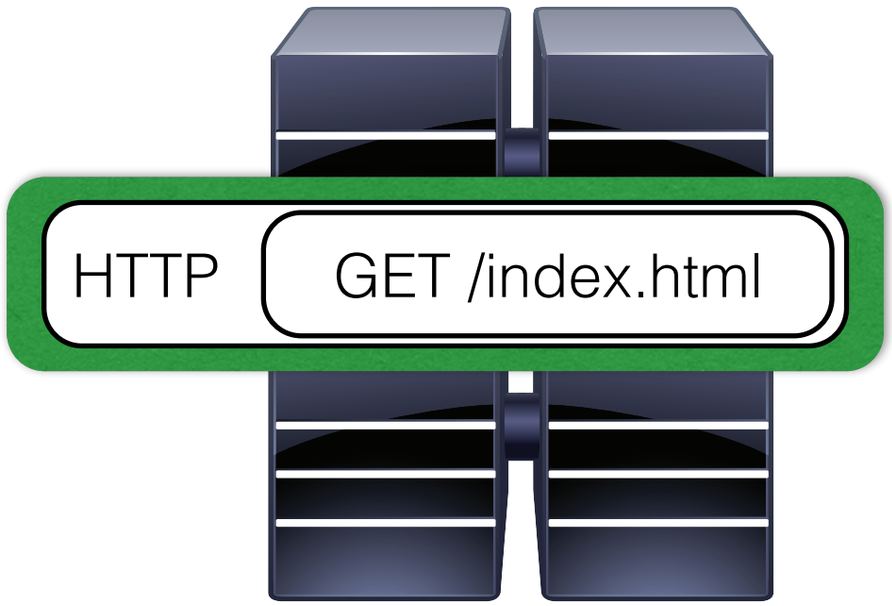
Adding state to HTTP

- Two approaches:
  - Client sends **session identifier** with every request, e.g. encoded in URL or as data in POST
  - Server sets a **cookie,** client transmits the cookie with **every future request**



Adding state to HTTP

- Two approaches:
  - Client sends **session identifier** with every request, e.g. encoded in URL or as data in POST
  - Server sets a **cookie,** client transmits the cookie with **every future request**

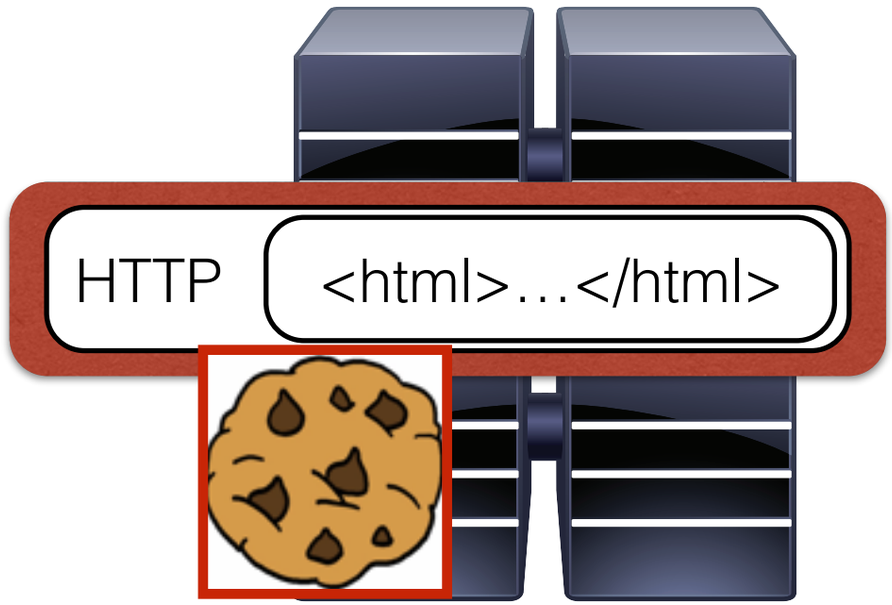


Adding state to HTTP

- Two approaches:
  - Client sends **session identifier** with every request, e.g. encoded in URL or as data in POST
  - Server sets a **cookie,** client transmits the cookie with **every future request**

Adding state to HTTP

- Two approaches:
  - Client sends **session identifier** with every request, e.g. encoded in URL or as data in POST
  - Server sets a **cookie,** client transmits the cookie with **every future request**



Adding state to HTTP

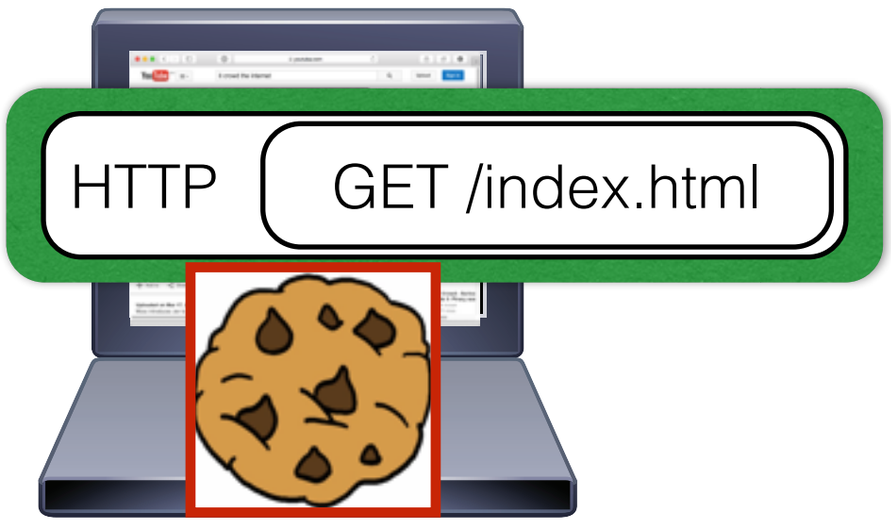
- Two approaches:
  - Client sends **session identifier** with every request, e.g. encoded in URL or as data in POST
  - Server sets a **cookie,** client transmits the cookie with **every future request**

Adding state to HTTP

- Two approaches:
  - Client sends **session identifier** with every request, e.g. encoded in URL or as data in POST
  - Server sets a **cookie,** client transmits the cookie with **every future request**

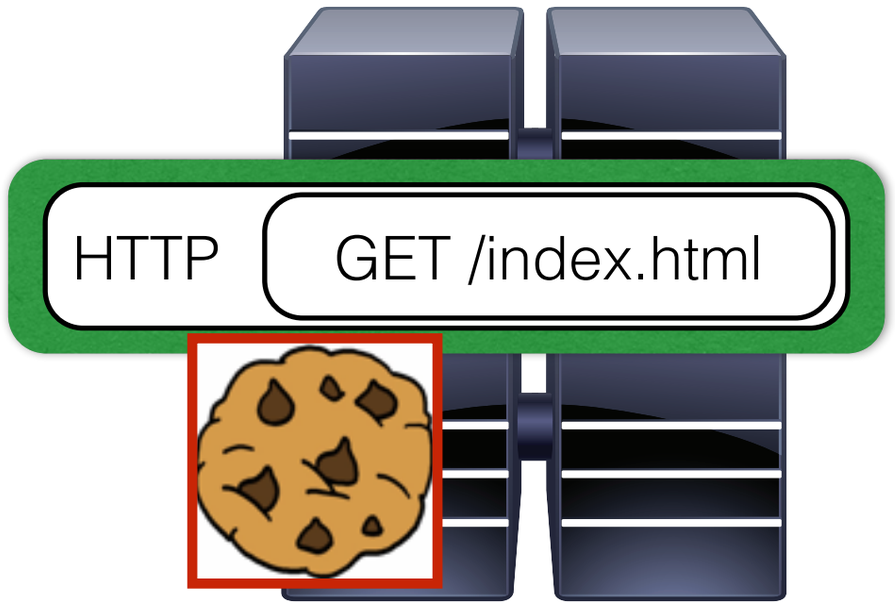
Adding state to HTTP

- Two approaches:
  - Client sends **session identifier** with every request, e.g. encoded in URL or as data in POST
  - Server sets a **cookie,** client transmits the cookie with **every future request**



Adding state to HTTP

- Two approaches:
  - Client sends **session identifier** with every request, e.g. encoded in URL or as data in POST
  - Server sets a **cookie,** client transmits the cookie with **every future request**



HTML

- **H**yper**T**ext **M**arkup **L**anguage
- Plain text document *annotated* with *tags* that describe how to format the file
- First version also developed by Berners-Lee in 1990

HTML

- **H**yper**T**ext **M**arkup **L**anguage
- Plain text document *annotated* with *tags* that describe how to format the file
- First version also developed by Berners-Lee in 1990
- Example:

```
<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
  <a href="http://www.w3c.org">W3C</a>
</body>
</html>
```

HTML

- **H**yper**T**ext **M**arkup **L**anguage
- Plain text document *annotated* with *tags* that describe how to format the file
- First version also developed by Berners-Lee in 1990
- Example:

```
<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
  <a href="http://www.w3c.org">W3C</a>
</body>
</html>
```

tag

HTML

- **H**yper**T**ext **M**arkup **L**anguage
- Plain text document *annotated* with *tags* that describe how to format the file
- First version also developed by Berners-Lee in 1990
- Example:

```
<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
  <a href="http://www.w3c.org">W3C</a>
</body>
</html>
```

tag

link to other asset

HTML

- **H**yper**T**ext **M**arkup **L**anguage
- Plain text document *annotated* with *tags* that describe how to format the file
- First version also developed by Berners-Lee in 1990
- Example:

```
<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
  <a href="http://www.w3c.org">W3C</a>
</body>
</html>
```

tag

link to other asset

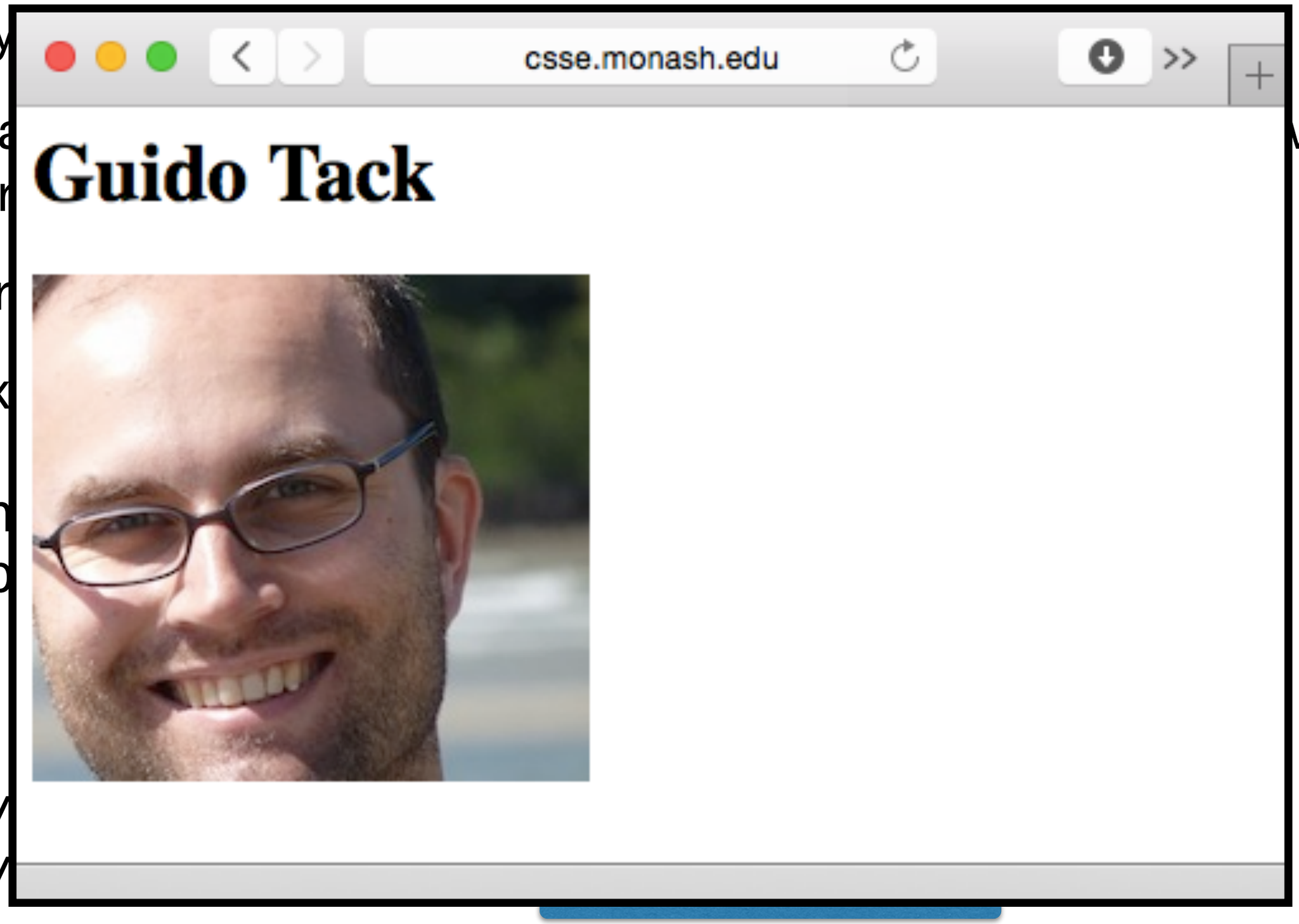
link to other page

HTML

- **Hy**
- Pla w to
  for
- Fir
- Ex

<h
<b

</
</



HTML

- **H**yper**T**ext **M**arkup **L**anguage
- Plain text document *annotated* with *tags* that describe how to format the file
- First version also developed by Berners-Lee in 1990
- Example:

```
<html>
<body>
  <h1>Guido Tack</h1>
  <img src="images/guido3.jpg">
  <a href="http://www.w3c.org">W3C</a>
</body>
</html>
```

tag

link to other asset

link to other page

HTML + CSS

- Separation of **structure** and **layout**:
  - HTML defines *what* each element of the page *is*
  - CSS defines *how* each element type should be *displayed*

HTML + CSS

- Separation of **structure** and **layout**:
  - HTML defines *what* each element of the page *is*
  - CSS defines *how* each element type should be *displayed*

- Example:

```
h1 {
   font-family: sans-serif;
   color: #006020;
   float: left;
}
img { float: right; }
```

HTML + CSS

- Se
  -
  -
- Ex

```
h1

}
im
```

HTML + CSS

- Separation of **structure** and **layout**:
  - HTML defines *what* each element of the page *is*
  - CSS defines *how* each element type should be *displayed*

- Example:

```
h1 {
   font-family: sans-serif;
   color: #006020;
   float: left;
}
img { float: right; }
```

Electronic Mail

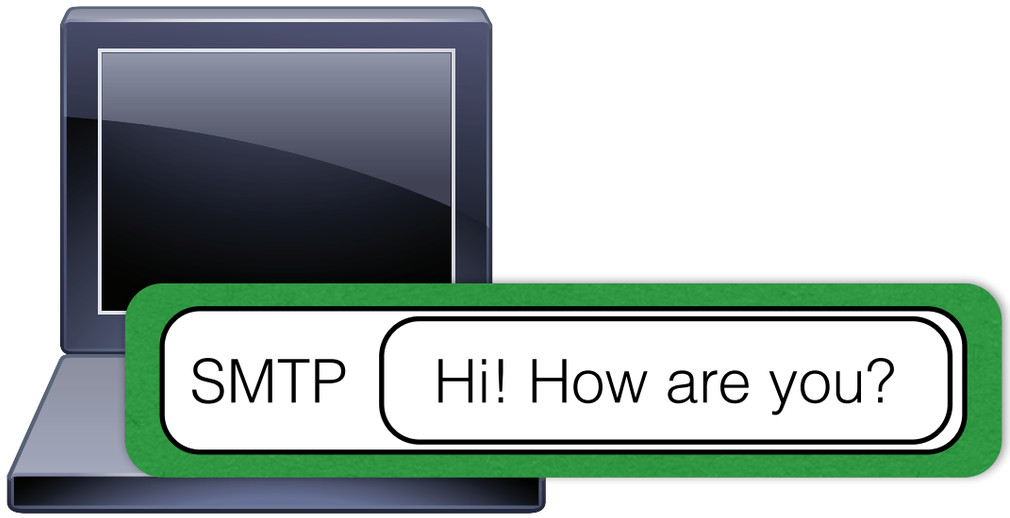
alice@hotmail.com

smtp.live.com

Internet

smtp.gmail.com

bob@gmail.com



alice@hotmail.com

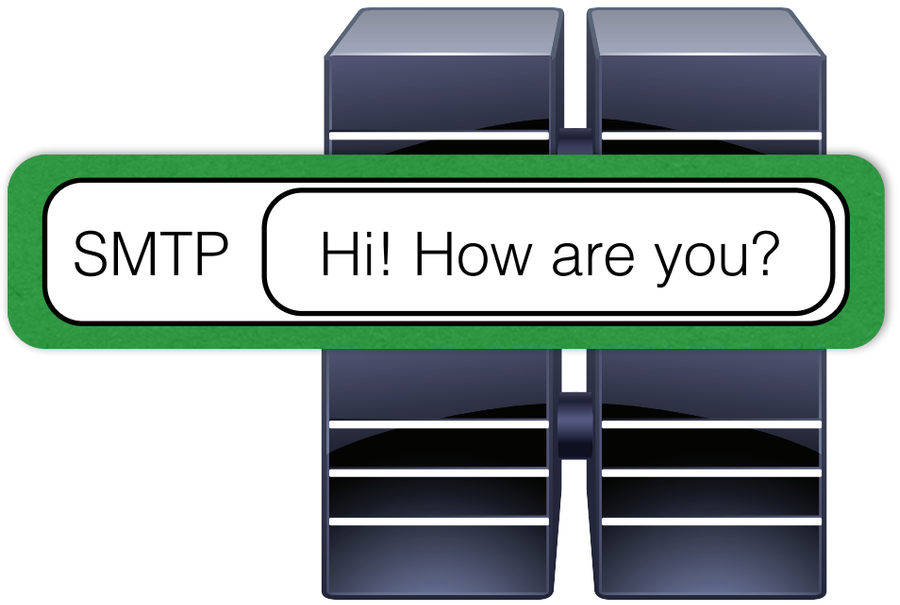
smtp.live.com

Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com



smtp.live.com

Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

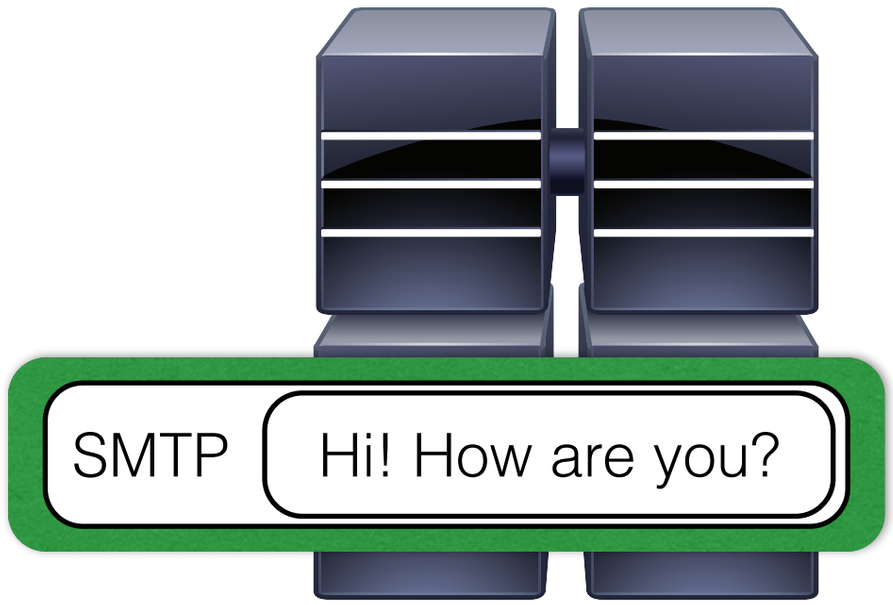
smtp.live.com

Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com



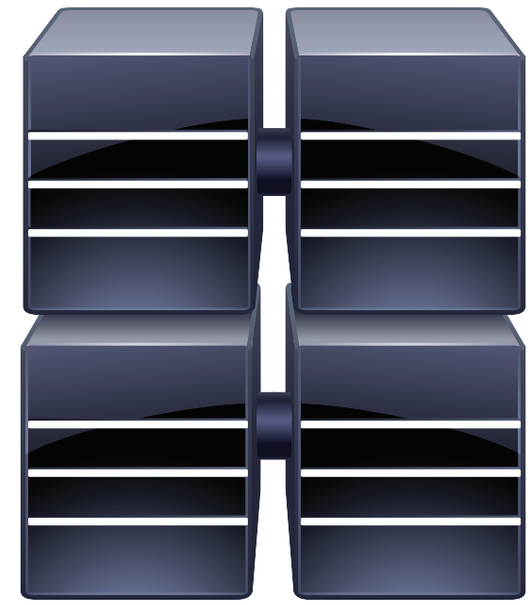
smtp.live.com

Internet

smtp.gmail.com

bob@gmail.com

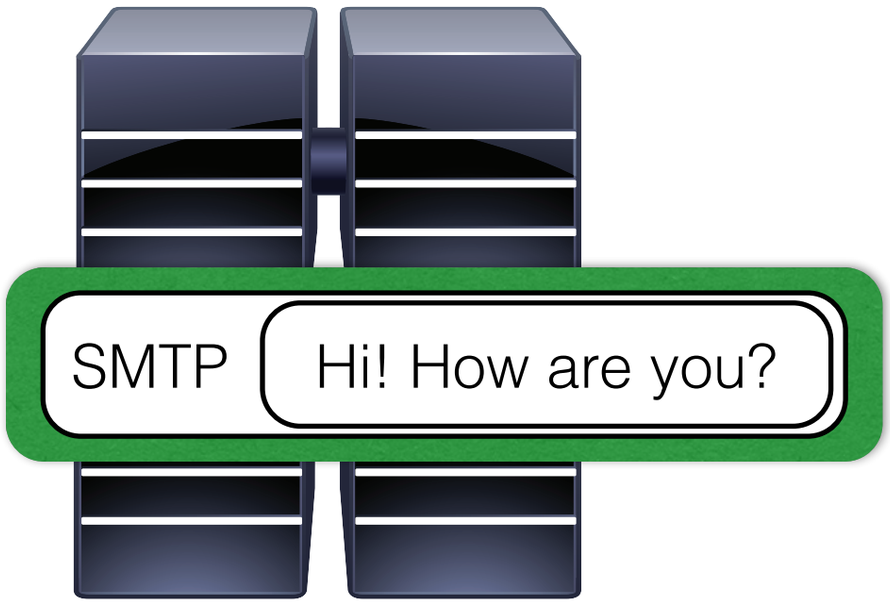
alice@hotmail.com



smtp.live.com

Internet

smtp.gmail.com



bob@gmail.com

alice@hotmail.com

smtp.live.com

Internet

smtp.gmail.com

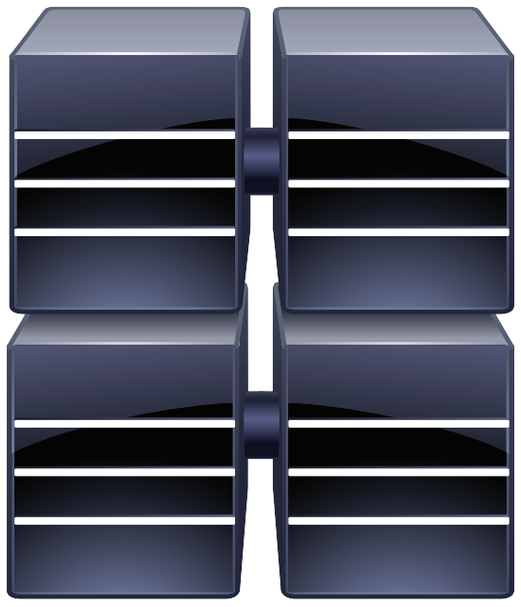
bob@gmail.com

alice@hotmail.com

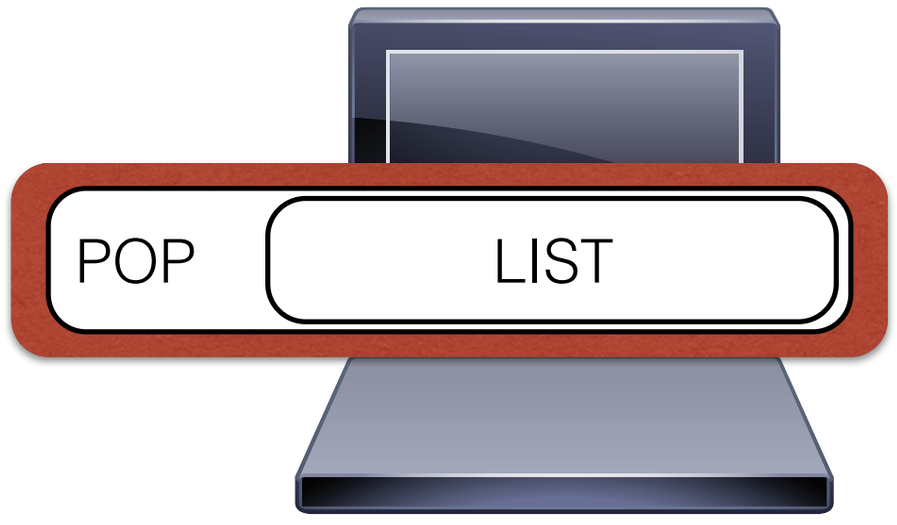
smtp.live.com

Internet

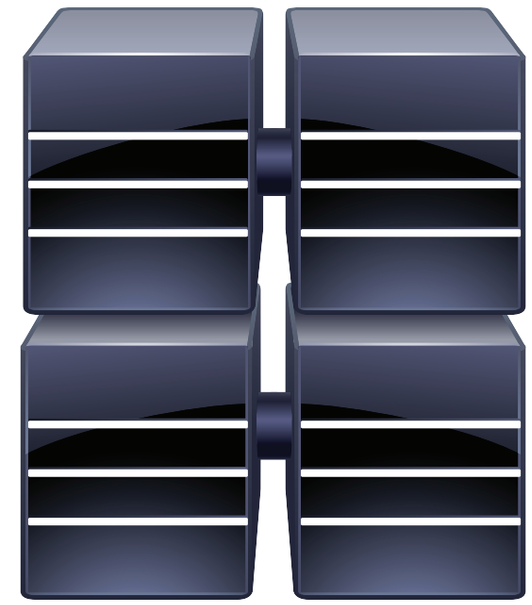
smtp.gmail.com



bob@gmail.com



alice@hotmail.com



smtp.live.com

Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

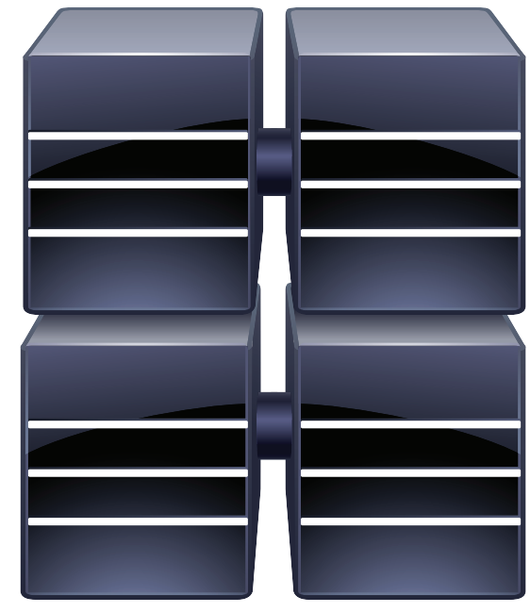
smtp.live.com

Internet

smtp.gmail.com

bob@gmail.com

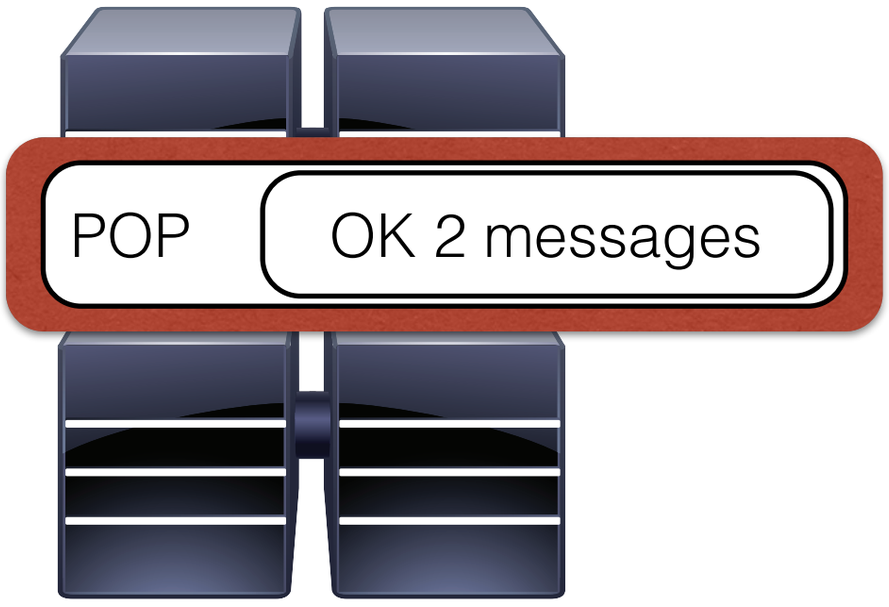
alice@hotmail.com



smtp.live.com

Internet

smtp.gmail.com



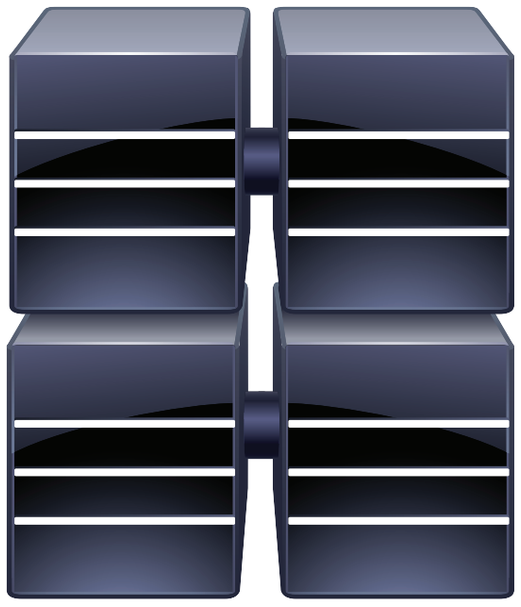
bob@gmail.com

alice@hotmail.com

smtp.live.com

Internet

smtp.gmail.com



bob@gmail.com

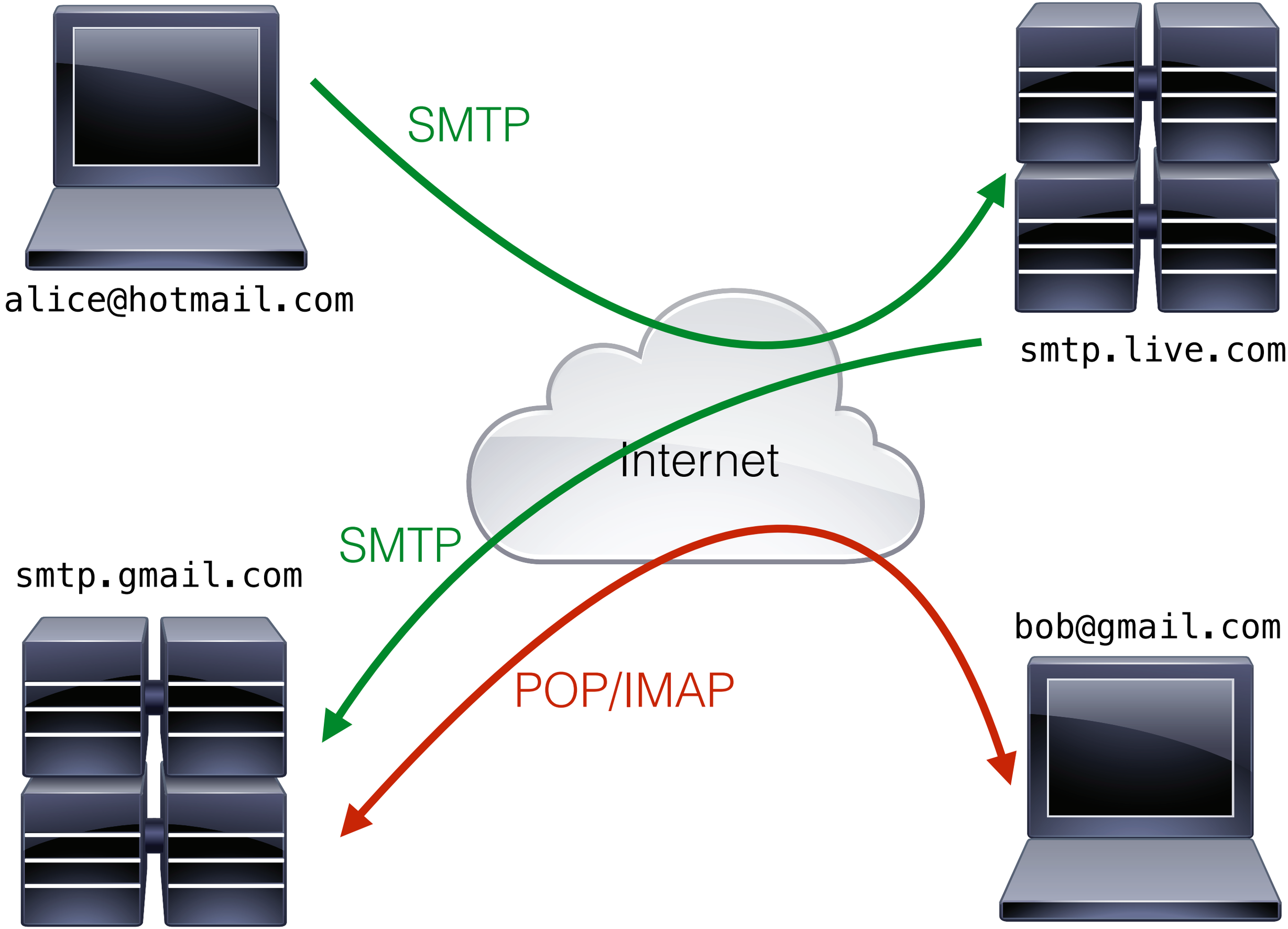
alice@hotmail.com

smtp.live.com

Internet

smtp.gmail.com

bob@gmail.com


alice@hotmail.com
SMTP
smtp.live.com
Internet
SMTP
smtp.gmail.com
POP/IMAP
bob@gmail.com

Email Protocols

- Simple Mail Transfer Protocol **(SMTP)**
  - Handles transfer of text messages between email client and mail server, and between mail servers

Email Protocols

- Simple Mail Transfer Protocol **(SMTP)**
  - Handles transfer of text messages between email client and mail server, and between mail servers
- Post Office Protocol **(POP)**
  - Messages are downloaded onto client and deleted from server

Email Protocols

- Simple Mail Transfer Protocol **(SMTP)**
  - Handles transfer of text messages between email client and mail server, and between mail servers
- Post Office Protocol **(POP)**
  - Messages are downloaded onto client and deleted from server
- Internet Message Access Protocol **(IMAP)**
  - Messages remain on server
  - Multiple clients can be connected simultaneously to same mailbox

Example SMTP Session

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
```

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
```

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
250 MyMailServer Hello laptop [192.168.1.5]
```

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
250 MyMailServer Hello laptop [192.168.1.5]
MAIL FROM:<alice@mymail.com>
```

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
250 MyMailServer Hello laptop [192.168.1.5]
MAIL FROM:<alice@mymail.com>
250 OK
```

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
250 MyMailServer Hello laptop [192.168.1.5]
MAIL FROM:<alice@mymail.com>
250 OK
RCPT TO:<guido.tack@monash.edu>
```

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
250 MyMailServer Hello laptop [192.168.1.5]
MAIL FROM:<alice@mymail.com>
250 OK
RCPT TO:<guido.tack@monash.edu>
250 Accepted
```

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
250 MyMailServer Hello laptop [192.168.1.5]
MAIL FROM:<alice@mymail.com>
250 OK
RCPT TO:<guido.tack@monash.edu>
250 Accepted
DATA
```

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
250 MyMailServer Hello laptop [192.168.1.5]
MAIL FROM:<alice@mymail.com>
250 OK
RCPT TO:<guido.tack@monash.edu>
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
```

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
250 MyMailServer Hello laptop [192.168.1.5]
MAIL FROM:<alice@mymail.com>
250 OK
RCPT TO:<guido.tack@monash.edu>
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
From: "Alice" <alice@mymail.com>
To: "Guido Tack" <guido.tack@monash.edu>
Date: Mon, 09 Mar 2015 19:24:00 +1000
Subject: test message

Hi Guido!
This is just a test.

Cheers,
Alice

.
```

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
250 MyMailServer Hello laptop [192.168.1.5]
MAIL FROM:<alice@mymail.com>
250 OK
RCPT TO:<guido.tack@monash.edu>
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
From: "Alice" <alice@mymail.com>
To: "Guido Tack" <guido.tack@monash.edu>
Date: Mon, 09 Mar 2015 19:24:00 +1000
Subject: test message

Hi Guido!
This is just a test.

Cheers,
Alice

.
250 OK id=1YUBBf-0003Ch-M1
```

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
250 MyMailServer Hello laptop [192.168.1.5]
MAIL FROM:<alice@mymail.com>
250 OK
RCPT TO:<guido.tack@monash.edu>
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
From: "Alice" <alice@mymail.com>
To: "Guido Tack" <guido.tack@monash.edu>
Date: Mon, 09 Mar 2015 19:24:00 +1000
Subject: test message

Hi Guido!
This is just a test.

Cheers,
Alice

.
250 OK id=1YUBBf-0003Ch-M1
QUIT
```

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
250 MyMailServer Hello laptop [192.168.1.5]
MAIL FROM:<alice@mymail.com>
250 OK
RCPT TO:<guido.tack@monash.edu>
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
From: "Alice" <alice@mymail.com>
To: "Guido Tack" <guido.tack@monash.edu>
Date: Mon, 09 Mar 2015 19:24:00 +1000
Subject: test message

Hi Guido!
This is just a test.

Cheers,
Alice

.
250 OK id=1YUBBf-0003Ch-M1
QUIT
221 MyMailServer closing connection
```

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
250 MyMailServer Hello laptop [192.168.1.5]
MAIL FROM:<alice@mymail.com>
250 OK
RCPT TO:<guido.tack@monash.edu>
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
From: "Alice" <alice@mymail.com>
To: "Guido Tack" <guido.tack@monash.edu>
Date: Mon, 09 Mar 2015 19:24:00 +1000
Subject: test message

Hi Guido!
This is just a test.

Cheers,
Alice


.
250 OK id=1YUBBf-0003Ch-M1
QUIT
221 MyMailServer closing connection
```

Header

Example SMTP Session

```
220 MyMailServer ESMTP Exim 4.82 Ubuntu Sat, 07 Mar 2015 20:37:24 +1100
HELO my.laptop
250 MyMailServer Hello laptop [192.168.1.5]
MAIL FROM:<alice@mymail.com>
250 OK
RCPT TO:<guido.tack@monash.edu>
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
From: "Alice" <alice@mymail.com>
To: "Guido Tack" <guido.tack@monash.edu>
Date: Mon, 09 Mar 2015 19:24:00 +1000
Subject: test message

Hi Guido!
This is just a test.

Cheers,
Alice

.
250 OK id=1YUBBf-0003Ch-M1
QUIT
221 MyMailServer closing connection
```

Header

Body

MIME

- **M**ulti-Purpose **I**nternet **M**ail **E**xtensions
- Remember: SMTP handles **plain text** email

MIME

- **M**ulti-Purpose **I**nternet **M**ail **E**xtensions
- Remember: SMTP handles **plain text** email
- How do you attach other file types?
- MIME specifies an **encoding**
- Supports **character sets** (e.g. Unicode) to send emails with non-latin characters
- Supports **non-text attachments**
- Supports **multi-part** message bodies

MIME Example

```
--Apple-Mail=_DC544A01-B885-421C-B475-2DBBCF2DEE37
Content-Transfer-Encoding: base64
Content-Disposition: inline;
   filename=guido3.jpg
Content-Type: image/jpeg;
   name="guido3.jpg"
Content-Id: <B073584A-EAF2-4A30-9ACC-1368C9C2E846@iiNet>

/9j/4AAQSkZJRgABAQEASABIAAD/4gVASUNDX1BST0ZJTEUAAQEAAAUwYXBwbAIgAABtbnRyUkdC
IFhZWiAH2QACABkACwAaAAthY3NwQVBQTAAAAABhcHBsAAAAAAAAAAAAAAAAAAAAAAAA9tYAAQAA
AADTLWFwcGwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAtk
c2NtAAABCAAAAvJkZXNjAAAD/AAAAG9nWFlaAAAEbAAAABR3dHB0AAAEgAAAABRyWFlaAAAElAAA
ABRiWFlaAAAEqAAAABRyVFJDAAAEvAAAAA5jcHJ0AAAEzAAAADhjaGFkAAAFBAAAACxnVFJDAAAE
vAAAAA5iVFJDAAAEvAAAAA5tbHVjAAAAAAAAABEAAAAMZW5VUwAAACYAAAJ+ZXNFUwAAACYAAAGC
ZGFESwAAAC4AAAHqZGVERQAAACwAAAGoZmlGSQAAACgAAADcZnJGVQAAACgAAAEqaXRJVAAAACgA
AAJWbmxOTAAAACgAAAIYbmJOTwAAACYAAAEEcHRCUgAAACYAAAGCc3ZTRQAAACYAAAEEamFKUAAA
ABoAAAFSa29LUgAAABYAAAJAemhUVwAAABYAAAFsemhDTgAAABYAAAHUcnVSVQAAACIAAAKkcGxQ
TAAAACwAAALGAFkAbABlAGkAbgBlAG4AIABSAEcAQgAtAHAAcgBvAGYAaQBpAGwAaQBHAGUAbgBl
AHIAaQBzAGsAIABSAEcAQgAtAHAAcgBvAGYAaQBsAFAAcgBvAGYAaQBsACAARwDpAG4A6QByAGkA
cQB1AGUAIABSAFYAQk4AgiwAIABSAEcAQgAgMNcw7TDVMKEwpDDrkBp1KAAgAFIARwBCACCCcl9p
Y8+P8ABQAGUAcgBmAGkAbAAgAFIARwBCACAARwBlAG4A6QByAGkAYwBvAEEAbABsAGcAZQBtAGUA
aQBuAGUAcwAgAFIARwBCAC0AUAByAG8AZgBpAGxmbpAaACAAUgBHAEIAIGPPj/Blh072AEcAZQBu
AGUAcgBlAGwAIABSAEcAQgAtAGIAZQBzAGsAcgBpAHYAZQBsAHMAZQBBAGwAZwBlAG0AZQBlAG4A
IABSAEcAQgAtAHAAcgBvAGYAaQBlAGzHfLwYACAAUgBHAEIAINUEuFzTDMd8AFAAcgBvAGYAaQBs
AG8AIABSAEcAQgAgAEcAZQBuAGUAcgBpAGMAbwBHAGUAbgBlAHIAaQBjACAAUgBHAEIAIAB0AHIA
```

MIME Example

```
--Apple-Mail=_DC544A01-B885-421C-B475-2DBBCF2DEE37
Content-Transfer-Encoding: base64
Content-Disposition: inline;
   filename=guido3.jpg
Content-Type: image/jpeg;
   name="guido3.jpg"
Content-Id: <B073584A-EAF2-4A30-9ACC-1368C9C2E846@iiNet>

/9j/4AAQSkZJRgABAQEASABIAAD/4gVASUNDX1BST0ZJTEUAAQEAAAUwYXBwbAIgAABtbnRyUkdC
IFhZWiAH2QACABkACwAaAAthY3NwQVBQTAAAAABhcHBsAAAAAAAAAAAAAAAAAAAAAAAA9tYAAQAA
AADTLWFwcGwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAtk
c2NtAAABCAAAAvJkZXNjAAAD/AAAAG9nWFlaAAAEbAAAABR3dHB0AAAEgAAAABRyWFlaAAAElAAA
ABRiWFlaAAAEqAAAABRyVFJDAAAEvAAAAA5jcHJ0AAAEzAAAADhjaGFkAAAFBAAAACxnVFJDAAAE
vAAAAA5iVFJDAAAEvAAAAA5tbHVjAAAAAAAAABEAAAAMZW5VUwAAACYAAAJ+ZXNFUwAAACYAAAGC
ZGFESwAAAC4AAAHqZGVERQAAACwAAAGoZmlGSQAAACgAAADcZnJGVQAAACgAAAEqaXRJVAAAACgA
AAJWbmxOTAAAACgAAAIYbmJOTwAAACYAAAEEcHRCUgAAACYAAAGCc3ZTRQAAACYAAAEEamFKUAAA
ABoAAAFSa29LUgAAABYAAAJAemhUVwAAABYAAAFsemhDTgAAABYAAAHUcnVSVQAAACIAAAKkcGxQ
TAAAACwAAALGAFkAbABlAGkAbgBlAG4AIABSAEcAQgAtAHAAcgBvAGYAaQBpAGwAaQBHAGUAbgBl
AHIAaQBzAGsAIABSAEcAQgAtAHAAcgBvAGYAaQBsAFAAcgBvAGYAaQBsACAARwDpAG4A6QByAGkA
cQB1AGUAIABSAFYAQk4AgiwAIABSAEcAQgAgMNcw7TDVMKEwpDDrkBp1KAAgAFIARwBCACCCcl9p
Y8+P8ABQAGUAcgBmAGkAbAAgAFIARwBCACAARwBlAG4A6QByAGkAYwBvAEEAbABsAGcAZQBtAGUA
aQBuAGUAcwAgAFIARwBCAC0AUAByAG8AZgBpAGxmbpAaACAAUgBHAEIAIGPPj/Blh072AEcAZQBu
AGUAcgBlAGwAIABSAEcAQgAtAGIAZQBzAGsAcgBpAHYAZQBsAHMAZQBBAGwAZwBlAG0AZQBlAG4A
IABSAEcAQgAtAHAAcgBvAGYAaQBlAGzHfLwYACAAUgBHAEIAINUEuFzTDMd8AFAAcgBvAGYAaQBs
AG8AIABSAEcAQgAgAEcAZQBuAGUAcgBpAGMAbwBHAGUAbgBlAHIAaQBjACAAUgBHAEIAIABQAHIA
```

MIME Example

```
--Apple-Mail=_DC544A01-B885-421C-B475-2DBBCF2DEE37
Content-Transfer-Encoding: base64
Content-Disposition: inline;
   filename=guido3.jpg
Content-Type: image/jpeg;
   name="guido3.jpg"
Content-Id: <B073584A-EAF2-4A30-9ACC-1368C9C2E846@iiNet>

/9j/4AAQSkZJRgABAQEASABIAAD/4gVASUNDX1BST0ZJTEUAAQEAAAUwYXBwbAIgAABtbnRyUkdC
IFhZWiAH2QACABkACwAaAAthY3NwQVBQTAAAAABhcHBsAAAAAAAAAAAAAAAAAAAAAAAA9tYAAQAA
AADTLWFwcGwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAtk
c2NtAAABCAAAAvJkZXNjAAAD/AAAAG9nWFlaAAAEbAAAABR3dHB0AAAEgAAAABRyWFlaAAAElAAA
ABRiWFlaAAAEqAAAABRyVFJDAAAEvAAAAA5jcHJ0AAAEzAAAADhjaGFkAAAFBAAAACxnVFJDAAAE
vAAAAA5iVFJDAAAEvAAAAA5tbHVjAAAAAAAAABEAAAAMZW5VUwAAACYAAAJ+ZXNFUwAAACYAAAGC
ZGFESwAAAC4AAAHqZGVERQAAACwAAAGoZmlGSQAAACgAAADcZnJGVQAAACgAAAEqaXRJVAAAACgA
AAJWbmxOTAAAACgAAAIYbmJOTwAAACYAAAEEcHRCUgAAACYAAAGCc3ZTRQAAACYAAAEEamFKUAAA
ABoAAAFSa29LUgAAABYAAAJAemhUVwAAABYAAAFsemhDTgAAABYAAAHUcnVSVQAAACIAAAKkcGxQ
TAAAACwAAALGAFkAbABlAGkAbgBlAG4AIABSAEcAQgAtAHAAcgBvAGYAaQBpAGwAaQBHAGUAbgBl
AHIAaQBzAGsAIABSAEcAQgAtAHAAcgBvAGYAaQBsAFAAcgBvAGYAaQBsACAARwDpAG4A6QByAGkA
cQB1AGUAIABSAFYAQk4AgiwAIABSAEcAQgAgMNcw7TDVMKEwpDDrkBp1KAAgAFIARwBCACCCcl9p
Y8+P8ABQAGUAcgBmAGkAbAAgAFIARwBCACAARwBlAG4A6QByAGkAYwBvAEEAbABsAGcAZQBtAGUA
aQBuAGUAcwAgAFIARwBCAC0AUAByAG8AZgBpAGxmbpAaACAAUgBHAEIAIGPPj/Blh072AEcAZQBu
AGUAcgBlAGwAIABSAEcAQgAtAGIAZQBzAGsAcgBpAHYAZQBsAHMAZQBBAGwAZwBlAG0AZQBlAG4A
IABSAEcAQgAtAHAAcgBvAGYAaQBlAGzHfLwYACAAUgBHAEIAINUEuFzTDMd8AFAAcgBvAGYAaQBs
AG8AIABSAEcAQgAgAEcAZQBuAGUAcgBpAGMAbwBHAGUAbgBlAHIAaQBjACAAUgBHAEIAIABQAHIA
```

MIME Example


```
d3DNaSQoWlkuRgqNu/GD/ERyB3zXIL41+F/1/WtSeCTS3Gh2Iw/7tYyA2SUyueGG3hThjAHtXjH1
H4I6zc6pHcLcX13BLJ5NrDcStLI+wYzuI7/Ng9lrui4Lc5MVVr1X
jWR5G2KSNhnBz3rstT+EniS3ge4tIsSgnbbvJiQKO3pnqPwrwzU
jGWzPFxLlDVoq3d85m3xb8scAdSPz7Vxl1O5vp7YqwmwVjDN8rH
MSsmwsSuCMAcn65/nXWoWR4Ves5M6PS2S/g8gbbSTyiC+3pwCwz
GSfzFctpEloZgGIL5GGYEMoxwc9weldV9rC2rebKvlxykox5ABX
TKgIwRySeh9cL2pmgLNP400/VEZ4LLTruORnIGBtIYH9OtczPPLeXckduP8ARw6s2Ry27gEA+5rt
dJs2XSljgcJJG/DshB5OSTzz15x2qqNoO4pJyP2oh1pJ7CCdZAySRK4OeCCM/wBaG1dc/e/KvDfC
utS3Hwt8PTNKrO2nRbipyCQgB/lWydUkB5c+1fQLLFLU+SqZhySaPV21ZccuKrPq6gn5h+Bry99U
kx941XbU5Dn5jVf2UjL+1D1P+2ACPm/M1C+tLz8w/OvLTqM394/TNMN9KzYyaf8AZkewf2melPrX
J+b9agbWAW+/+tee/aZTzk/nUTXUm7rS/s2Iv7SOna7lxxnPuag+1zbjnNFFe2oR7HlupLuYfiLT
P+Eh8OPZNIsFwGD2tw0YfyJOz4PXHp3GRXx9rmnX1h4r1Kymlie6hDTCIKQpy3yOemM8jaB8pJHv
RRXg55QhyqdtT6Lh7FVHUdO+hl3d/wCRYSxt56XRZmimQfc9MD05I/Cruj38i3p0t7N5JZbdriAy
3HlxrKCp81gcDAAbAHJJHYYoor49I+1Ume8eENTl1G8McsrtdtG01uWICsYyS5IHXkAAFlGe56V9
e+ANUlv9Bmja1nRZYwzwTOUWF06ktnJ68YBB/kUVzVYJtHqYOpJ6M9z8Jafpl/o589IXfdtRRcEI
GUjOSp4yoxjHBx3r0T+ydIF1HqFrpUF2YlbdckK0jJyRkjH3TkH1wQRyDRRXZRikkFWcnNq584fE
Dxxe3rnQrZ7eOK0k82yRVxHcAFs5PdSeCBg1wEutxP4cigS1LyPK8t0YzvmTIARQRzjOPy69aKK5
6mrPSilFKx4n4tv1MF42yzaZfmkdZN2AfQjqcYOO34V8OfEy0sdevUa2dTdFCFuY48ebgZ5xgEcY
z60UV04OKU0cmZybhY+fJbGUJJ5u9Fj4dkXcAe1UZQbe2U5eMOpPzH7yjAwD+NFFevKTPjZxRZtx
9mt5U2MiMoYORkKM5x+mM1qLcTXUu6SLZAikiMD7xIP5njrRRXHUd9Tektjo9NiggaOS6j8uQyAh
ZPYdOPcH2rudLWGSBiIUO6Tcyrzjt/jRRXPe502Ptj4bTfavgxorDpErxY9NrkD9K70RZGRRRX32
GbdOPofnGOgliJ+pG0RGc5NQsuBmiiuh7nIoorsxMnTP4VIOQOD+VFFZ3K5UKWx2qq/3+p/Ciisq
kncFE//Z
--Apple-Mail=_DC544A01-B885-421C-B475-2DBBCF2DEE37--
```


Represent image data as plain text!

Two-tier vs Three-tier mail

- **Two-tier:**
  - Client-server architecture
  - Client implements application logic, talks to server using SMTP and POP/IMAP

Two-tier vs Three-tier mail

- **Two-tier:**
  - Client-server architecture
  - Client implements application logic, talks to server using SMTP and POP/IMAP
- **Three-tier:**
  - Thin client accesses **web application**
  - Server handles application logic
  - Client accesses server using HTTP

alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

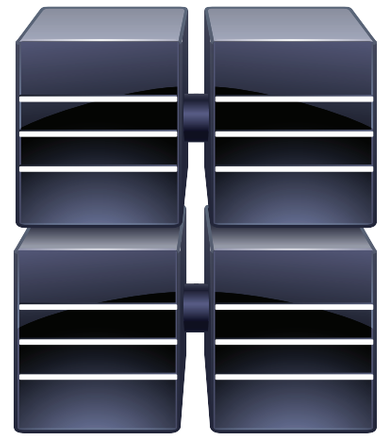
smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

Internet

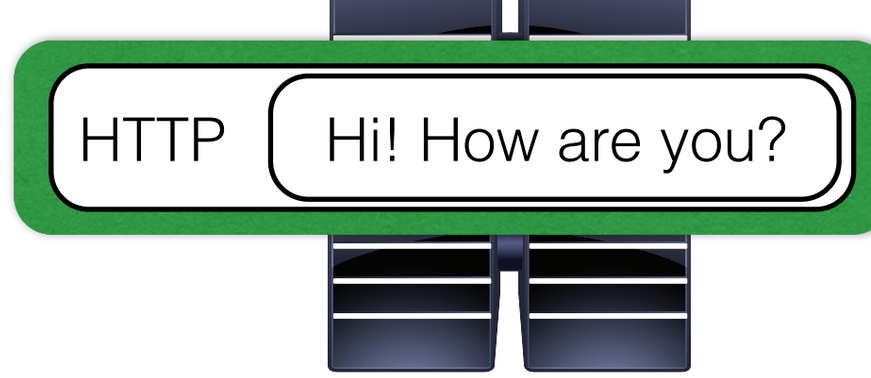


smtp.hotmail.com

smtp.gmail.com

bob@gmail.com

alice@hotmail.com



www.hotmail.com

Internet

smtp.hotmail.com

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

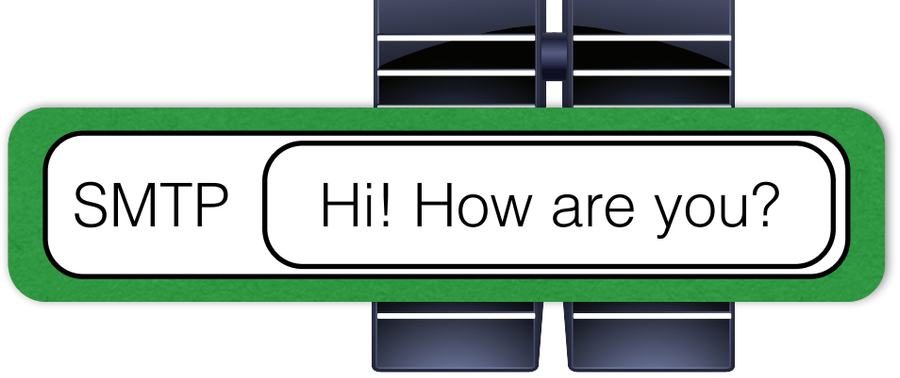
smtp.hotmail.com

Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com



www.hotmail.com

Internet

smtp.hotmail.com

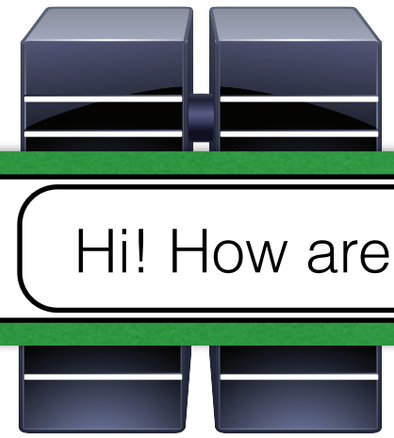
smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

Internet



SMTP

Hi! How are you?

smtp.hotmail.com

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

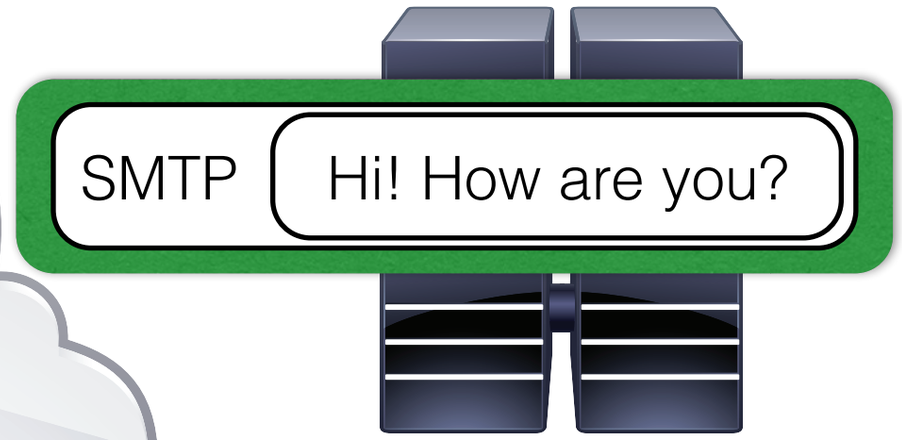
Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com



smtp.hotmail.com

Internet

smtp.gmail.com

bob@gmail.com

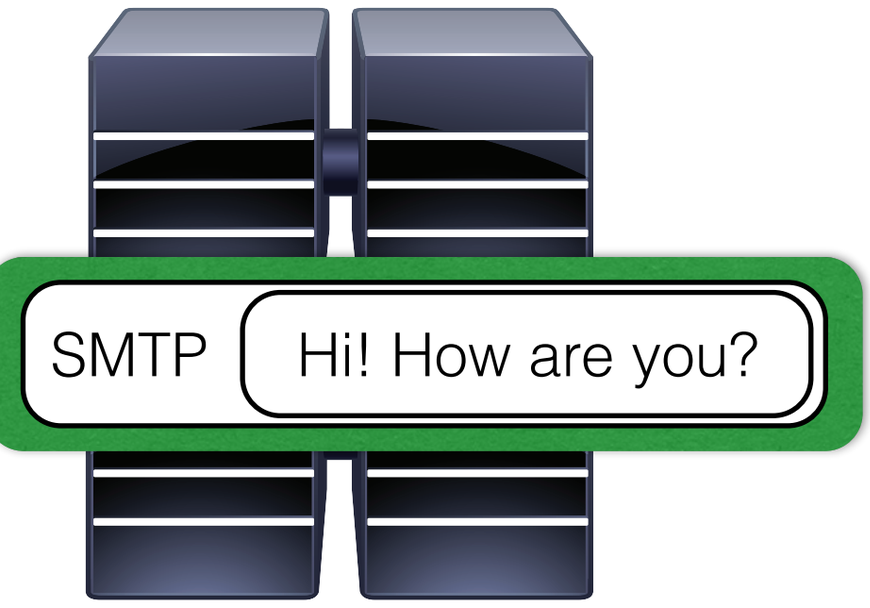
alice@hotmail.com

www.hotmail.com

Internet

smtp.hotmail.com

smtp.gmail.com



bob@gmail.com

alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

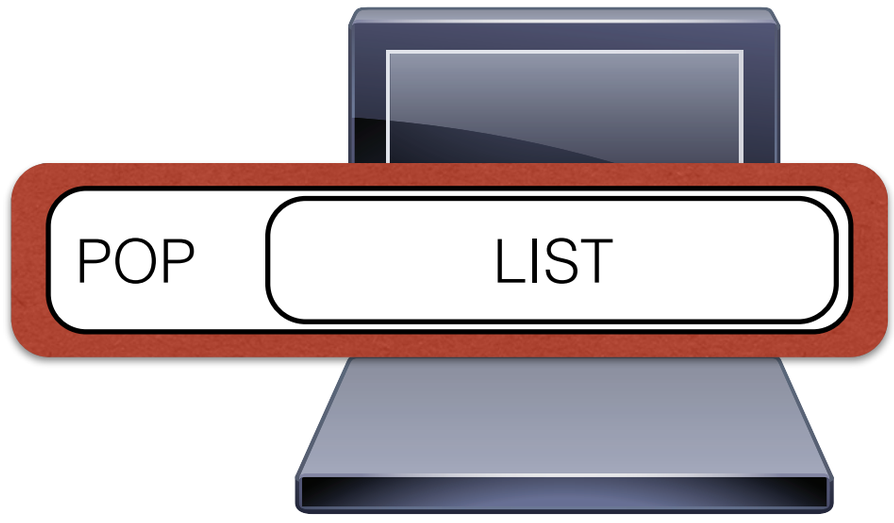
www.hotmail.com

smtp.hotmail.com

Internet

smtp.gmail.com

bob@gmail.com



alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

smtp.gmail.com

bob@gmail.com

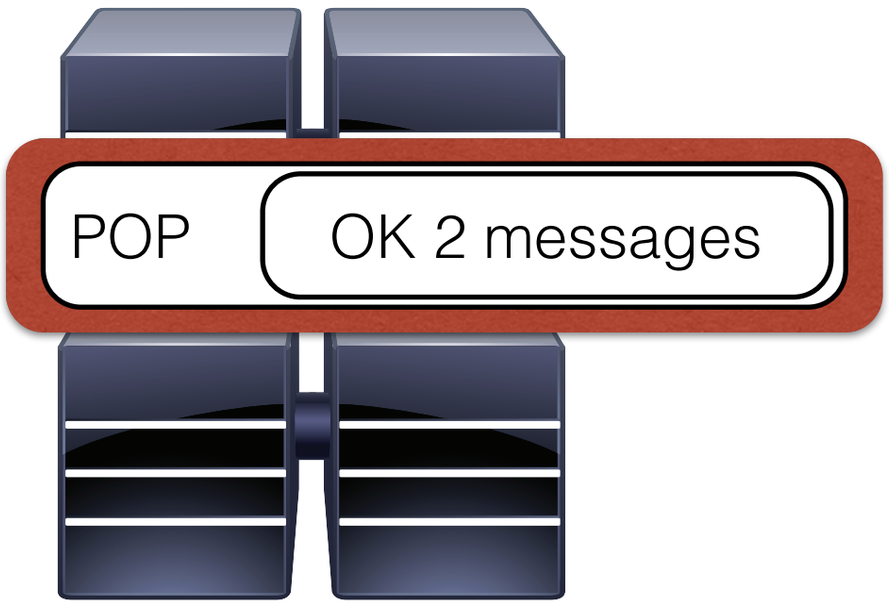
alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

smtp.gmail.com



bob@gmail.com

alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

Internet

smtp.hotmail.com

smtp.gmail.com

bob@gmail.com

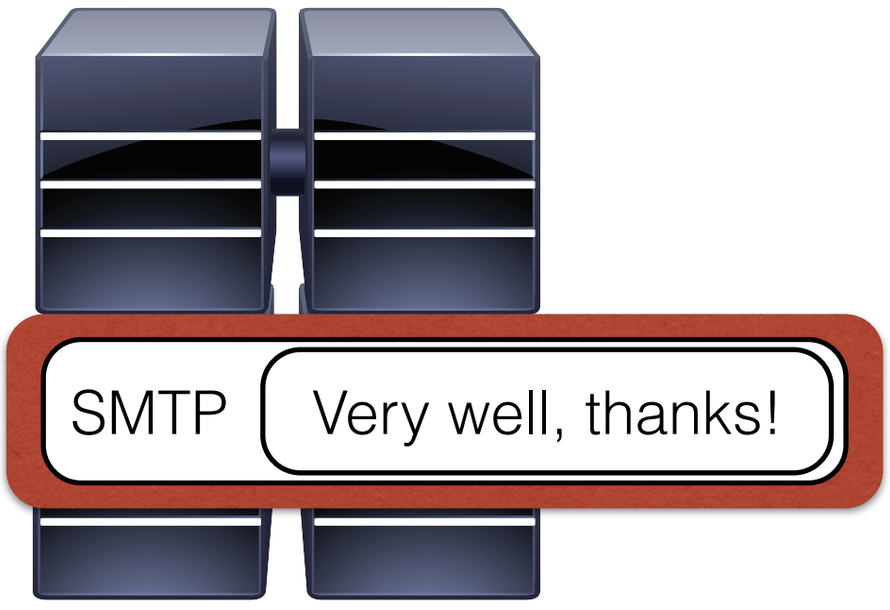
alice@hotmail.com

www.hotmail.com

Internet

smtp.hotmail.com

smtp.gmail.com



bob@gmail.com

alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

smtp.gmail.com

bob@gmail.com

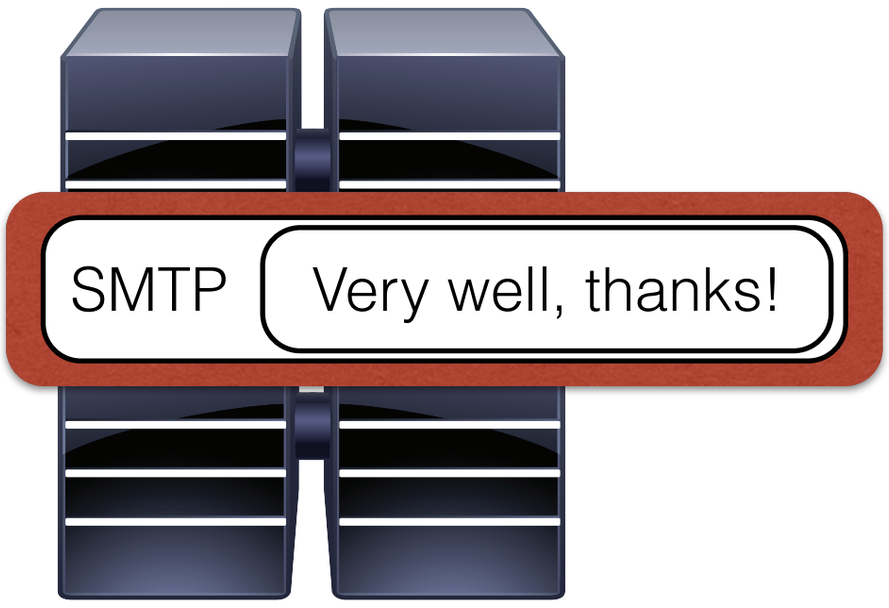
alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

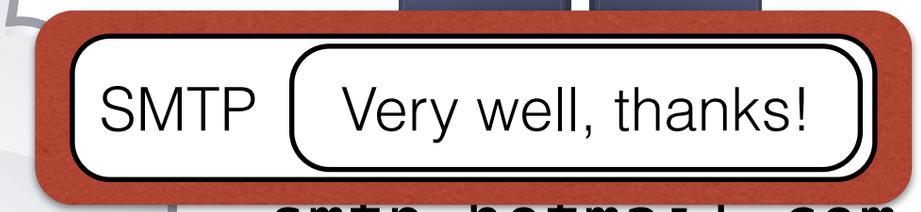
smtp.gmail.com



bob@gmail.com

alice@hotmail.com

www.hotmail.com



smtp.hotmail.com

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

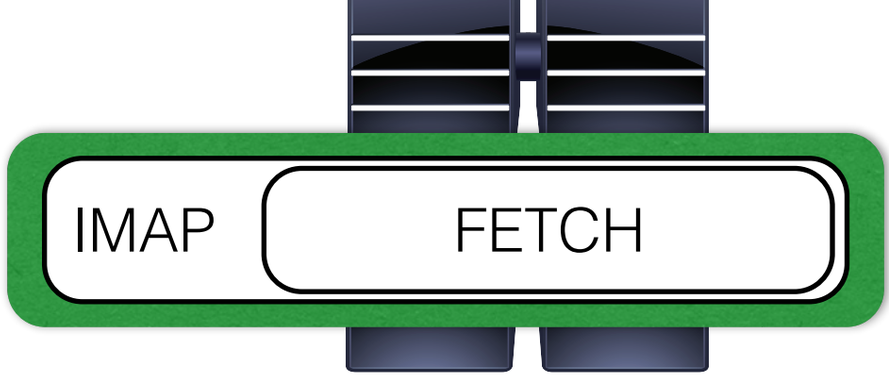
smtp.hotmail.com

Internet

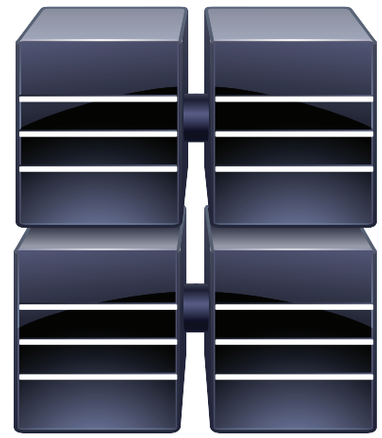
smtp.gmail.com

bob@gmail.com

alice@hotmail.com



www.hotmail.com



smtp.hotmail.com

Internet

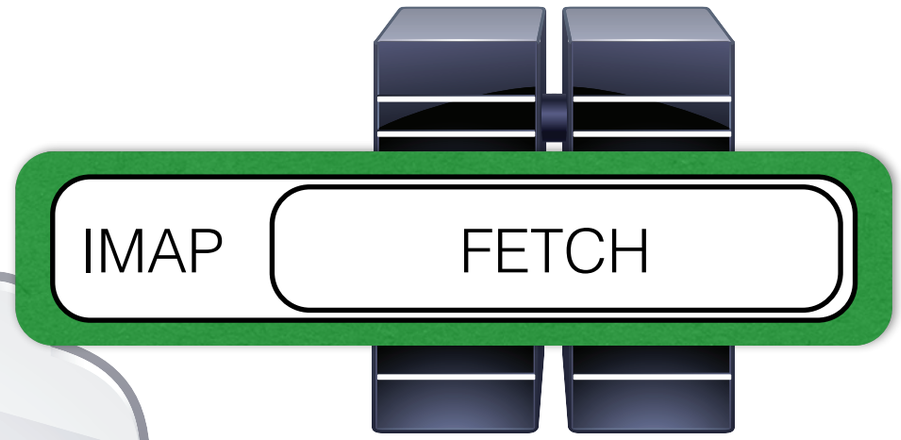
smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

Internet



smtp.hotmail.com

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

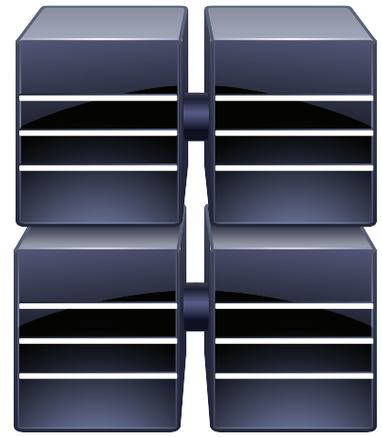
smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

Internet



smtp.hotmail.com

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

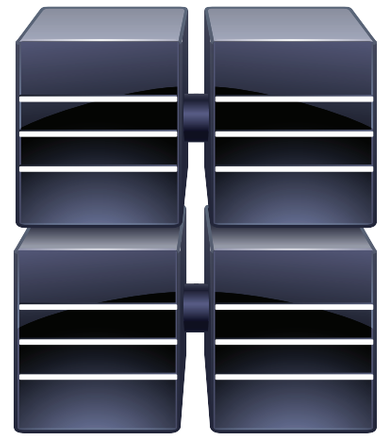
smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

Internet



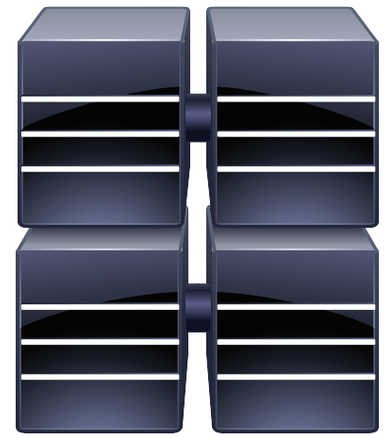
smtp.hotmail.com

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com



smtp.hotmail.com

Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

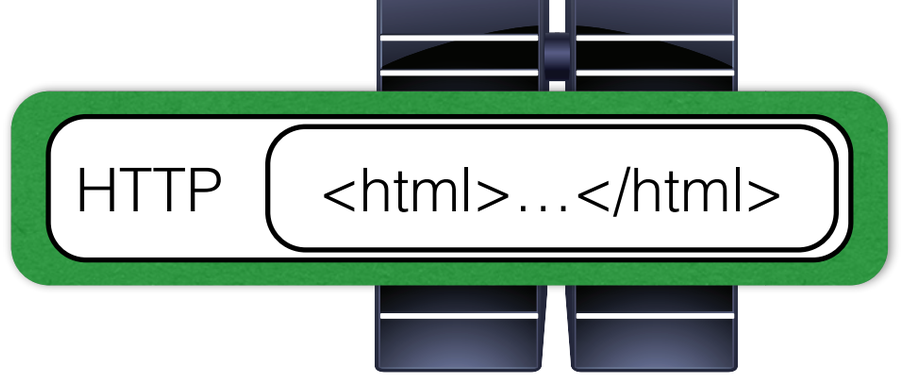
smtp.hotmail.com

Internet

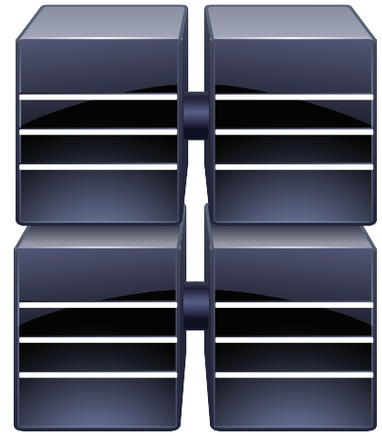
smtp.gmail.com

bob@gmail.com

alice@hotmail.com



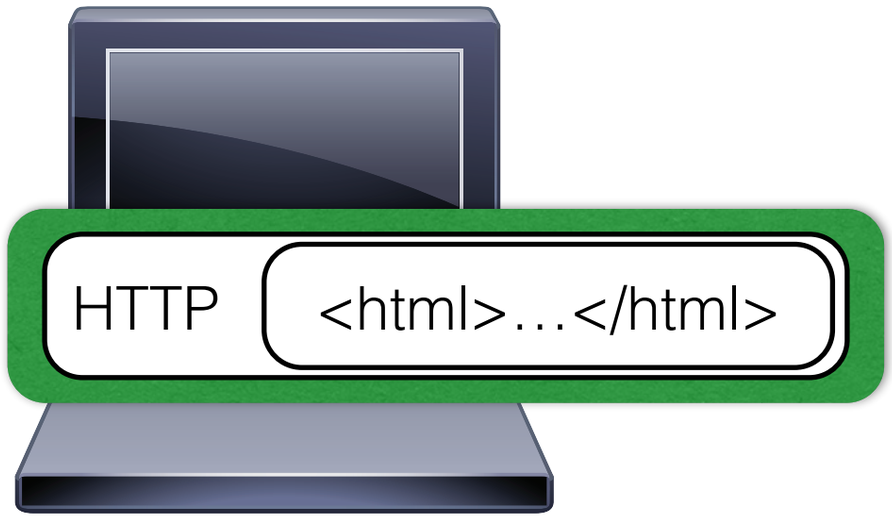
www.hotmail.com



smtp.hotmail.com

smtp.gmail.com

bob@gmail.com



alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

www.hotmail.com

smtp.hotmail.com

Internet

smtp.gmail.com

bob@gmail.com

alice@hotmail.com

HTTP

www.hotmail.com

SMTP/
POP/
IMAP

Internet

smtp.hotmail.com

smtp.gmail.com

SMTP

SMTP/POP/IMAP

bob@gmail.com

Summary

Summary

- **Application architecture** determines how client and server share the work load (server-based, client-based, client-server, thin-client-server, peer-to-peer)

Summary

- **Application architecture** determines how client and server share the work load (server-based, client-based, client-server, thin-client-server, peer-to-peer)
- **WWW**: based on HTML hypertext and URLs, communicates using HTTP

Summary

- **Application architecture** determines how client and server share the work load (server-based, client-based, client-server, thin-client-server, peer-to-peer)
- **WWW**: based on HTML hypertext and URLs, communicates using HTTP
- **Email**: SMTP, POP, IMAP, thin-client web mail

Summary

- **Application architecture** determines how client and server share the work load (server-based, client-based, client-server, thin-client-server, peer-to-peer)
- **WWW**: based on HTML hypertext and URLs, communicates using HTTP
- **Email**: SMTP, POP, IMAP, thin-client web mail
- Important: **standard protocols enable interoperability**