The Transport Layer

Antonio Carzaniga

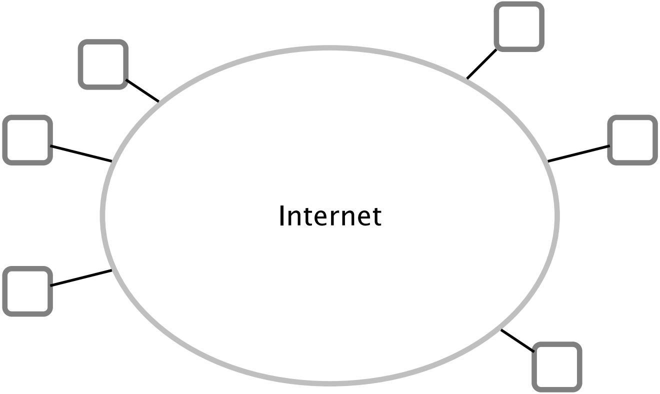
Faculty of Informatics
University of Lugano

October 24, 2014

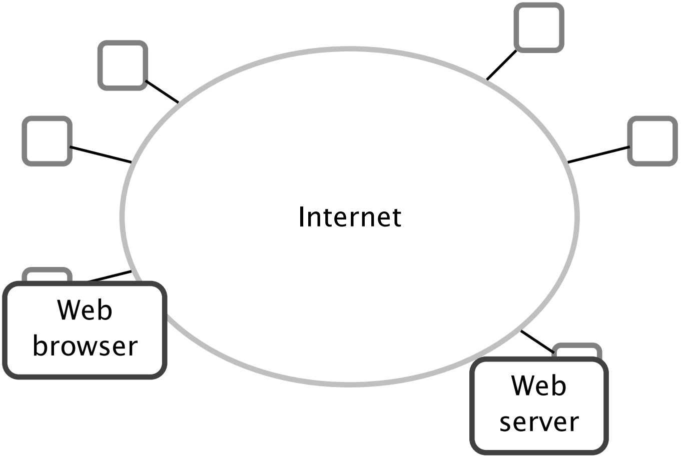
Outline

- Basic concepts in transport-layer protocols
- Multiplexing/demultiplexing
- UDP message format
- Reliable transfer

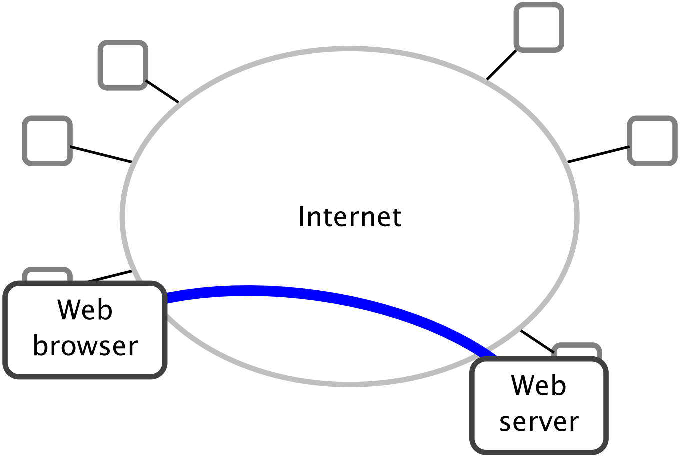
Transport Layer



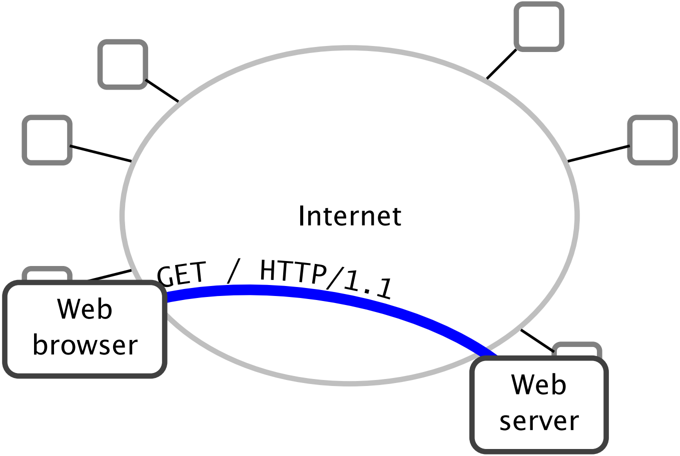
Transport Layer



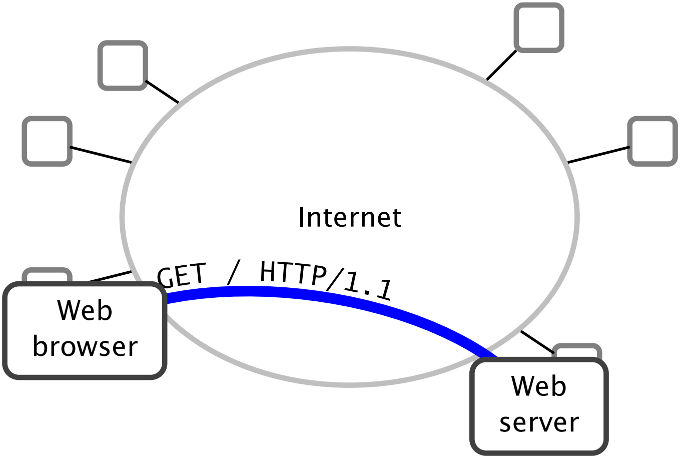
Transport Layer



Transport Layer



Transport Layer



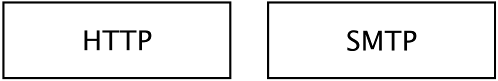
Primitive communication between applications

Type of Service

Type of Service

HTTP

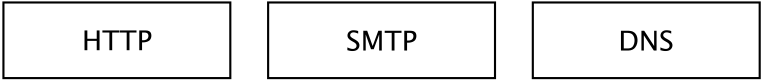
Type of Service



HTTP

SMTP

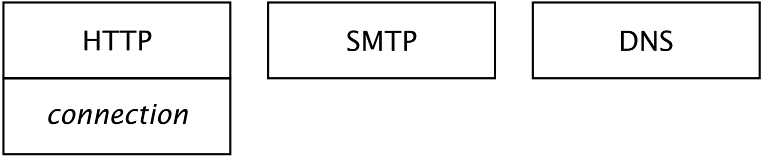
Type of Service


HTTP

SMTP

DNS

Type of Service



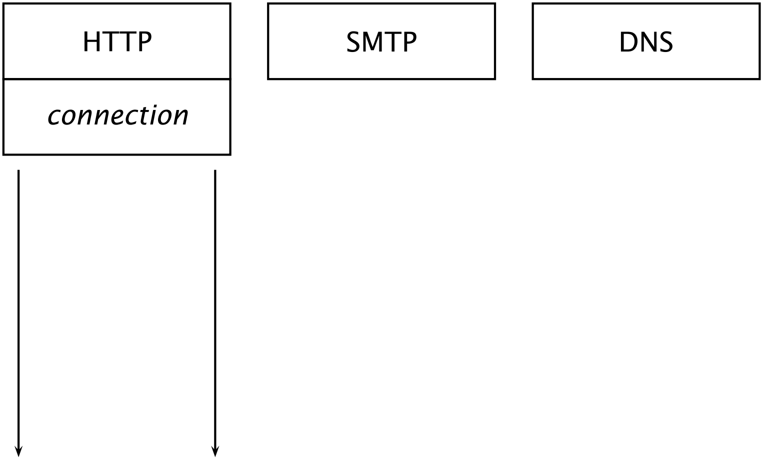
HTTP

SMTP

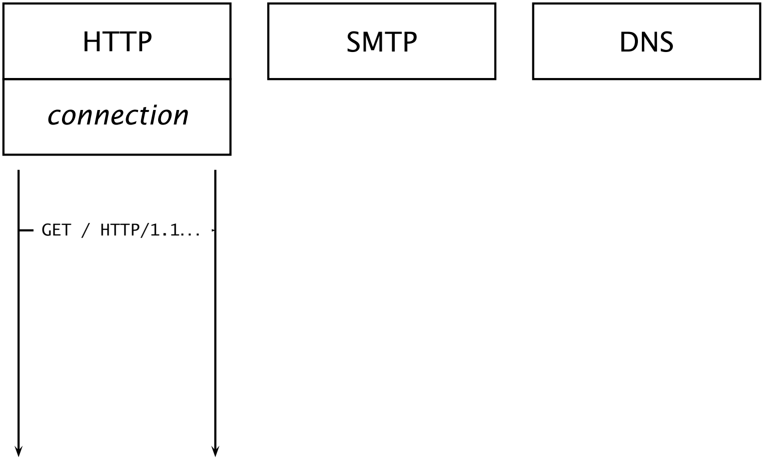
DNS

connection

Type of Service



Type of Service



HTTP

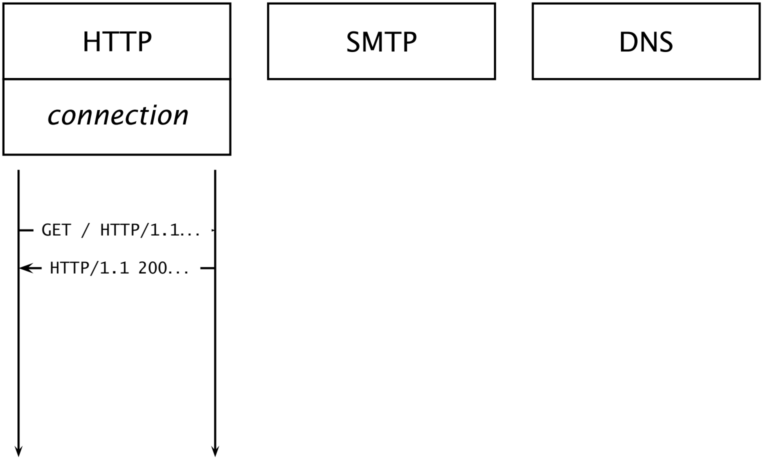
SMTP

DNS

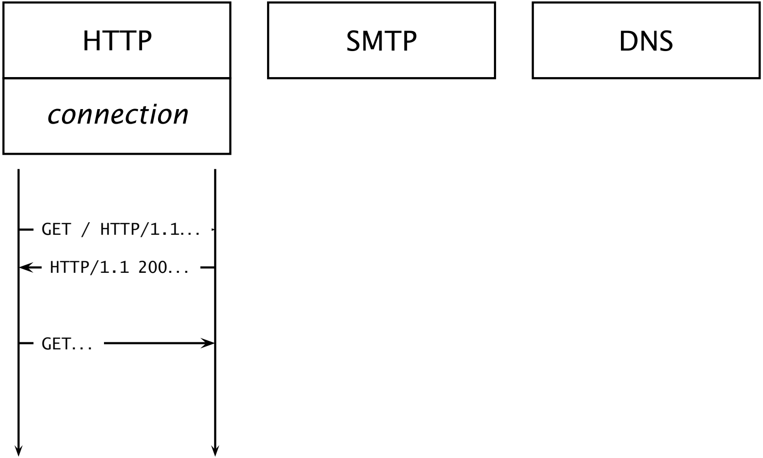
connection

```
GET / HTTP/1.1...
```

Type of Service



Type of Service



HTTP

connection

SMTP

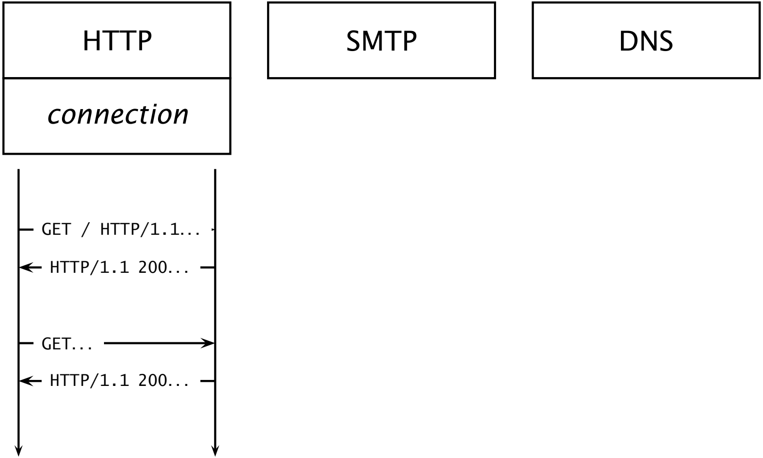
DNS

GET / HTTP/1.1...

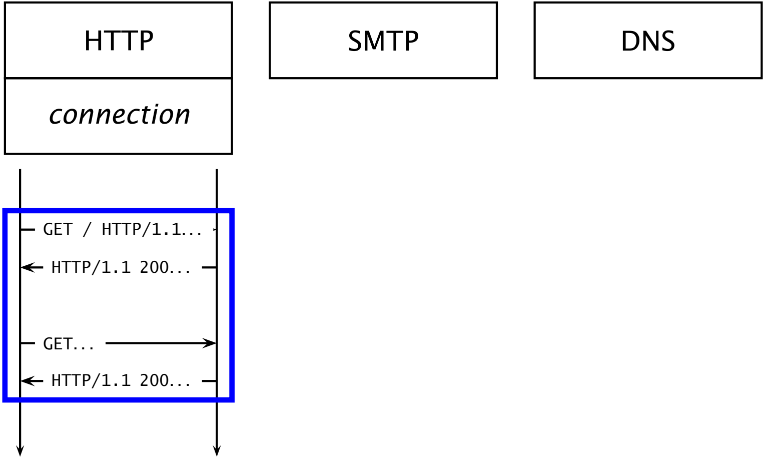
HTTP/1.1 200...

GET...

Type of Service



Type of Service



HTTP

SMTP

DNS

connection

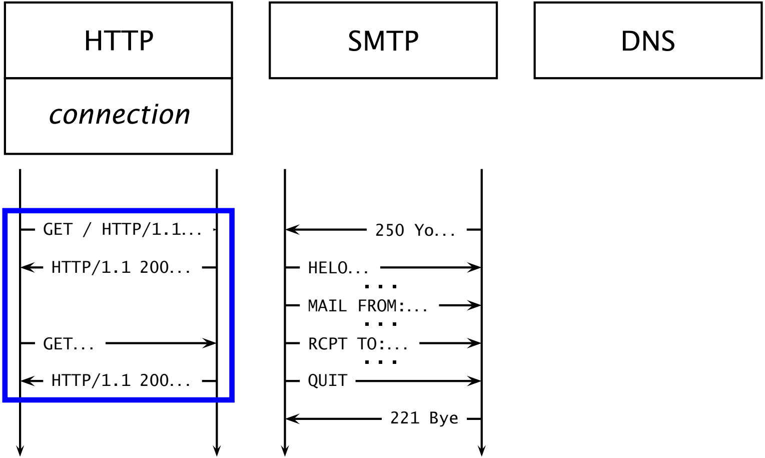
GET / HTTP/1.1...

HTTP/1.1 200...

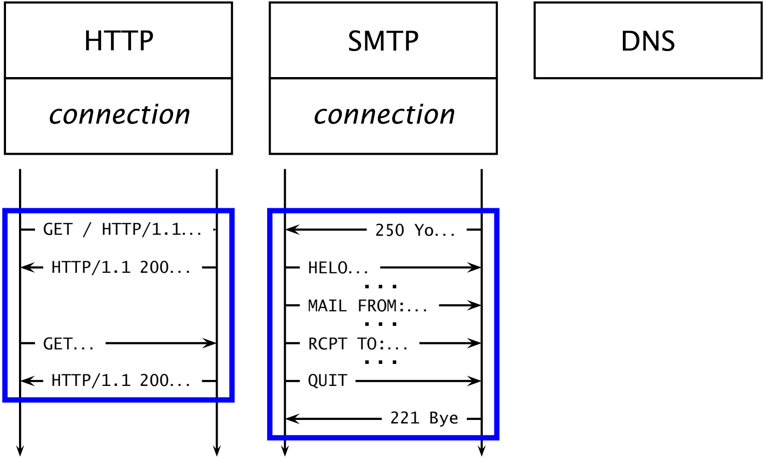
GET...

HTTP/1.1 200...

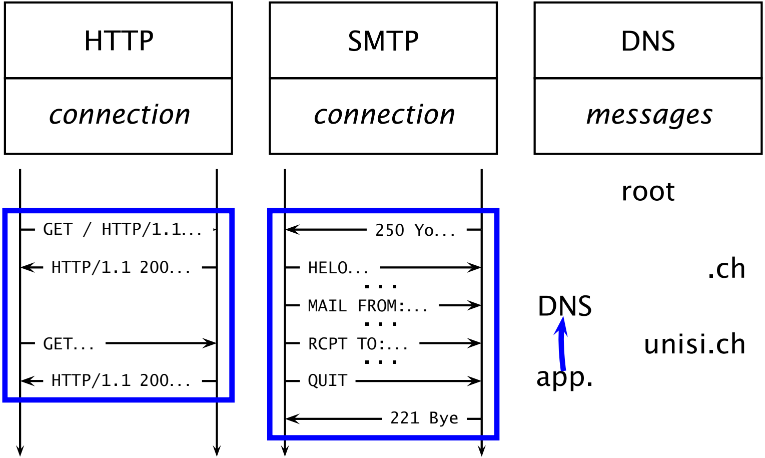
Type of Service



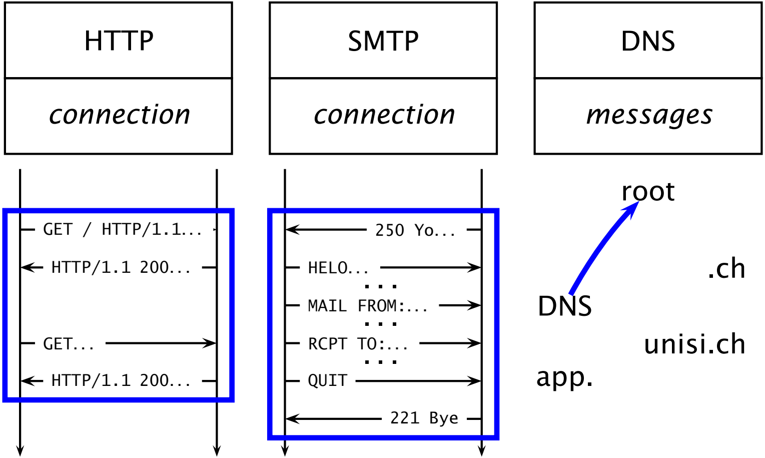
Type of Service



Type of Service



Type of Service



HTTP

connection

GET / HTTP/1.1...

HTTP/1.1 200...

GET...

HTTP/1.1 200...

SMTP

connection

250 Yo...

HELO...

...

MAIL FROM:...

...

RCPT TO:...

...

QUIT

221 Bye

DNS

messages

root

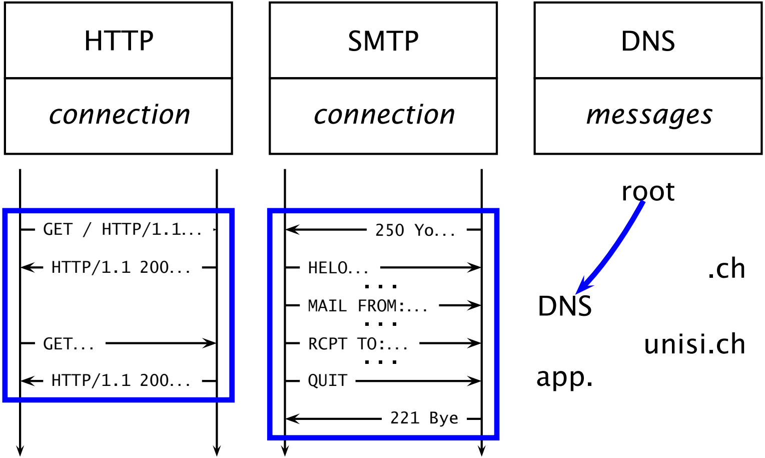
.ch

DNS

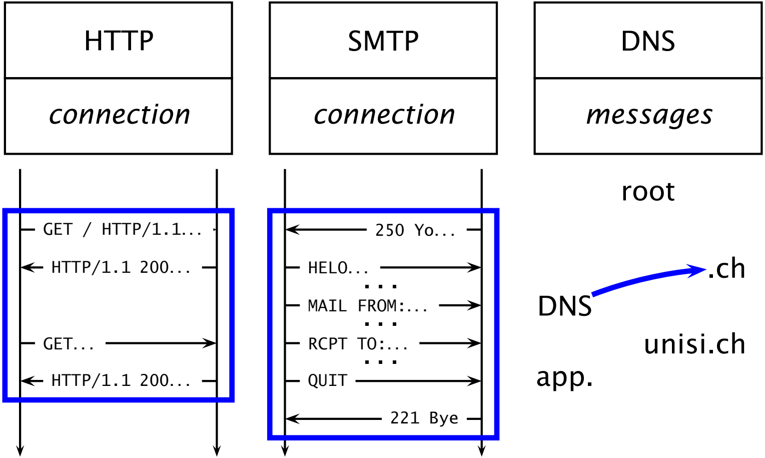
unisi.ch

app.

Type of Service



Type of Service



HTTP

connection

GET / HTTP/1.1...

HTTP/1.1 200...

GET...

HTTP/1.1 200...

SMTP

connection

250 Yo...

HELO...

...

MAIL FROM:...

...

RCPT TO:...

...

QUIT

221 Bye

DNS

messages

root

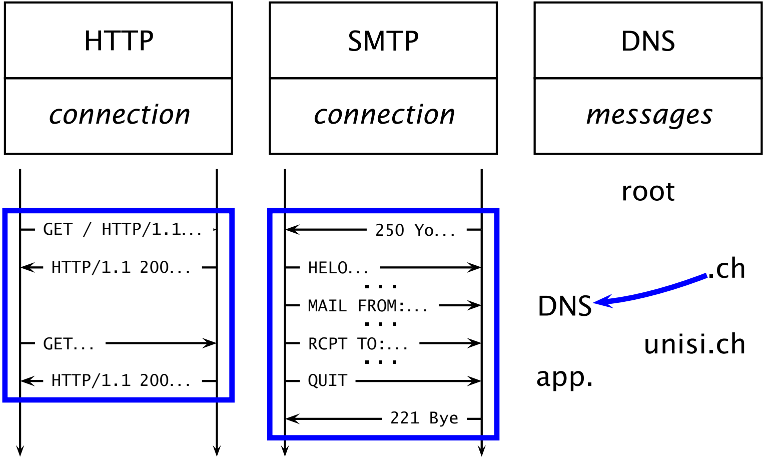
.ch

DNS

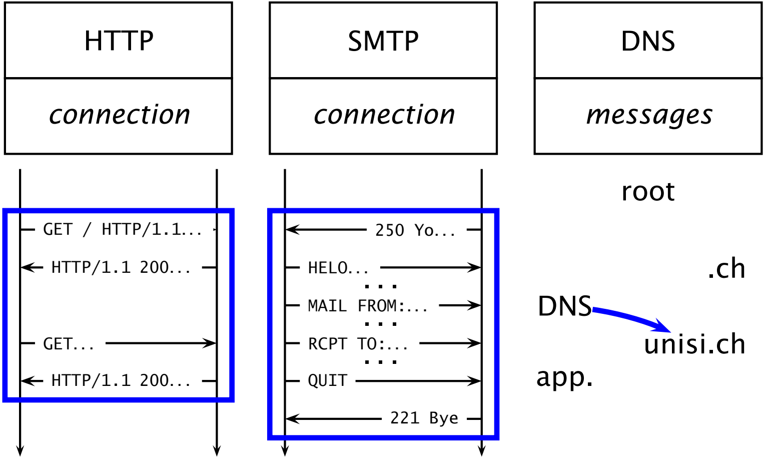
unisi.ch

app.

Type of Service



Type of Service



HTTP

connection

SMTP

connection

DNS

messages

GET / HTTP/1.1...

HTTP/1.1 200...

GET...

HTTP/1.1 200...

250 Yo...

HELO...

...

MAIL FROM:...

...

RCPT TO:...

...

QUIT

221 Bye

root

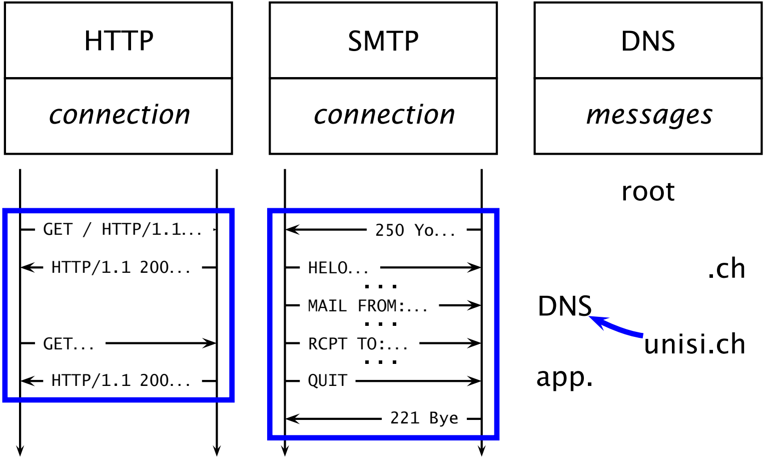
.ch

DNS

unisi.ch

app.

Type of Service



HTTP

connection

SMTP

connection

DNS

messages

GET / HTTP/1.1...

HTTP/1.1 200...

GET...

HTTP/1.1 200...

250 Yo...

HELO...

...

MAIL FROM:...

...

RCPT TO:...

...

QUIT

221 Bye

root

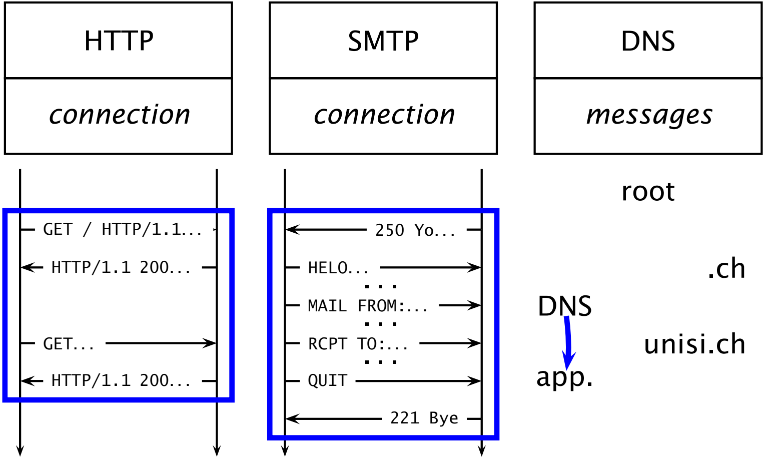
.ch

DNS

unisi.ch

app.

Type of Service



HTTP

connection

SMTP

connection

DNS

messages

```
GET / HTTP/1.1...
HTTP/1.1 200...
GET...
HTTP/1.1 200...
```

```
250 Yo...
HELO...
...
MAIL FROM:...
...
RCPT TO:...
...
QUIT
221 Bye
```

root

.ch

DNS

unisi.ch

app.

Transport Layer in the Internet

Transport Layer in the Internet

- *Transport Control Protocol (TCP)*
  - conntection-oriented (i.e., “connections”)

Transport Layer in the Internet

- *Transport Control Protocol (TCP)*
  - conntection-oriented (i.e., “connections”)

- *User Datagram Protocol (UDP)*
  - connectionless (i.e., “messages”)

Transport Layer in the Internet

- *Transport Control Protocol (TCP)*
  - conntection-oriented (i.e., “connections”)
- *User Datagram Protocol (UDP)*
  - connectionless (i.e., “messages”)
- Terminology
  - transport-layer packets are called *segments*

Transport Layer in the Internet

- *Transport Control Protocol (TCP)*
  - conntection-oriented (i.e., “connections”)
- *User Datagram Protocol (UDP)*
  - connectionless (i.e., “messages”)
- Terminology
  - transport-layer packets are called *segments*
- Basic assumptions on the underlying network layer

Transport Layer in the Internet

- *Transport Control Protocol (TCP)*
  - conntection-oriented (i.e., “connections”)
- *User Datagram Protocol (UDP)*
  - connectionless (i.e., “messages”)
- Terminology
  - transport-layer packets are called *segments*
- Basic assumptions on the underlying network layer
  - every host has one unique *IP address*

Transport Layer in the Internet

- *Transport Control Protocol (TCP)*
  - conntection-oriented (i.e., “connections”)

- *User Datagram Protocol (UDP)*
  - connectionless (i.e., “messages”)

- Terminology
  - transport-layer packets are called *segments*

- Basic assumptions on the underlying network layer
  - every host has one unique *IP address*
  - best-effort delivery service
    - no guarantees on the integrity of segments
    - no guarantees on the order in which segments are delivered

Transport-Layer Value-Added Service

Transport-Layer Value-Added Service

- *Transport-layer multiplexing/demultiplexing*
  - i.e., connecting applications as opposed to hosts

Transport-Layer Value-Added Service

- *Transport-layer multiplexing/demultiplexing*
  - i.e., connecting applications as opposed to hosts

- *Reliable data transfer*
  - i.e., integrity and possibly ordered delivery

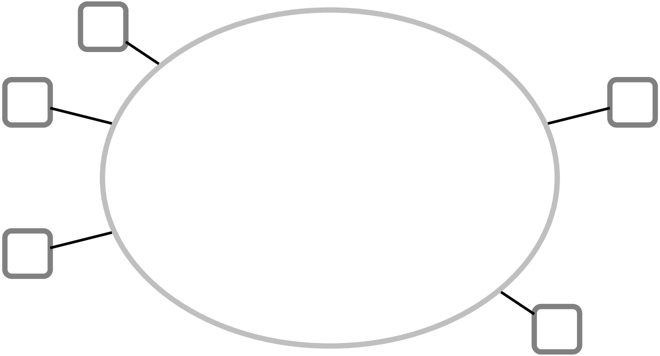
Transport-Layer Value-Added Service

- *Transport-layer multiplexing/demultiplexing*
    - i.e., connecting applications as opposed to hosts
- *Reliable data transfer*
    - i.e., integrity and possibly ordered delivery
- *Connections*
    - i.e., streams
    - can be seen as the same as ordered delivery

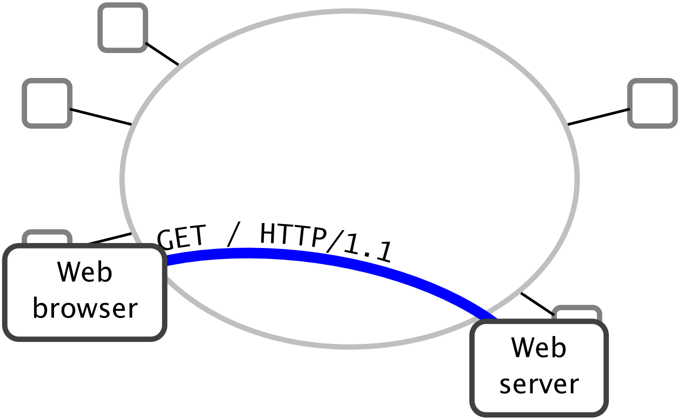
Transport-Layer Value-Added Service

- *Transport-layer multiplexing/demultiplexing*
  - i.e., connecting applications as opposed to hosts
- *Reliable data transfer*
  - i.e., integrity and possibly ordered delivery
- *Connections*
  - i.e., streams
  - can be seen as the same as ordered delivery
- *Congestion control*
  - i.e., end-to-end traffic (admission) control so as to avoid destructive congestions within the network

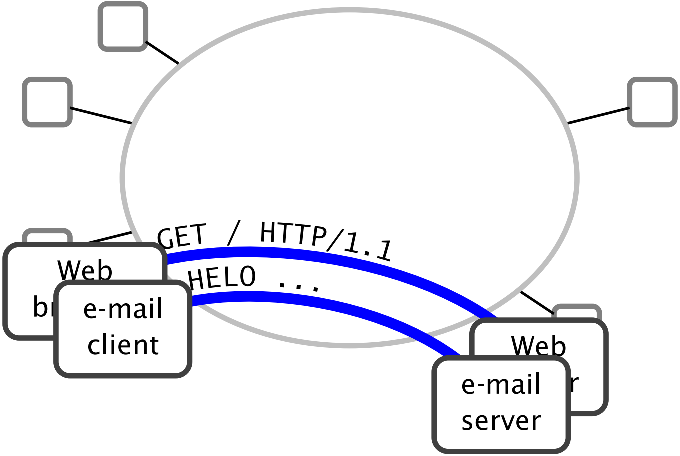
Multiplexing/Demultiplexing



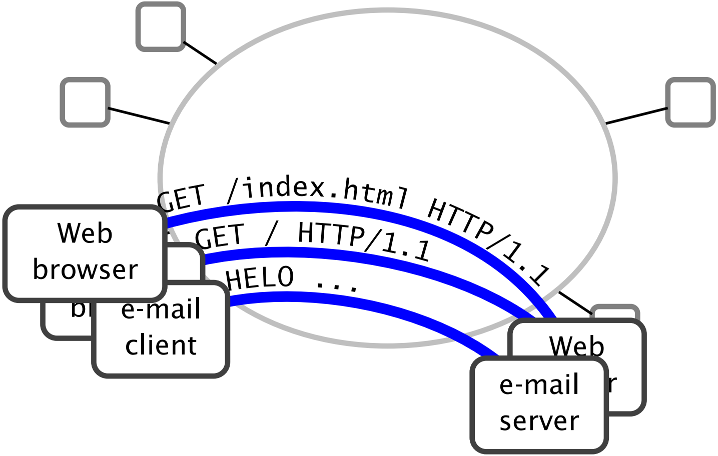
Multiplexing/Demultiplexing



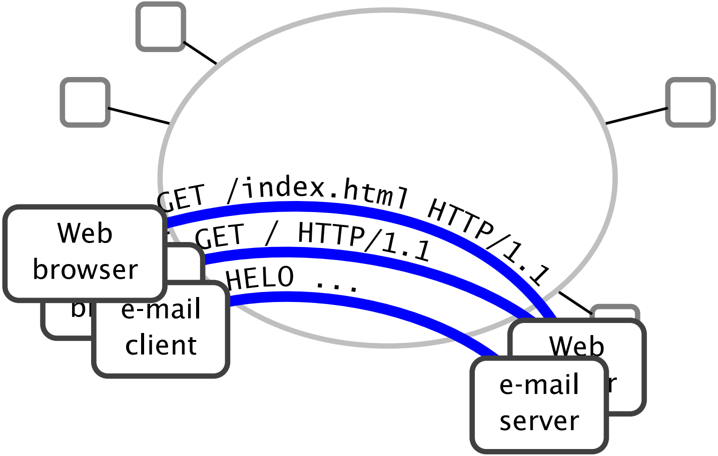
Multiplexing/Demultiplexing



Multiplexing/Demultiplexing



Multiplexing/Demultiplexing



How do we distinguish all these “connections”?

Ports

Ports

- Each application running on a host is identified (within that host) by a unique *port number*
  - port numbers are simply cross-platform process identifiers

Ports

- Each application running on a host is identified (within that host) by a unique *port number*
  - port numbers are simply cross-platform process identifiers
- How do we identify a “connection”?

Ports

- Each application running on a host is identified (within that host) by a unique *port number*
  - port numbers are simply cross-platform process identifiers

- How do we identify a “connection”?
  - two pairs of *host* and *application* identifiers
  - i.e., two pairs *(IP-address, port)*

Ports

- Each application running on a host is identified (within that host) by a unique *port number*
  - port numbers are simply cross-platform process identifiers
- How do we identify a “connection”?
  - two pairs of *host* and *application* identifiers
  - i.e., two pairs *(IP-address, port)*
- How do we find out which application (host and port number) to connect to?

Ports

- Each application running on a host is identified (within that host) by a unique *port number*
  - port numbers are simply cross-platform process identifiers

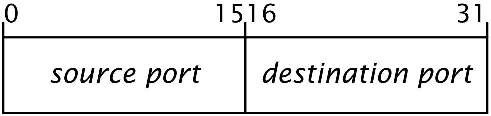
- How do we identify a "connection"?
  - two pairs of *host* and *application* identifiers
  - i.e., two pairs *(IP-address, port)*

- How do we find out which application (host and port number) to connect to?
  - outside the scope of the definition of the transport layer
  - but of course we can have "well-known" service numbers

Ports

Ports

- The message format of both UDP and TCP starts with the source and destination port numbers

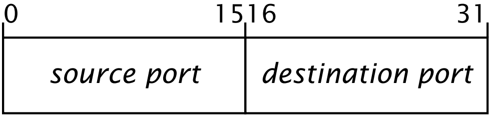


0 … 15	16 … 31
source port	*destination port*

. . .

Ports

- The message format of both UDP and TCP starts with the source and destination port numbers

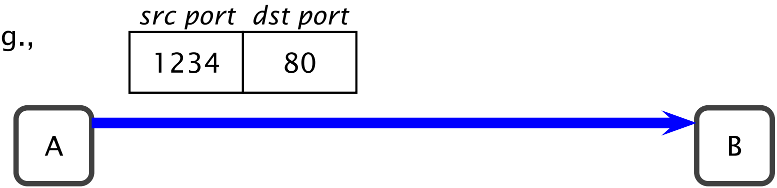


0 15 16 31

source port *destination port*

. . .

- E.g.,



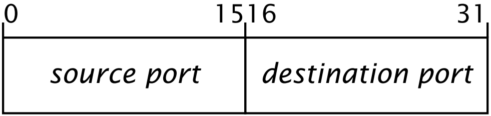
src port *dst port*

1234 80

A B

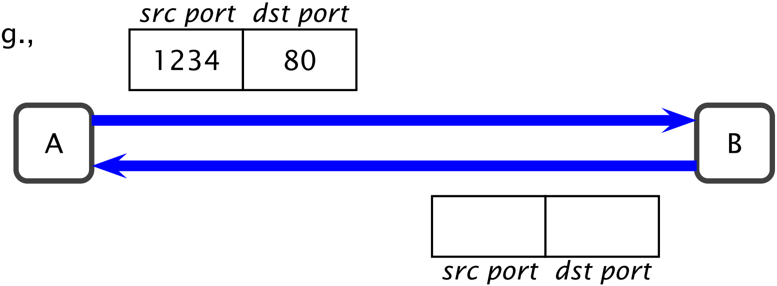
Ports

- The message format of both UDP and TCP starts with the source and destination port numbers



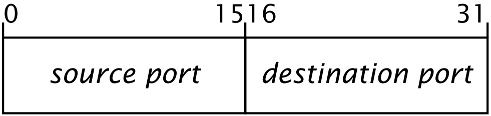
...

- E.g.,



Ports

- The message format of both UDP and TCP starts with the source and destination port numbers

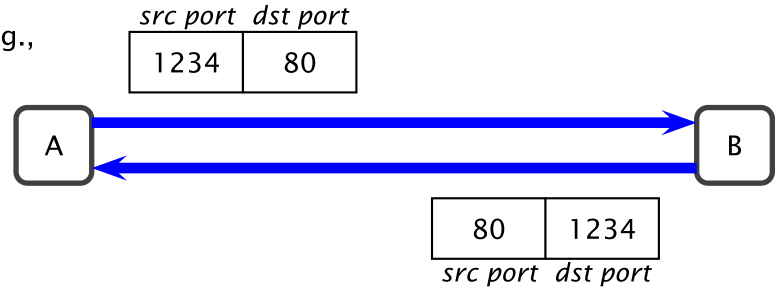


0 15 16 31

source port *destination port*

...

- E.g.,



src port *dst port*

1234 80

A

B

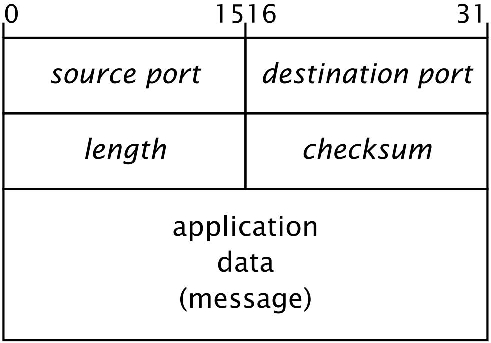
80 1234

src port *dst port*

UDP Packet Format

UDP Packet Format

- The UDP message format is very simple



0 15 16 31

source port *destination port*

length *checksum*

application
data
(message)

UDP Features

UDP Features

- UDP provides only the two most basic functionalities of a transport protocol

UDP Features

- UDP provides only the two most basic functionalities of a transport protocol
  - application identification (multiplexing/demultiplexing)

UDP Features

- UDP provides only the two most basic functionalities of a transport protocol
  - application identification (multiplexing/demultiplexing)
  - integrity check by means of a CRC-type checksum

UDP Features

- UDP provides only the two most basic functionalities of a transport protocol
  - application identification (multiplexing/demultiplexing)
  - integrity check by means of a CRC-type checksum

- What if there is no application at the other end?

UDP Features

- UDP provides only the two most basic functionalities of a transport protocol
  - application identification (multiplexing/demultiplexing)
  - integrity check by means of a CRC-type checksum
- What if there is no application at the other end?
- How is the checksum computed?
  - which parts of the segment does it cover?

UDP Features

- UDP provides only the two most basic functionalities of a transport protocol
  - application identification (multiplexing/demultiplexing)
  - integrity check by means of a CRC-type checksum
- What if there is no application at the other end?
- How is the checksum computed?
  - which parts of the segment does it cover?
- What should happen when the checksum doesn’t check?