

# Internet Electronic Mail

Antonio Carzaniga

Faculty of Informatics  
Università della Svizzera italiana

October 19, 2018

- General concepts
- Transport protocol: SMTP
- Basic message format
- MIME format

# A Postal Service for the Internet

# A Postal Service for the Internet



Alice

# A Postal Service for the Internet



Alice

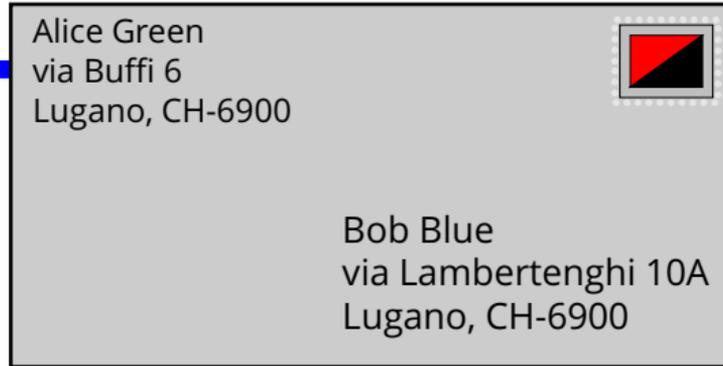


Bob

# A Postal Service for the Internet



Alice

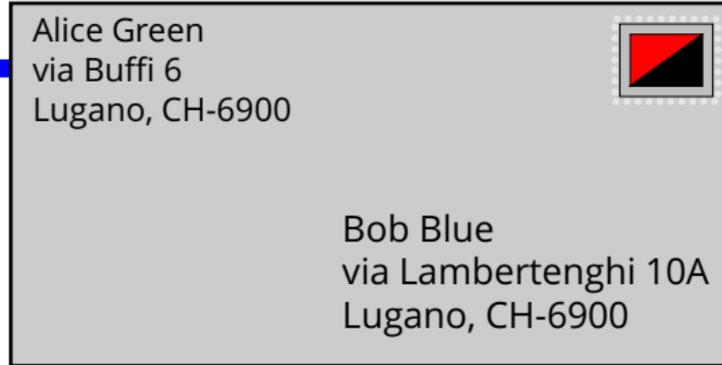


Bob

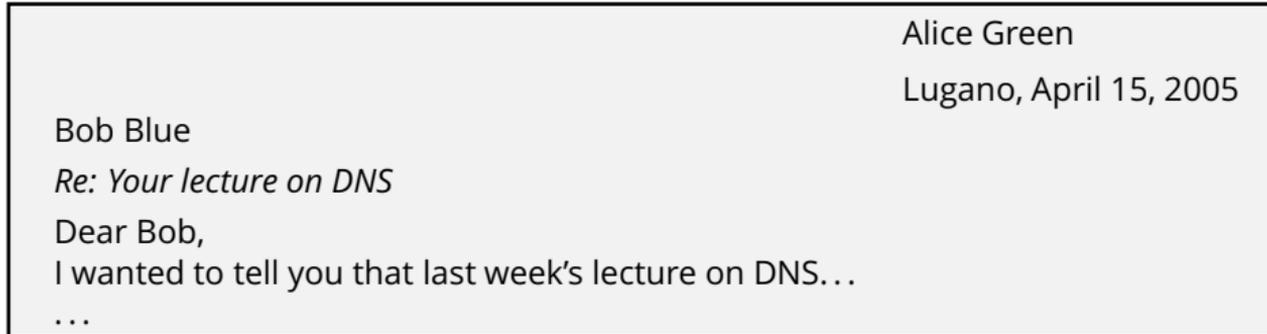
# A Postal Service for the Internet



Alice



Bob



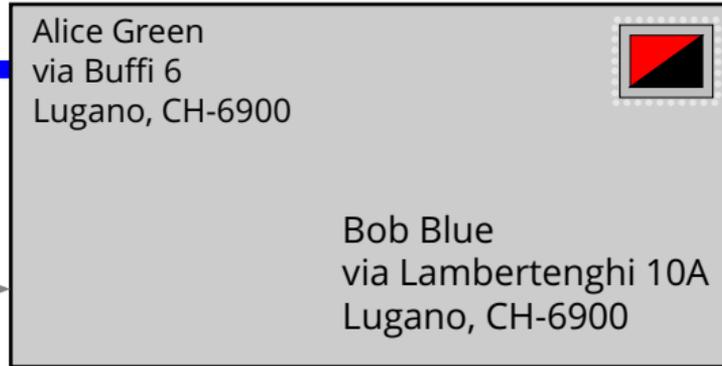
# A Postal Service for the Internet



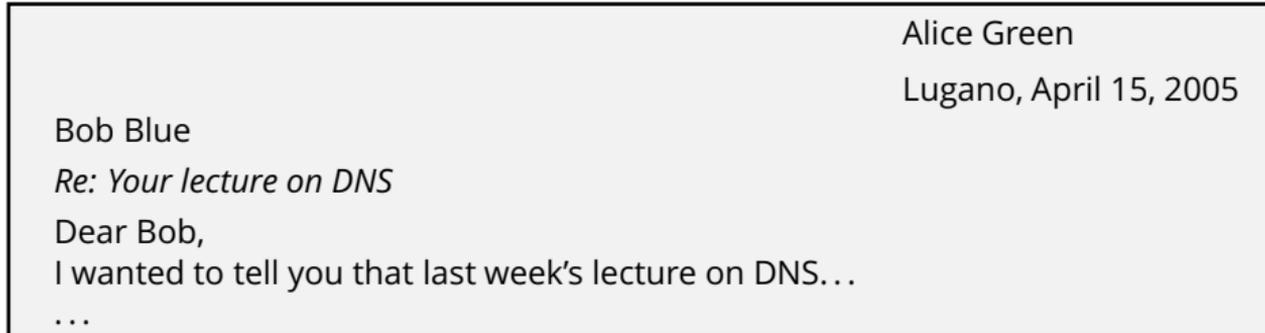
Alice



*envelope* →



Bob



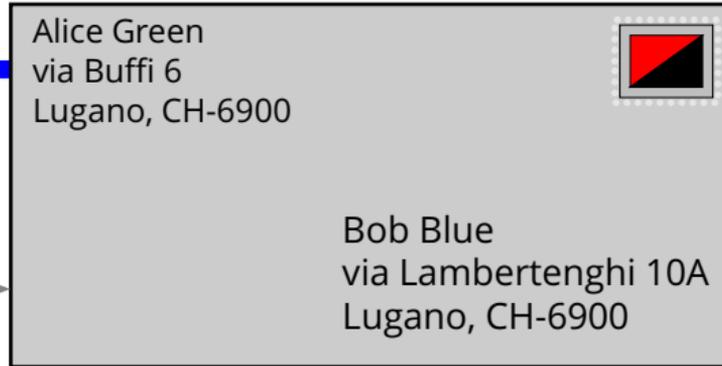
# A Postal Service for the Internet



Alice



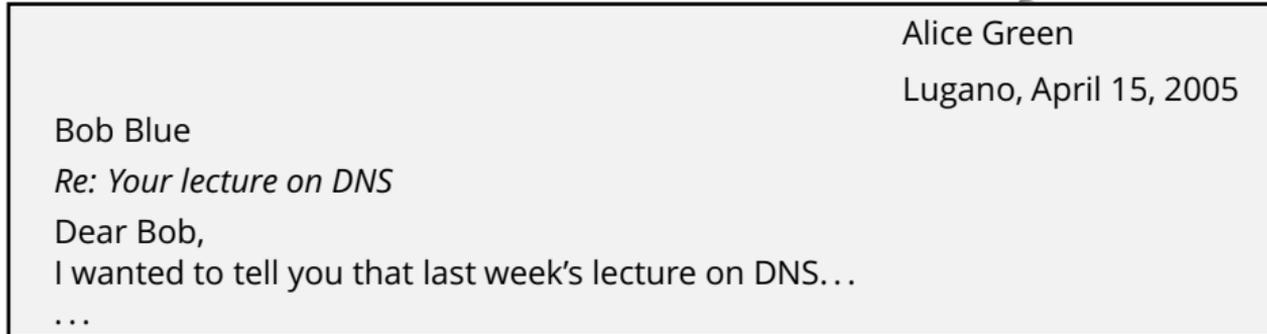
*envelope* →



*message* ←



Bob





- Asynchronous communication

- ▶ Alice sends a message when it is convenient to her
- ▶ Bob reads Alice's message whenever he has time to do that

## ■ Asynchronous communication

- ▶ Alice sends a message when it is convenient to her
- ▶ Bob reads Alice's message whenever he has time to do that

## ■ One-to-many communication

- ▶ Alice can send a message to Bob and Charlie
- ▶ a mailing list sends messages to several receivers

## ■ Asynchronous communication

- ▶ Alice sends a message when it is convenient to her
- ▶ Bob reads Alice's message whenever he has time to do that

## ■ One-to-many communication

- ▶ Alice can send a message to Bob and Charlie
- ▶ a mailing list sends messages to several receivers

## ■ Multi-media content

- ▶ images and all sorts of attachments as well as normal text



- No authentication

- ▶ Bob can not know for sure that the message he reads was actually written by Alice
- ▶ messages can be modified
- ▶ messages can be forged

## ■ No authentication

- ▶ Bob can not know for sure that the message he reads was actually written by Alice
- ▶ messages can be modified
- ▶ messages can be forged

## ■ No confidentiality

- ▶ Alice can not make sure that only Bob will read the message
- ▶ the message can be read by others

## ■ No authentication

- ▶ Bob can not know for sure that the message he reads was actually written by Alice
- ▶ messages can be modified
- ▶ messages can be forged

## ■ No confidentiality

- ▶ Alice can not make sure that only Bob will read the message
- ▶ the message can be read by others

## ■ Little or no delivery guarantees

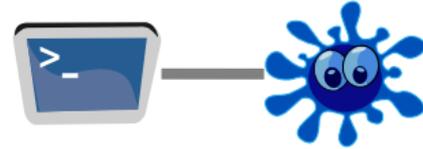
- ▶ Alice has no idea whether the messages was in fact receiver (much less read!) by Bob
- ▶ messages can be accidentally lost or intentionally blocked
- ▶ no reliable acknowledgement system

# Architecture and Terminology

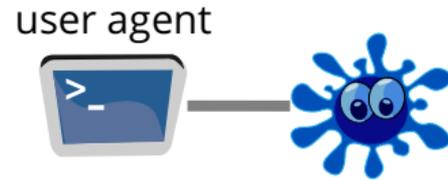
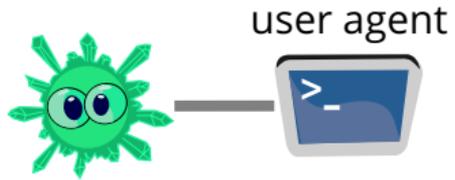
# Architecture and Terminology



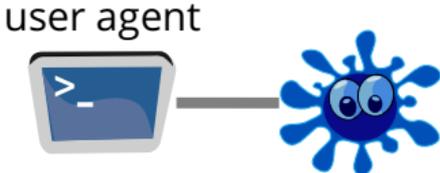
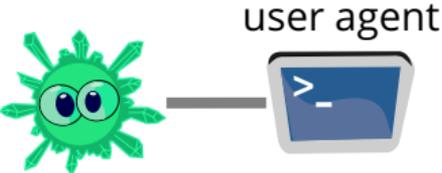
# Architecture and Terminology



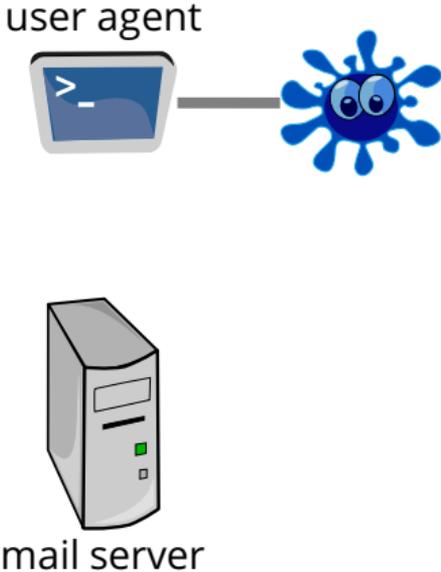
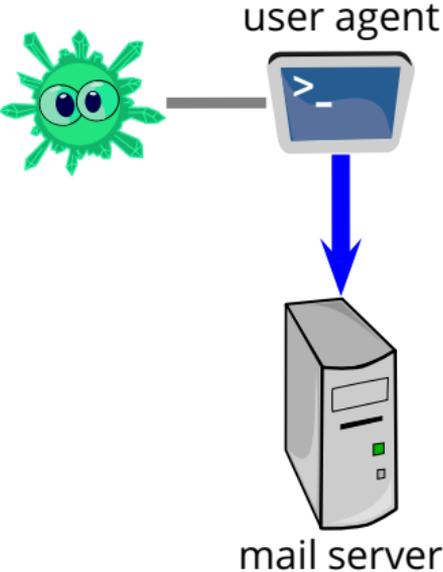
# Architecture and Terminology



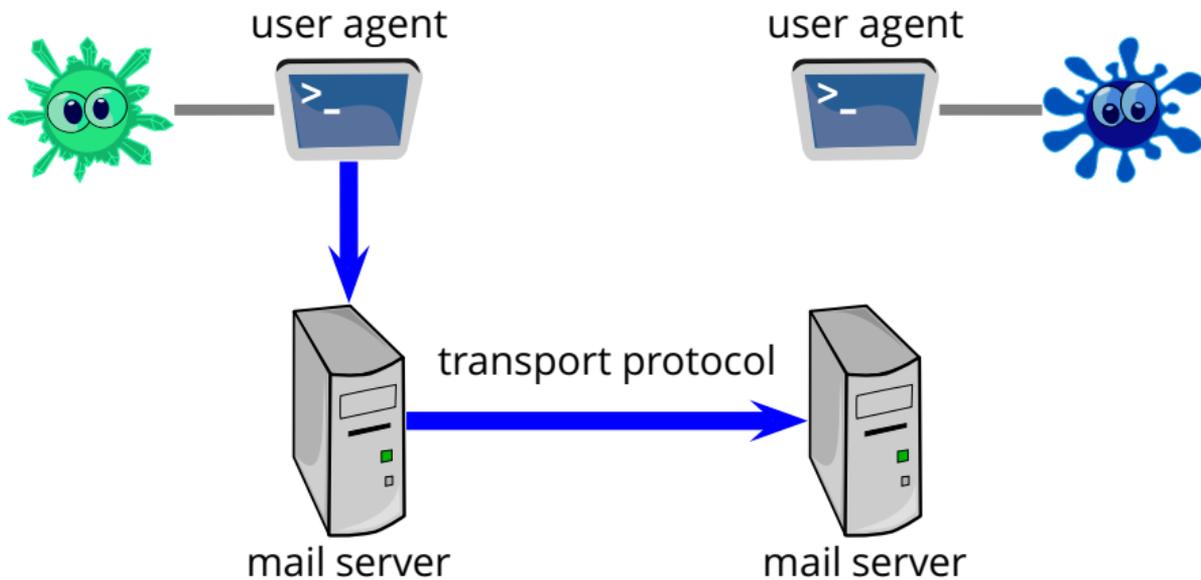
# Architecture and Terminology



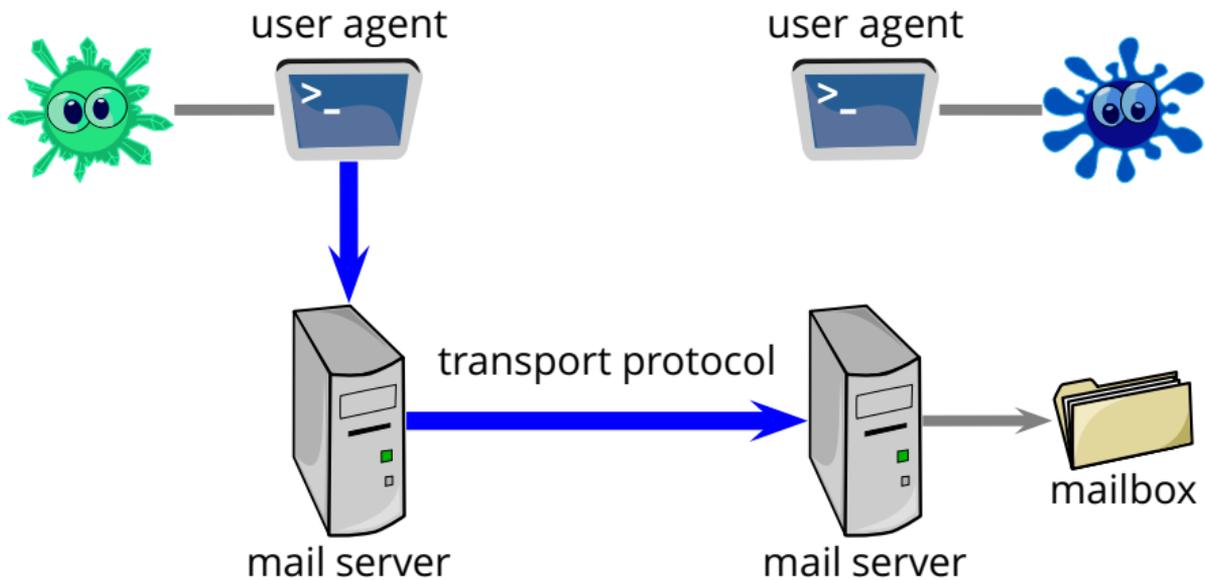
# Architecture and Terminology



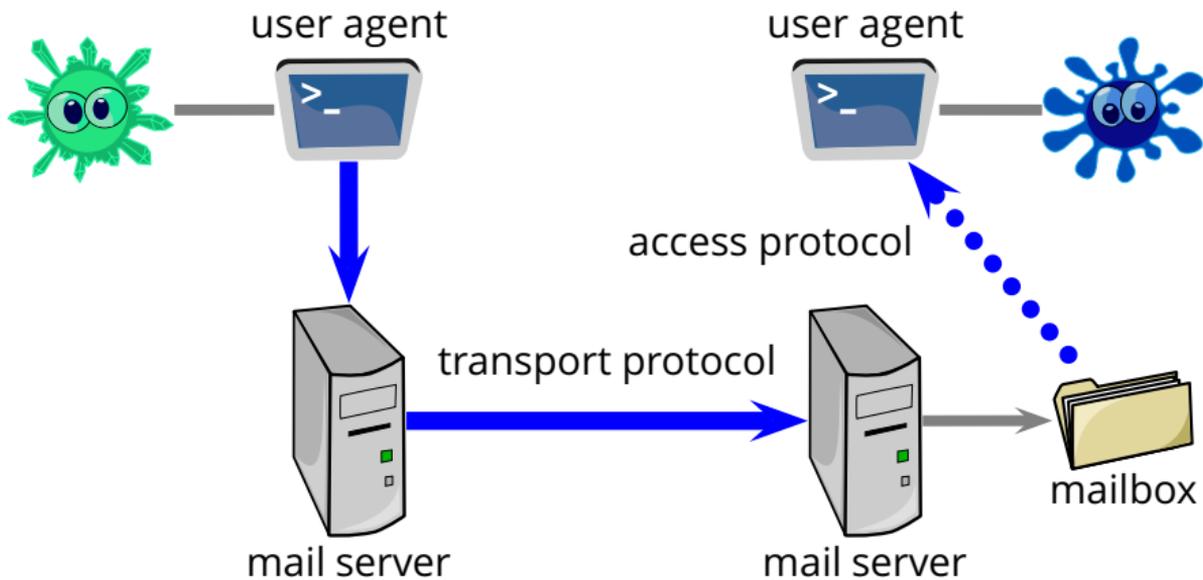
# Architecture and Terminology



# Architecture and Terminology



# Architecture and Terminology



# Architecture and Terminology

## ■ *User agent*

- ▶ allows a user to read, compose, reply to, send, and forward messages
- ▶ and also to save, classify, sort, search, ...

## ■ *User agent*

- ▶ allows a user to read, compose, reply to, send, and forward messages
- ▶ and also to save, classify, sort, search, ...

## ■ *Mail servers*

- ▶ accept messages for *remote delivery*
  - ▶ store messages in a local persistent queue
  - ▶ deliver messages to a remote (destination) server using the *transport protocol*
- ▶ accept messages for *local delivery*
  - ▶ save messages in some local persistent mailbox
- ▶ allow user agents to *access local mailboxes*
  - ▶ user agents can retrieve and/or delete messages
  - ▶ this is done through an *access protocol*



- ***Simple Mail Transfer Protocol*** (defined in RFC 2821)

- ***Simple Mail Transfer Protocol*** (defined in RFC 2821)
- Connection-oriented protocol

- ***Simple Mail Transfer Protocol*** (defined in RFC 2821)
- Connection-oriented protocol
- It is “simple”
  - ▶ indeed its simplicity is a reason for its success

- ***Simple Mail Transfer Protocol*** (defined in RFC 2821)
- Connection-oriented protocol
- It is “simple”
  - ▶ indeed its simplicity is a reason for its success
- It is an old protocol, compared to HTTP; the first RFCs date back to the early 80s
  - ▶ it has some archaic characteristics. E.g., it is restricted to 7-bit characters

# SMTP Abstract Example

usi.ch

*client*

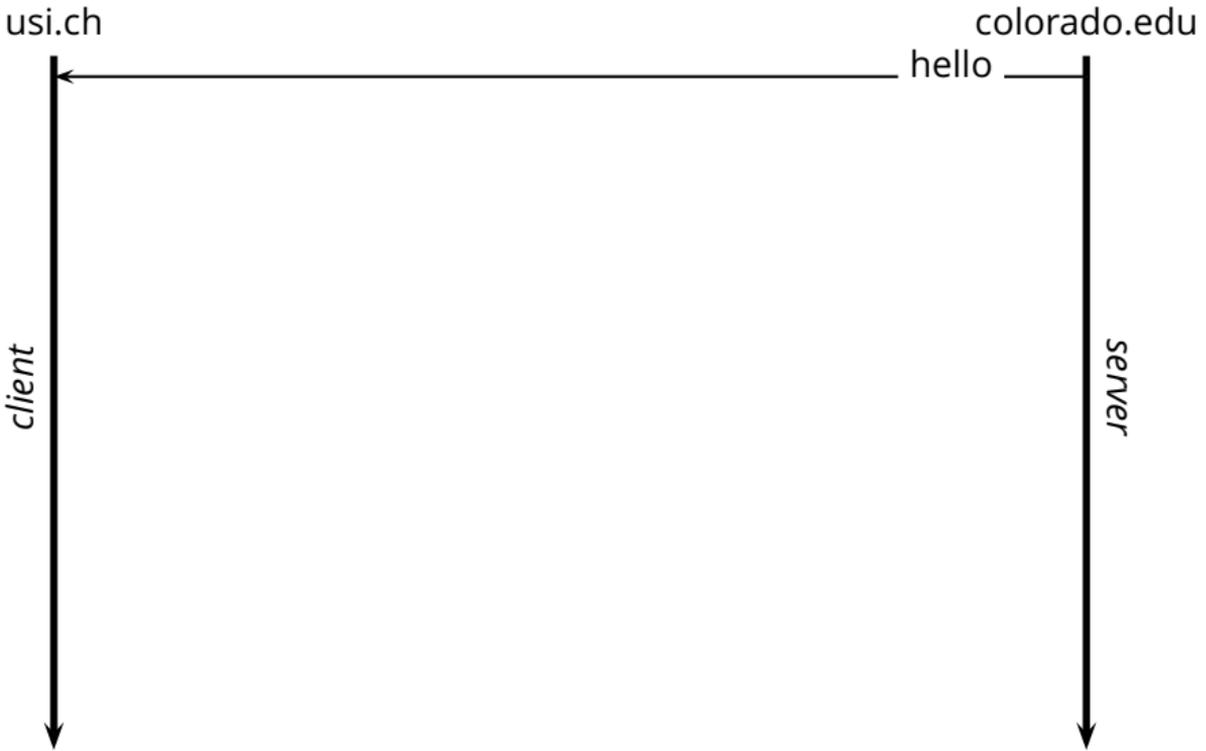


colorado.edu

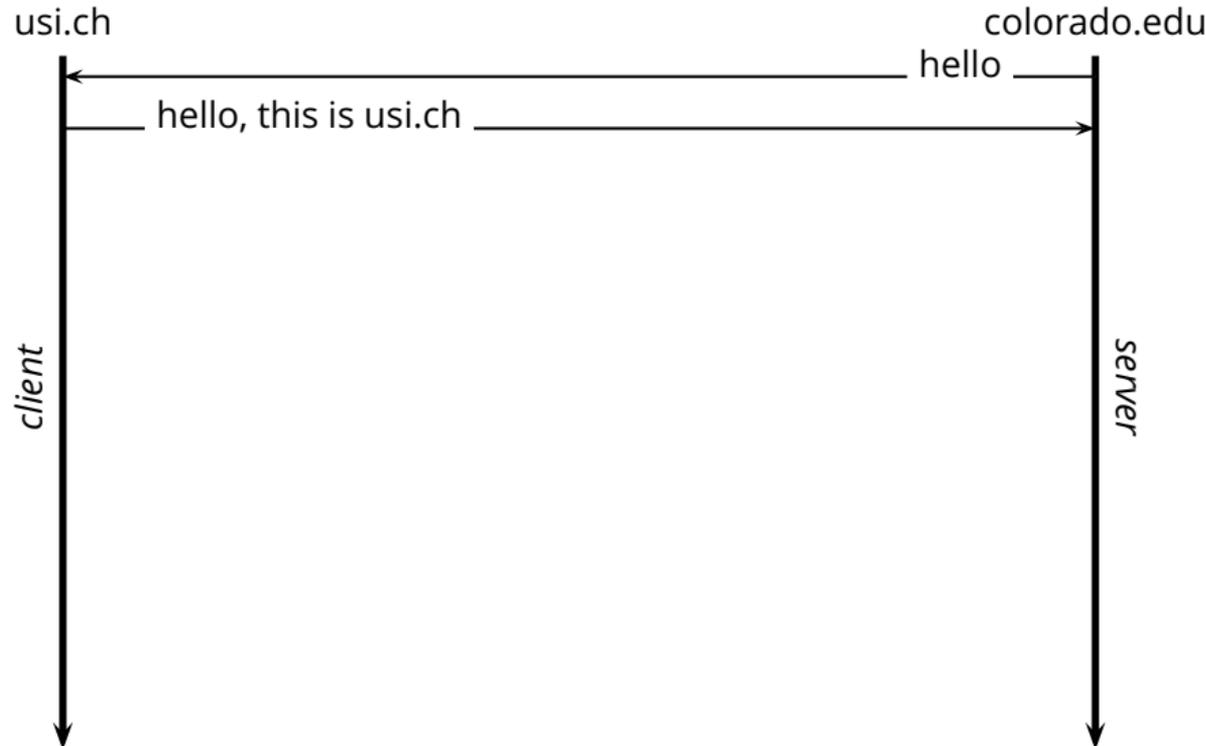
*server*



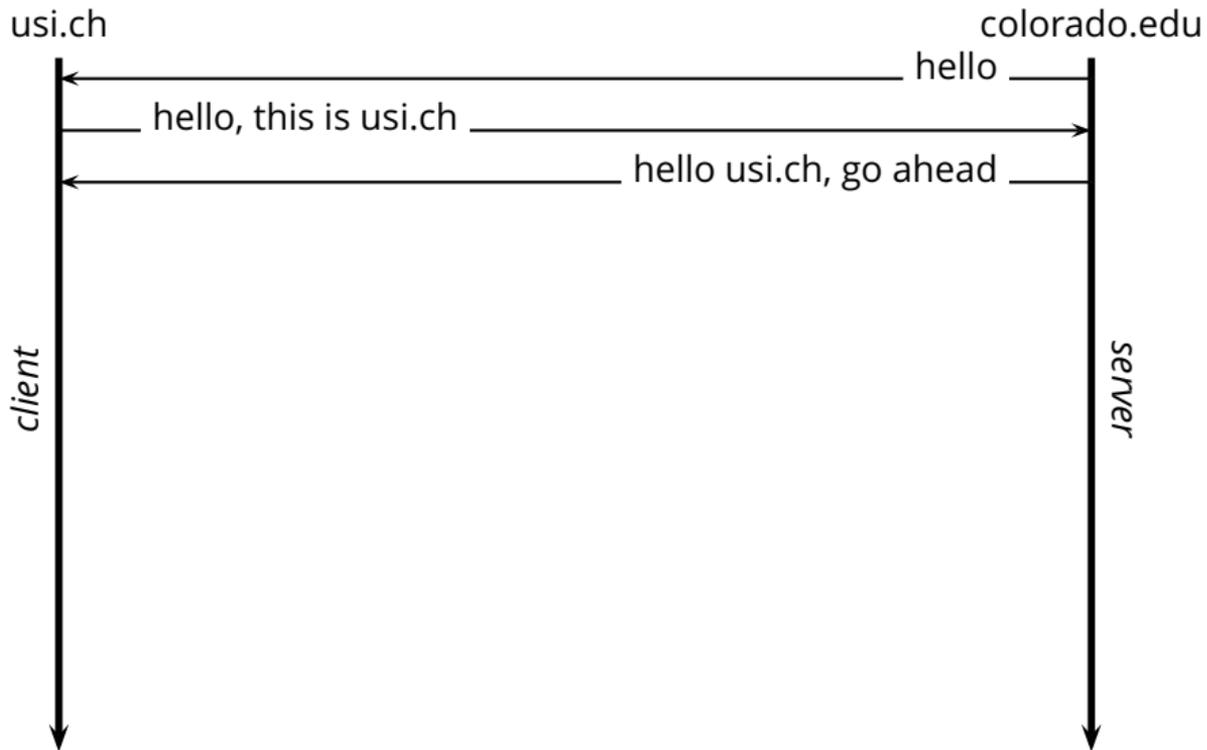
# SMTP Abstract Example



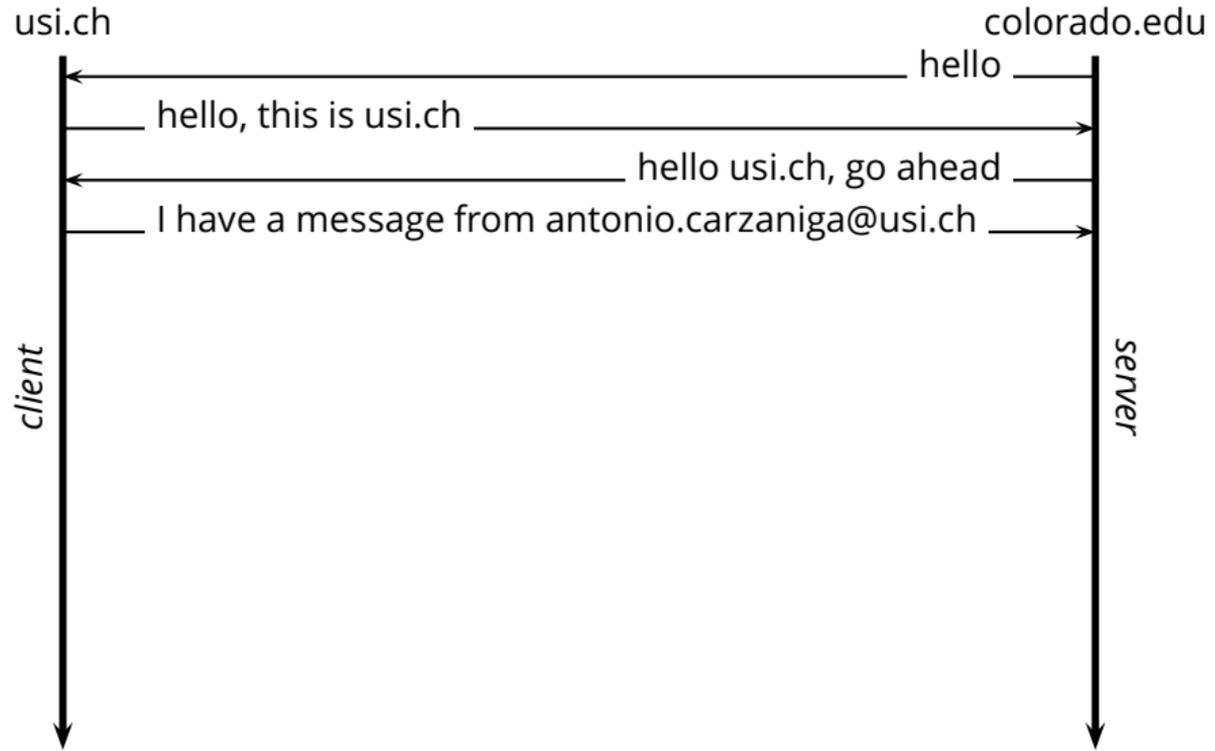
# SMTP Abstract Example



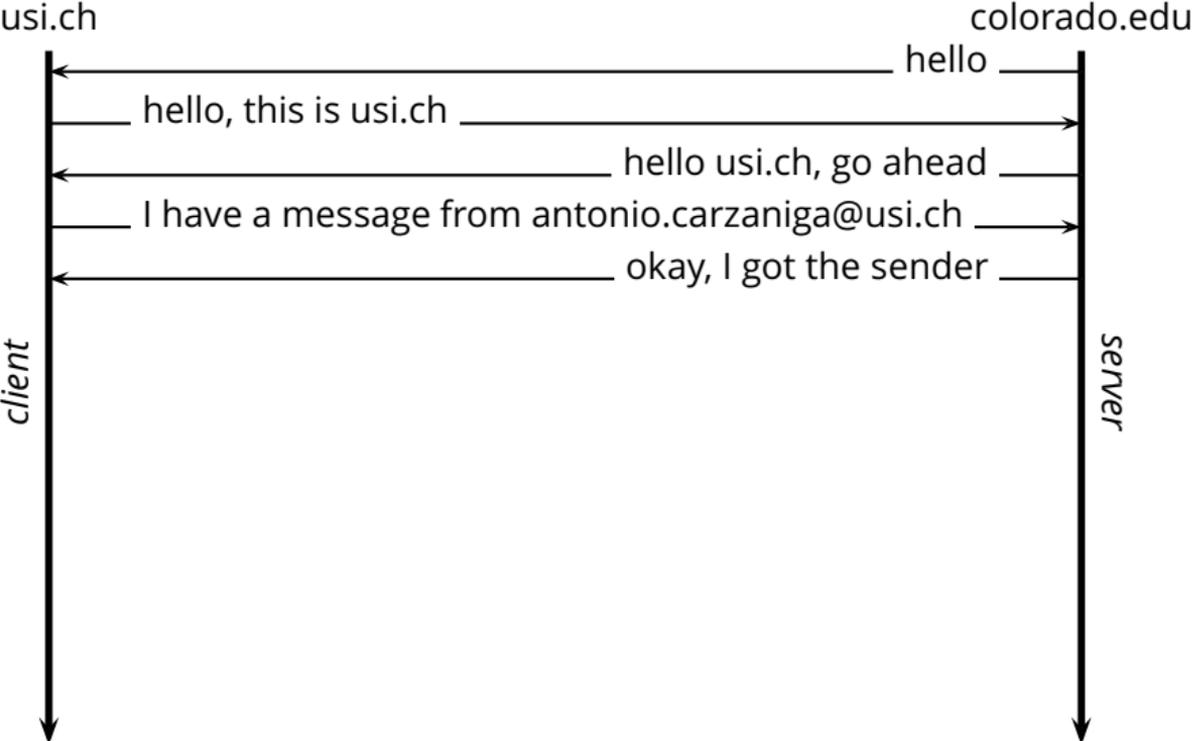
# SMTP Abstract Example



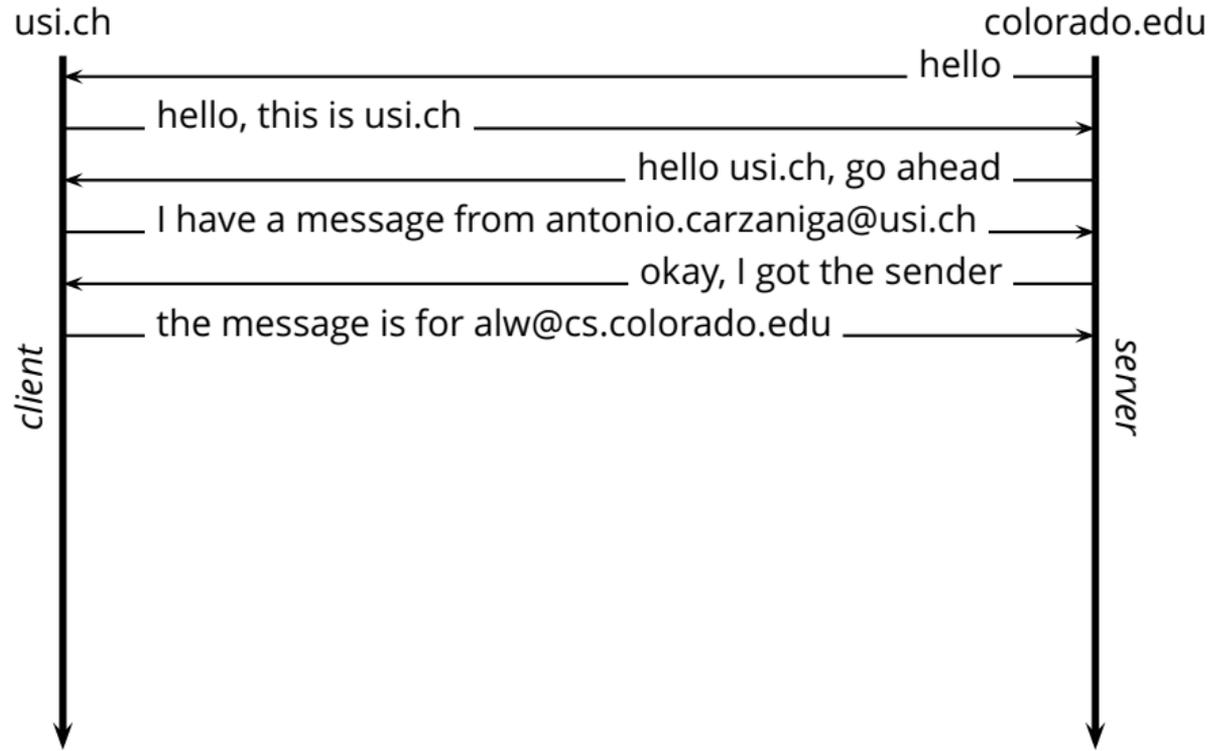
# SMTP Abstract Example



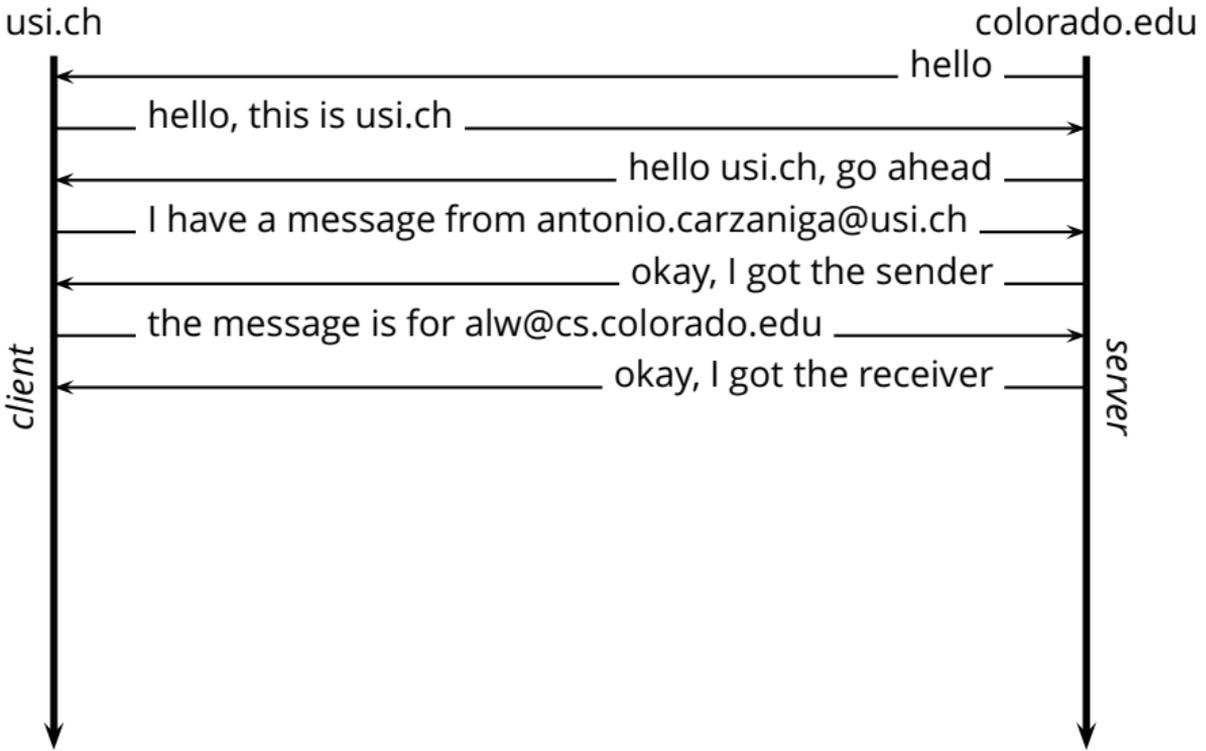
# SMTP Abstract Example



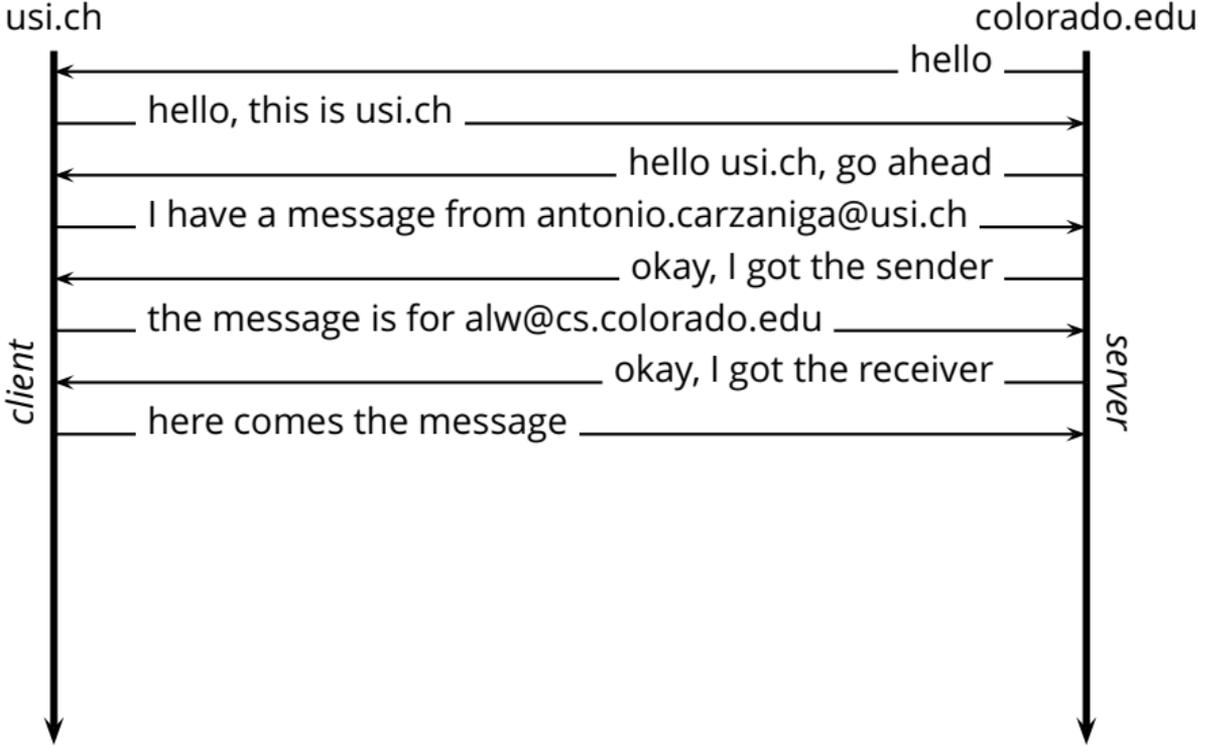
# SMTP Abstract Example



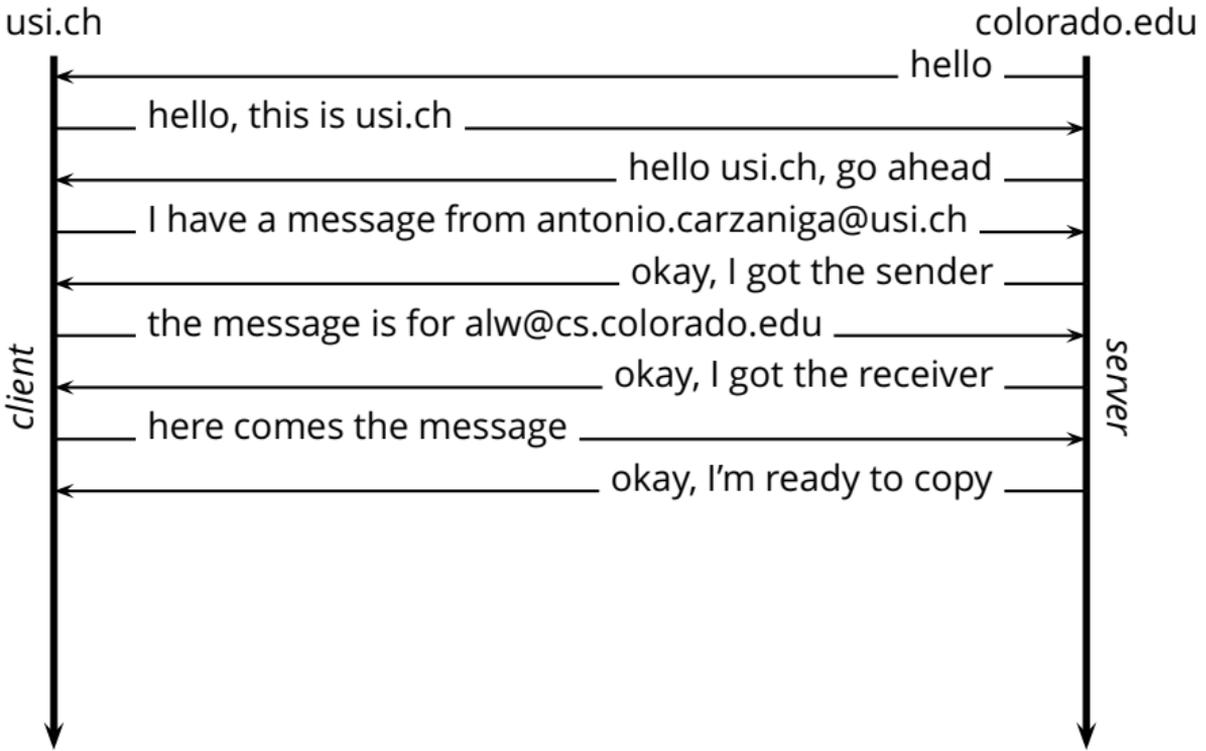
# SMTP Abstract Example



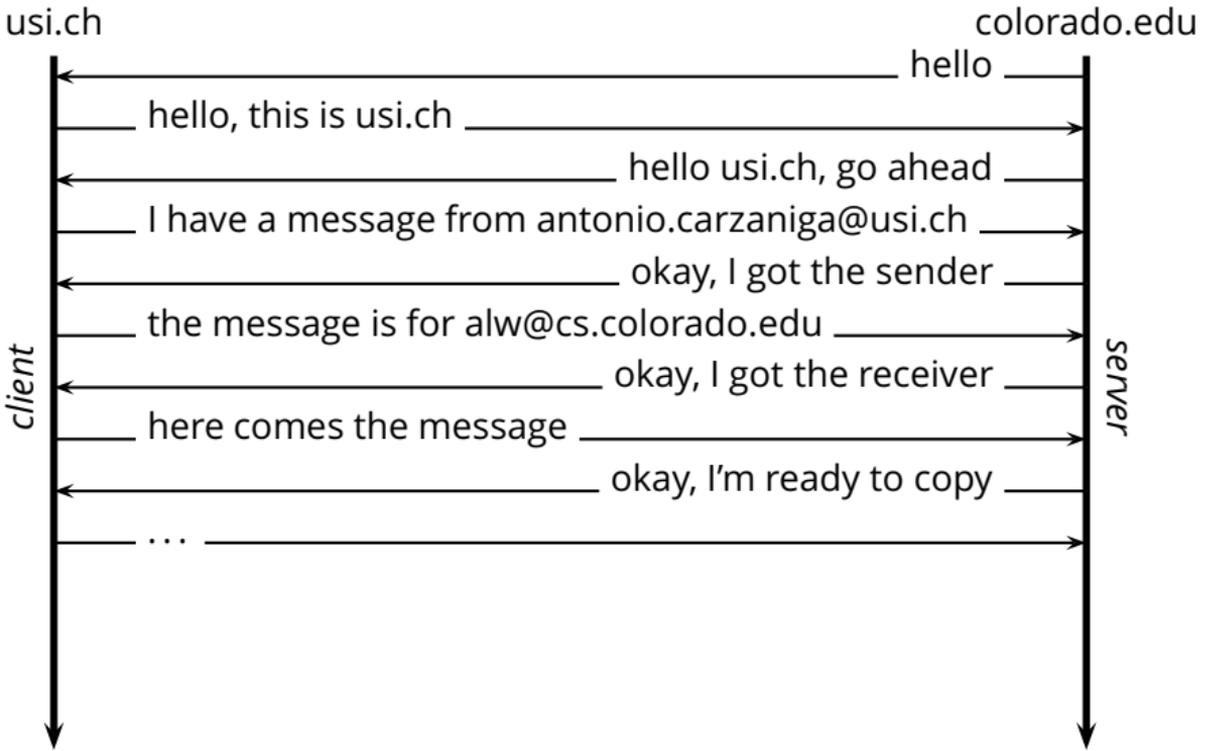
# SMTP Abstract Example



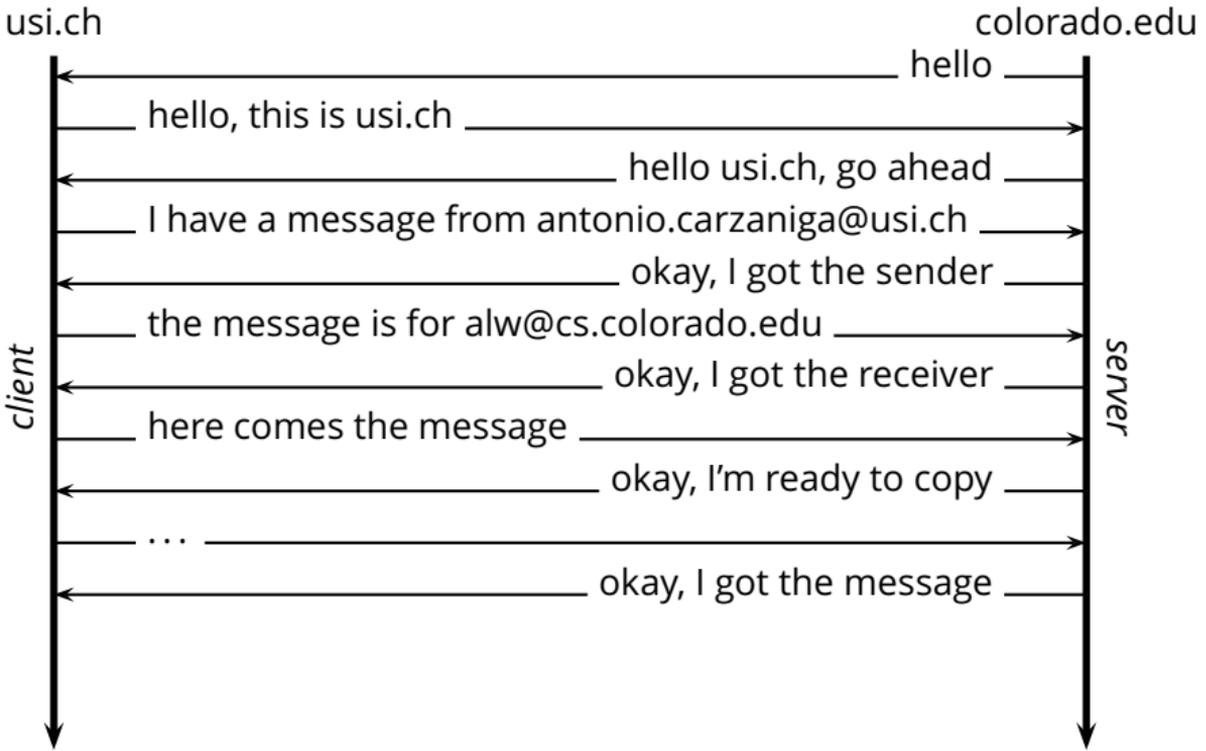
# SMTP Abstract Example



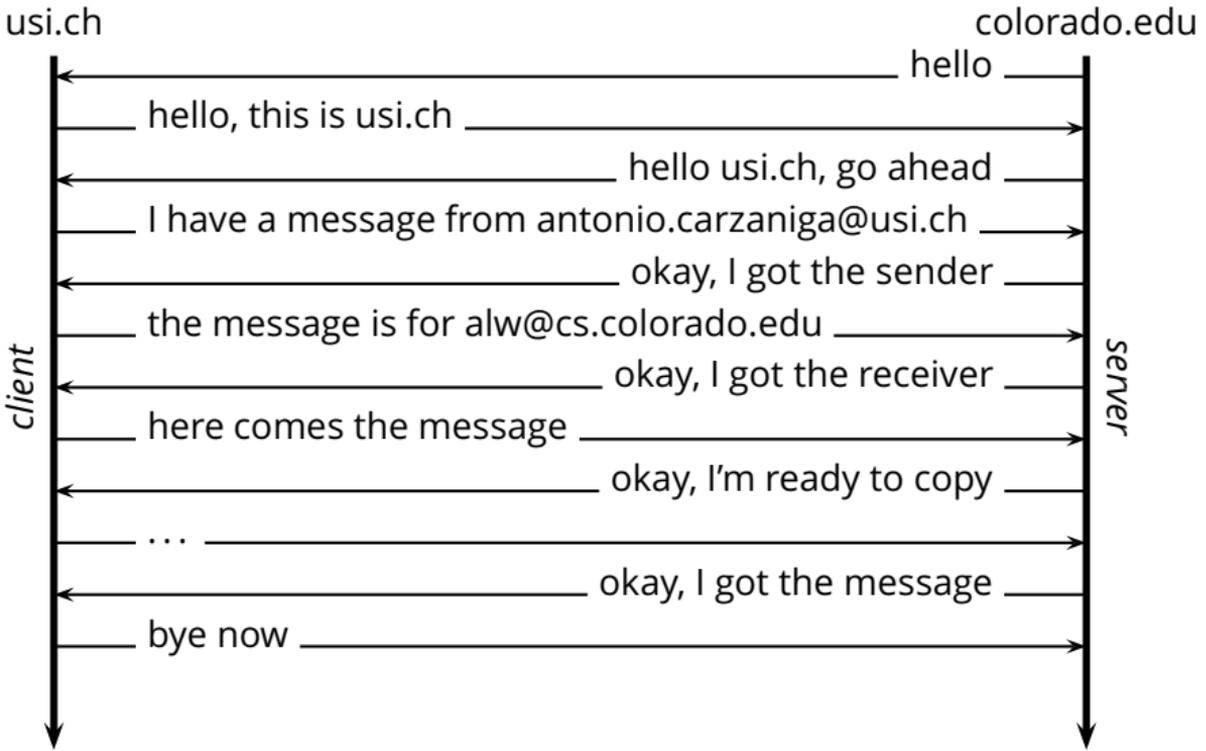
# SMTP Abstract Example



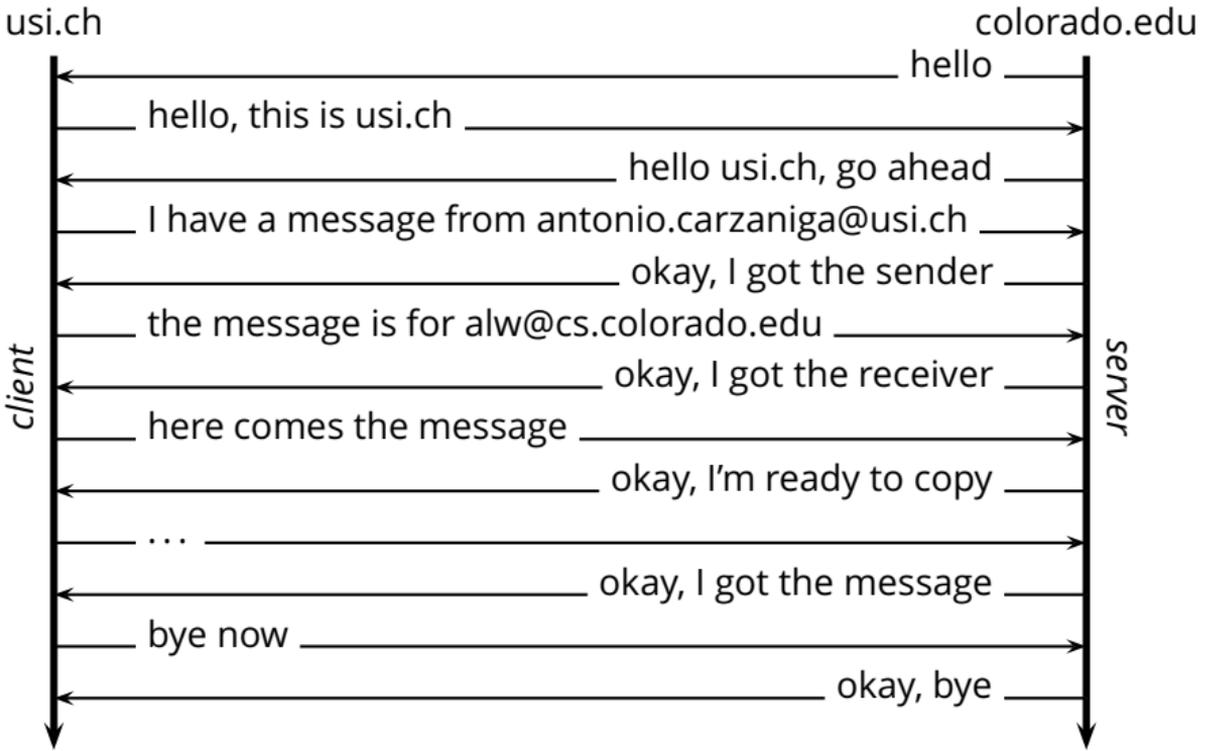
# SMTP Abstract Example



# SMTP Abstract Example



# SMTP Abstract Example



# SMTP Concrete Example

usi.ch

*client*

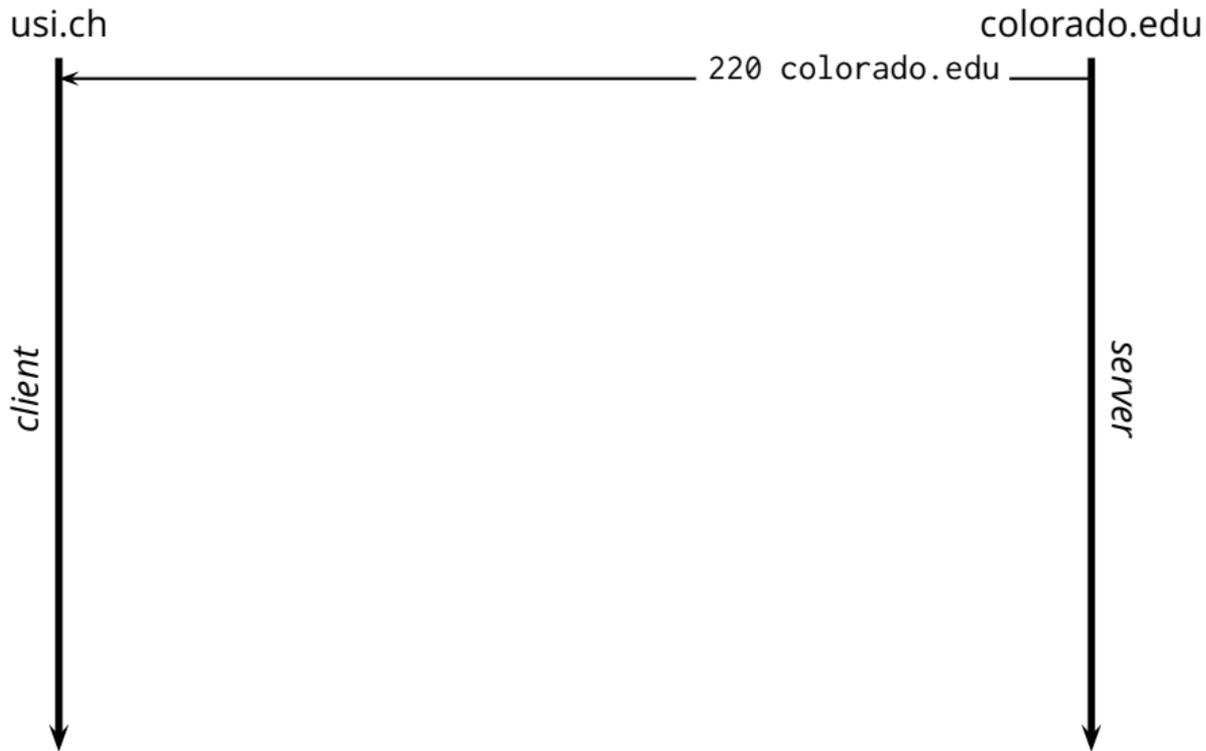


colorado.edu

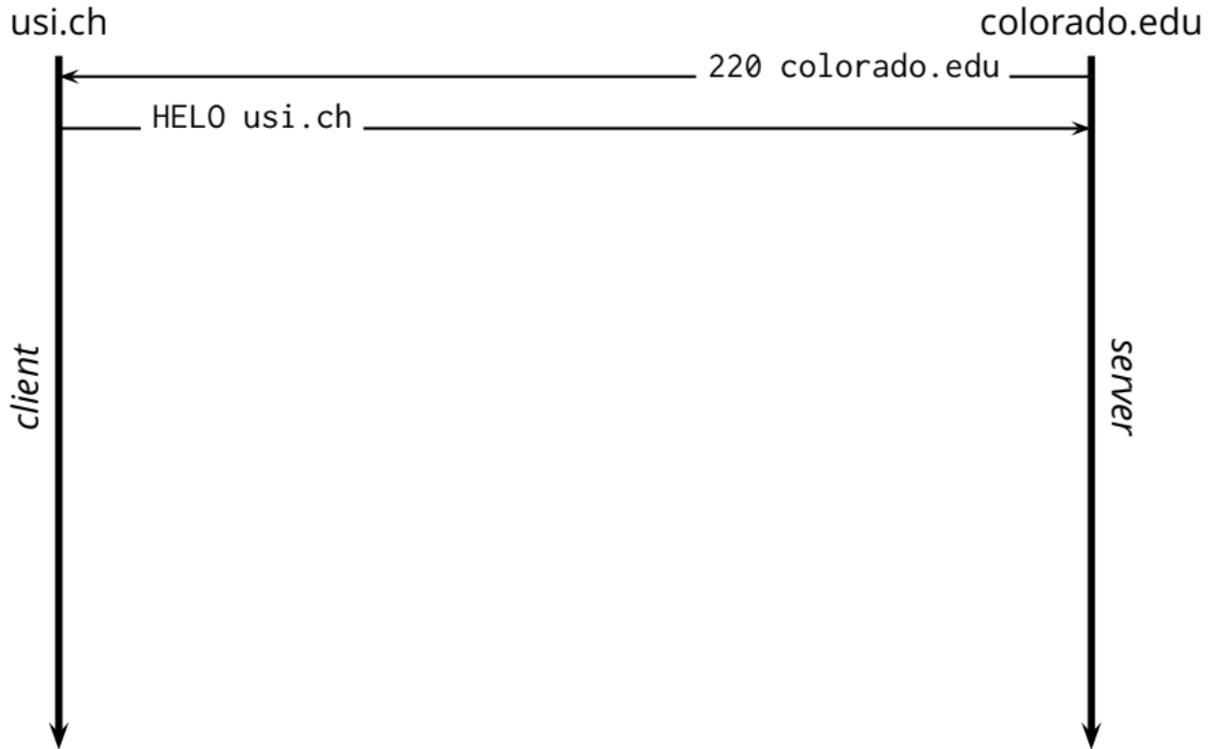
*server*



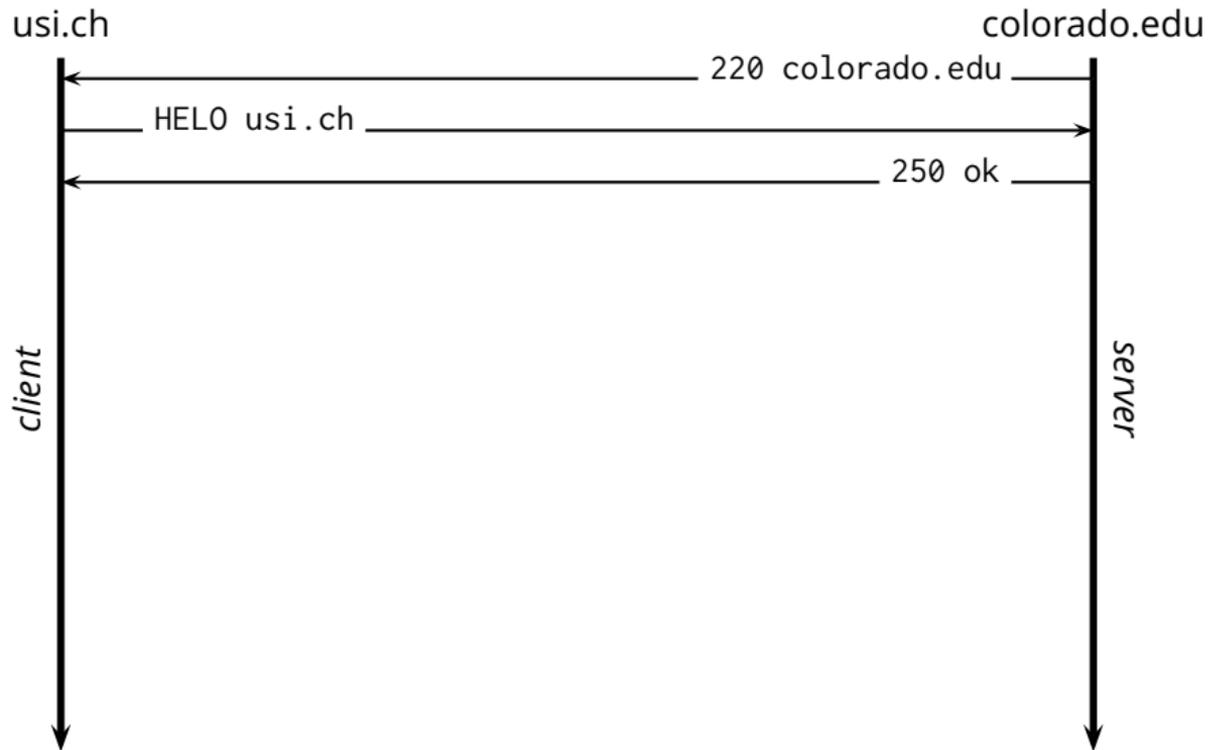
# SMTP Concrete Example



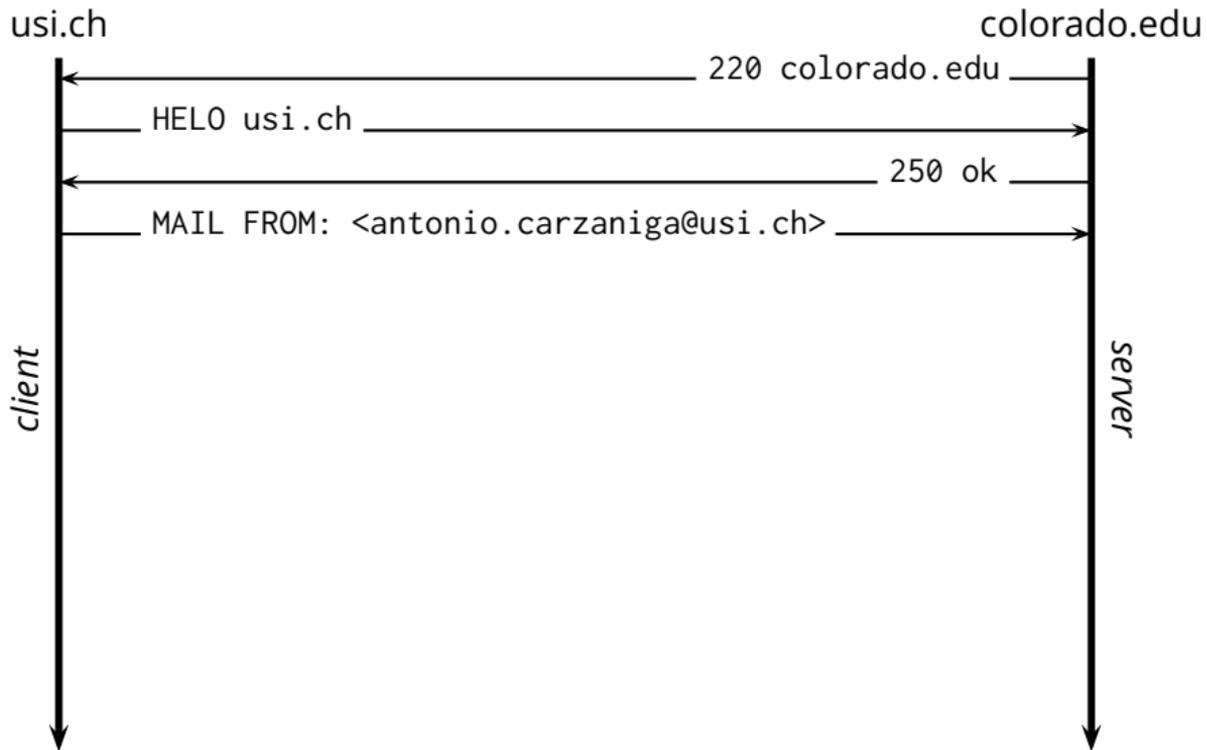
# SMTP Concrete Example



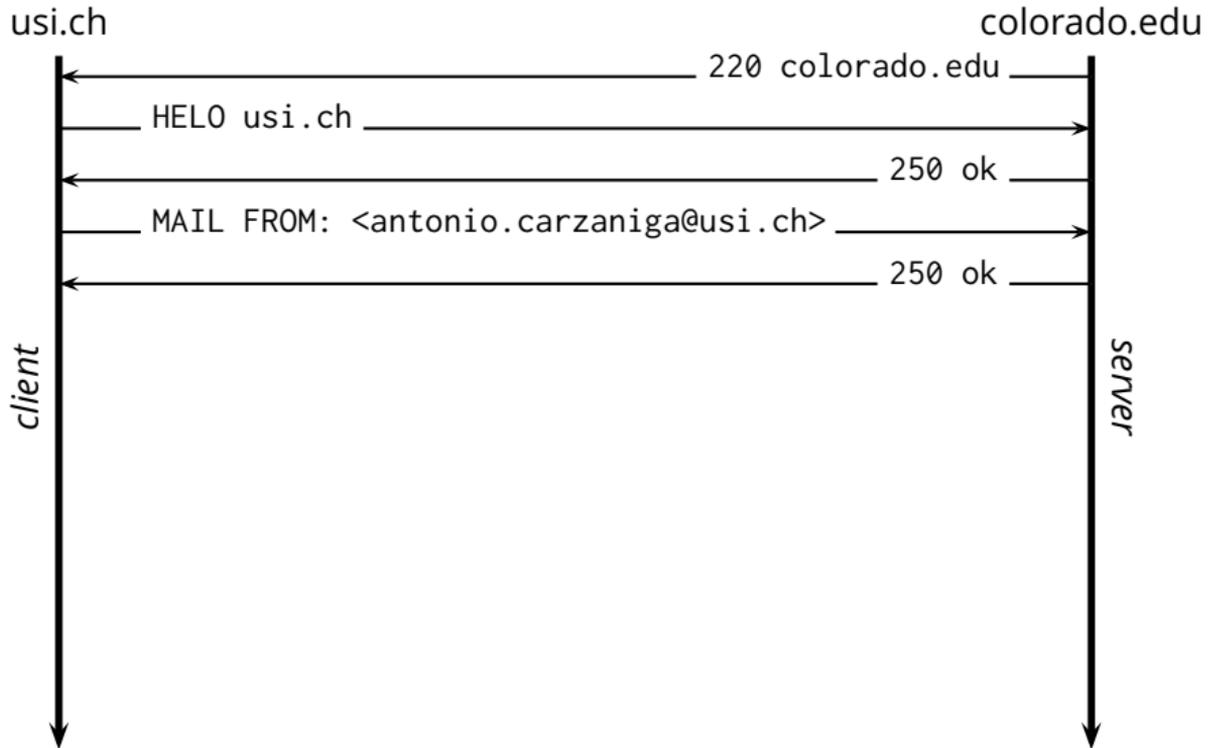
# SMTP Concrete Example



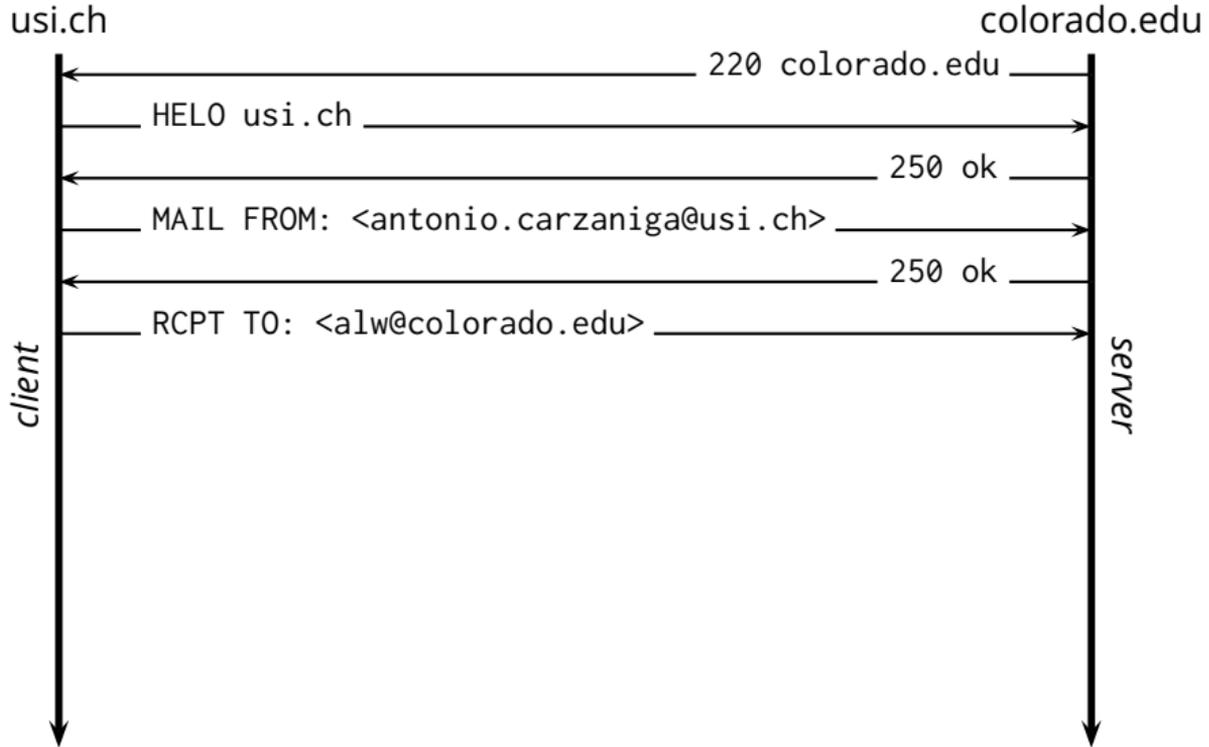
# SMTP Concrete Example



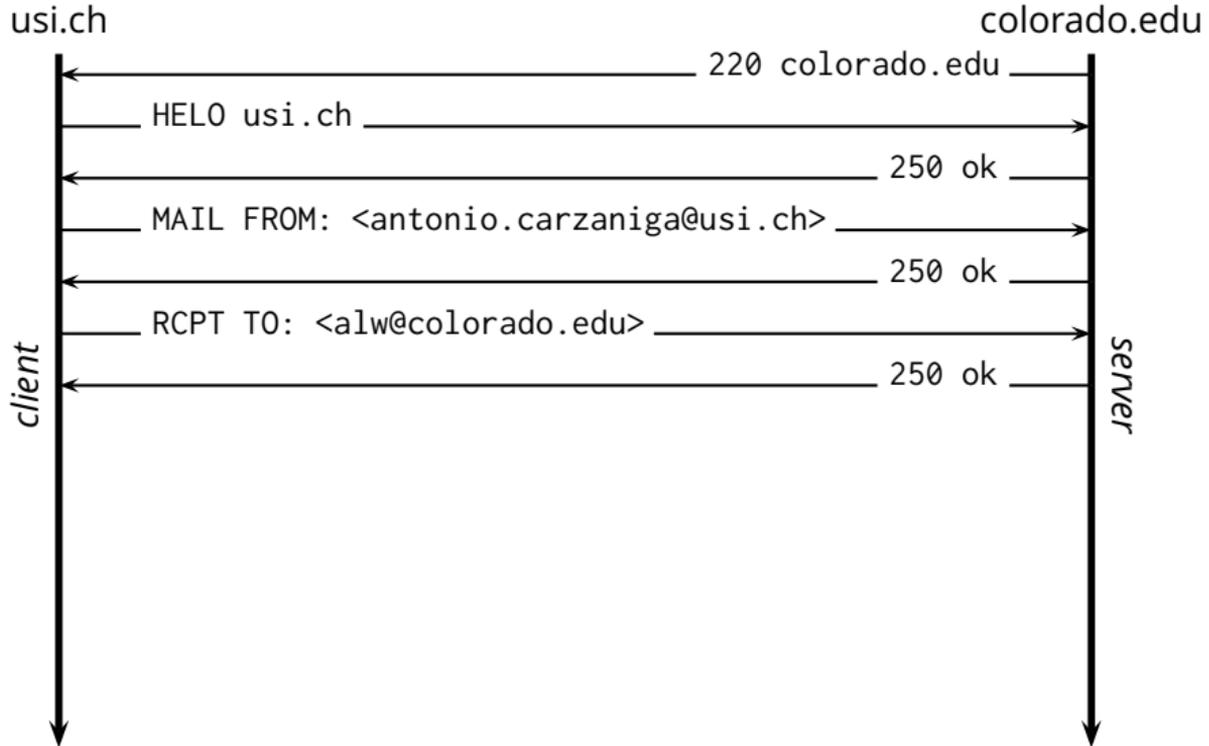
# SMTP Concrete Example



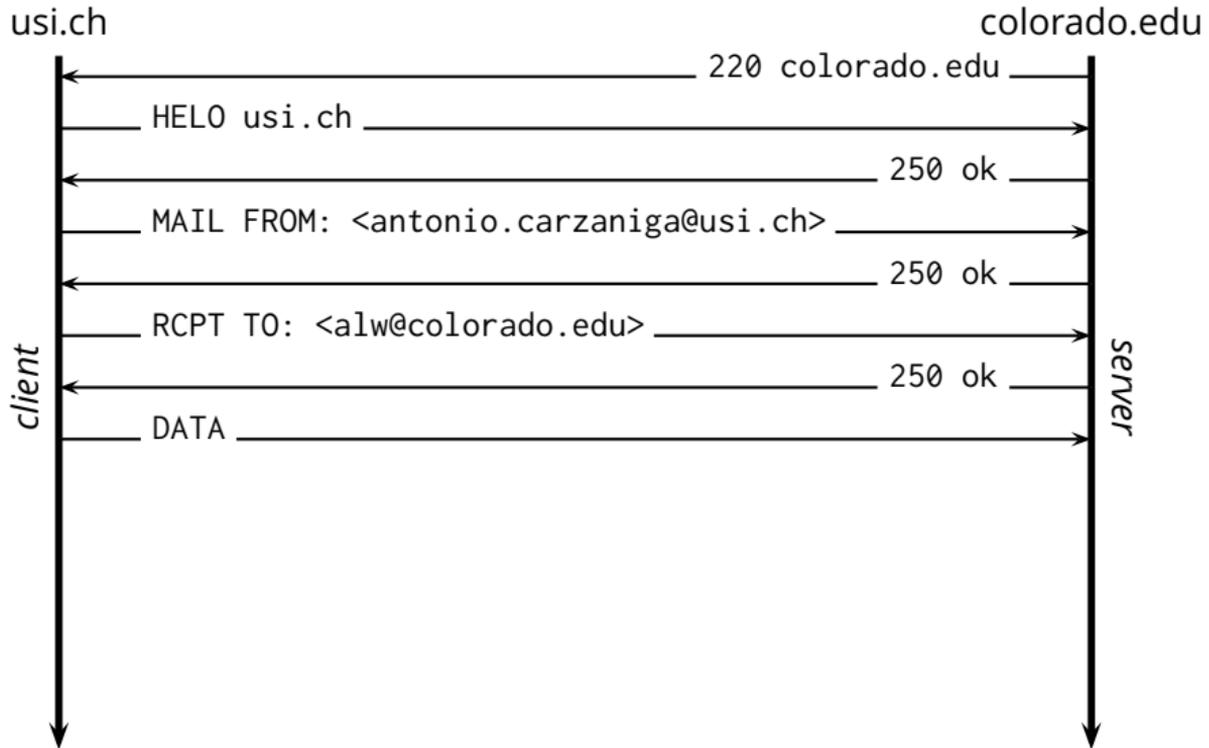
# SMTP Concrete Example



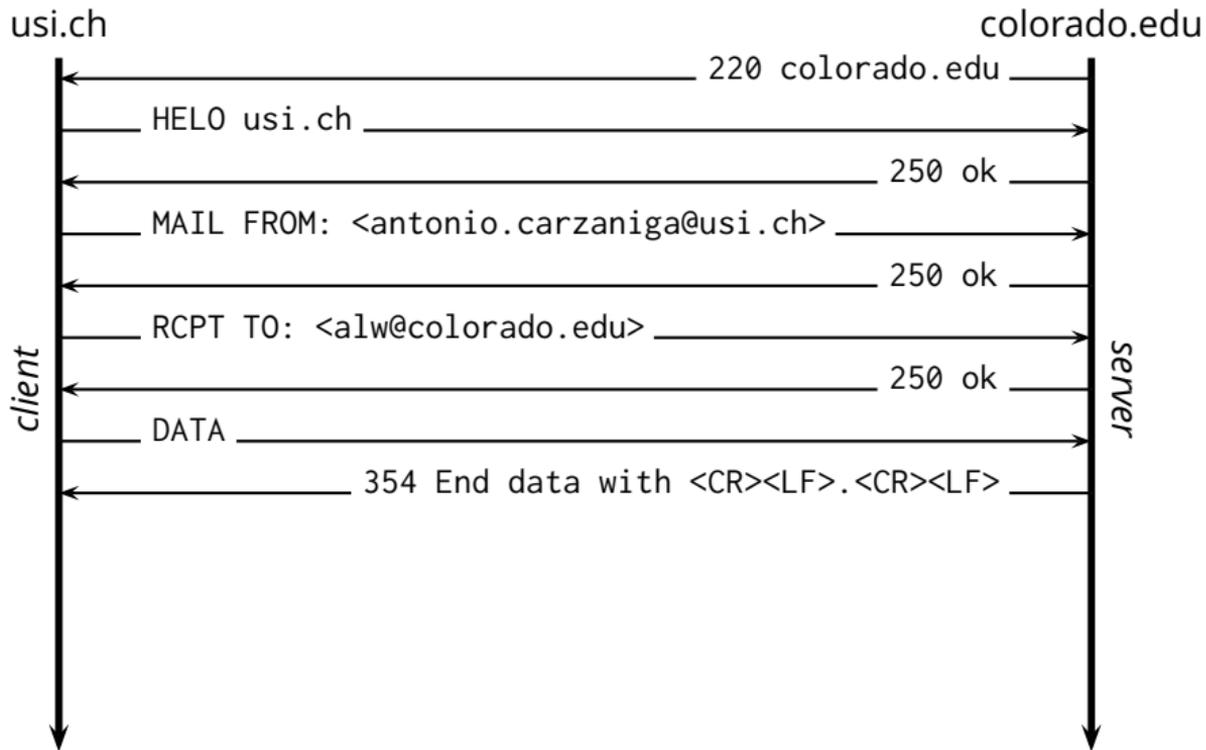
# SMTP Concrete Example



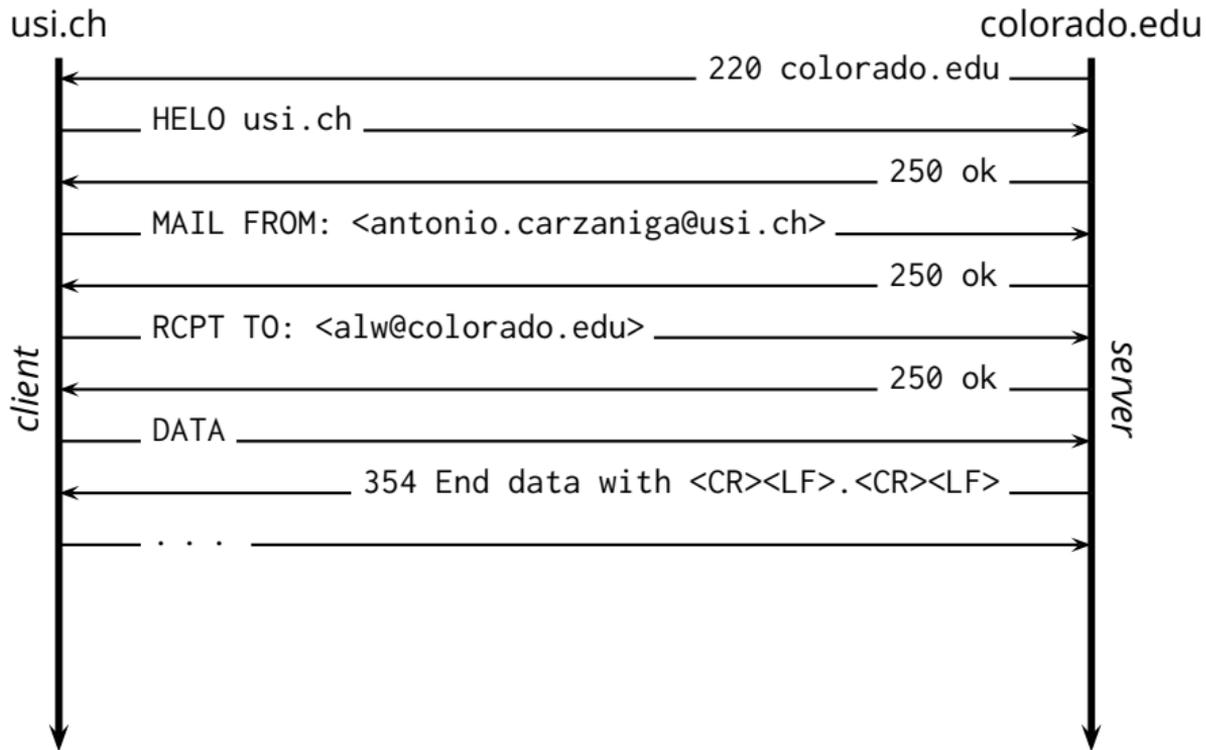
# SMTP Concrete Example



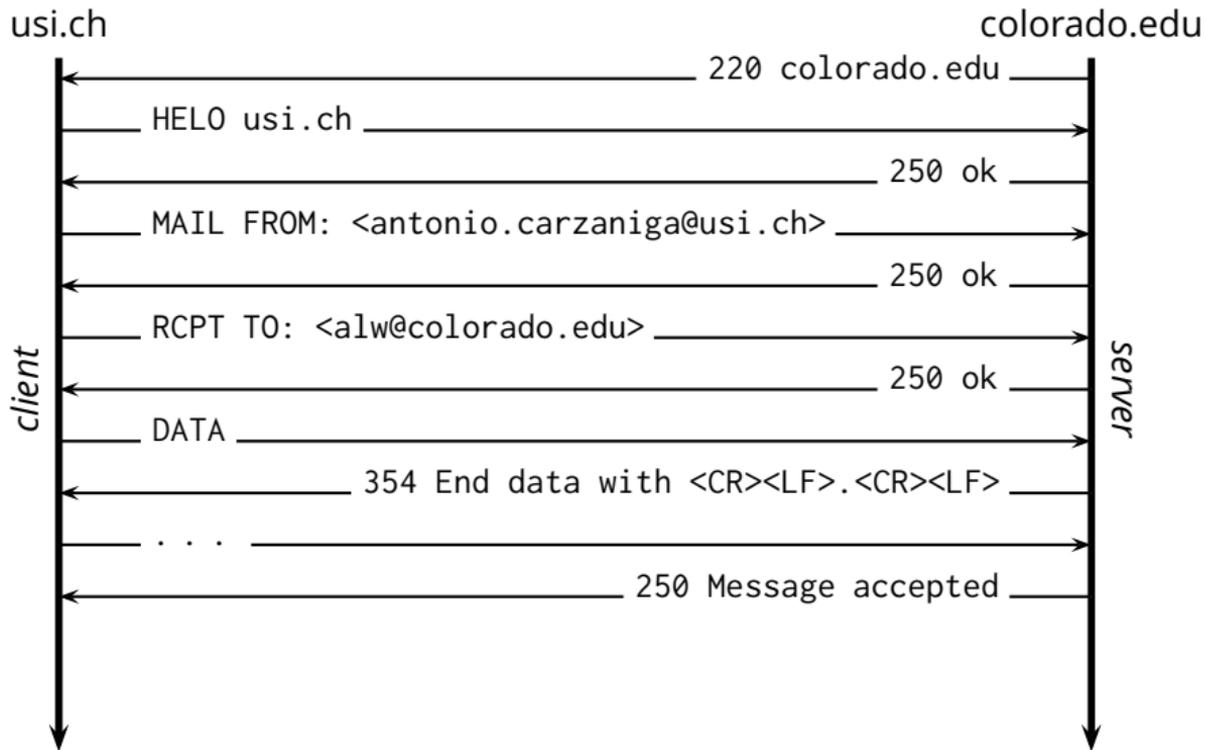
# SMTP Concrete Example



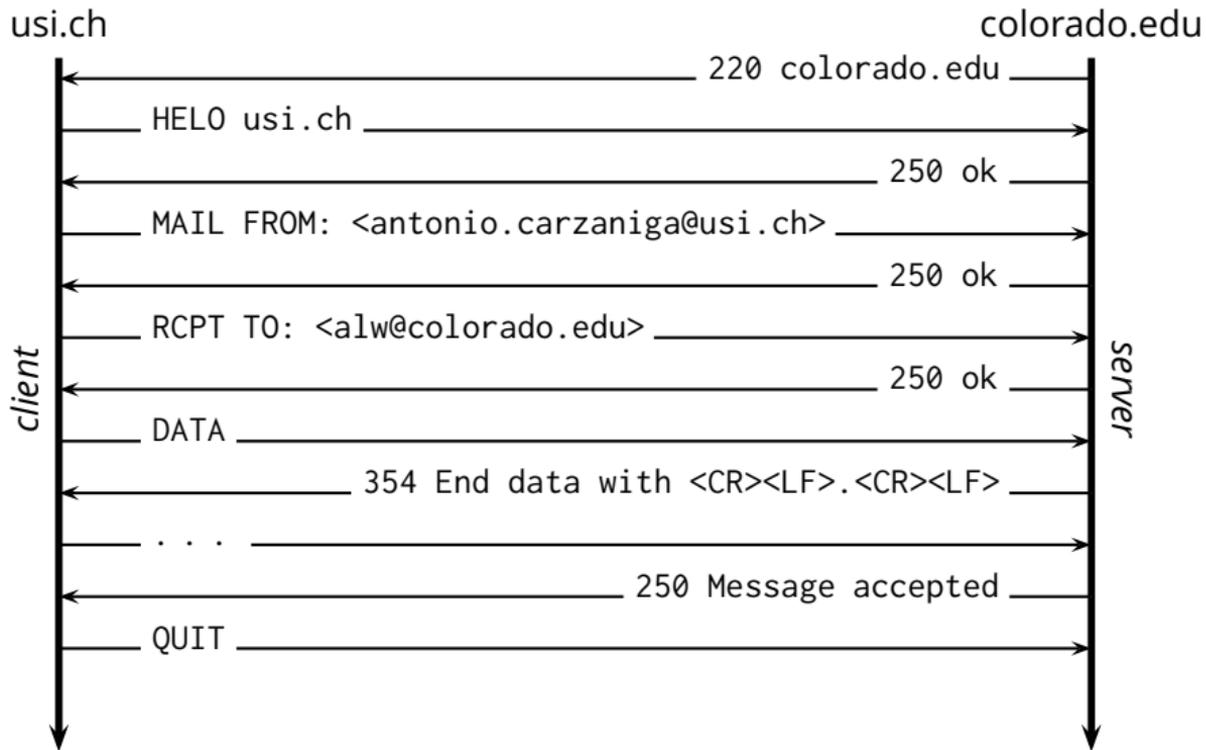
# SMTP Concrete Example



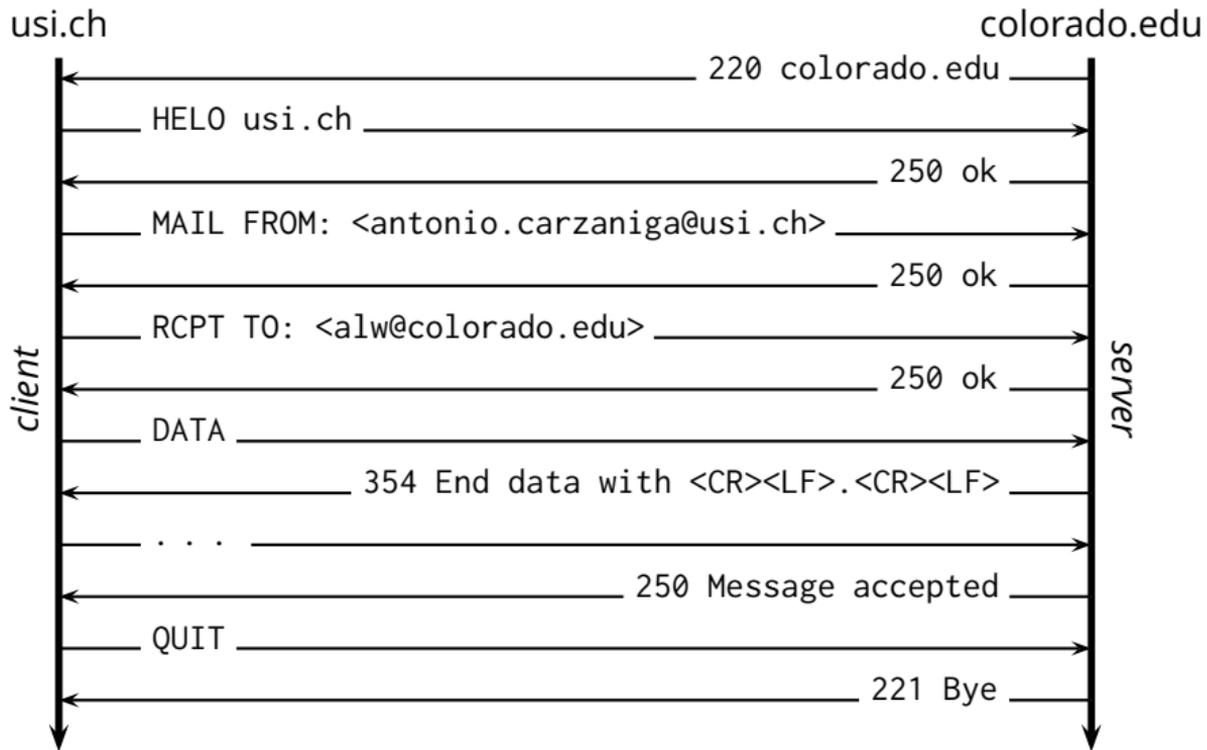
# SMTP Concrete Example



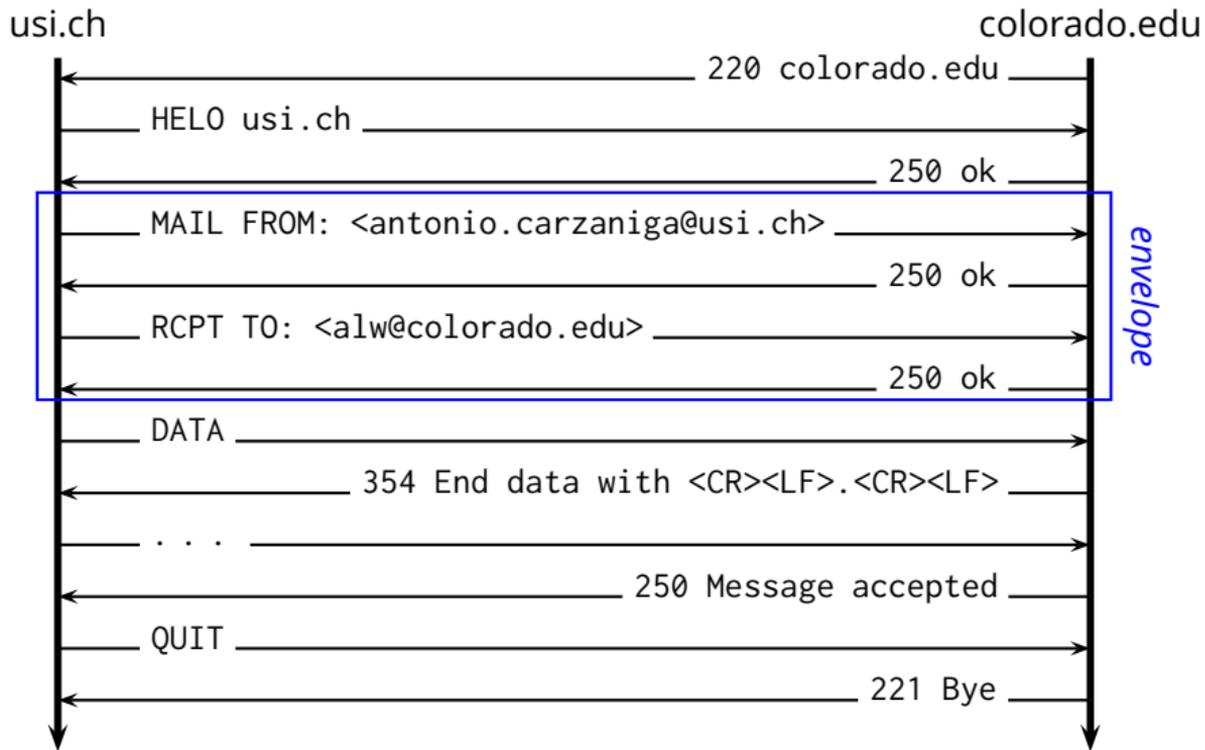
# SMTP Concrete Example



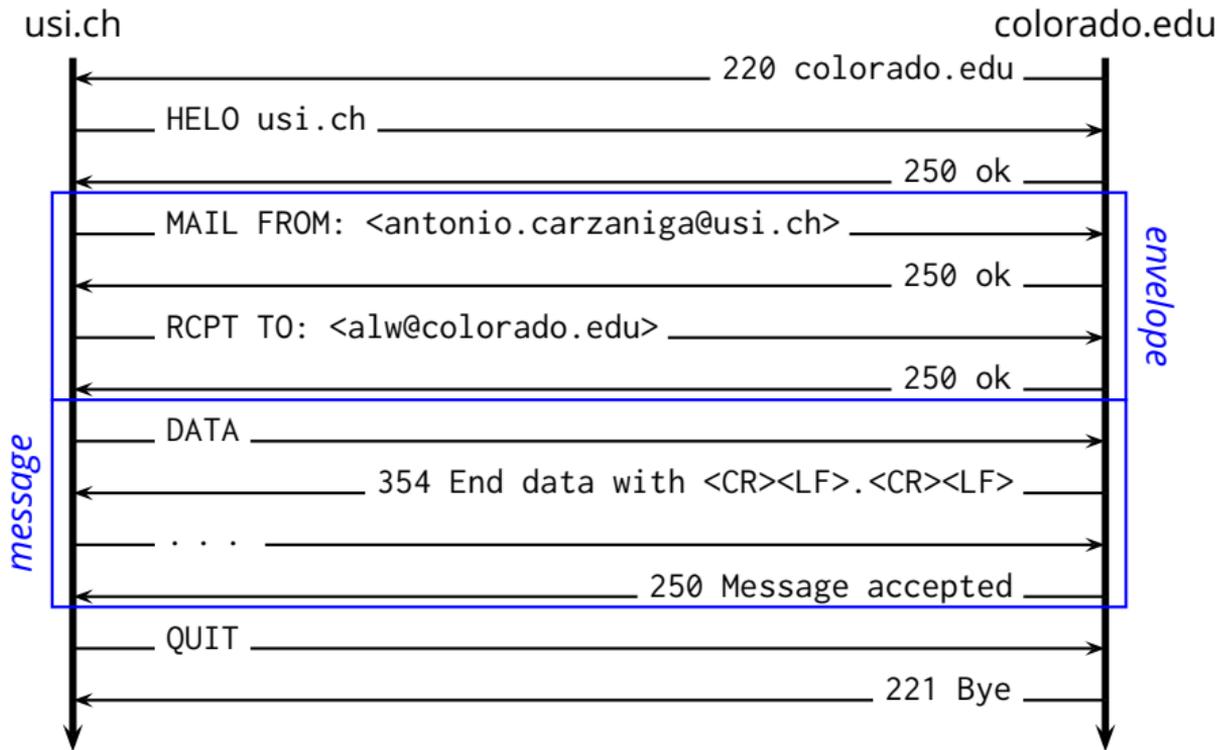
# SMTP Concrete Example



# SMTP Concrete Example



# SMTP Concrete Example





## Message Format

<p>From: antonio.carzaniga@usi.ch Date: Mon, 3 Apr 2005 16:48:22 -0600 (MDT) To: carzanig@cs.colorado.edu Subject: how to send fake e-mail messages</p>	
---	--

<p>Hey Dude, I heard this story about forging messages. Do you know anything about that? ...</p>	
--	--

# Message Format

From: antonio.carzaniga@usi.ch  
Date: Mon, 3 Apr 2005 16:48:22 -0600 (MDT)  
To: carzanig@cs.colorado.edu  
Subject: how to send fake e-mail messages

header  
lines

Hey Dude,  
I heard this story about forging messages.  
Do you know anything about that?  
...

# Message Format

<p>From: antonio.carzaniga@usi.ch Date: Mon, 3 Apr 2005 16:48:22 -0600 (MDT) To: carzanig@cs.colorado.edu Subject: how to send fake e-mail messages</p>	header lines
	empty line
<p>Hey Dude, I heard this story about forging messages. Do you know anything about that? ...</p>	

# Message Format

From: antonio.carzaniga@usi.ch Date: Mon, 3 Apr 2005 16:48:22 -0600 (MDT) To: carzanig@cs.colorado.edu Subject: how to send fake e-mail messages	header lines
	empty line
Hey Dude, I heard this story about forging messages. Do you know anything about that? ...	message body

- SMTP is almost completely oblivious to the content of a message. One exception is the Received: header.
- Every receiving SMTP server must add a Received: header.

- SMTP is almost completely oblivious to the content of a message. One exception is the Received: header.
- Every receiving SMTP server must add a Received: header.

```
Received: from mroe.cs.colorado.edu (mroe-fs.cs.colorado.edu
[128.138.242.197])
  by ser1.cs.colorado.edu (Postfix) with ESMTP id 9AC463D07
  for <carzanig@ser1.cs.colorado.edu>; Mon, 3 Apr 2006 13:39:28 -0600
Received: from max.colorado.edu (max.colorado.edu [128.138.129.234])
  by mroe.cs.colorado.edu (Postfix) with ESMTP id 541C8577A
  for <carzanig@cs.colorado.edu>; Mon, 3 Apr 2006 13:43:59 -0600
Received: from cs.colorado.edu (host132-91.pool82107.interbusiness.it
[82.107.91.132])
  by max.colorado.edu (8.13.6/8.13.6/Hesiod+SSL) with ESMTP id . . .
  for <carzanig@cs.colorado.edu>; Mon, 3 Apr 2006 13:38:12 -0600
```

## Message vs. Envelope

Consider the following SMTP client directives

Consider the following SMTP client directives

1. MAIL FROM: <antonio.carzaniga@usi.ch>

Consider the following SMTP client directives

1. MAIL FROM: <antonio.carzaniga@usi.ch>
2. RCPT TO: <carzanig@cs.colorado.edu>

Consider the following SMTP client directives

1. MAIL FROM: <antonio.carzaniga@usi.ch>
2. RCPT TO: <carzanig@cs.colorado.edu>
3. From: Barak H. Obama <president@whitehouse.gov>  
To: Deserters <all@iobject.org>  
Subject: warning...

You can run, but you can't hide!

Consider the following SMTP client directives

1. MAIL FROM: <antonio.carzaniga@usi.ch>
2. RCPT TO: <carzanig@cs.colorado.edu>
3. From: Barak H. Obama <president@whitehouse.gov>  
To: Deserters <all@iobject.org>  
Subject: warning...

You can run, but you can't hide!

- Anything wrong with this exchange?

## Message vs. Envelope

- The MAIL FROM: and RCPT TO: *SMTP* messages specify ***envelope addresses***

## Message vs. Envelope

- The MAIL FROM: and RCPT TO: *SMTP messages* specify ***envelope addresses***
- From: and To: (and Cc:) *headers within a message* define ***message addresses***

# Message vs. Envelope

- The MAIL FROM: and RCPT TO: *SMTP messages* specify ***envelope addresses***
- From: and To: (and Cc:) *headers within a message* define ***message addresses***
- There are many situations in which it is perfectly legitimate to have envelope addresses that don't match up with the message addresses

# Message vs. Envelope

- The MAIL FROM: and RCPT TO: *SMTP messages* specify ***envelope addresses***
- From: and To: (and Cc:) *headers within a message* define ***message addresses***
- There are many situations in which it is perfectly legitimate to have envelope addresses that don't match up with the message addresses
  - ▶ a message from a mailing list

# Message vs. Envelope

- The MAIL FROM: and RCPT TO: *SMTP messages* specify ***envelope addresses***
- From: and To: (and Cc:) *headers within a message* define ***message addresses***
- There are many situations in which it is perfectly legitimate to have envelope addresses that don't match up with the message addresses
  - ▶ a message from a mailing list
  - ▶ a "blind" copy

# Message vs. Envelope

- The MAIL FROM: and RCPT TO: *SMTP messages* specify ***envelope addresses***
- From: and To: (and Cc:) *headers within a message* define ***message addresses***
- There are many situations in which it is perfectly legitimate to have envelope addresses that don't match up with the message addresses
  - ▶ a message from a mailing list
  - ▶ a "blind" copy
  - ▶ a message to multiple receivers (To: and/or Cc:)

# Message vs. Envelope

- The MAIL FROM: and RCPT TO: *SMTP messages* specify **envelope addresses**
- From: and To: (and Cc:) *headers within a message* define **message addresses**
- There are many situations in which it is perfectly legitimate to have envelope addresses that don't match up with the message addresses
  - ▶ a message from a mailing list
  - ▶ a "blind" copy
  - ▶ a message to multiple receivers (To: and/or Cc:)
  - ▶ a forwarded (or re-sent) message

# Limitations of the Message Format

- The standard message format has some serious limitations

# Limitations of the Message Format

- The standard message format has some serious limitations
  - ▶ 7-bit (text) content

# Limitations of the Message Format

- The standard message format has some serious limitations
  - ▶ 7-bit (text) content
  - ▶ only text

# Limitations of the Message Format

- The standard message format has some serious limitations
  - ▶ 7-bit (text) content
  - ▶ only text
  - ▶ essentially good exclusively for the English language

# Limitations of the Message Format

- The standard message format has some serious limitations
  - ▶ 7-bit (text) content
  - ▶ only text
  - ▶ essentially good exclusively for the English language
  - ▶ monolithic data

# Limitations of the Message Format

- The standard message format has some serious limitations
  - ▶ 7-bit (text) content
  - ▶ only text
  - ▶ essentially good exclusively for the English language
  - ▶ monolithic data
- The ***Multipurpose Internet Mail Extensions (MIME)*** specification (RFC 2045 and RFC 2046) defines *extensions* of the basic message format that support all of the above

- Supports multimedia content

- Supports multimedia content
- Supports different encodings for text (different from ASCII)

- Supports multimedia content
- Supports different encodings for text (different from ASCII)
- Supports messages consisting of multiple parts
  - E.g.,
    - ▶ a message containing some text and an image
    - ▶ a message containing a binary attachment (e.g., an executable program, a document, etc.)
    - ▶ a message containing another message
    - ▶ a message containing some Italian text plus another message containing German text
    - ▶ a message containing another message, containing another message, ...

The primary mechanism used by MIME extensions consists of added *MIME headers*

- `MIME-Version: 1.0`  
signals a user agent that this message uses MIME extensions, version 1.0

The primary mechanism used by MIME extensions consists of added *MIME headers*

- `MIME-Version: 1.0`  
signals a user agent that this message uses MIME extensions, version 1.0
- `Content-Type: ...`  
specifies the content of the message. Valid types include:

The primary mechanism used by MIME extensions consists of added *MIME headers*

- `MIME-Version: 1.0`  
signals a user agent that this message uses MIME extensions, version 1.0
- `Content-Type: ...`  
specifies the content of the message. Valid types include:
  - ▶ `text/plain` — this is a normal ASCII message

The primary mechanism used by MIME extensions consists of added *MIME headers*

- `MIME-Version: 1.0`  
signals a user agent that this message uses MIME extensions, version 1.0
  
- `Content-Type: ...`  
specifies the content of the message. Valid types include:
  - ▶ `text/plain` — this is a normal ASCII message
  - ▶ `text/html` — this is an HTML-formatted message

The primary mechanism used by MIME extensions consists of added *MIME headers*

- `MIME-Version: 1.0`  
signals a user agent that this message uses MIME extensions, version 1.0
  
- `Content-Type: ...`  
specifies the content of the message. Valid types include:
  - ▶ `text/plain` — this is a normal ASCII message
  - ▶ `text/html` — this is an HTML-formatted message
  - ▶ `image/jpeg` — this message contains (only) an image file

The primary mechanism used by MIME extensions consists of added *MIME headers*

- `MIME-Version: 1.0`  
signals a user agent that this message uses MIME extensions, version 1.0
  
- `Content-Type: ...`  
specifies the content of the message. Valid types include:
  - ▶ `text/plain` — this is a normal ASCII message
  - ▶ `text/html` — this is an HTML-formatted message
  - ▶ `image/jpeg` — this message contains (only) an image file
  - ▶ `multipart/mixed` — this message consists of multiple parts

The primary mechanism used by MIME extensions consists of added *MIME headers*

- **MIME-Version: 1.0**

signals a user agent that this message uses MIME extensions, version 1.0

- **Content-Type: ...**

specifies the content of the message. Valid types include:

- ▶ `text/plain` — this is a normal ASCII message
- ▶ `text/html` — this is an HTML-formatted message
- ▶ `image/jpeg` — this message contains (only) an image file
- ▶ `multipart/mixed` — this message consists of multiple parts
- ▶ ...

An Internet mail message must contain only 7-bit characters, therefore any content that does not fit the 7-bit (ASCII) character set must be *encoded*

An Internet mail message must contain only 7-bit characters, therefore any content that does not fit the 7-bit (ASCII) character set must be *encoded*

- Content-Transfer-Encoding:  
defines the encoding for the message content (or a part thereof). Common values are:

An Internet mail message must contain only 7-bit characters, therefore any content that does not fit the 7-bit (ASCII) character set must be *encoded*

- Content-Transfer-Encoding:  
defines the encoding for the message content (or a part thereof). Common values are:
  - ▶ base64

An Internet mail message must contain only 7-bit characters, therefore any content that does not fit the 7-bit (ASCII) character set must be *encoded*

- Content-Transfer-Encoding:  
defines the encoding for the message content (or a part thereof). Common values are:
  - ▶ base64
  - ▶ Quoted-Printable

- Several functionalities of the MIME extensions depend on the ability to carry multiple “parts” within the same message
  - ▶ e.g., to implement “attachments”

# MIME Multipart Messages

- Several functionalities of the MIME extensions depend on the ability to carry multiple “parts” within the same message
  - ▶ e.g., to implement “attachments”
- Content-Type: multipart/mixed;  
boundary="----\_=\_NextPart\_001\_01C539DF.6607A632"

# MIME Multipart Messages

- Several functionalities of the MIME extensions depend on the ability to carry multiple “parts” within the same message
  - ▶ e.g., to implement “attachments”
- Content-Type: multipart/mixed;  
boundary="----\_=\_NextPart\_001\_01C539DF.6607A632"
- The message consists of a list of *parts* (e.g., the main message text and an attached document)
  1. parts are separated by a *boundary line*
  2. parts are introduced (right after the separator line) by a set of specific headers that define that part
  3. the list is terminated by a terminator line

- Several functionalities of the MIME extensions depend on the ability to carry multiple “parts” within the same message
  - ▶ e.g., to implement “attachments”
- Content-Type: multipart/mixed;  
boundary="----\_=\_NextPart\_001\_01C539DF.6607A632"
- The message consists of a list of *parts* (e.g., the main message text and an attached document)
  1. parts are separated by a *boundary line*
  2. parts are introduced (right after the separator line) by a set of specific headers that define that part
  3. the list is terminated by a terminator line
- This format is naturally recursive

# MIME Example

MIME-Version: 1.0  
Content-Type: multipart/mixed; boundary="gJ7ppttFJL"  
Content-Transfer-Encoding: 7bit  
Date: Fri, 15 Apr 2005 15:24:31 +0200  
From: Antonio Carzaniga <antonio.carzaniga@usi.ch>  
To: Antonio Carzaniga <carzanig@cs.colorado.edu.ch>  
Subject: Immagini e testo

--gJ7ppttFJL  
Content-Type: text/plain; charset=iso-8859-15  
Content-Description: message body text  
Content-Transfer-Encoding: quoted-printable

questo =E8 un esempio di un messaggio che usa il formato MIME.

-A

--gJ7ppttFJL  
Content-Type: image/png  
Content-Disposition: inline;  
filename="anto.png"  
Content-Transfer-Encoding: base64

iVBORw0KGgoAAAANSUHEUgAAAMgAAADICAIAAAAi0jnJAAAACXBIWXMAAAIxAAACMQF3BQBZAAAA  
B3RJTUUH1AwdCiYGBdI1HQAAIABJREFUeNqEu1mMJteVJnaWe2P5l/wz88+tcqusnRT3fRMpiqOW  
Wt0De9w9bczAGAM2jPHAfVOLDcNvfVODAW8YeOC1DbeN7ullZnpa6kUttZqiKIoiWawiWaxibVIZ  
uVTu+a+x3Hv08UOq7cEAXsRTRCAiXuLDud928T/7L/6hoDZazYnu8kxnbqLdbbab7Xa71ZxoNZt5

. . .  
Ts2sSh8efiVxP3z3GtgR5/9Wz/8DNJKaidrd/8MAAAAASUVORK5CYII=

--gJ7ppttFJL--