Lecture 24

- Section 16.4 Case Study: Network Chat Room

Network chat room

Let's build software to support a network chat room.
It is a client-server model of communication.

Server: supports the chat room (more permanent connection, called *persistent connection*, than in previous examples)

Server cannot be solely devoted to handling only one connection, because it will be unresponsive to other connection attempts

Client: connects to the server and participates in a chat

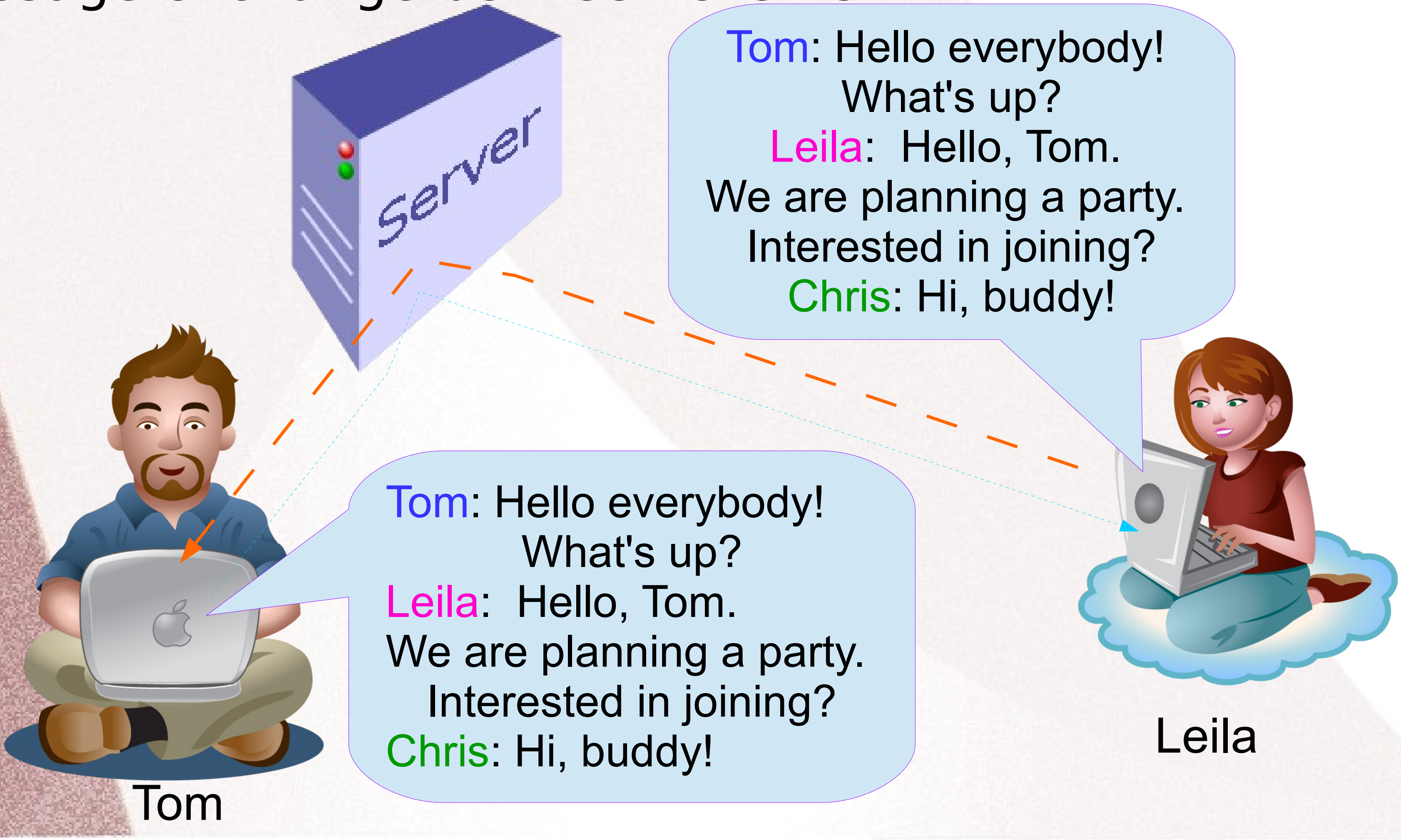
Client will be multi-tasking as well: monitoring the keyboard for user entering any commands and monitoring the socket to listen for transmissions from the server

We will have to develop a protocol for such communication.

Network chat room – communication protocol

While implementing a chat room we need to differentiate between:

- client-server software message/requests exchange, and
- message exchange between clients



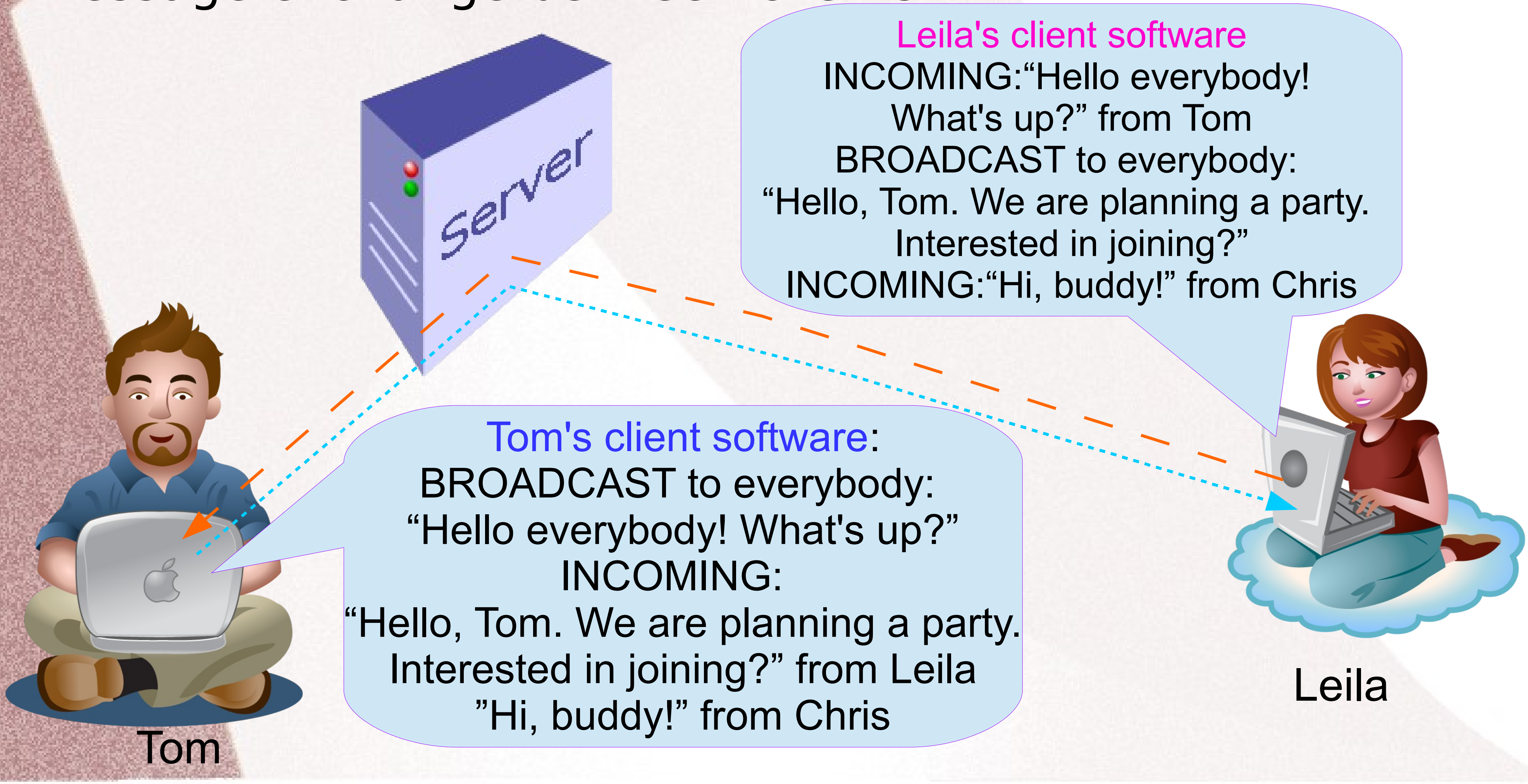
Tom: Hello everybody! What's up?
Leila: Hello, Tom. We are planning a party. Interested in joining?
Chris: Hi, buddy!

Tom: Hello everybody! What's up?
Leila: Hello, Tom. We are planning a party. Interested in joining?
Chris: Hi, buddy!

Network chat room – communication protocol

While implementing a chat room we need to differentiate between:

- client-server software message/requests exchange, and
- message exchange between clients



Leila's client software
INCOMING:“Hello everybody! What's up?” from Tom
BROADCAST to everybody:
“Hello, Tom. We are planning a party. Interested in joining?”
INCOMING:“Hi, buddy!” from Chris

Tom's client software:
BROADCAST to everybody:
“Hello everybody! What's up?”
INCOMING:
“Hello, Tom. We are planning a party. Interested in joining?” from Leila
”Hi, buddy!” from Chris

Network chat room – communication protocol

While implementing a chat room we need to differentiate between:

- client-server software message/requests exchange, and
- message exchange between clients

Message type	Format
join the room using given identity	`''ADD %s\n'' % screenName`
broadcast the message to everybody	`''MESSAGE %s\n'' % content`
send a private message	`''PRIVATE %s\n %s\n'' % (recipient, content)`
quit the chat room	`''QUIT\n''`

communication protocol for messages from client to server

Network chat room – communication protocol

While implementing a chat room we need to differentiate between:

- client-server software message/requests exchange, and
- message exchange between clients

Message type	Format
new user has joined	`''NEW %s\n'' % screenName`
message was broadcast message to everybody	`''MESSAGE %s\n %s\n'' % (sender,content)`
private message was sent to user	`''PRIVATE %s\n %s\n'' % (recipient, content)`
someone has left the room	`''LEFT %s\n'' % screenName`
acknowledges request to leave	`''GOODBYE\n''`

communication protocol for messages from server to client

A try on chat room

Download the program chatclient.py and run it.

Type in your name, press enter several times:
you will see info about you and other joining users.

Say hello to everybody (broadcast message)

Assume you are friends and are discussing ideas on how to spend winter break together. It may be the case that you decide to split into different groups (by interest).

Don't forget that you can send private messages.

End: as soon as you are in one of the groups (or maybe all of you decided to go to the same place) your task is accomplished.