SER 321 B Session

SI Session

Sunday, November 5th 2023

7:00 - 8:00 pm MST

Agenda

- Threading Pitfalls
  - Critical Sections
  - Traffic Analogy
- Threading your Server
- Serialization

SI Session Expectations

Thanks for coming to the **SER 321** SI session. We have a packed agenda and we are going to try to get through as many of our planned example problems as possible. This session will be recorded and shared with others.

- If after this you want to see additional examples, please visit the drop-in tutoring center.
- We will post the link in the chat now and at the end of the session.
  - tutoring.asu.edu
- Please keep in mind we are recording this session and it will be made available for you to review 24-48 hours after this session concludes.
- Finally, please be respectful to each other during the session.

Interact with us:

Zoom Features

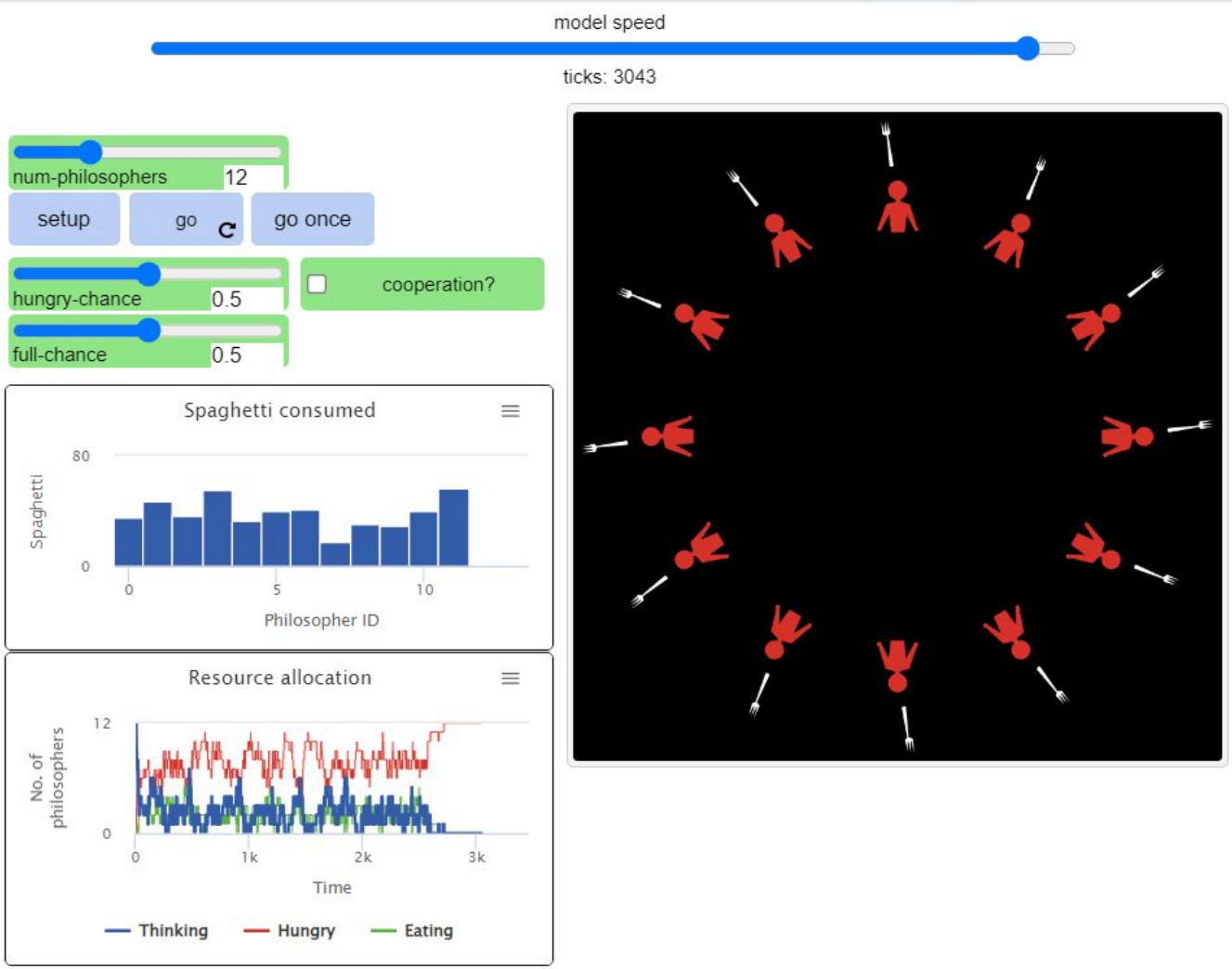
Zoom Chat

- Use the chat feature to interact with the presenter and respond to presenter's questions.
- Annotations are encouraged

SER 321

Threading Pitfalls!

Dining Philosophers



Austin Walter's Traffic Comparison

SER 321

Threading Pitfalls

- Race Condition

More than one thread accesses a single resource at one time

- Starvation

One thread never gets access to the resource it needs

- Deadlock

A thread is only able to acquire access to part of its resources

Austin Walter's Traffic Comparison

SER 321

Threading Pitfalls

- Race Condition → Crash

More than one thread accesses a single resource at one time

- Starvation

One thread never gets access to the resource it needs

- Deadlock

A thread is only able to acquire access to part of its resources

Austin Walter's Traffic Comparison

SER 321

Threading Pitfalls

- Race Condition → Crash

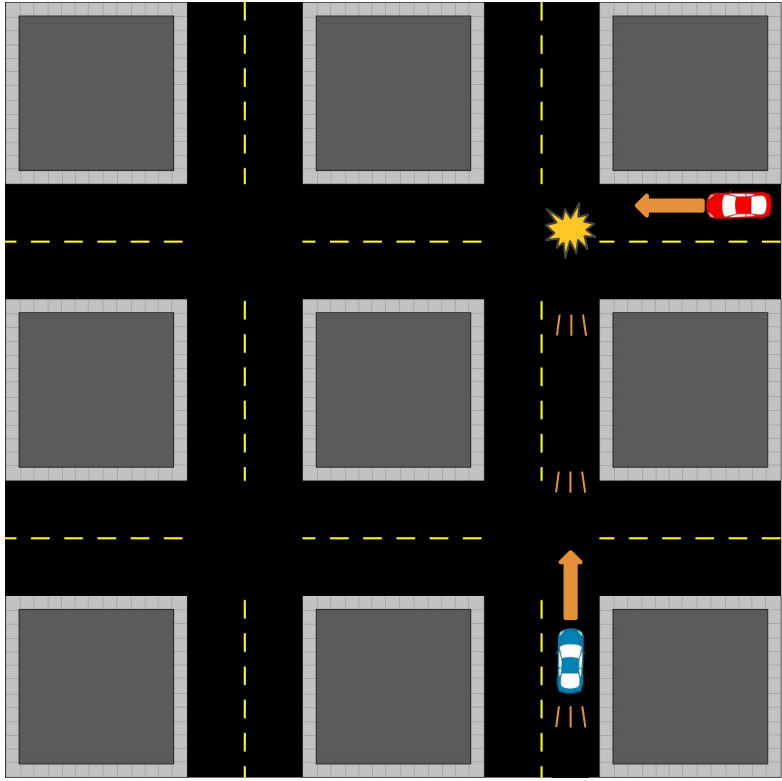
More than one thread accesses a single resource at one time

- Starvation

One thread never gets access to the resource it needs

- Deadlock

A thread is only able to acquire access to part of its resources



SER 321

Threading Pitfalls

- Race Condition → Crash

More than one thread accesses a single resource at one time

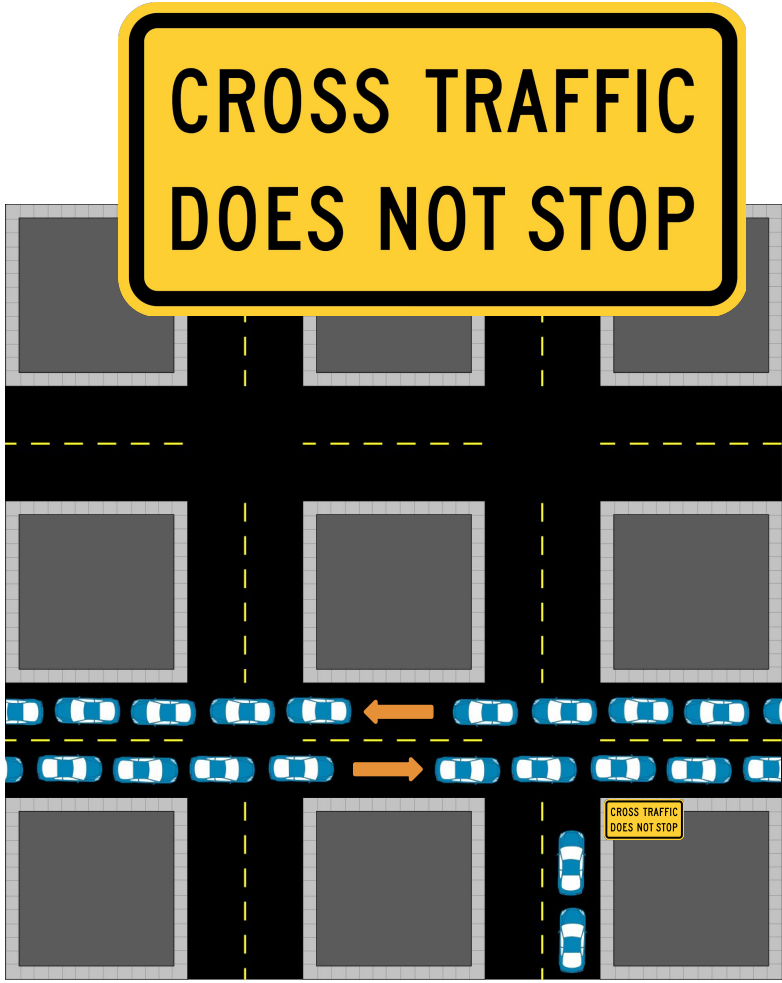
- Starvation → Cross Traffic

One thread never gets access to the resource it needs

- Deadlock

A thread is only able to acquire access to part of its resources

Austin Walter's Traffic Comparison



Austin Walter's Traffic Comparison

SER 321

Threading Pitfalls

- Race Condition → Crash

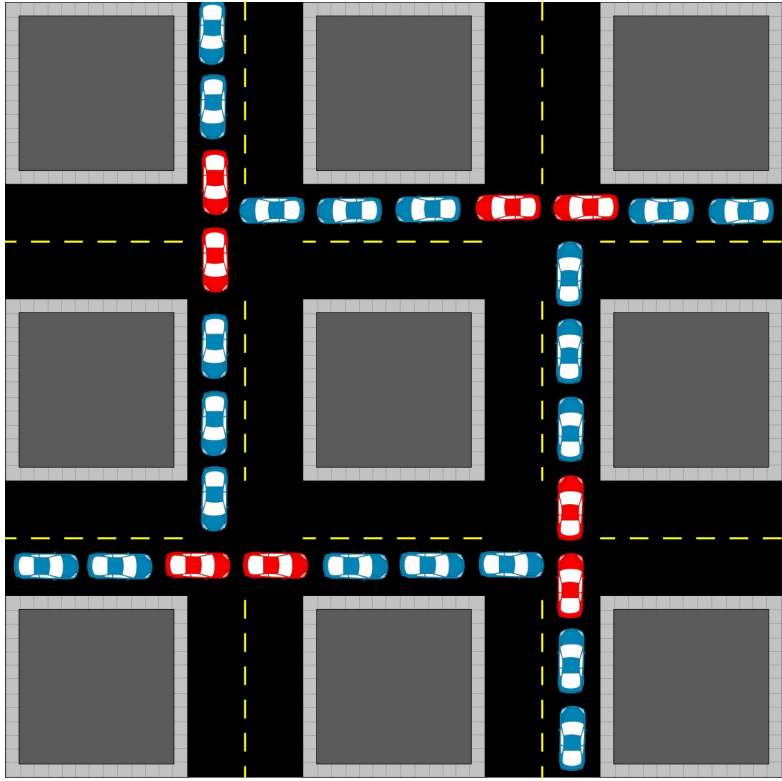
More than one thread accesses a single resource at one time

- Starvation → Cross Traffic

One thread never gets access to the resource it needs

- Deadlock → Gridlock

A thread is only able to acquire access to part of its resources



SER 321

Threading your Server

Standard Server Steps

1. Define Params
2. Make socket
3. Establish connection
4. Handle Connection
5. Close Connection

Okay so how do we thread this?

```java
public static void main (String args[]) {
  Socket sock;
  try {
    //open socket
    ServerSocket serv = new ServerSocket( port: 8888); // create server socket on port 8888
    System.out.println("Server ready for 3 connections");
    // only does three connections then closes
    // NOTE: SINGLE-THREADED, only one connection at a time
    for (int rep = 0; rep < 3; rep++){
      System.out.println("Server waiting for a connection");
      sock = serv.accept(); // blocking wait
      // setup the object reading channel
      ObjectInputStream in = new ObjectInputStream(sock.getInputStream());

      // read in one object, the message. we know a string was written only by knowing what the client sent.
      // must cast the object from Object to desired type to be useful
      String s = (String) in.readObject();
      System.out.println("Received the String "+s);
      // read in the number, we know it's an integer because that's the second thing sent by the client.
      Integer i = (Integer) in.readObject();
      System.out.println("Received the Integer "+ i);

      // generate an output
      // get output channel
      OutputStream out = sock.getOutputStream();
      // create an object output writer (Java only)
      ObjectOutputStream os = new ObjectOutputStream(out);
      // write the whole message
      os.writeObject("Got it!");
      // make sure it wrote and doesn't get cached in a buffer
      os.flush();
    }
  } catch(Exception e) {e.printStackTrace();}
}
```

SockServer from JavaSimpleSock2 in examples Repo

SER 321

Threading your Server

Threaded Server Steps

0. `Server extends Thread`
1. Define Params
2. Make socket
3. Establish connection
4.
5. Handle Connection
6. Close Connection

```java
public class ThreadedSockServer extends Thread {
```

SockServer in JavaSimpleSock2

```java
Socket sock;
try {
  //open socket
  ServerSocket serv = new ServerSocket( port: 8888); // create server socket on port 8888
  System.out.println("Server ready for 3 connections");
  // only does three connections then closes
  // NOTE: SINGLE-THREADED, only one connection at a time
  for (int rep = 0; rep < 3; rep++){
    System.out.println("Server waiting for a connection");
    sock = serv.accept(); // blocking wait
    // setup the object reading channel
    ObjectInputStream in = new ObjectInputStream(sock.getInputStream());
```

```java
ServerSocket serv = new ServerSocket(portNo);
while (true) {
  System.out.println("Threaded server waiting for connects on port " + portNo);
  sock = serv.accept();
  System.out.println("Threaded server connected to client-" + id);
```

JavaThreadedSock in Sockets

SER 321

Threading your Server

Threaded Server Steps

0. `Server extends Thread`
1. Define Params
2. Make socket
3. Establish connection
4. Start Thread!
5. Handle Connection
6. Close Connection

```
public class ThreadedSockServer extends Thread {
```

```
Socket sock
try {
  //open so
  ServerSoc
  System.ou
  // only does three connections then closes
  // NOTE: SINGLE-THREADED, only one connection at a time
  for (int rep = 0; rep < 3; rep++){
    System.out.println("Server waiting for a connection");
    sock = serv.accept(); // blocking wait
    // setup the object reading channel
    ObjectInputStream in = new ObjectInputStream(sock.getInputStream());
```

```
public ThreadedSockServer(Socket sock, int id) {
    this.conn = sock;
    this.id = id;
}
```

t 8888

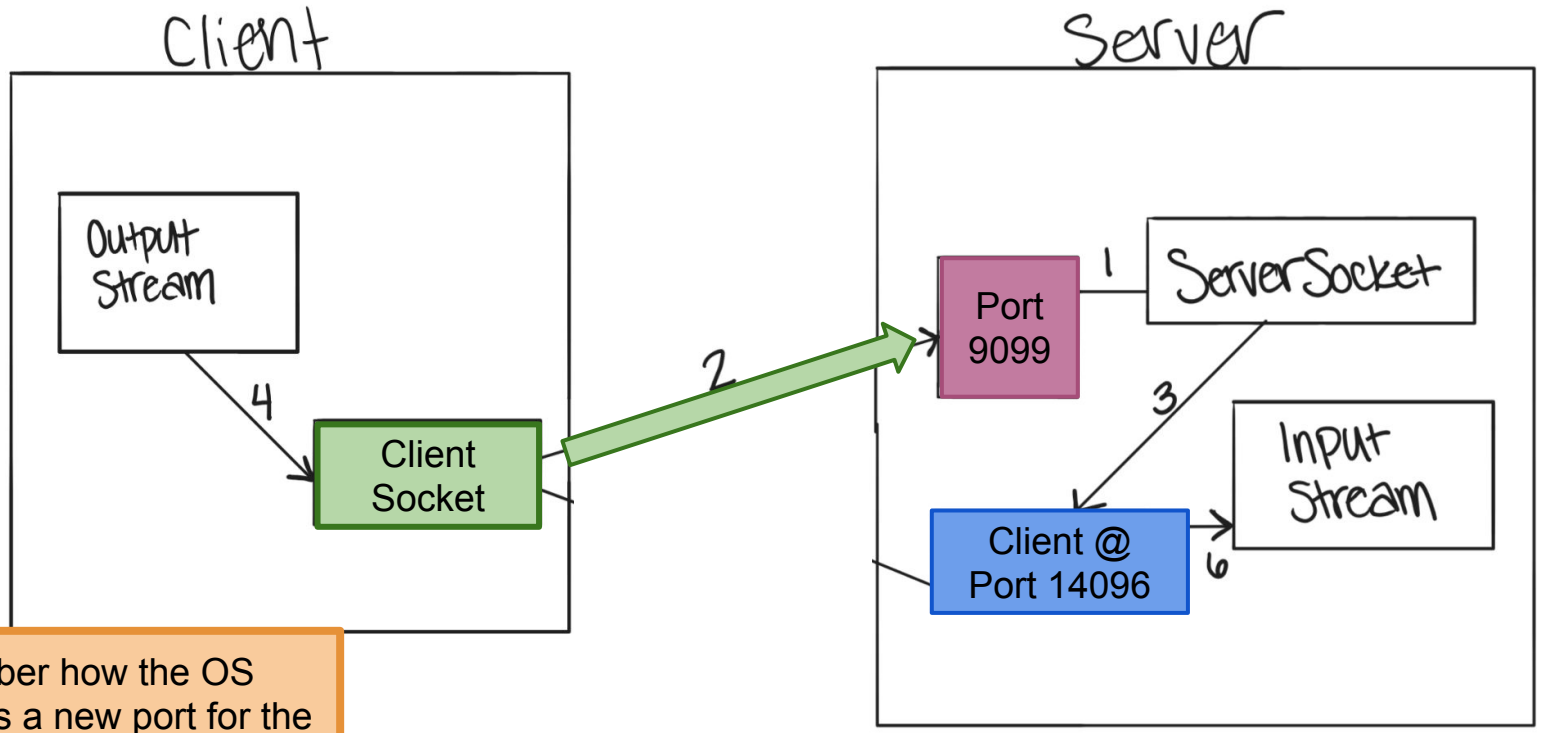
```
ServerSocket serv = new ServerSocket(portNo);
while (true) {
  System.out.println("Threaded server waiting for connects on port " + portNo);
  sock = serv.accept();
  System.out.println("Threaded server connected to client-" + id);
  // create thread
  ThreadedSockServer myServerThread = new ThreadedSockServer(sock, id++);
  // run thread and don't care about managing it
  myServerThread.start();
}
```

```
public void run() {
```

JavaThreadedSock in Sockets

SER 321
Client Socket

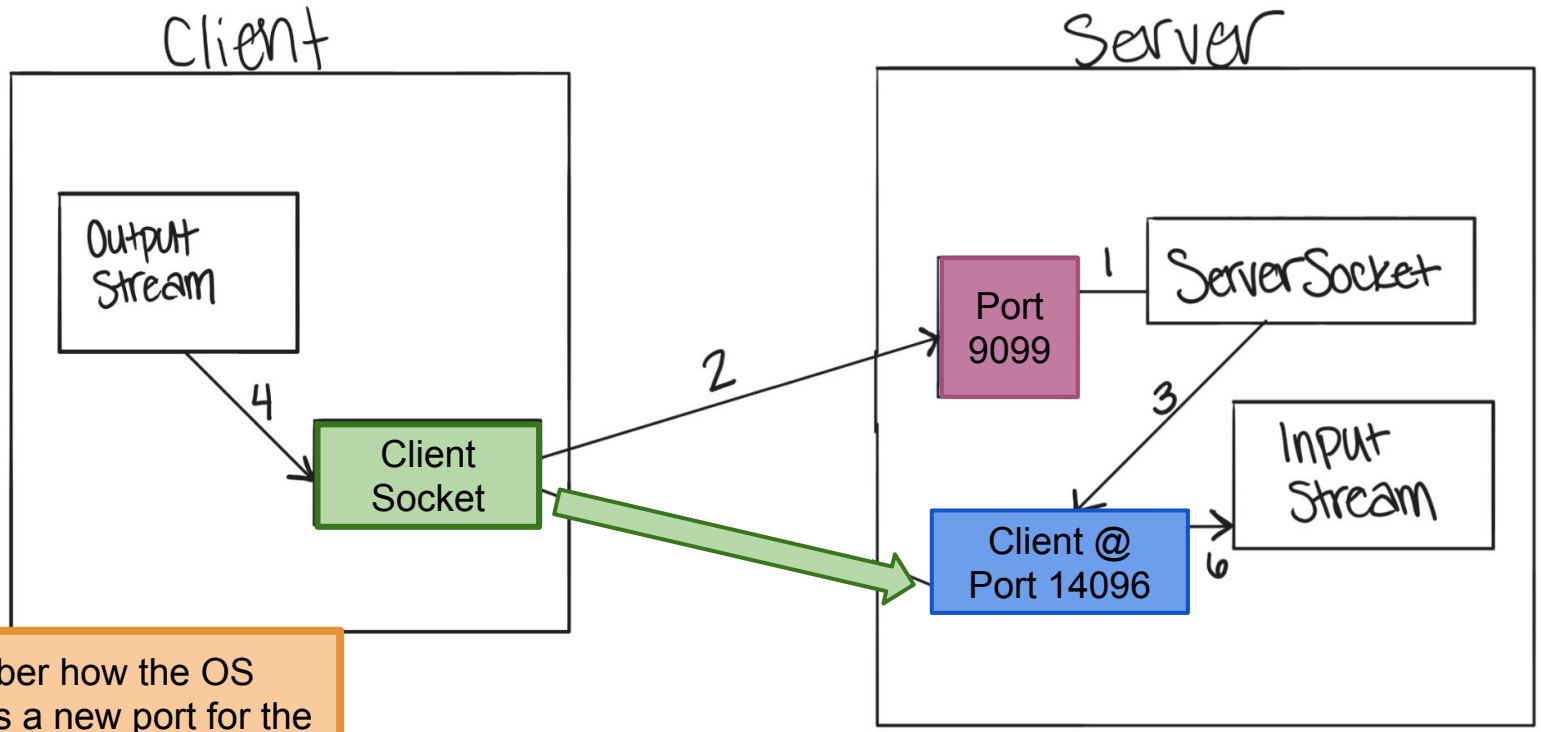
```
Values of the Client Socket Object after Connection:
Inet Address: /127.0.0.1
Local Address: /127.0.0.1
Local Port: 9099
Allocated Client Socket (Remote Port): 14096
```



Remember how the OS allocates a new port for the client?

SER 321
Client Socket

```
Values of the Client Socket Object after Connection:
Inet Address: /127.0.0.1
Local Address: /127.0.0.1
Local Port: 9099
Allocated Client Socket (Remote Port): 14096
```

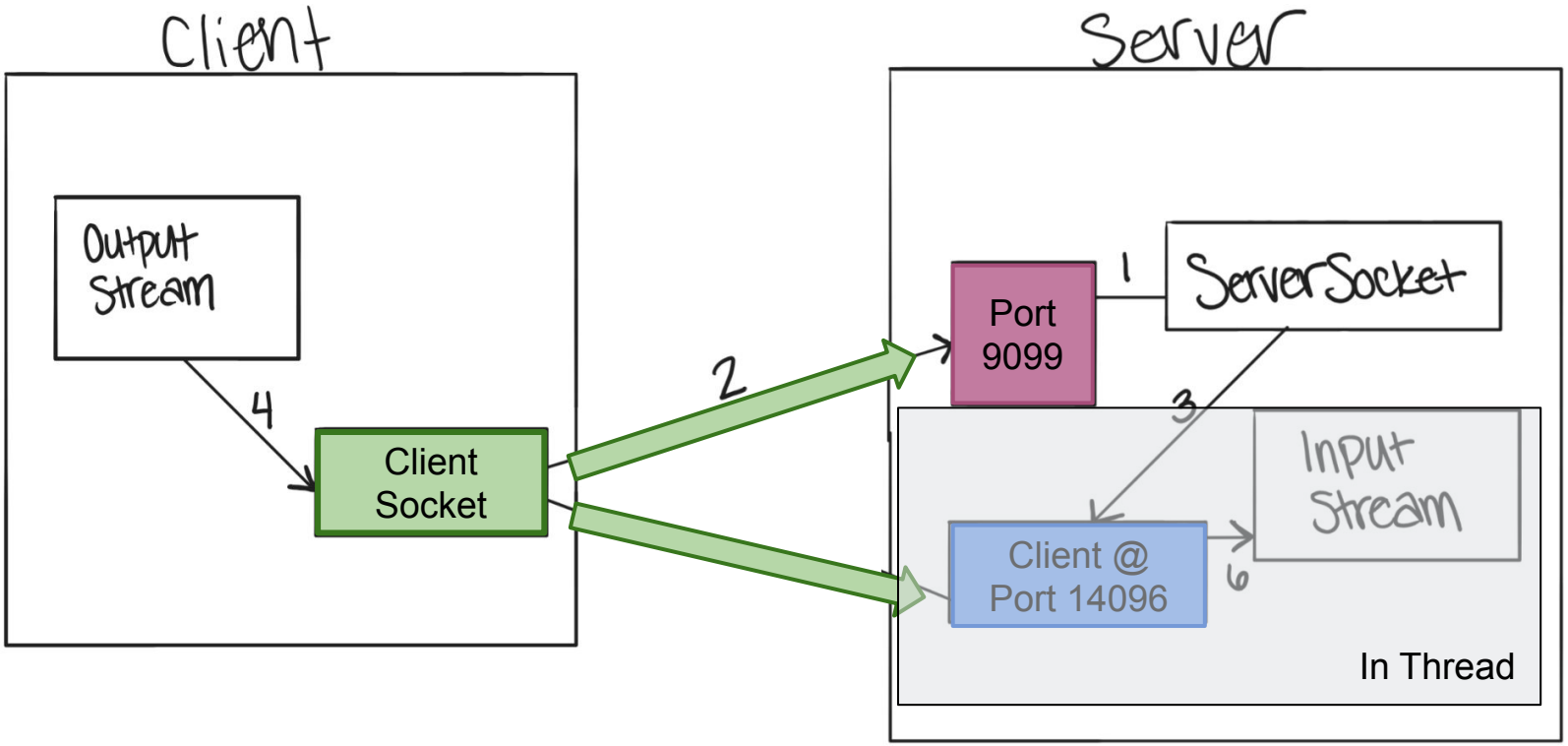


Remember how the OS allocates a new port for the client?

SER 321

Threaded Server Communication

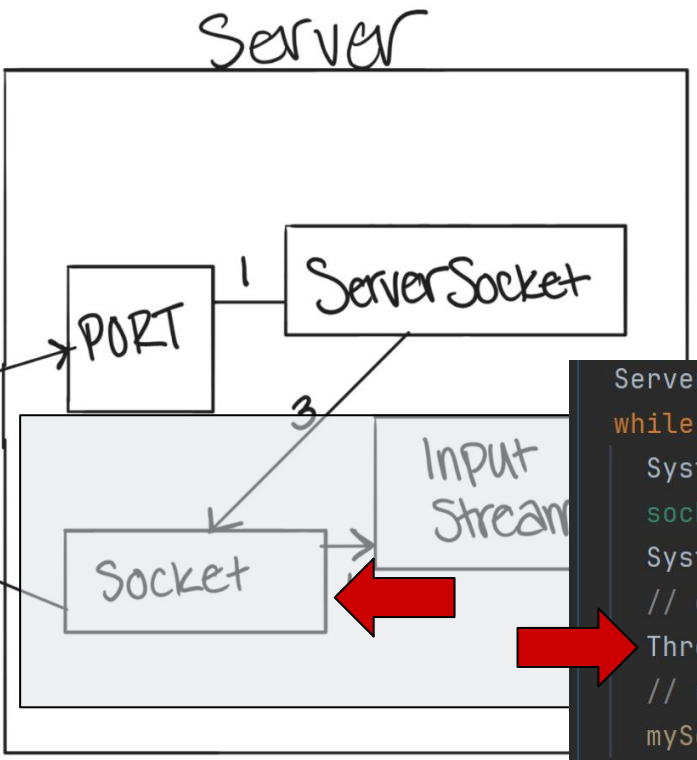
How do we do that in a threaded server?



SER 321

Threaded Server Communication

So how does the communication get passed from the main thread into the Client Thread?




```java
sock = serv.accept();
```

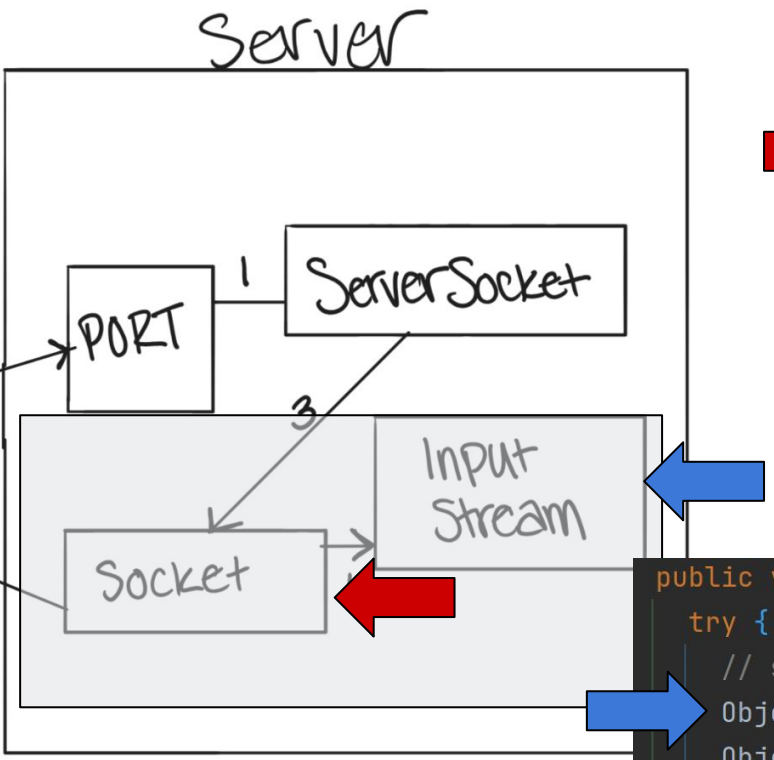
```java
public ThreadedSockServer(Socket sock, int id) {
  this.conn = sock;
  this.id = id;
}
```

```java
ServerSocket serv = new ServerSocket(portNo);
while (true) {
  System.out.println("Threaded server waiting for connects on port " + portNo);
  sock = serv.accept();
  System.out.println("Threaded server connected to client-" + id);
  // create thread
  ThreadedSockServer myServerThread = new ThreadedSockServer(sock, id++);
  // run thread and don't care about managing it
  myServerThread.start();
}
```

SER 321

Threaded Server Communication

So how does the communication get passed from the main thread into the Client Thread?




```
sock = serv.accept();
```

```
public ThreadedSockServer(Socket sock, int id) {
  this.conn = sock;
  this.id = id;
}
```

Now we can just handle the connection as we have been!

```
public void run() {
  try {
    // setup read/write channels for connection
    ObjectInputStream in = new ObjectInputStream(conn.getInputStream());
    ObjectOutputStream out = new ObjectOutputStream(conn.getOutputStream());
```

Questions?

Survey:

https://bit.ly/asn_survey

19

Upcoming Events

SI Sessions:

- Monday, November 6th 2023 at 4:00 pm MST
- Thursday, November 9th 2023 at 7:00 pm MST
- Sunday, November 12th 2023 at 7:00 pm MST

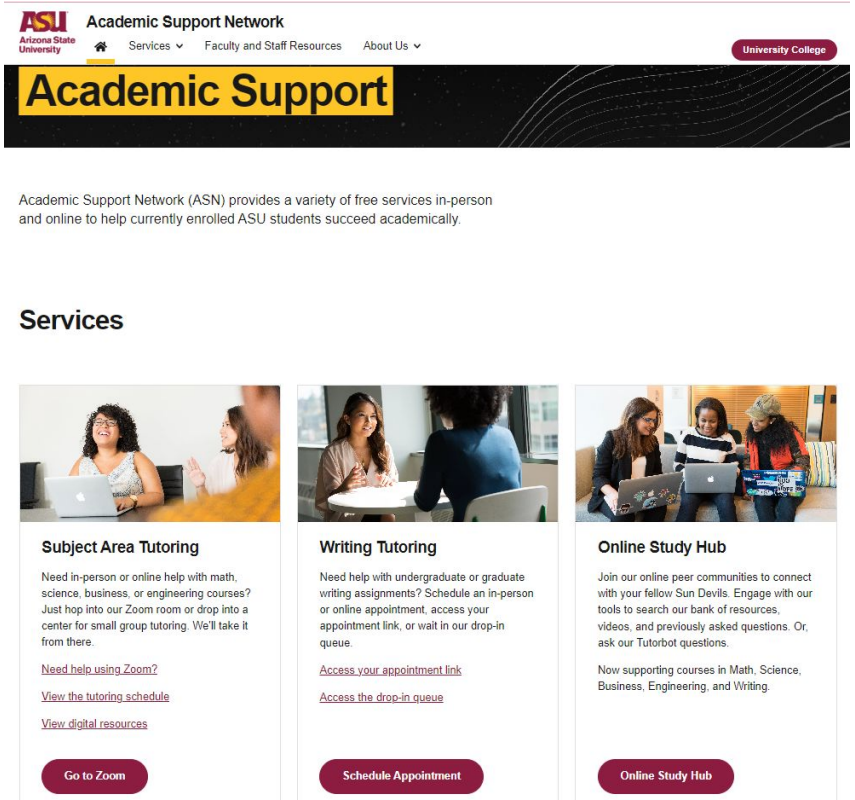
Review Sessions:

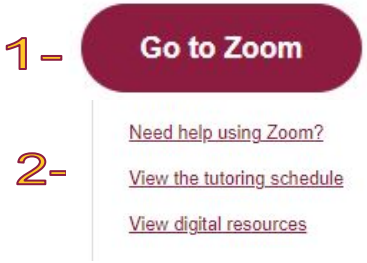
- Planning to post a poll this week!

More Questions?

Check out our other resources!

tutoring.asu.edu



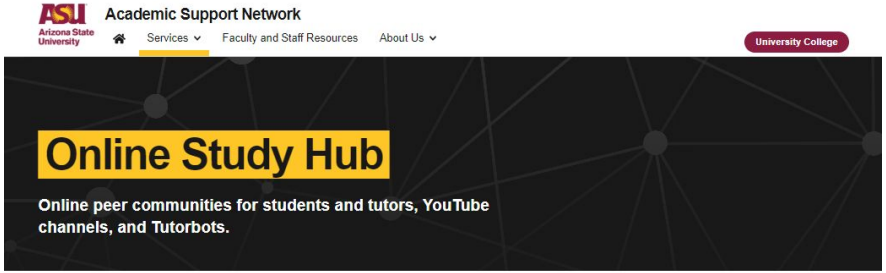


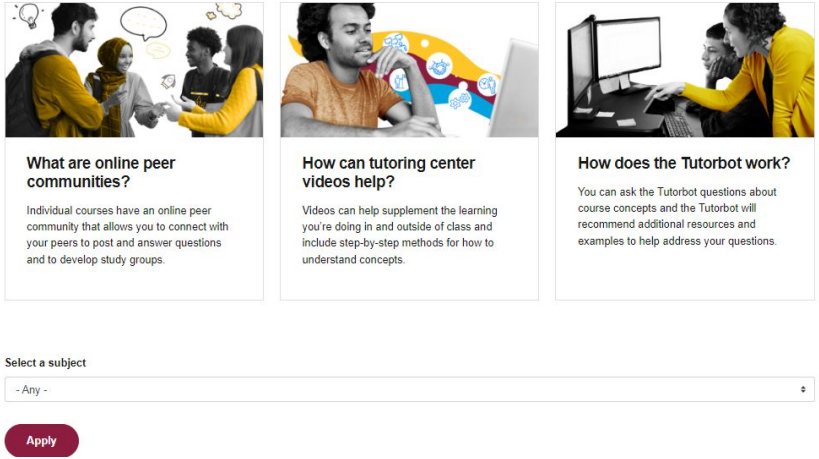
1. **Click on 'Go to Zoom' to log onto our Online Tutoring Center.**
2. **Click on 'View the tutoring schedule' to see when tutors are available for specific courses.**

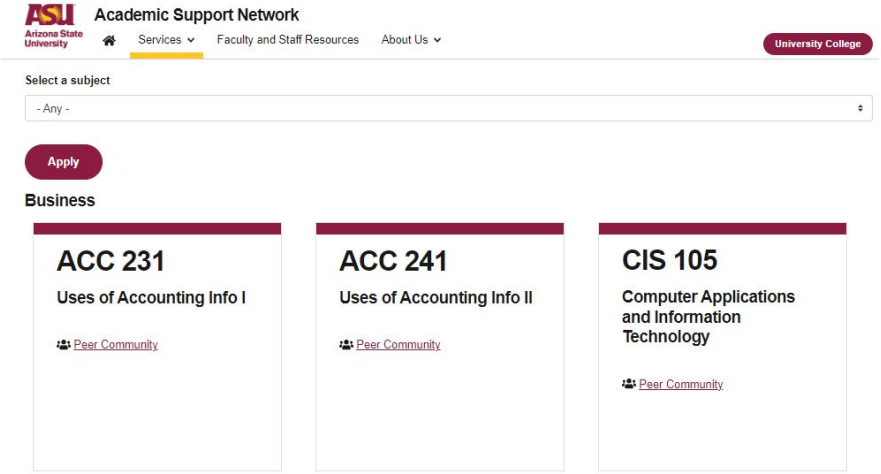
More Questions?

Check out our other resources!

tutoring.asu.edu/online-study-hub







Don't forget to check out the Online Study Hub for additional resources!

Additional Resources

CoureRepo

org.json API Docs

JSON Helper

Dining Philosophers Interactive

Austin Walter’s Traffic Comparison