SER 321 A Session

SI Session

Sunday September 24th 2023

6:00 - 7:00 pm MST

Agenda

- Sockets & Threading Review
- This Assignment
  - Client vs. Server
  - Communication Structure
- Revisit RAFT

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 A Session

Socket Server - No Threads

Make Socket

Wait for connections

Handle the connection

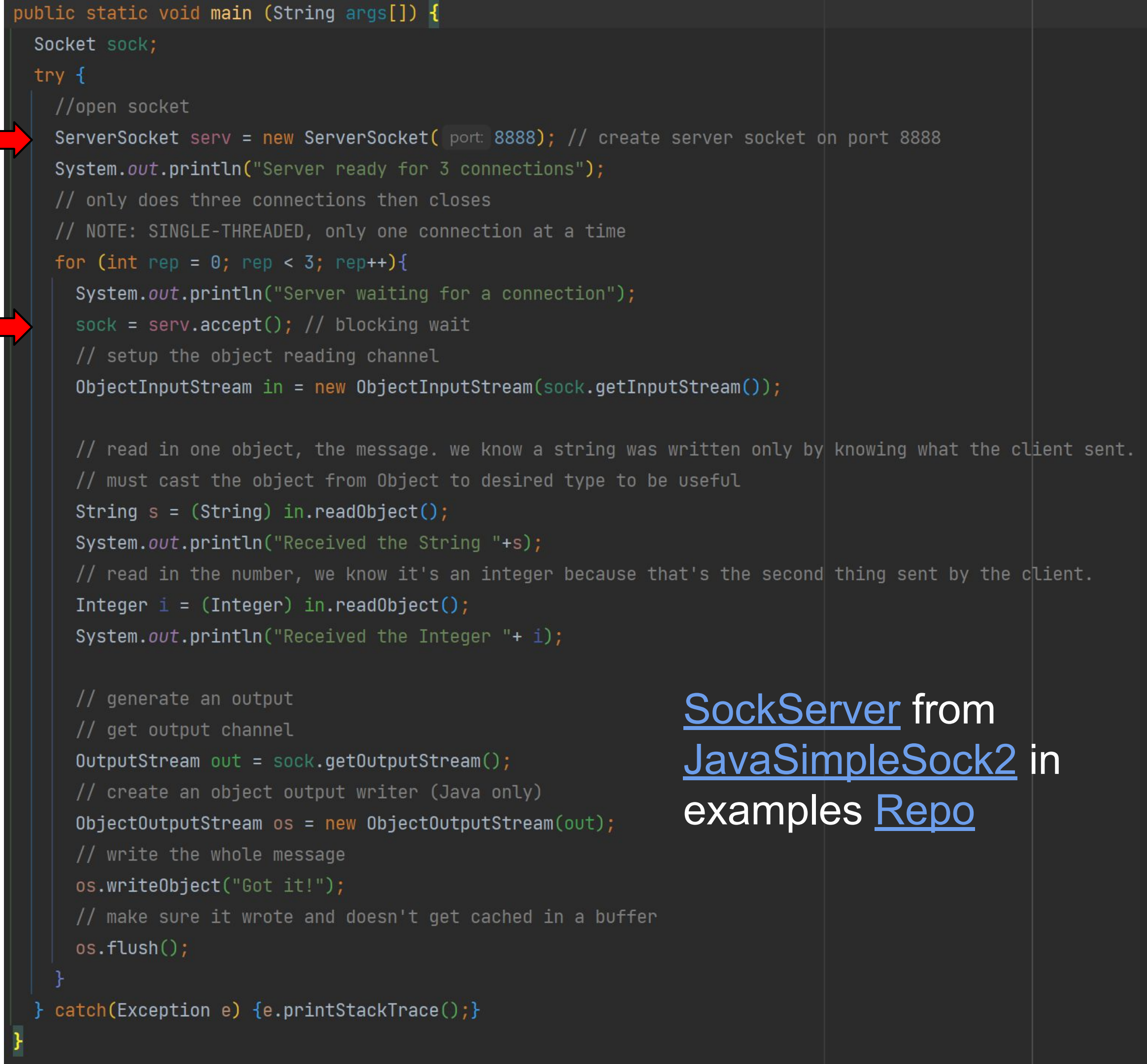
Perform the task

Clean up - what is that again?

in.close();

out.close();

sock.close();



```
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 A Session

Threading your Server

Make Socket

Wait for connections

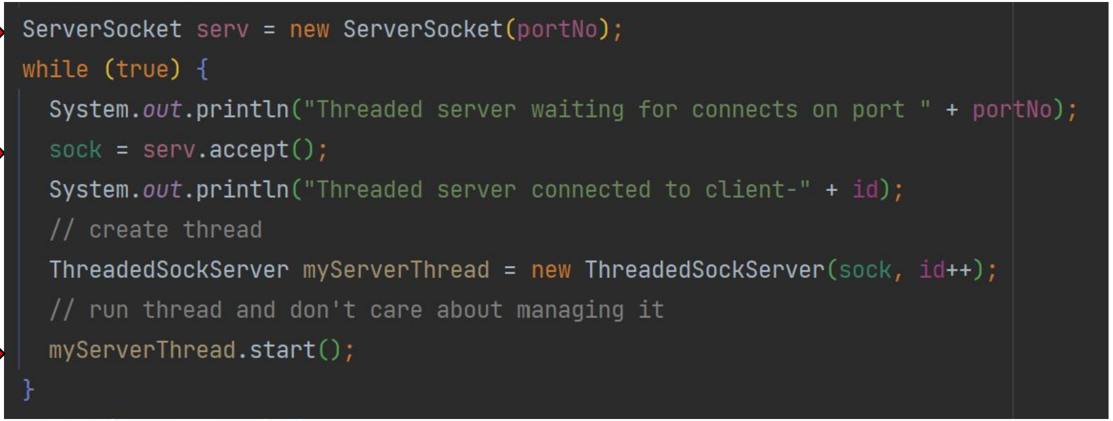
Start Thread

Handle the connection

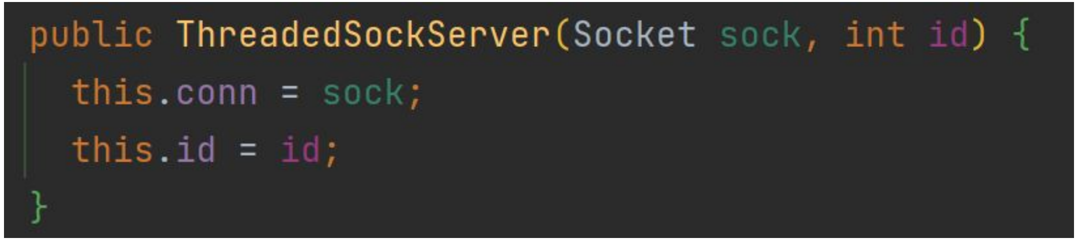
Perform the task

Clean up

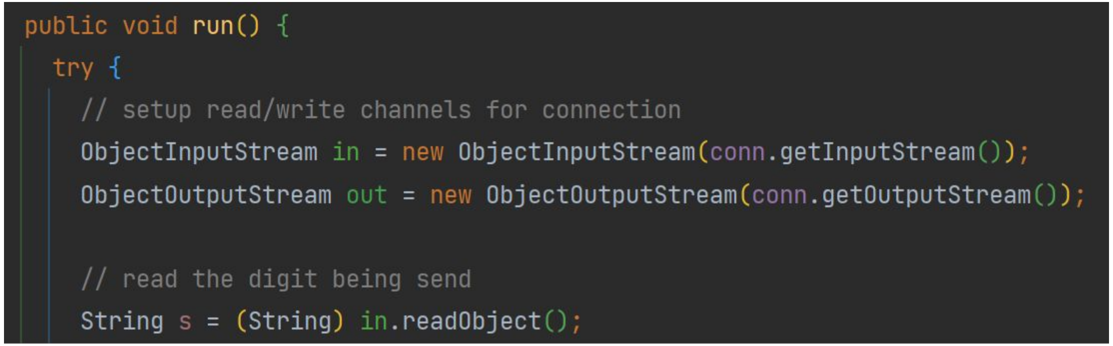
JavaThreadedSock in Sockets



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



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

    // read the digit being send
    String s = (String) in.readObject();
```

SER 321 A Session

Threading

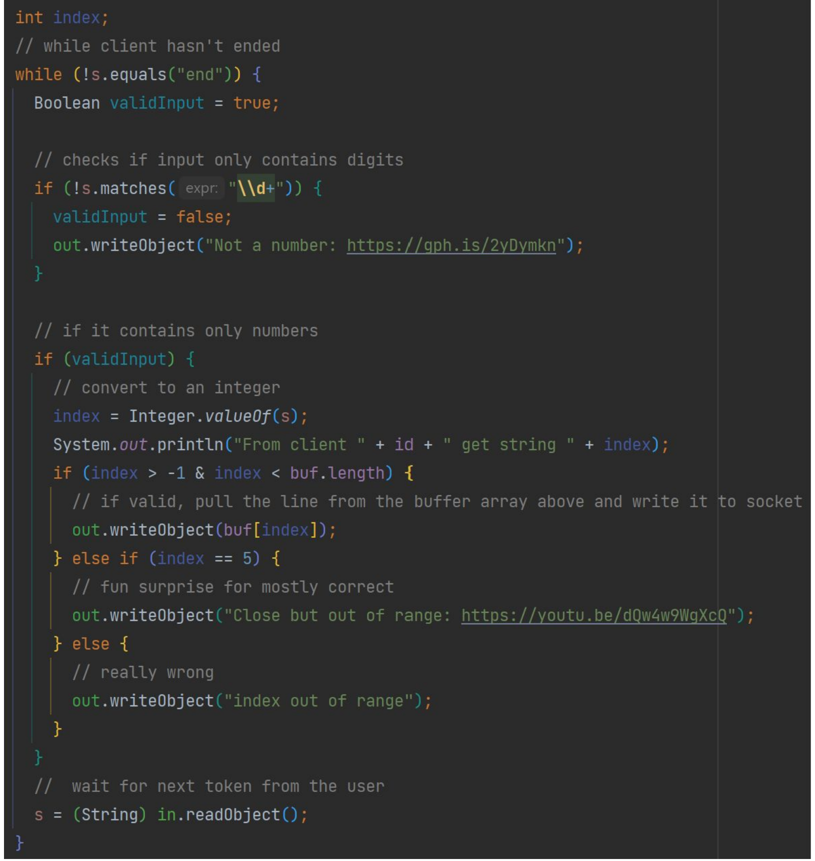
Make Socket

Wait for connections

Start Thread

Handle the connection

Perform the task

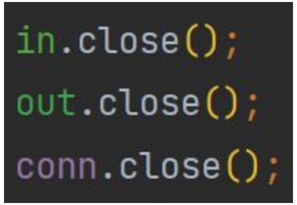


```
int index;
// while client hasn't ended
while (!s.equals("end")) {
  Boolean validInput = true;

  // checks if input only contains digits
  if (!s.matches( expr: "\\d+")) {
    validInput = false;
    out.writeObject("Not a number: https://gph.is/2yDymkn");
  }

  // if it contains only numbers
  if (validInput) {
    // convert to an integer
    index = Integer.valueOf(s);
    System.out.println("From client " + id + " get string " + index);
    if (index > -1 & index < buf.length) {
      // if valid, pull the line from the buffer array above and write it to socket
      out.writeObject(buf[index]);
    } else if (index == 5) {
      // fun surprise for mostly correct
      out.writeObject("Close but out of range: https://youtu.be/dQw4w9WgXcQ");
    } else {
      // really wrong
      out.writeObject("index out of range");
    }
  }
  //  wait for next token from the user
  s = (String) in.readObject();
}
```

Clean up

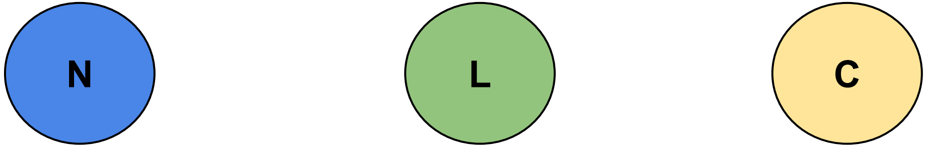


```
in.close();
out.close();
conn.close();
```

SER 321

Sockets and Threading - Assignment

Which one is a server?



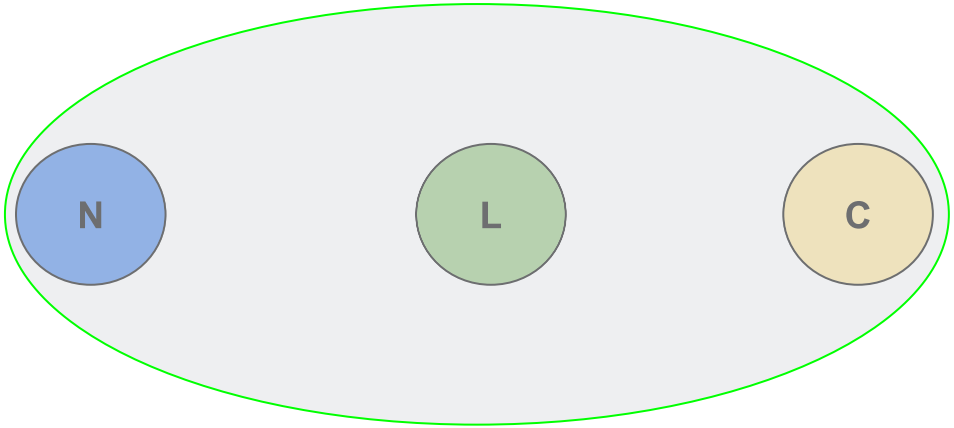
Which one is a client?

Check out the recording for the solution!

SER 321

Sockets and Threading - Assignment

Where are the threads?



Focused around the Leader!

Communication with the nodes will be in **threads** - simulating many nodes across the globe

What if a node connects at the same time the client sends a message?

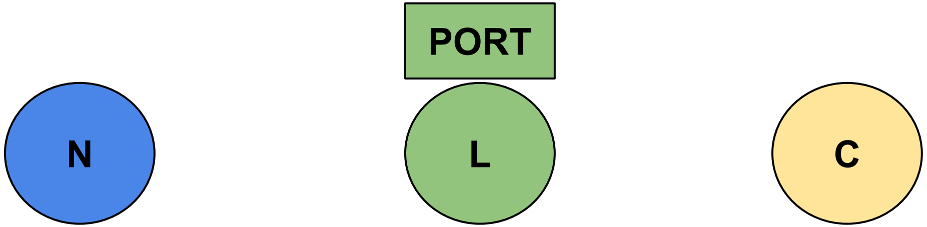
How is that supposed to work?

Check out the recording for the solution!

SER 321

Sockets and Threading - Assignment

Server is listening for connections…



Start a node

>> Server accepts the Node's connection

What? How?

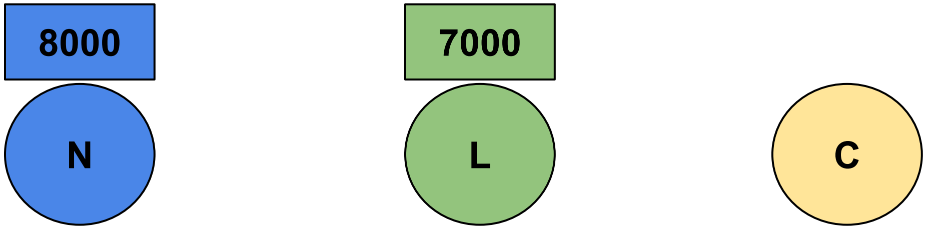
Node needs two things on startup

- Its own port
- Port of the leader

SER 321

Sockets and Threading - Assignment

Example:
Leader is listening on port **7000**



Let’s start a node and tell it to listen on port **8000**

Is that enough?

NO!

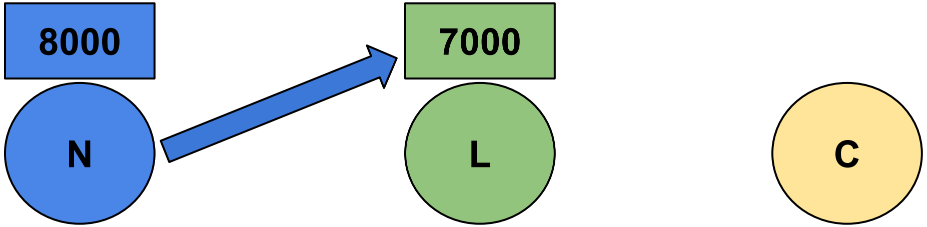
Why not?

Leader and node are both listening for outside communication on their assigned port

SER 321

Sockets and Threading - Assignment

Example:
Leader is listening on port **7000**



Let’s start a node and tell it to listen on port **8000**

Is that enough? **NO!**

Why not?

Leader and node are both listening for outside communication on their assigned port

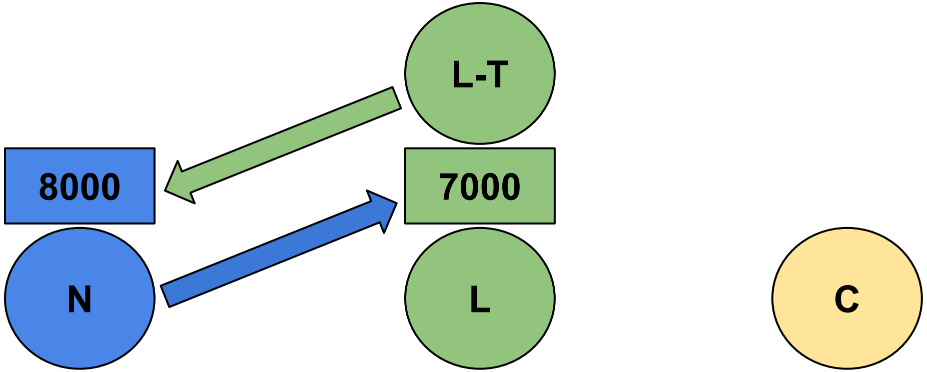
Must know the port that you want to communicate with!

Remember how a client needs the host and port of the server?

SER 321

Sockets and Threading - Assignment

Back to the outline…
Server is listening for connections…



Start a node

>> Server accepts the Node's connection

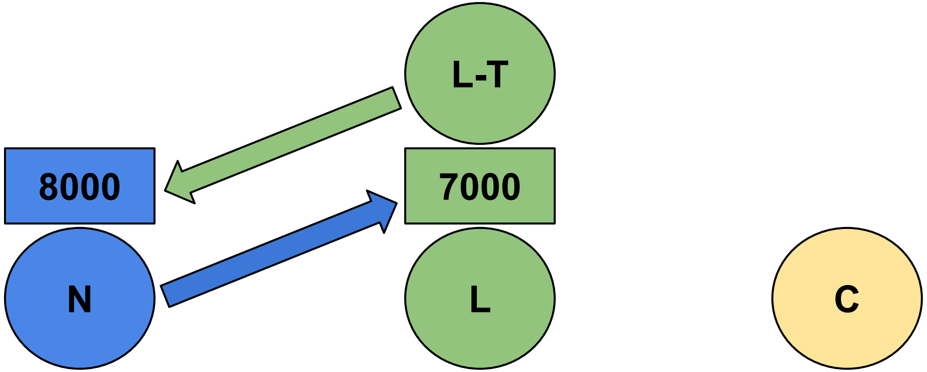
>> What does the server do next?

Obtains the port of that connection - SocketInfo in PeerToPeer

Starts a thread to handle communication

SER 321

Sockets and Threading - Assignment



You need at least three nodes

Leader will need to **keep track of the nodes and their threads**

Need to send data to nodes for encryption

Need to track individual nodes for consensus

And don't forget the client!

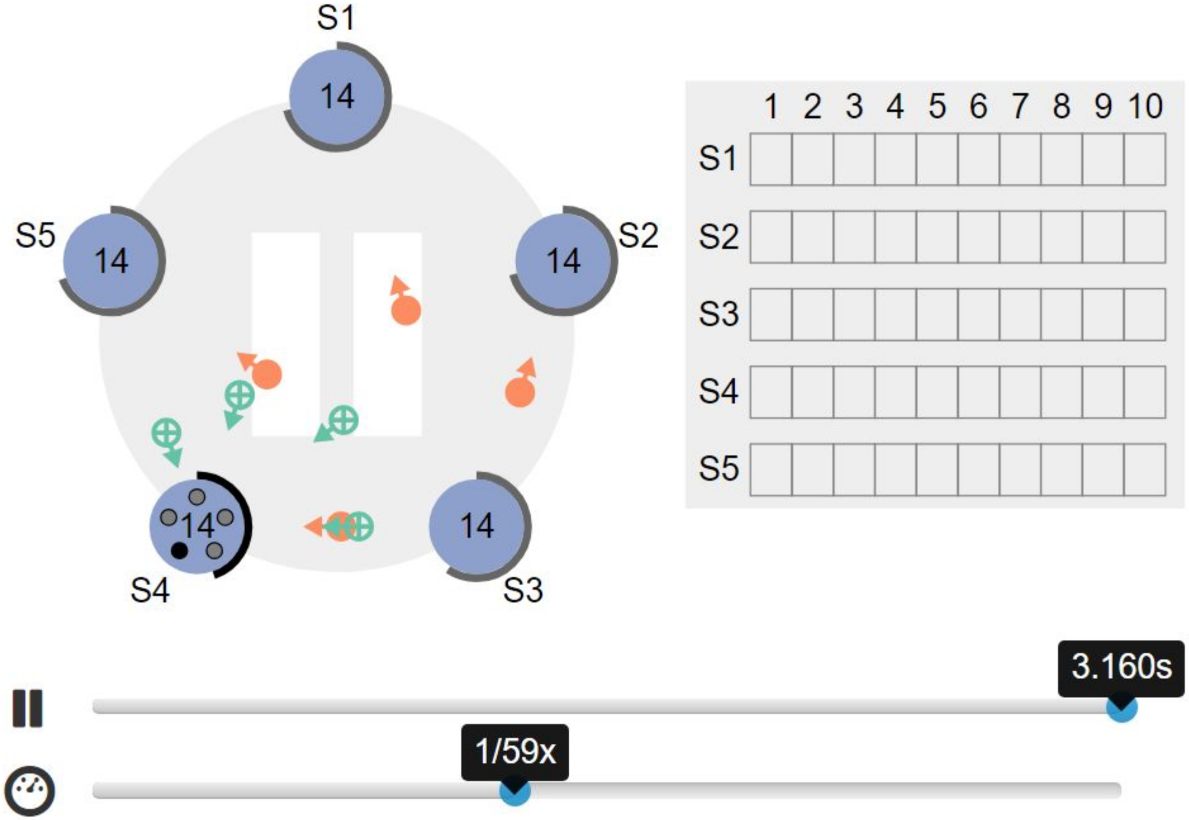
SER 321

RAFT

RAFT

Leader Election

Log Replication



	1	2	3	4	5	6	7	8	9	10
S1										
S2										
S3										
S4										
S5										

Questions?

Survey:

https://bit.ly/asn_survey

Upcoming Events

SI Sessions:

- Monday September 25th 2023 6:00 pm MST

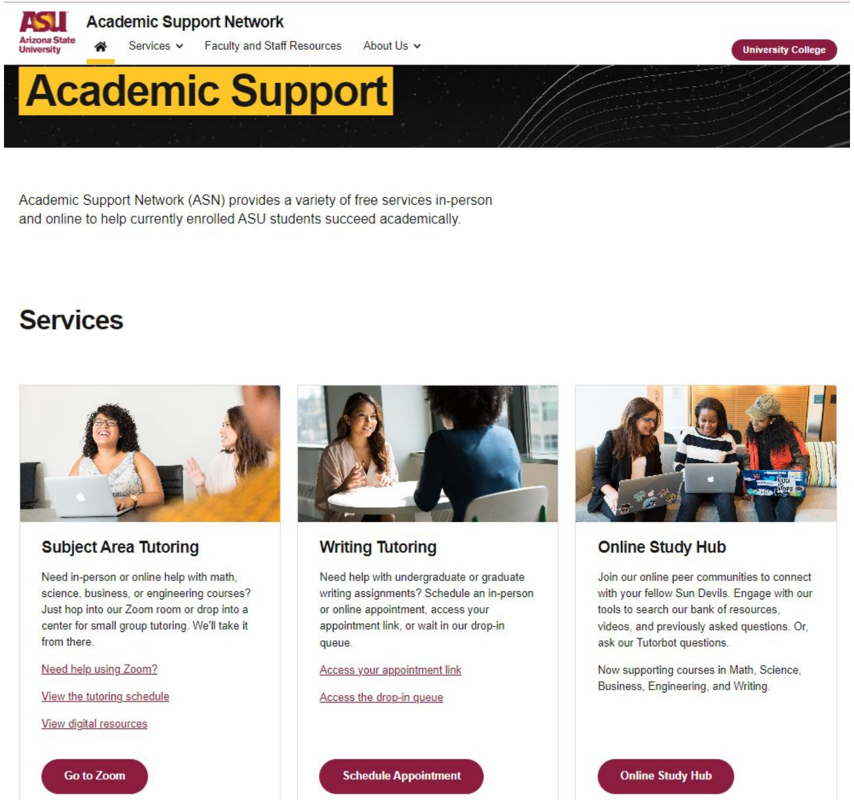
Review Sessions:

- TBD

More Questions?

Check out our other resources!

tutoring.asu.edu



ASU Arizona State University

Academic Support Network

Services ⌄ Faculty and Staff Resources About Us ⌄

University College

Academic Support

Academic Support Network (ASN) provides a variety of free services in-person and online to help currently enrolled ASU students succeed academically.

Services

Subject Area Tutoring

Need in-person or online help with math, science, business, or engineering courses? Just hop into our Zoom room or drop into a center for small group tutoring. We'll take it from there.

Need help using Zoom?

View the tutoring schedule

View digital resources

Go to Zoom

Writing Tutoring

Need help with undergraduate or graduate writing assignments? Schedule an in-person or online appointment, access your appointment link, or wait in our drop-in queue.

Access your appointment link

Access the drop-in queue

Schedule Appointment

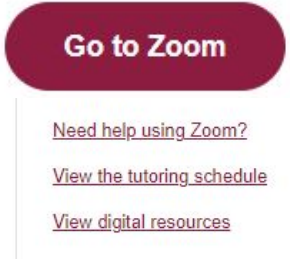
Online Study Hub

Join our online peer communities to connect with your fellow Sun Devils. Engage with our tools to search our bank of resources, videos, and previously asked questions. Or, ask our Tutorbot questions.

Now supporting courses in Math, Science, Business, Engineering, and Writing.

Online Study Hub

1 –



Go to Zoom

2-

Need help using Zoom?

View the tutoring schedule

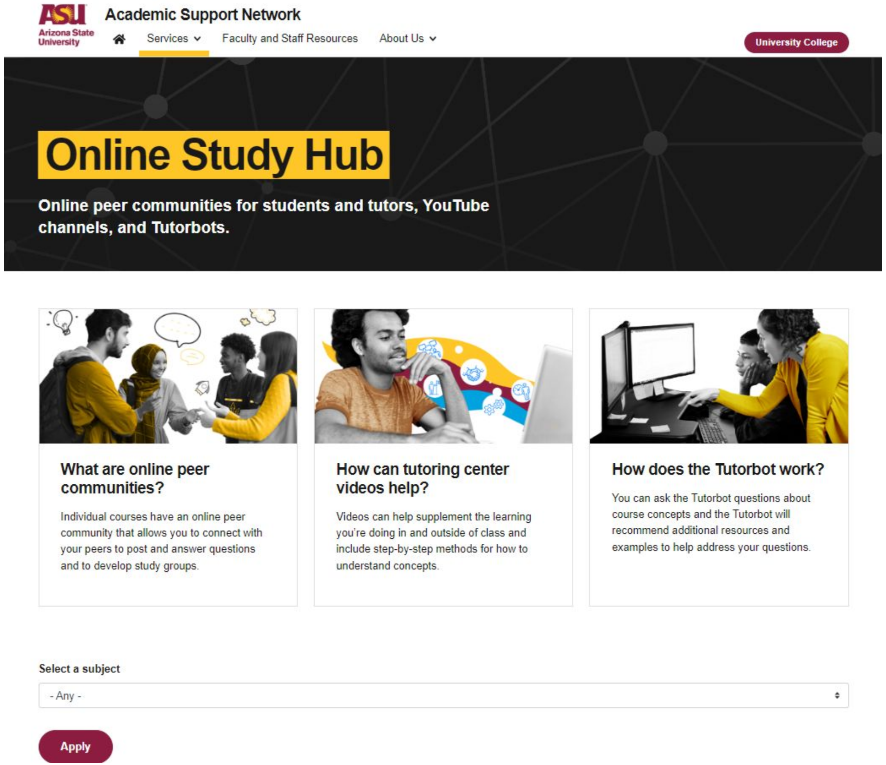
View digital resources

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



Online Study Hub

Online peer communities for students and tutors, YouTube channels, and Tutorbots.

What are online peer communities?

Individual courses have an online peer community that allows you to connect with your peers to post and answer questions and to develop study groups.

How can tutoring center videos help?

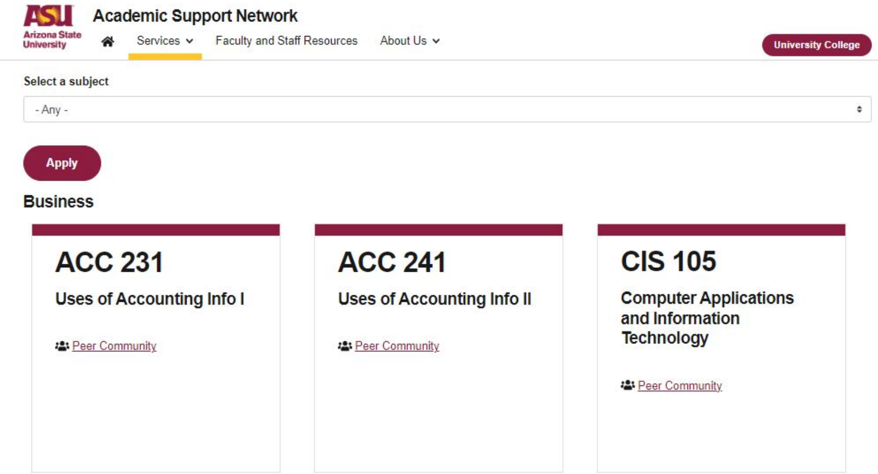
Videos can help supplement the learning you're doing in and outside of class and include step-by-step methods for how to understand concepts.

How does the Tutorbot work?

You can ask the Tutorbot questions about course concepts and the Tutorbot will recommend additional resources and examples to help address your questions.

Select a subject

- Any -



Select a subject

- Any -

Business

ACC 231
Uses of Accounting Info I
Peer Community

ACC 241
Uses of Accounting Info II
Peer Community

CIS 105
Computer Applications and Information Technology
Peer Community

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

Additional Resources

CoureRepo

PeerToPeer

Dining Philosophers Interactive

Raft Interactive