SER 321 A Session

SI Session

Wednesday September 27th 2023

6:00 - 7:00 pm MST

Agenda

- Middleware
  - What is it?
  - Why do I care?
- Assignment 6

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

Intro to Middleware video

SER 321

OSI Model - Middleware

So far we have been focused *pretty much* on a single layer

We are *manually* creating specifications and *manually* constructing our payloads

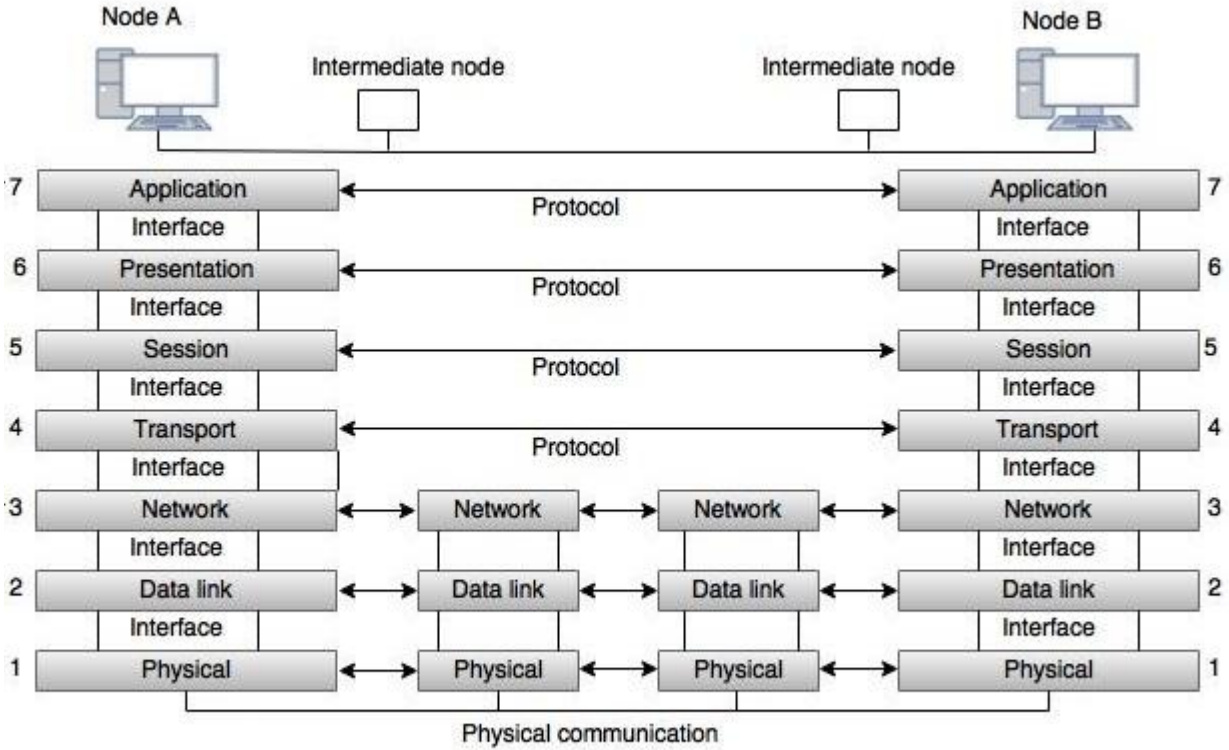


Fig: OSI Model

Intro to Middleware video

SER 321

OSI Model - Middleware

Client:

- Construct JSON request
- Send to server
- Wait for response
- Parse JSON response

Server:

- Receive JSON request
- Parse JSON request
- Handle accordingly
- Create JSON response
- Send to Client

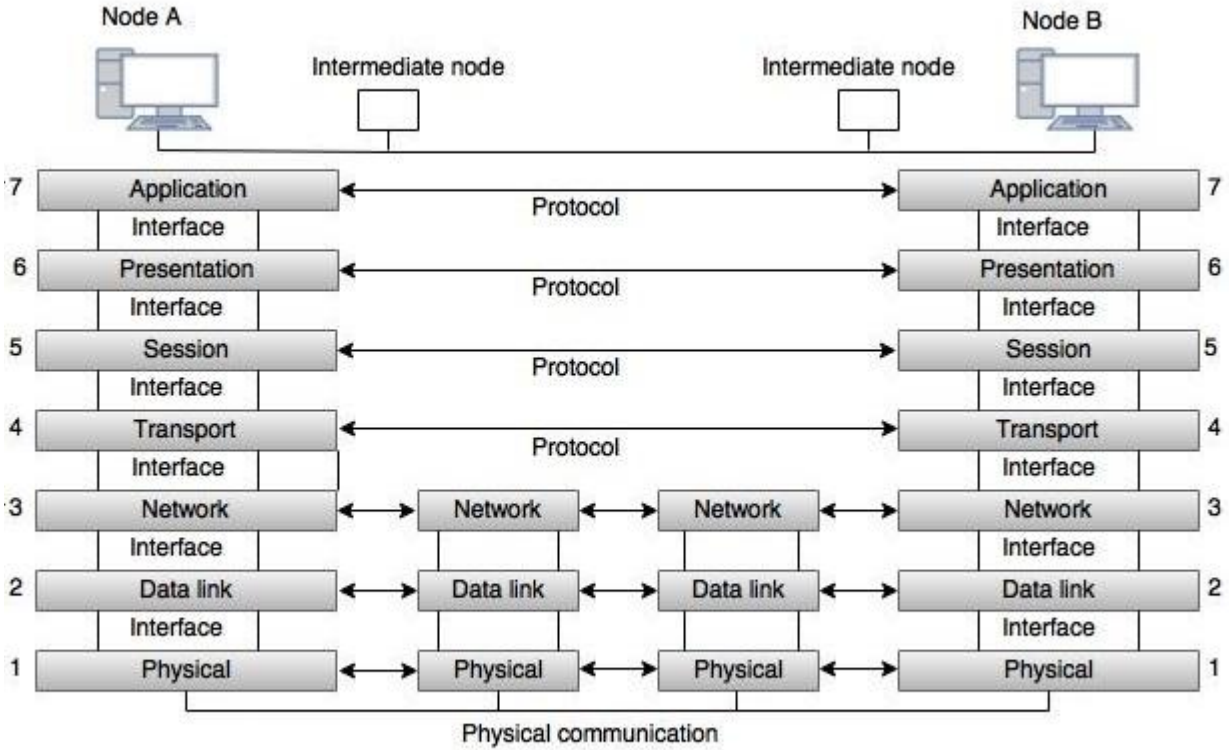


Fig: OSI Model

SER 321

OSI Model - Middleware

Intro to Middleware video

Yeah that's what we've been doing, so what?

Middleware is a set of services that allow us to *target methods on the server directly*

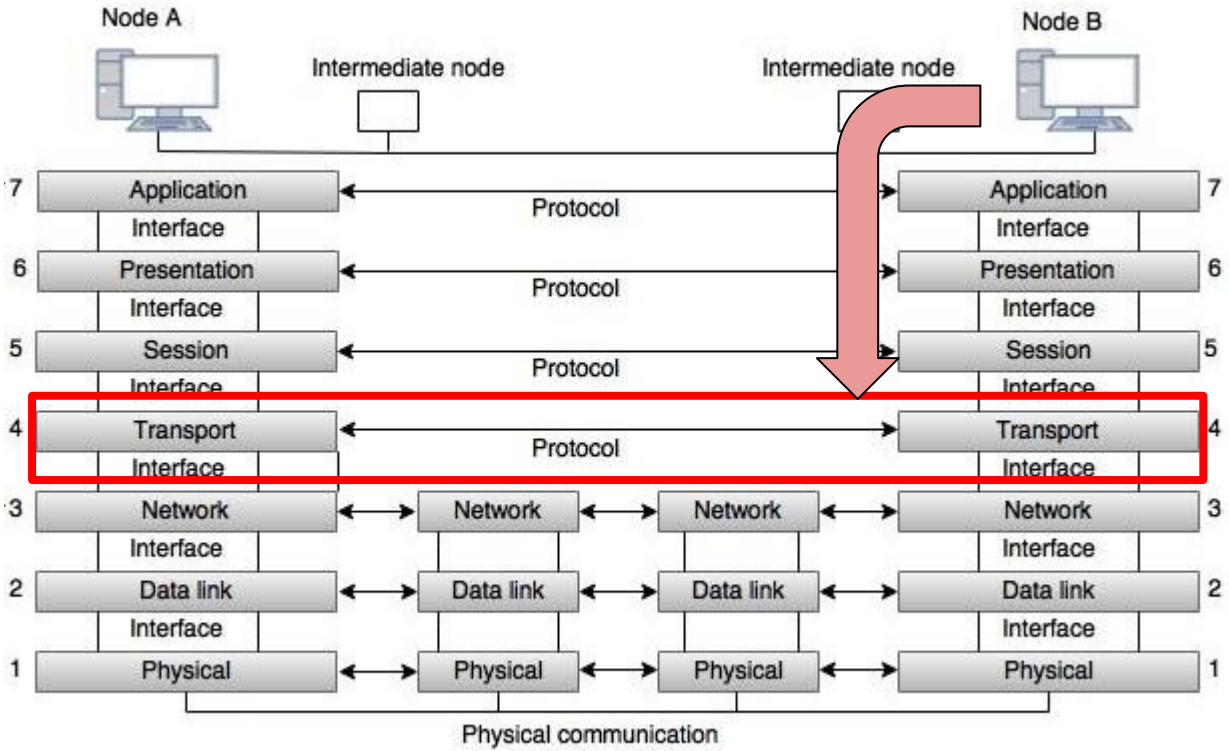


Fig: OSI Model

Intro to Middleware video

SER 321

OSI Model - Middleware

Yeah that's what we've been doing, so what?

Middleware is a set of services that allow us to *target methods on the server directly*

It handles the formatting and transmission of data for us!

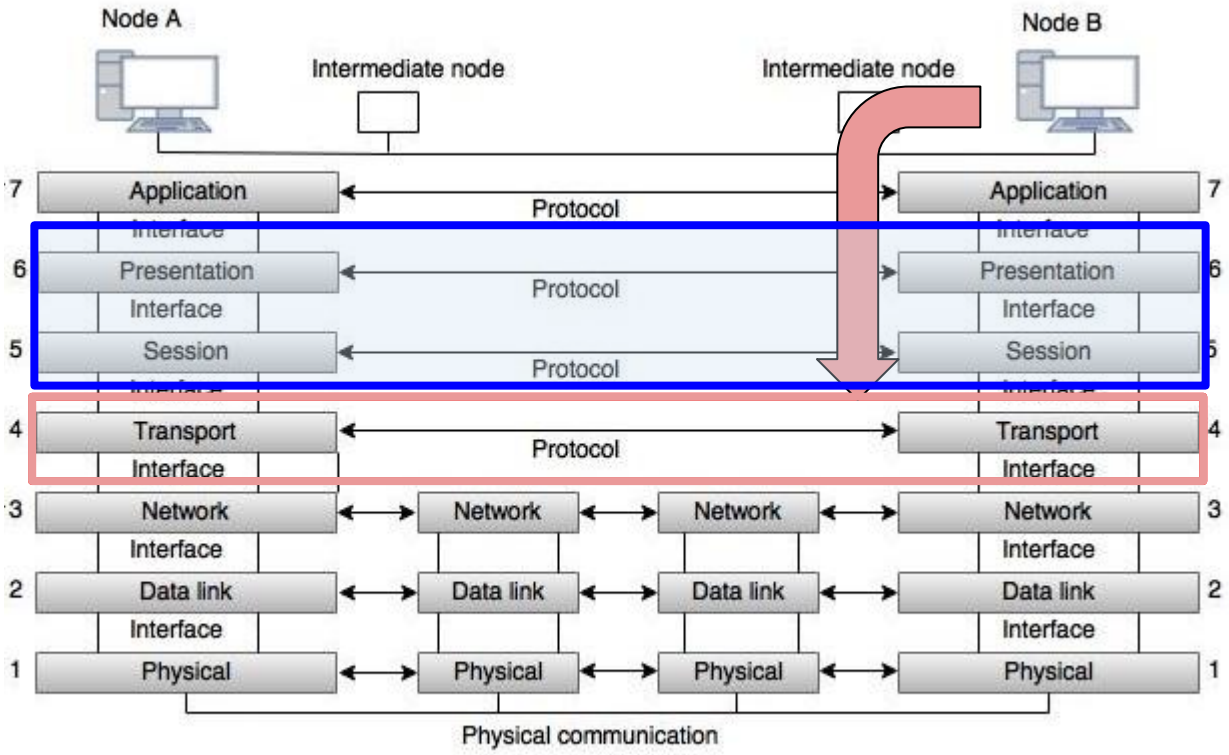


Fig: OSI Model

SER 321
Middleware

Essentially defines a common interface for communication with the server

Want to talk to the server?

Use middleware to avoid the overhead!

Enables flexibility and portability in terms of usage

Language independent

Increased scalability - any system can join the network

Can you think of examples?

- RPC
- MOM
- API

SER 321

This Assignment

Working with Protobuf again - *very* similar to the steps in Assignment 4

Will use a *builder* to construct the request or response, and use **`.build()`** when done

BUT the transmission handling is different!

Will use a *StreamObserver* to work with the stream

Everything else is handled for you!

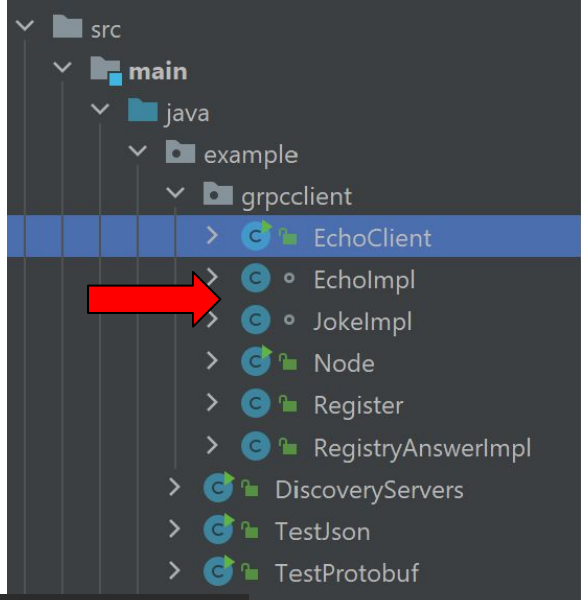
```
@Override
public void getJoke(JokeReq req, StreamObserver<JokeRes> responseObserver) {

    System.out.println("Received from client: " + req.getNumber());
    JokeRes.Builder response = JokeRes.newBuilder();
    for (int i=0; i < req.getNumber(); i++){
        if(!jokes.empty()) {
            response.addJoke(jokes.pop()); // yes, I take the joke out when it was used already
        }
        else {
            response.addJoke( value: "I am out of jokes..."); // this is more of a hack
            break;
        }
    }
    JokeRes resp = response.build();
    responseObserver.onNext(resp);
    responseObserver.onCompleted();
}
```

SER 321

This Assignment

The Implementation files hold the services you are providing



How do you provide the services to the client?

```
private void start() throws IOException {
  /* The port on which the server should run */
  // Here we are adding the different services that a client can call
  ArrayList<String> services = new ArrayList<>();
  server = ServerBuilder.forPort(port) ServerBuilder<capture of ?>
      .addService(new EchoImpl()) capture of ?
      .addService(new JokeImpl())
      .addService(new RegistryAnswerImpl(services)).build().start();
```

SER 321

This Assignment

```
public void askServerToParrot(String message) {

  try {

  } catch (Exception e) {
    System.err.println("RPC failed: " + e.getMessage());
    return;
  }
  System.out.println("Received from server: " + response.getMessage());
}
```

What about the client? How do we *use* these services?

We build the request that we want to send

Call the server method with a *blockingStub*

Waits for the server to give response or exception

SER 321
This Assignment

```
// create client
EchoClient client = new EchoClient(channel, regChannel);
```

The client uses a *channel* to communicate the server

And we create a blocking stub using the channel

```
2 usages
private final EchoGrpc.EchoBlockingStub blockingStub;
3 usages
private final JokeGrpc.JokeBlockingStub blockingStub2;
4 usages
private final RegistryGrpc.RegistryBlockingStub blockingStub3;
2 usages
private final RegistryGrpc.RegistryBlockingStub blockingStub4;
```

```
public EchoClient(Channel channel, Channel regChannel) {
  // 'channel' here is a Channel, not a ManagedChannel, so it is
  // responsibility to
  // shut it down.

  // Passing Channels to code makes code easier to test and makes it easier to
  // reuse Channels.
  blockingStub = EchoGrpc.newBlockingStub(channel);
  blockingStub2 = JokeGrpc.newBlockingStub(channel);
  blockingStub3 = RegistryGrpc.newBlockingStub(regChannel);
  blockingStub4 = RegistryGrpc.newBlockingStub(channel);
}
```

SER 321
Summary

What is middleware?

-

Katie’s thoughts:

An interface for communication between client and server. Client is able to make method calls “directly” to the server and obtain results.

Questions?

Survey:

https://bit.ly/asn_survey

15

Upcoming Events

SI Sessions:

- Sunday October 1st 2023 at 6:00 pm MST

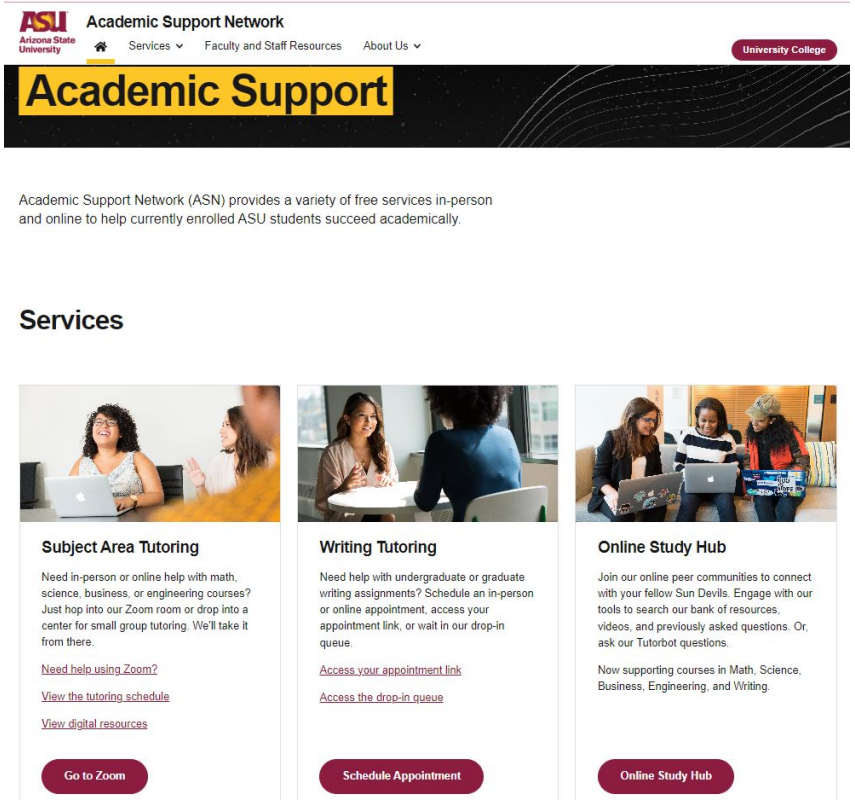
Review Sessions:

- I’m leaning towards planning our Review Session for Sunday or Monday

More Questions?

Check out our other resources!

tutoring.asu.edu

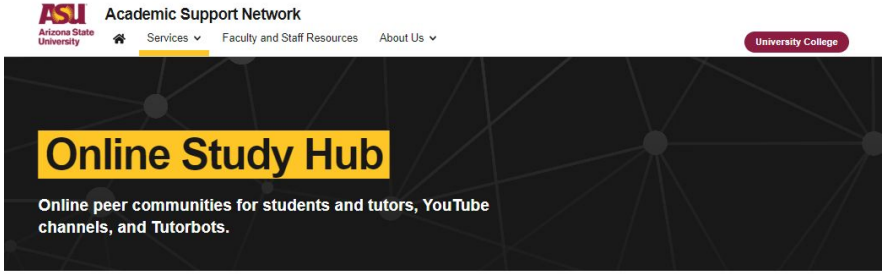


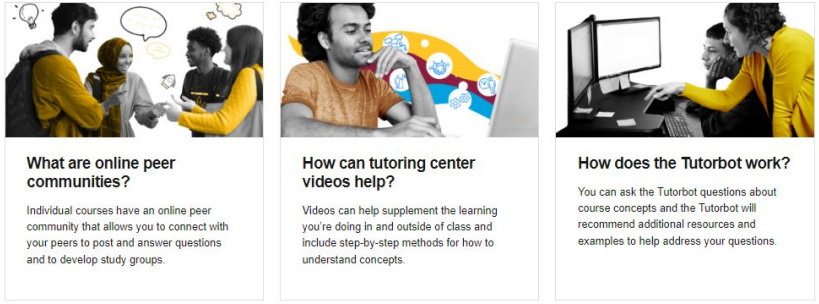
1. **Click on ‘Go to Zoom’ to log onto our Online Tutoring Center.**
2. **Click on ‘View the tutoring schedule’ to see when tutors are available for specific courses.**

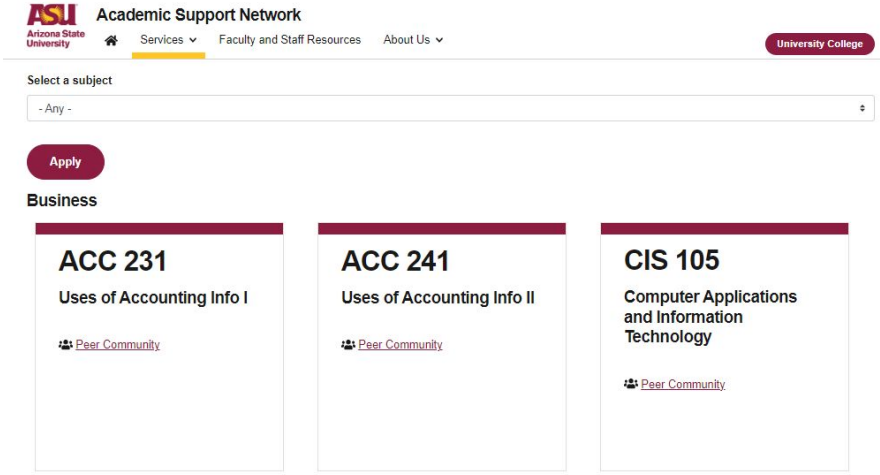
More Questions?

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

Dining Philosophers Interactive

Raft Interactive