

# SER 321 B Session

**SI Session**

**Monday, April 22nd 2024**

*7:00 pm - 8:00 pm MST*

# Agenda



Middleware

GRPC

Assignment 6

Q&A

# 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](https://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**

**Exam Information**

[Exam Info Page](#)

80 minutes

Similar to the  
quizzes

**Opens:** Wednesday  
April 24th  
@ 12:01 AM

**Closes:** Friday  
April 26th  
@ 11:59 PM

Front and Back!

MUST BE *Handwritten*



# SER 321

## Sockets!

Sockets allow our client and server to communicate!

Location

Connection  
Semantics

Message Format

Need to define **3 properties** before usage

IP or DNS

142.251.46.206

www.google.com

TCP or UDP

Connection  
Oriented

Connectionless

Protocol Specs

Synchronous

Asynchronous

Stateless

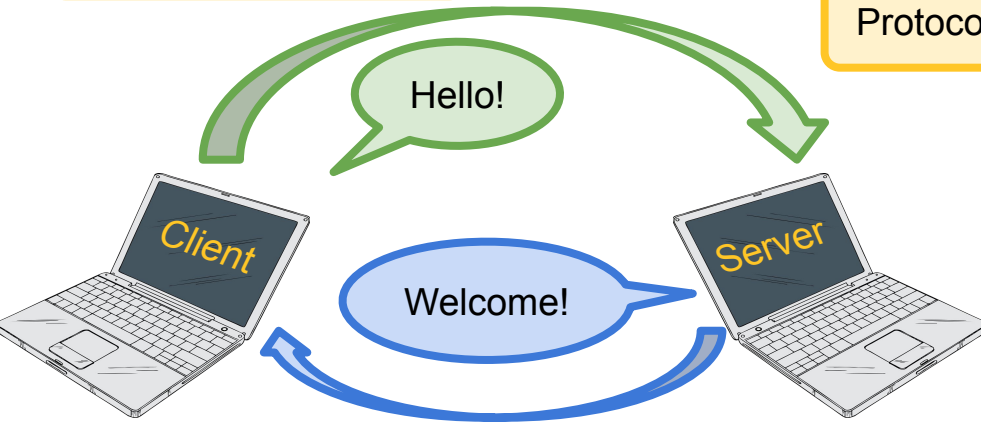
Stateful

Binary

Text

Headers

No Headers



# SER 321

## Middleware

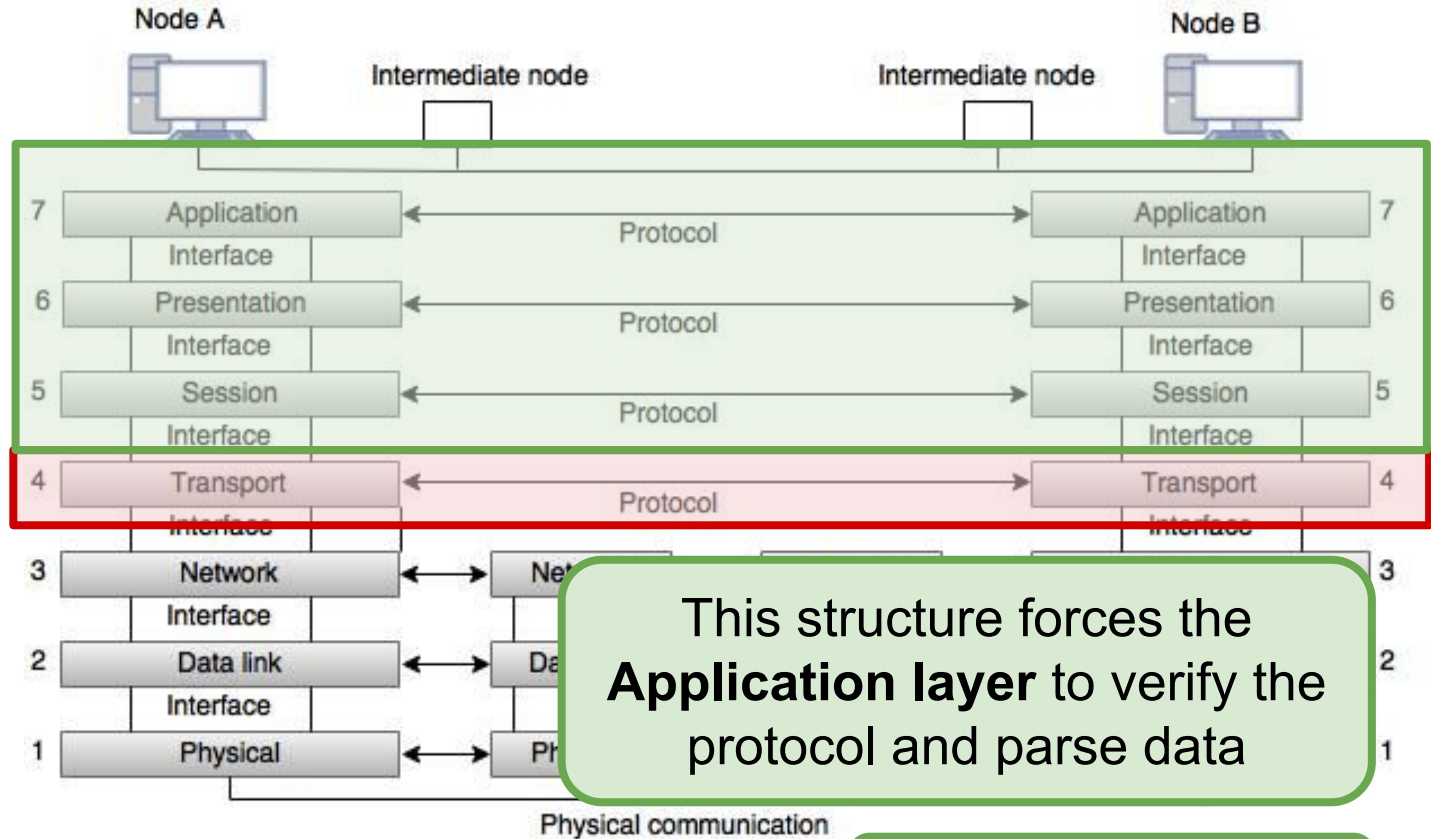
We have been:

Constructing  
(valid) Messages

Serializing  
Messages

Sending  
Message

Handle Message



This structure forces the **Application layer** to verify the protocol and parse data

Not really its job...

# SER 321

## Middleware

Middleware:

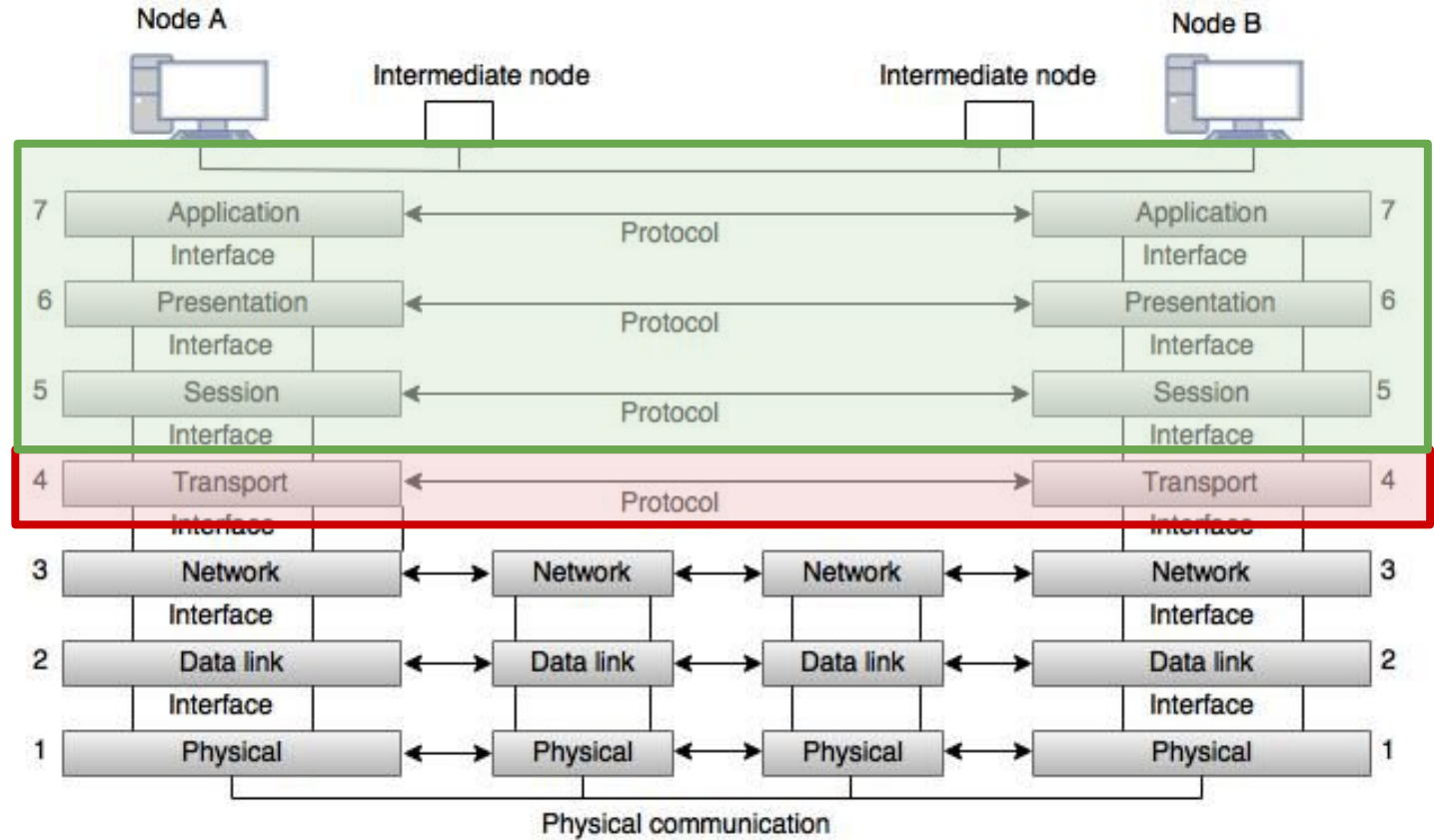


Fig: OSI Model



# SER 321

## Middleware

Middleware:

All that is  
handled  
*within* the  
middleware!

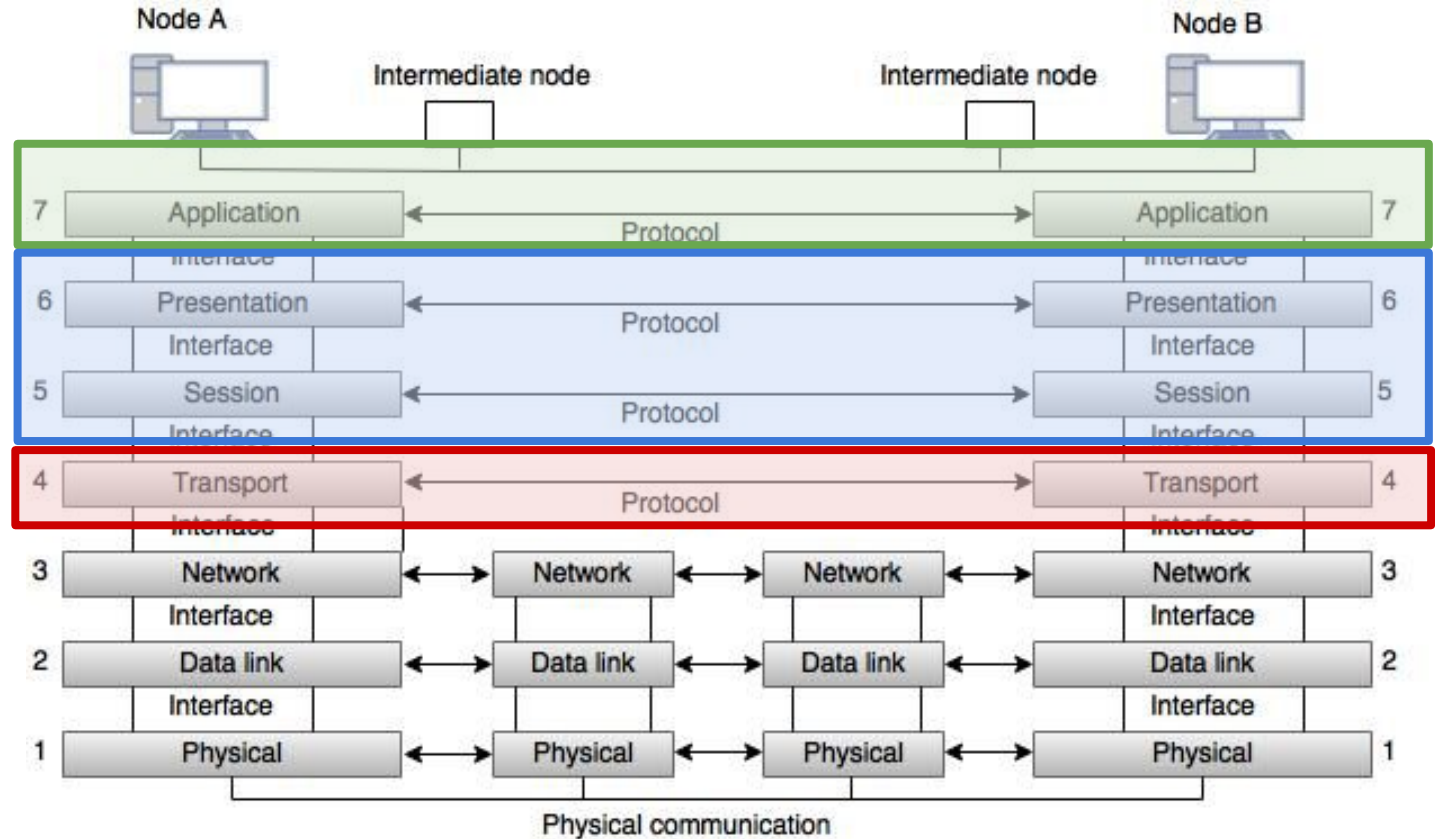


Fig: OSI Model

# SER 321

## Middleware

Middleware:

*Session Layer Responsibilities:*

Authentication

Authorization

Session Management

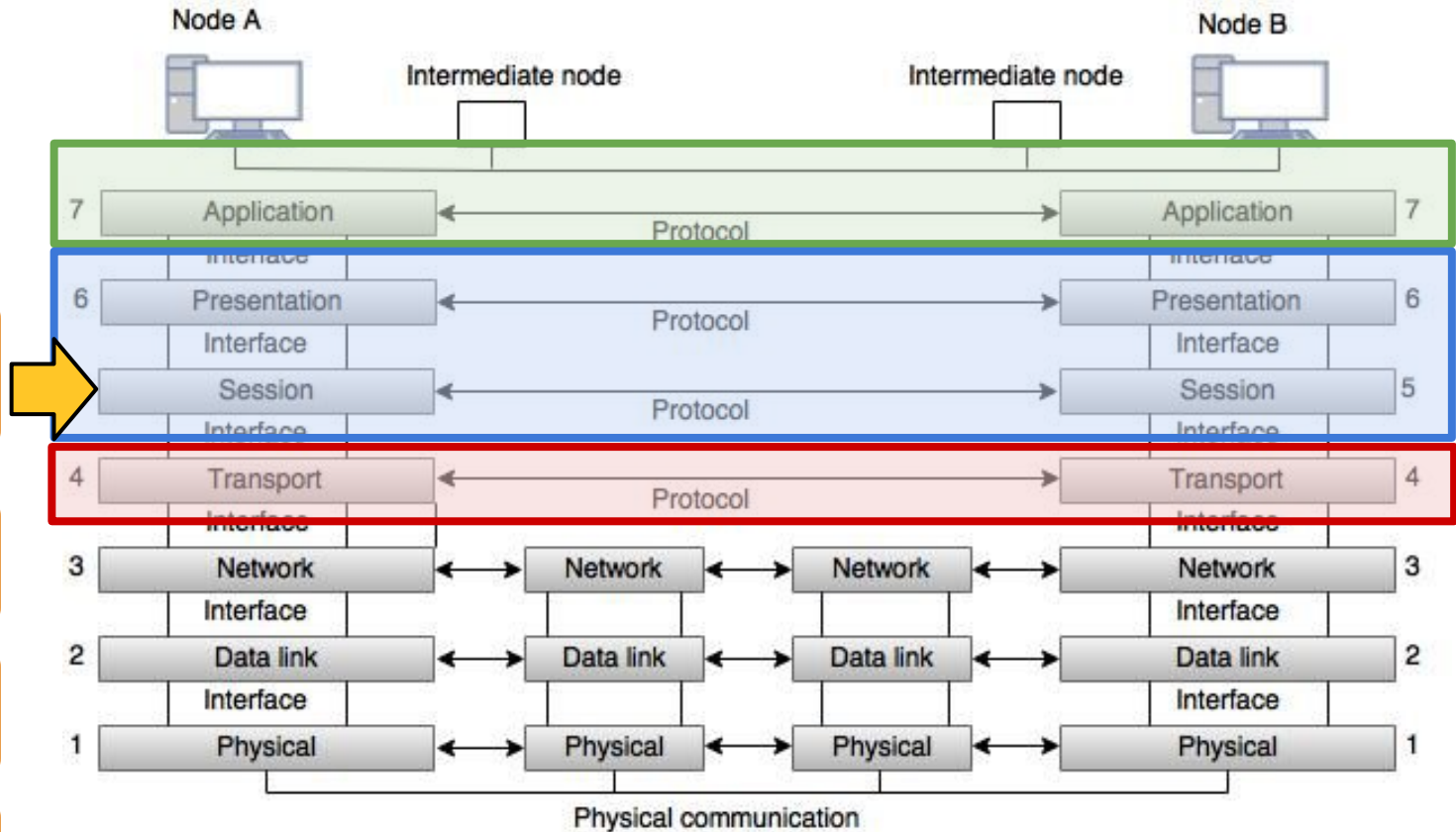


Fig: OSI Model

# SER 321

## Middleware

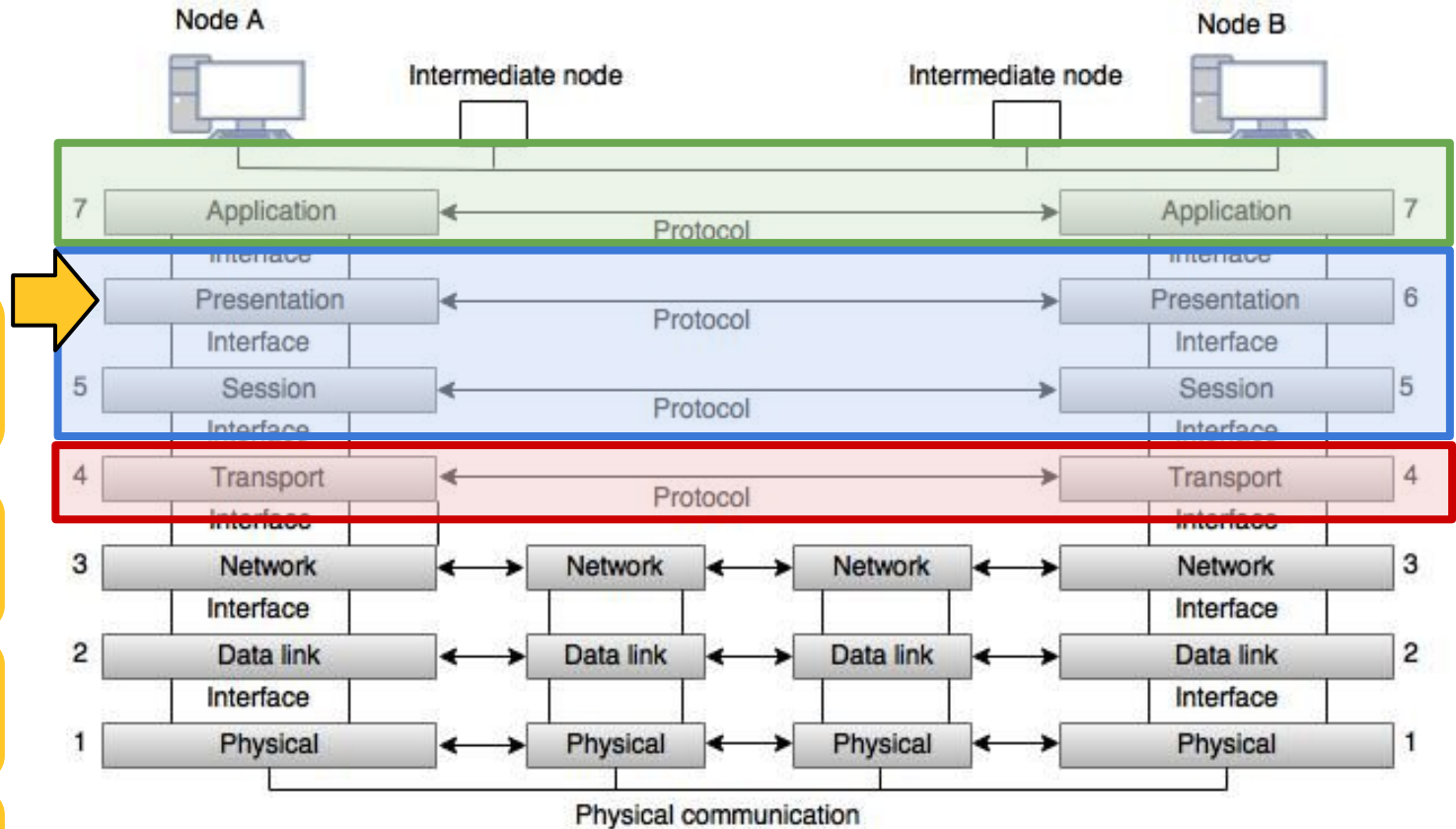
Middleware:

*Presentation  
Layer  
Responsibilities:*

Translation

Compression

Encryption



**Fig: OSI Model**

# SER 321

## Middleware

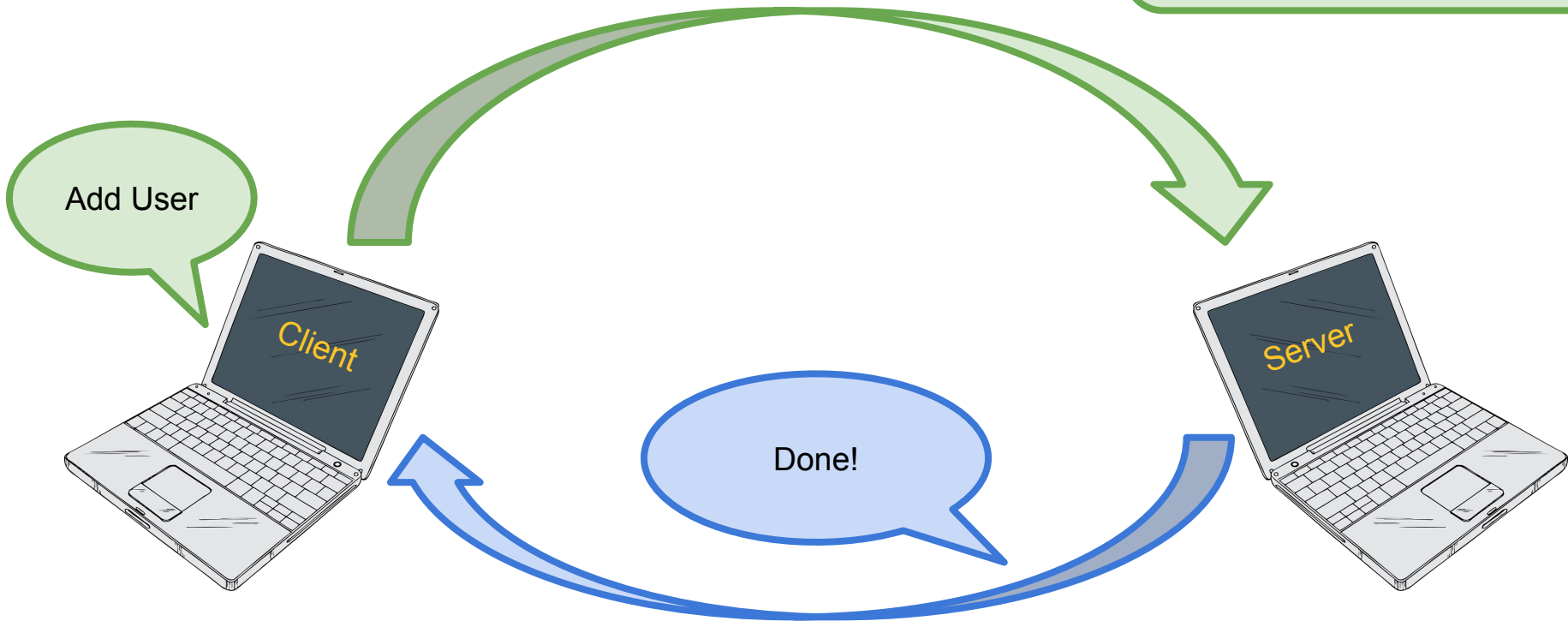
```
{  
  "type" : "addUser",  
  "name" : "katie",  
  "password" : "password"  
}
```

Add User

Client

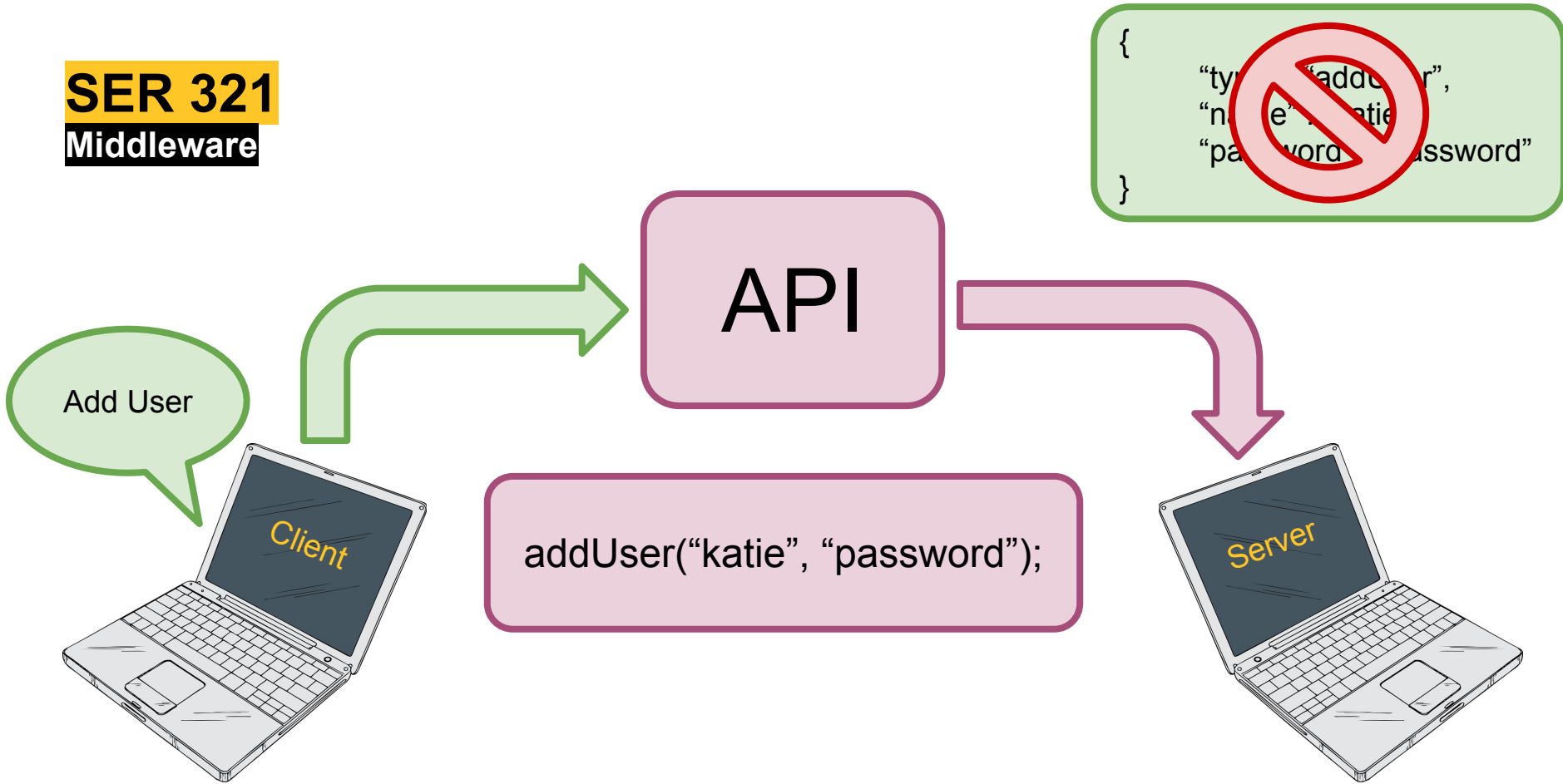
Done!

Server

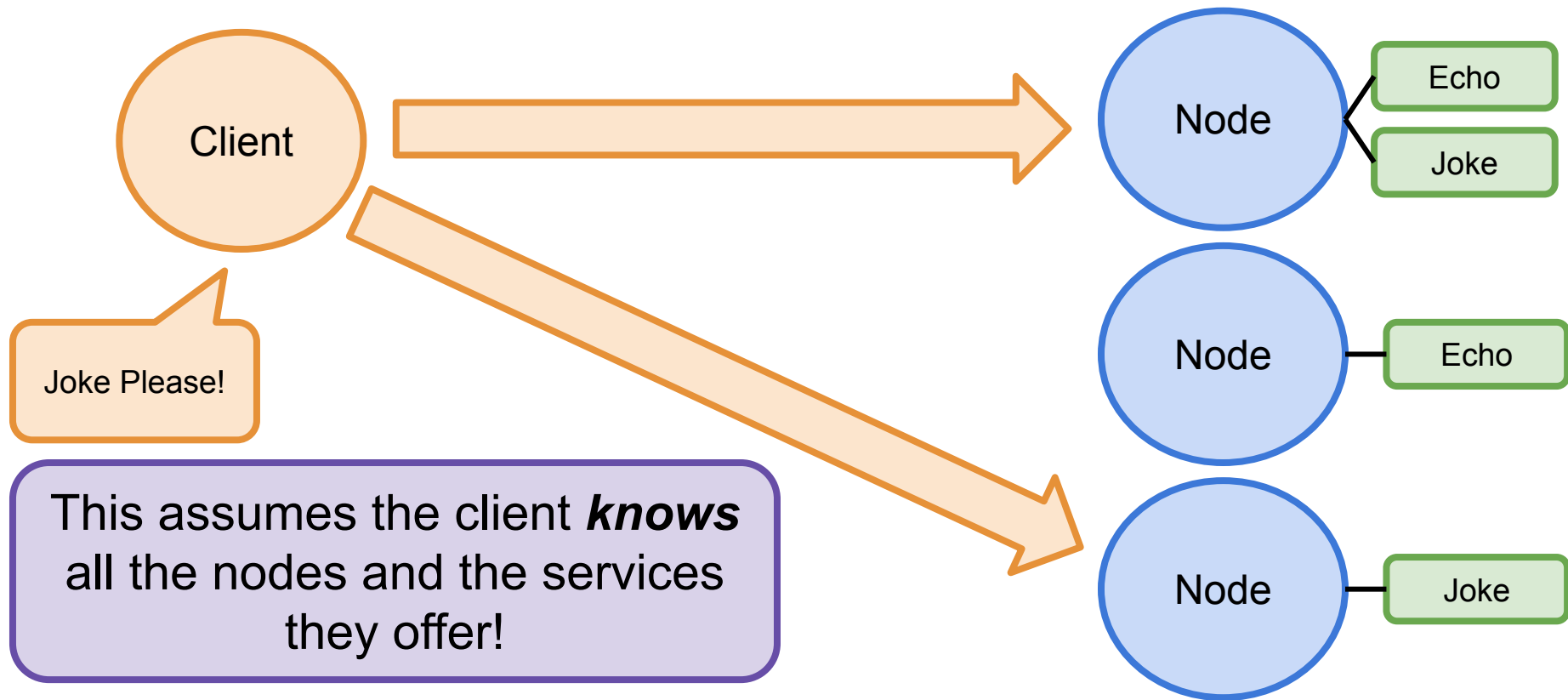


# SER 321

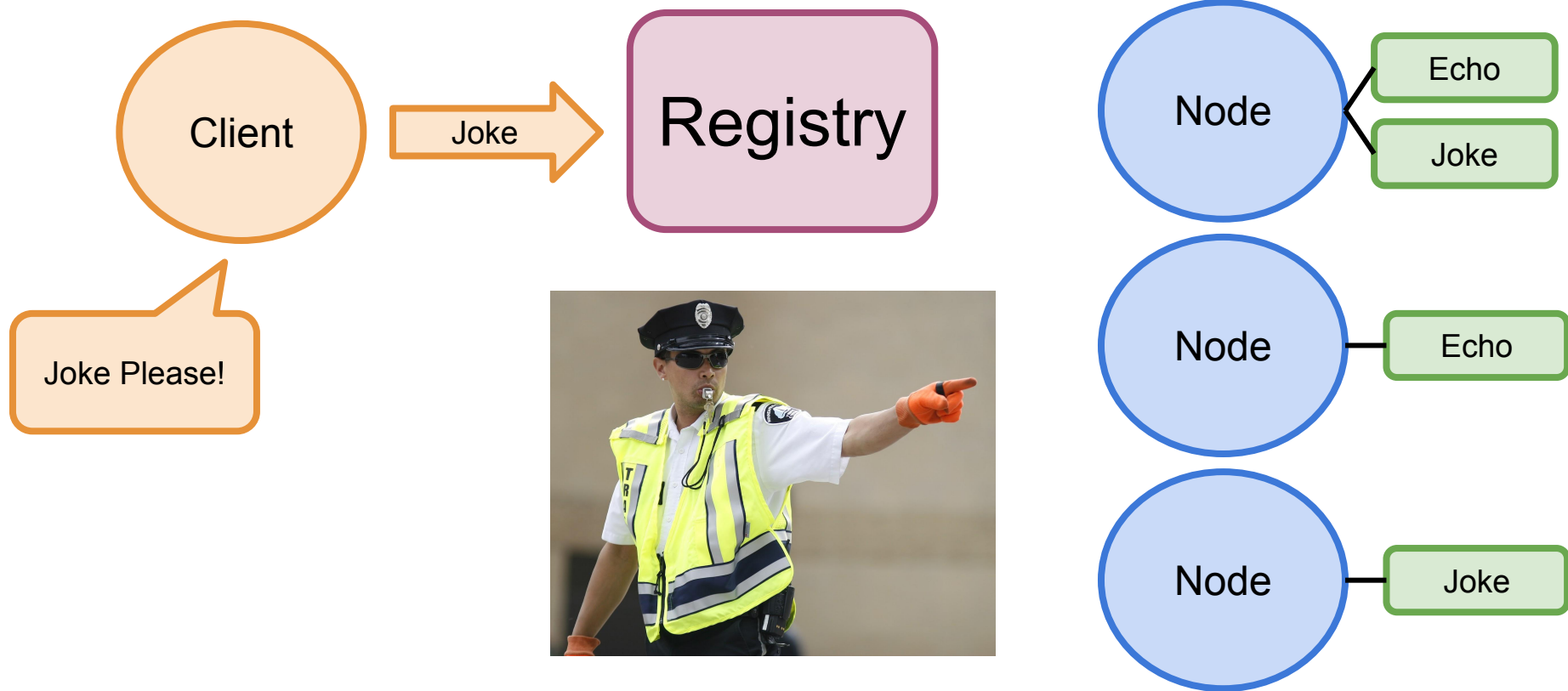
## Middleware



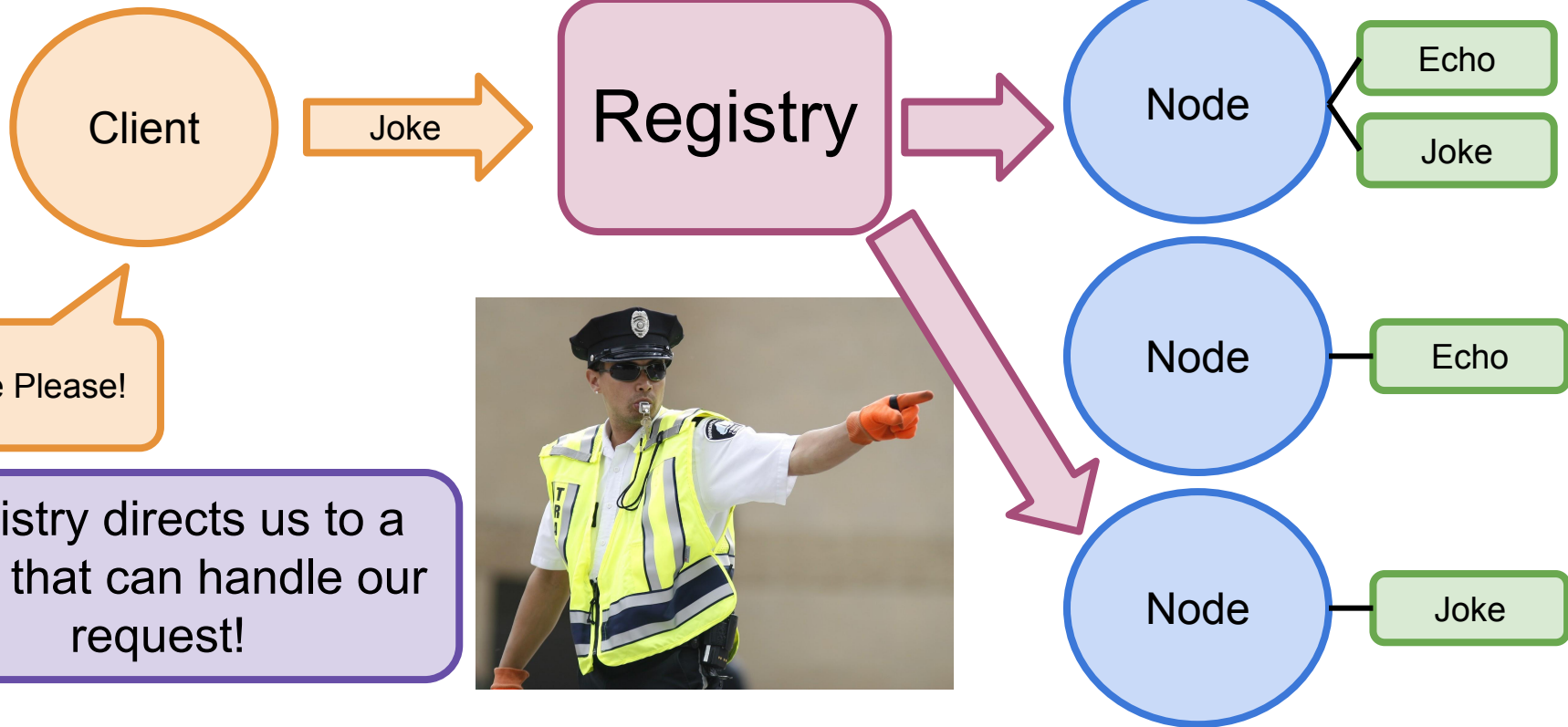
## Previously...



# With GRPC...



# With GRPC...





# SER 321

## Assign 6 Setup

You can think of channels like sockets!

```
String target = host + ":" + port;
```

```
ManagedChannel channel = ManagedChannelBuilder.forTarget(target)  
    .usePlaintext().build();
```

```
String regTarget = regHost + ":" + regPort;
```

```
ManagedChannel regChannel = ManagedChannelBuilder.forTarget(regTarget)  
    .usePlaintext().build();
```

```
Client client = new Client(channel, regChannel);
```

Targets the *REGISTRY*

## SER 321

### Assign 6 Setup

Blocking Stubs allow us to “use” the services on a node without interruption

```
public Client(Channel channel, Channel regChannel) {  
    blockingStub = EchoGrpc.newBlockingStub(channel);  
    blockingStub2 = JokeGrpc.newBlockingStub(channel);  
    blockingStub3 = RegistryGrpc.newBlockingStub(regChannel);  
    blockingStub4 = RegistryGrpc.newBlockingStub(channel);  
}
```

I think of it as **blocking** incoming requests until the node has finished the current request

I'm busy right now, try again later.

# SER 321

## Assign 6 Setup

```
Client client = new Client(channel, regChannel);

// call the parrot service on the server
client.askServerToParrot(message);
```

```
public void askServerToParrot(String message) {

    ClientRequest request = ClientRequest.newBuilder().setMessage(message).build();
    ServerResponse response;
    try {
        blockingStub = EchoGrpc.newBlockingStub(channel);
        response = blockingStub.parrot(request);
    } catch (Exception e) {
        System.err.println("RPC failed: " + e.getMessage());
        return;
    }
    System.out.println("Received from server: " + response.getMessage());
}
```

# SER 321

## Assign 6 Setup

```
public void askServerToParrot(String message) {  
  
    ClientRequest request = ClientRequest.newBuilder().setMessage(message).build();  
    ServerResponse response;  
    try {  
        response = blockingStub.parrot(request);  
    } catch (Exception e) {  
        System.err.println("RPC failed: " + e.getMessage());  
        return;  
    }  
    System.out.println("Received from server: " + response.getMessage());  
}
```

```
public void parrot(ClientRequest req, StreamObserver<ServerResponse> responseObserver) {
```

```
    System.out.println("Received from client: " + req.getMessage());  
    ServerResponse.Builder response = ServerResponse.newBuilder();  
    if (req.getMessage().isEmpty()) {  
        response.setIsSuccess(false).setError("No message provided");  
    } else {  
        response.setIsSuccess(true).setMessage(req.getMessage());  
    }  
    responseObserver.onNext(response.build());  
    responseObserver.onCompleted();  
}
```

# SER 321

## Assign 6 Setup

```
public void invoke(Req request, io.grpc.stub.StreamObserver<Resp> responseObserver) {  
    switch (methodId) {  
        case METHODID_PARROT:  
            serviceImpl.parrot((service.ClientRequest) request,  
                               (io.grpc.stub.StreamObserver<service.ServerResponse>) responseObserver);  
            break;  
        default:  
            throw new AssertionError();  
    }  
}  
  
response.setSuccess(true).setMessage(req.getMessage());  
responseObserver.onNext(response.build());  
responseObserver.onCompleted();  
}
```

```
public void askServerToParrot(String message) {  
  
    ClientRequest request = ClientRequest.newBuilder().setMessage(message).build();  
    ServerResponse response;  
    try {  
        response = blockingStub.parrot(request);  
    } catch (Exception e) {  
    }  
}
```

Found in the Generated Code

GRPC doing the work we  
were doing before!

**SER 321**

**Scratch Space**

# Questions?

## Survey:

<http://bit.ly/ASN2324>



## Upcoming Events

### SI Sessions:

- ~~Thursday, April 25th at 7:00 pm MST~~ - **CANCELLED**
- Sunday, April 28th at 7:00 pm MST - **FINAL SESSION**

### Review Sessions:

- N/A



# More Questions?

Check out our other resources!

tutoring.asu.edu



## 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](https://tutoring.asu.edu/online-study-hub)

 **Academic Support Network**

Services Faculty and Staff Resources About Us

University College

## 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 -

Apply



Academic Support Network



Services

Faculty and Staff Resources

About Us

University College

Select a subject

- Any -

Apply

Business

### ACC 231

Uses of Accounting Info I

Peer Community

### ACC 241

Uses of Accounting Info II

Peer Community

### CIS 105

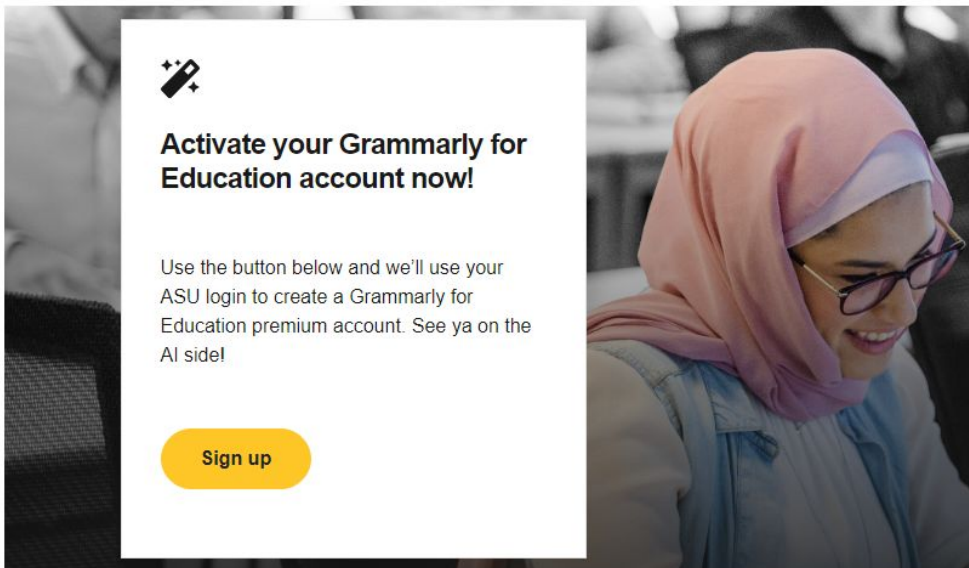
Computer Applications and Information Technology


Peer Community

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

# Expanded Writing Support Available

Including Grammarly for Education, at no cost!





**Activate your Grammarly for Education account now!**

Use the button below and we'll use your ASU login to create a Grammarly for Education premium account. See ya on the AI side!

[Sign up](#)



[tutoring.asu.edu/expanded-writing-support](https://tutoring.asu.edu/expanded-writing-support)

\*Available slots for this pilot are limited

## Additional Resources

- [Course Repo](#)
- [Gradle Documentation](#)
- [GitHub SSH Help](#)
- [Linux Man Pages](#)
- [OSI Interactive](#)
- [MDN HTTP Docs](#)
  - [Requests](#)
  - [Responses](#)
- [JSON Guide](#)
- [org.json Docs](#)
- [javax.swing package API](#)
- [Swing Tutorials](#)
- [Dining Philosophers Interactive](#)
- [Austin G Walters Traffic Comparison](#)
- [RAFT](#)