

# SER 321 B Session

**SI Session**

**Tuesday, December 3rd 2024**

*10:00 am - 11:00 am MST*

# Agenda



Assignment 6

Registry

RPC

Middleware Review

Distribution Review

# 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

## Assign 6

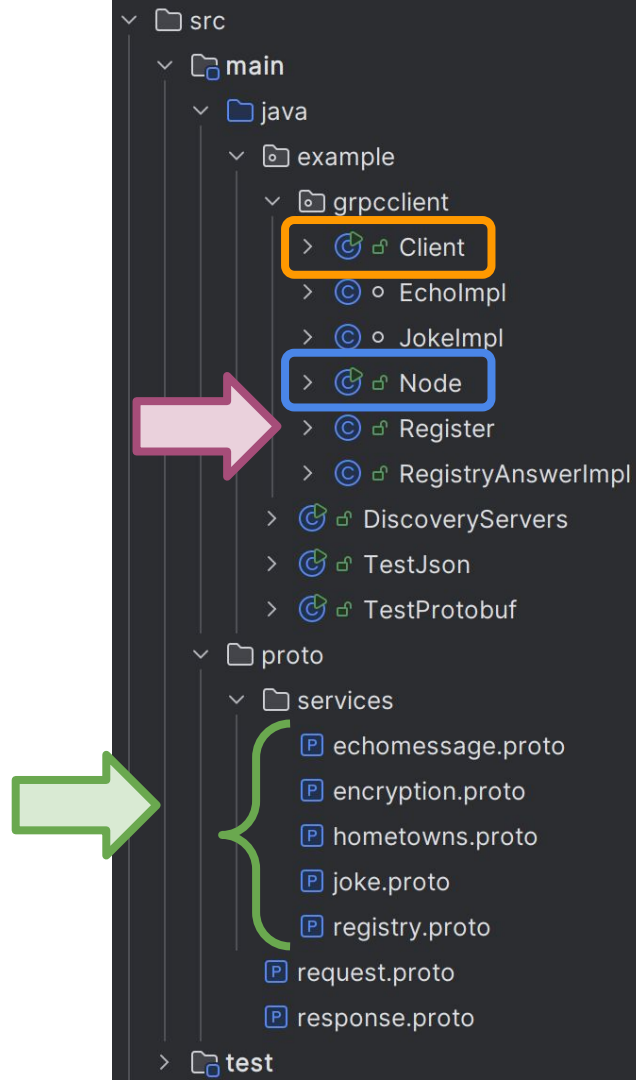
Client

Node

Registry

Protocol Buffers!

Service



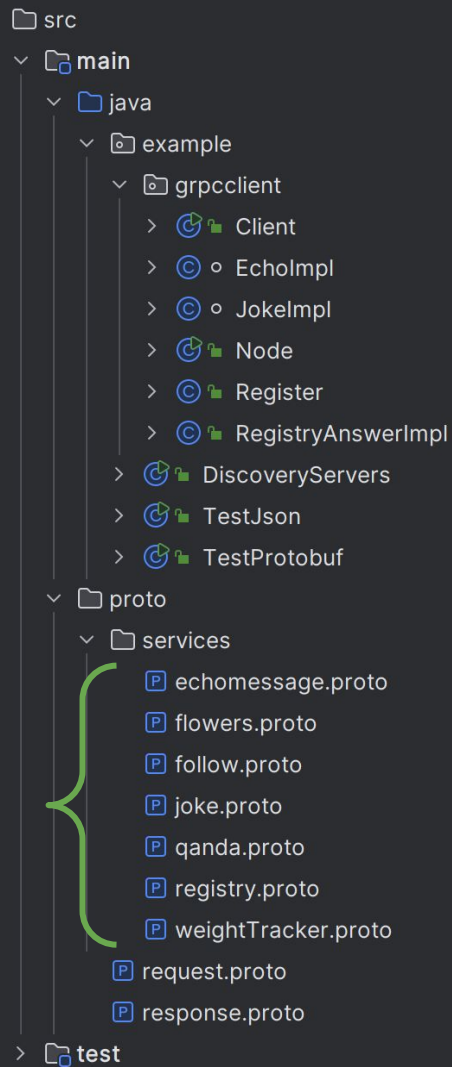
# SER 321

## Protobuf Review

All nodes and clients have agreed to these contracts

So ***DON'T CHANGE THEM!***

Think of these as a *contract*



# SER 321

## Protobuf Review

### joke.proto

```
// We are reading how many jokes the clients wants and put them in a list to send back to client
@Override 1 usage
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()) {
            // yes, I take the joke out when it was used already,
            // should probably be done differently since this way
            // a joke cannot be told twice even to different clients
            response.addJoke(jokes.pop());
        }
        else {
            // this is more of a hack, better would be to either
            // check the number at the beginning and say right away
            // if you do not have enough. Or send an error code or
            // similar as well.
            response.addJoke(value: "I am out of jokes...");
            break;
        }
    }
    JokeRes resp = response.build();
    responseObserver.onNext(resp);
    responseObserver.onCompleted();
}
```

```
// We take the joke the user wants to set and put it in our set of jokes
@Override 1 usage
public void setJoke(JokeSetReq req, StreamObserver<JokeSetRes> responseObserver) {

    System.out.println("Received from client: " + req.getJoke());
    JokeSetRes.Builder response = JokeSetRes.newBuilder();
    if (req.getJoke().isEmpty()) { // we do not want to add empty jokes
        response.setOk(false);
    } else {
        jokes.add(req.getJoke());
        response.setOk(true);
    }

    JokeSetRes resp = response.build();
    responseObserver.onNext(resp);
    responseObserver.onCompleted();
}
```

```
syntax = "proto3";

option java_multiple_files = true;
option java_package = "service";
option java_outer_classname = "JokeProto";

package services;

service Joke {
    rpc getJoke (JokeReq) returns (JokeRes) {}
    rpc setJoke (JokeSetReq) returns (JokeSetRes) {}
}

// The request message
message JokeReq {
    int32 number = 1;
```

# SER 321

## Protobuf Review

Use a **Builder** to construct the proto object

Fill with *setters*

Build when done!

joke.proto

How do we use Protobufs again?

```
syntax = "proto3";

option java_multiple_files = true;
option java_package = "service";
option java_outer_classname = "JokeProto";

package services;

service Joke {
  rpc getJoke (JokeReq) returns (JokeRes) {}
  rpc setJoke (JokeSetReq) returns (JokeSetRes) {}
}

// The request message
message JokeReq {
  int32 number = 1;
```

```
// We take the joke the user wants to set and put it in our set of jokes
@Override 1usage
public void setJoke(JokeSetReq req, StreamObserver<JokeSetRes> responseObserver) {

    System.out.println("Received from client: " + req.getJoke());
    JokeSetRes.Builder response = JokeSetRes.newBuilder();
    if (req.getJoke().isEmpty()) { // we do not want to add empty jokes
        response.setOk(false);
    } else {
        jokes.add(req.getJoke());
        response.setOk(true);
    }

    JokeSetRes resp = response.build();
    responseObserver.onNext(resp);
    responseObserver.onCompleted();
}
```



**SER 321**

**Assign 6**

***Two*** new concepts!

Registry

RPC

# SER 321

## Assign 6

Previously...

Registry

Client

Node

Echo

Joke

Node

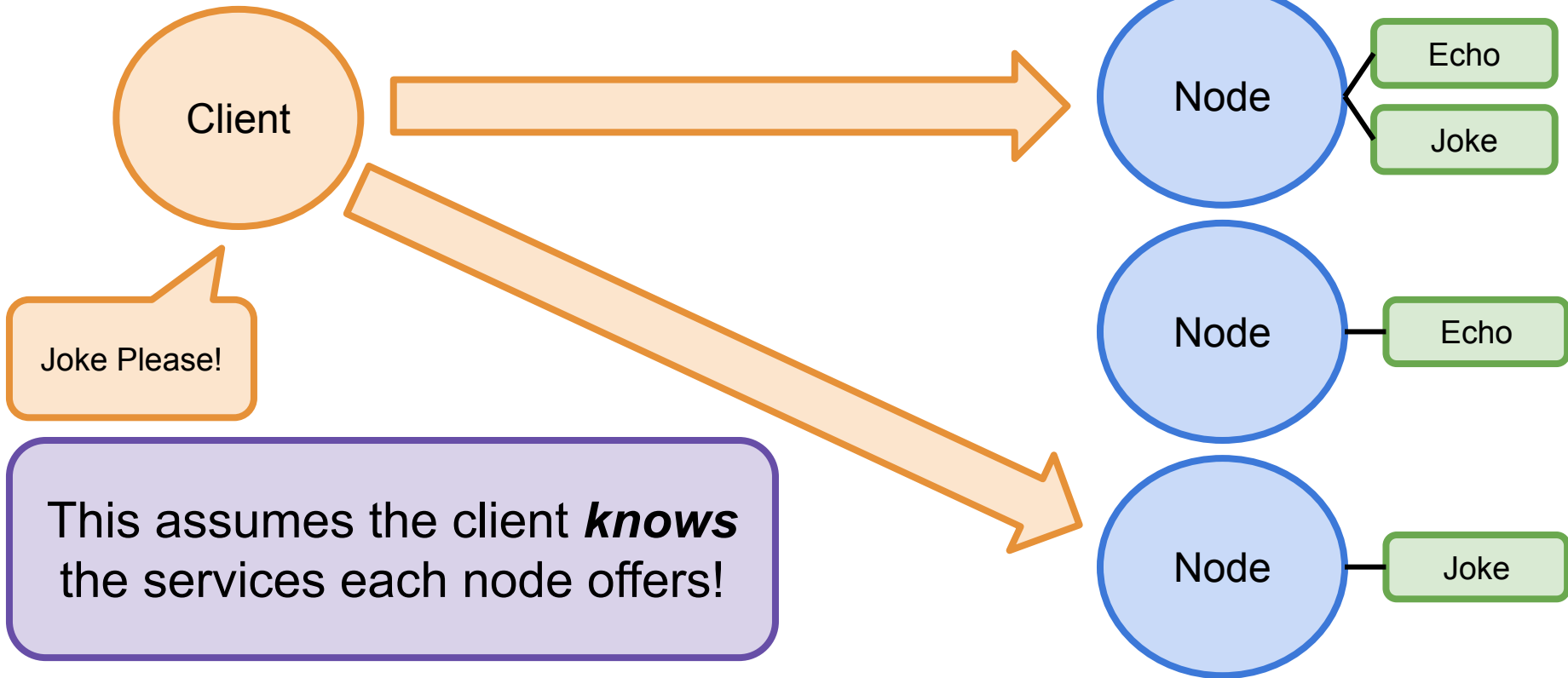
Echo

Node

Joke

Joke Please!

This assumes the client ***knows***  
the services each node offers!



**SER 321**

**Assign 6**

With the Registry...

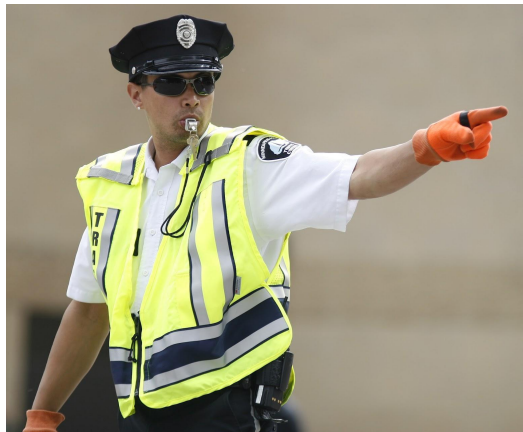
Registry

Client

Joke

Registry

Joke Please!



Node

Echo

Joke

Node

Echo

Node

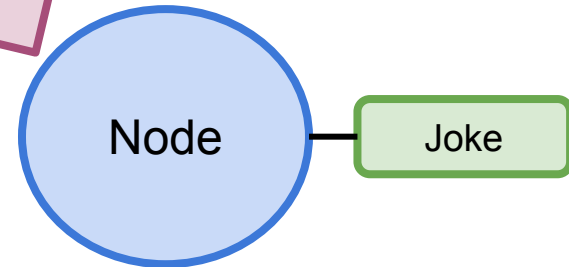
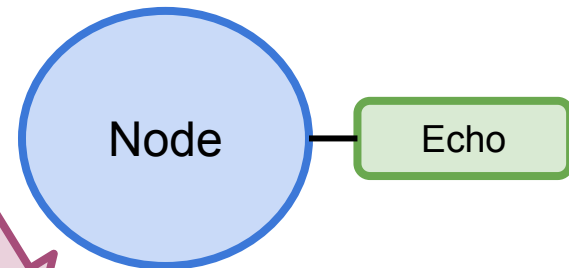
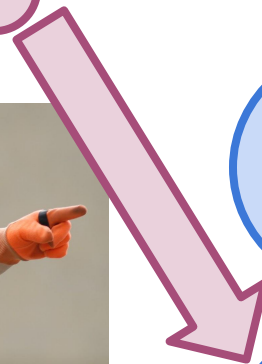
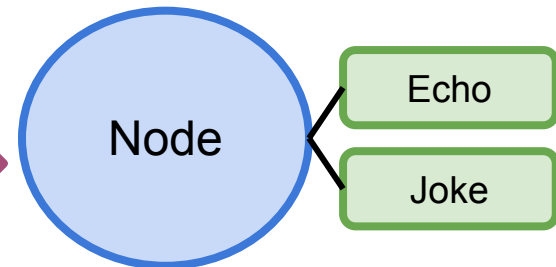
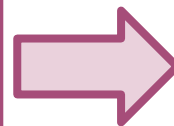
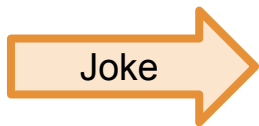
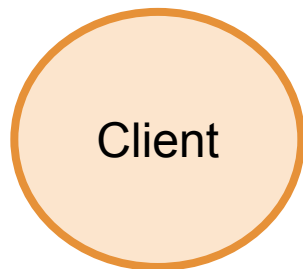
Joke

# SER 321

## Assign 6

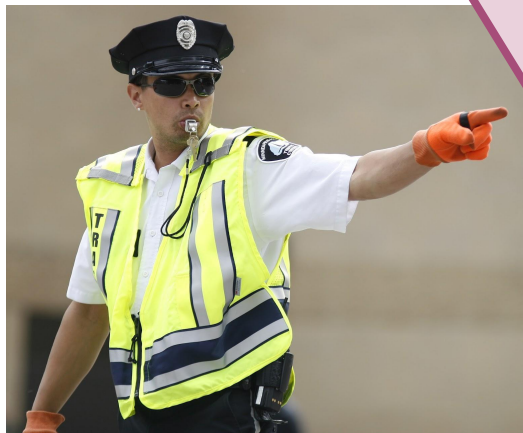
### With the Registry...

Registry



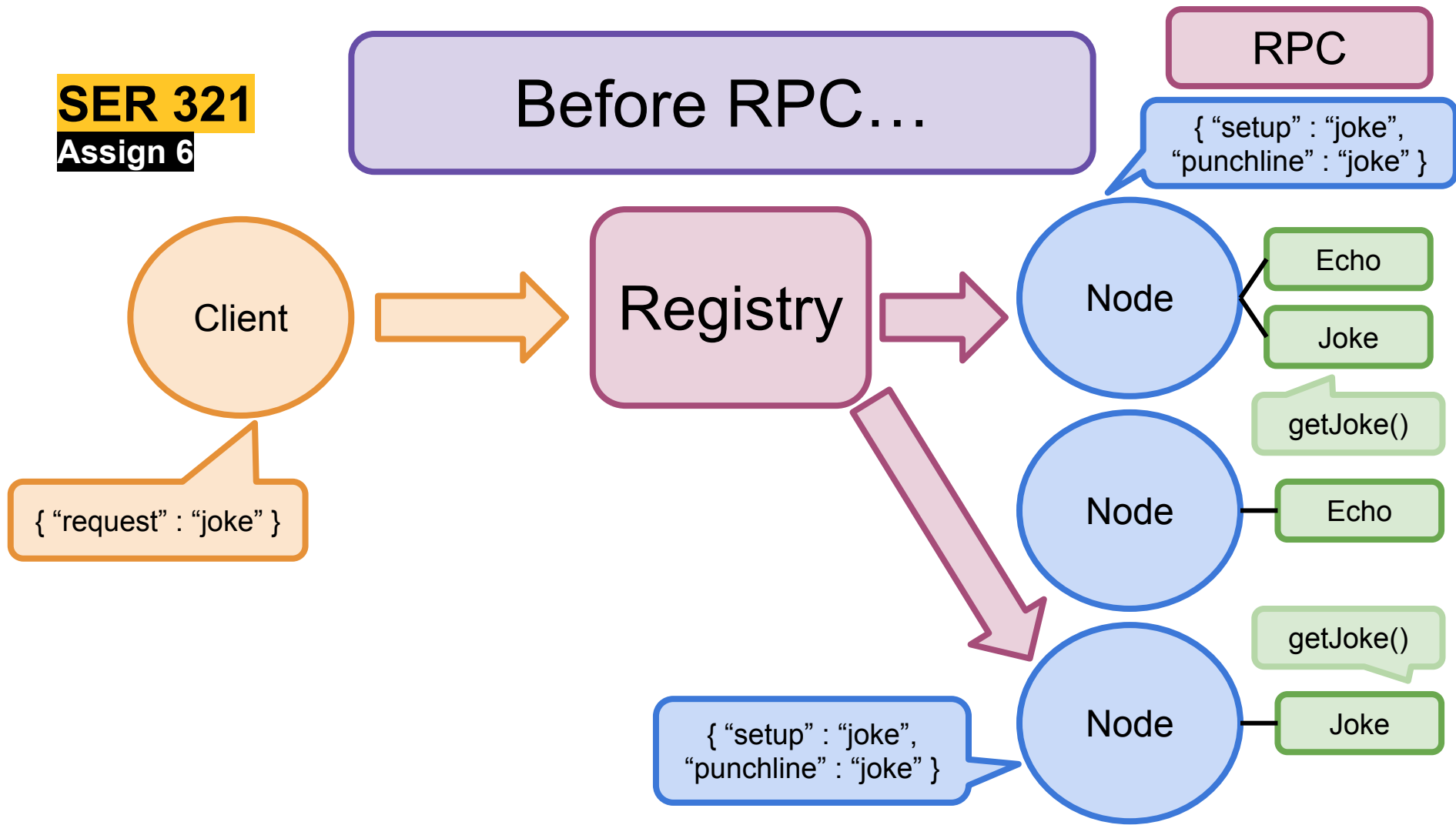
Joke Please!

Registry directs us to a node that can handle our request!



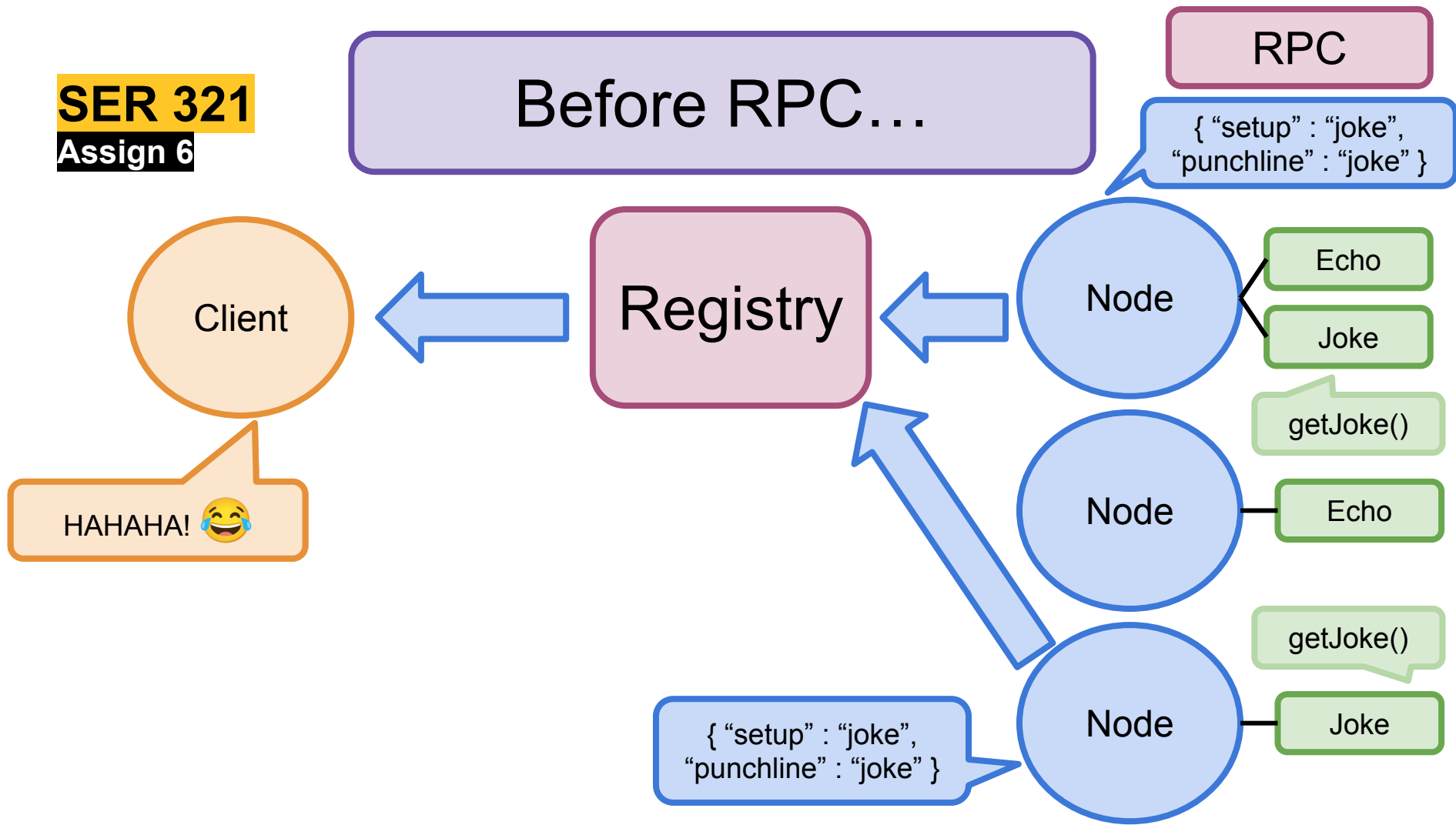
# SER 321

## Assign 6



# SER 321

## Assign 6



**SER 321**

**Assign 6**

*Using RPC...*

RPC

fwd:response

Client

getJoke()

Registry

Node

Echo

Joke

getJoke()

Node

Echo

getJoke()

Node

Joke

fwd:response

# SER 321

## Assign 6

*Using RPC...*

RPC

fwd:response

Client

Registry

Node

Echo

Joke

getJoke()

Node

Echo

getJoke()

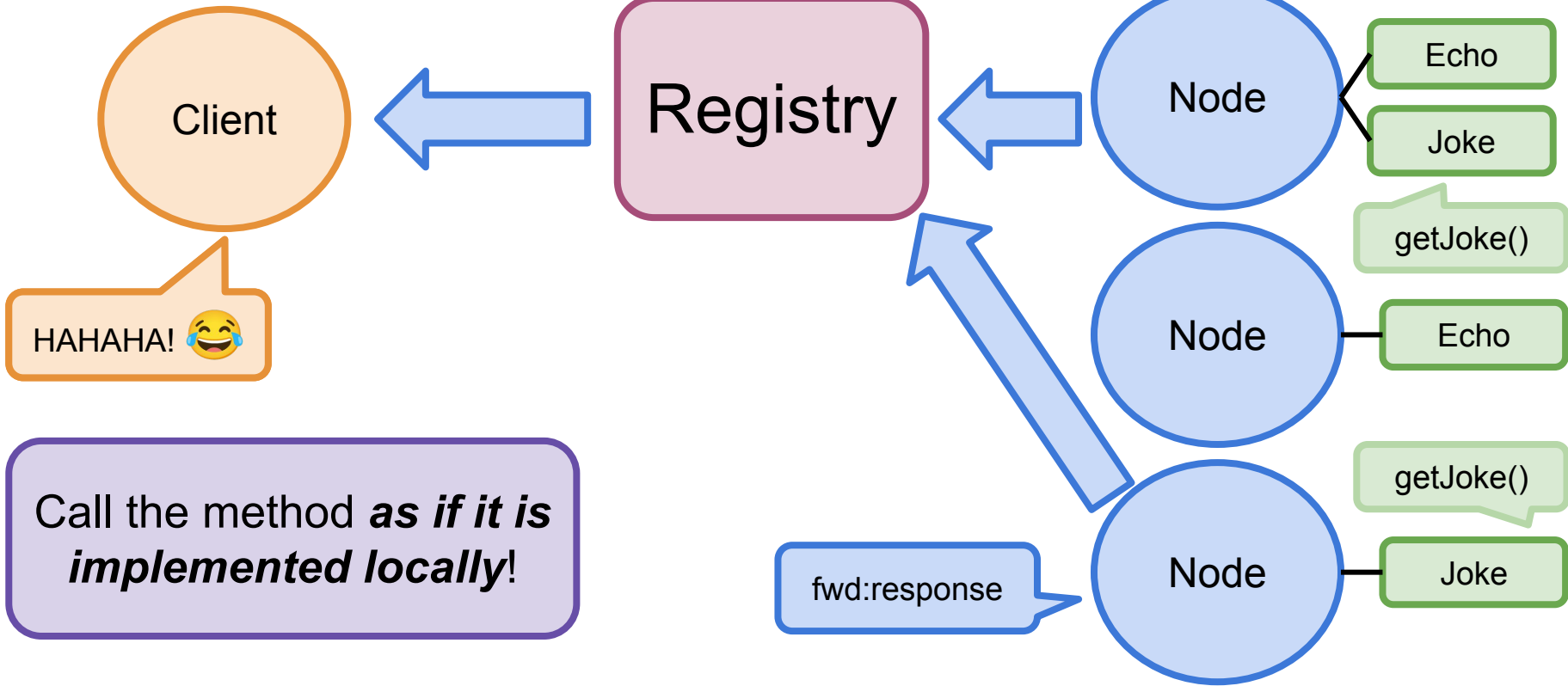
Node

Joke

HAHAHA! 😂

Call the method *as if it is implemented locally!*

fwd:response





# SER 321

## Assign 6

Okay so how do we actually *use* this setup?

```
public void setJoke(String joke) {
    JokeSetReq request = JokeSetReq.newBuilder()
        .setJoke(joke).build();
    JokeSetRes response;

    try {
        response = blockingStub2.setJoke(request);
        System.out.println(response.getOk());
    } catch (Exception e) {
        System.err.println("RPC failed: " + e);
        return;
    }
}
```

Client.java

Looking at **SetJoke**

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

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

```
// ask the user for input how many jokes the user wants
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
```

```
// Reading data using readLine
System.out.println("How many jokes would you like?"); // NO ERROR handling of wrong input here.
String num = reader.readLine();
```

```
// calling the joked service from the server with num from user input
client.askForJokes(Integer.valueOf(num));
```

```
// adding a joke to the server
client.setJoke("I made a pencil with two erasers. It was pointless.");
```

```
// showing 6 joked
client.askForJokes(Integer.valueOf(6));
```

```
// W
@Ove

public void setJoke(JokeSetReq req, StreamObserver<JokeSetRes> responseObserver) {

    System.out.println("Received from client: " + req.getJoke());
    JokeSetRes.Builder response = JokeSetRes.newBuilder();
    if (req.getJoke().isEmpty()) { // we do not want to add empty jokes
        response.setOk(false);
    } else {
        jokes.add(req.getJoke());
        response.setOk(true);
    }

    JokeSetRes resp = response.build();
    responseObserver.onNext(resp);
    responseObserver.onCompleted();
}
```

Client.java (Main)

JokeImpl.java

## SER 321

### Assign 6

Okay so how do we actually *use* this setup?

```
public void setJoke(String joke) {  
    JokeSetReq request = JokeSetReq.newBuilder()  
        .setJoke(joke).build();  
    JokeSetRes response;  
  
    try {  
        response = blockingStub2.setJoke(request);  
        System.out.println(response.getOk());  
    } catch (Exception e) {  
        System.err.println("RPC failed: " + e);  
        return;  
    }  
}
```

Client.java

Client provides  
the info

Client creates  
request

*Everything else we have  
had to do is handled in the  
Implementation Class!*

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

```
// Implement the joke service. It has two services getJokes and setJoke  
class JokeImpl extends JokeGrpc.JokeImplBase { 1 usage
```

JokeImpl.java

```
// having a global set of jokes  
Stack<String> jokes = new Stack<>(); 7 usages
```

```
public JokeImpl(){ 1 usage  
    super();  
    // copying some dad jokes  
    jokes.add("How do you get a squirrel to like you? Act like a nut.");  
    jokes.add("I don't trust stairs. They're always up to something.");  
    jokes.add("What do you call someone with no body and no nose? Nobody knows.");  
    jokes.add("Did you hear the rumor about butter? Well, I'm not going to spread it!");  
}
```

```
// When the client asks for jokes  
@Override  
client.askForJokes(Integer.valueOf(6));
```

```
public void setJoke(JokeSetReq req, StreamObserver<JokeSetRes> responseObserver) {  
  
    System.out.println("Received from client: " + req.getJoke());  
    JokeSetRes.Builder response = JokeSetRes.newBuilder();  
    if (req.getJoke().isEmpty()) { // we do not want to add empty jokes  
        response.setOk(false);  
    } else {  
        jokes.add(req.getJoke());  
        response.setOk(true);  
    }  
  
    JokeSetRes resp = response.build();  
    responseObserver.onNext(resp);  
    responseObserver.onCompleted();  
}
```

JokeImpl.java

## SER 321

### Assign 6

What does that imply  
for the system?

```
public void setJoke(String joke) {  
    JokeSetReq request = JokeSetReq.newBuilder()  
        .setJoke(joke).build();  
    JokeSetRes response;
```

Client.java

```
try {  
    response = blockingStub2.setJoke(request);  
    System.out.println("Joke set successfully");  
} catch (Exception e) {  
    System.out.println("Error setting joke: " + e.getMessage());  
}
```

Client Class acts like a  
middleman!

*Everything else we have  
had to do is handled in the  
Implementation Class!*

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

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

```
// ask the user for input how many jokes the user wants  
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
```

```
// Reading data using readLine  
System.out.println("How many jokes would you like?"); // NO ERROR handling of wrong input here.  
String num = reader.readLine();
```

```
// calling the joked service from the server with num from user input  
client.askForJokes(Integer.valueOf(num));
```

```
// adding a joke to the server  
client.setJoke("I made a pencil with two erasers. It was pointless.");
```

```
// showing 6 joked  
client.askForJokes(Integer.valueOf(6));
```

Client.java (Main)

```
@Override  
public void setJoke(JokeSetReq req, StreamObserver<JokeSetRes> responseObserver) {
```

```
    System.out.println("Received from client: " + req.getJoke());  
    JokeSetRes.Builder response = JokeSetRes.newBuilder();  
    if (req.getJoke().isEmpty()) { // we do not want to add empty jokes  
        response.setOk(false);  
    } else {  
        jokes.add(req.getJoke());  
        response.setOk(true);  
    }  
}
```

JokeImpl.java

Implementations need to  
be robust and thorough!

# SER 321

## Middleware

We have been:

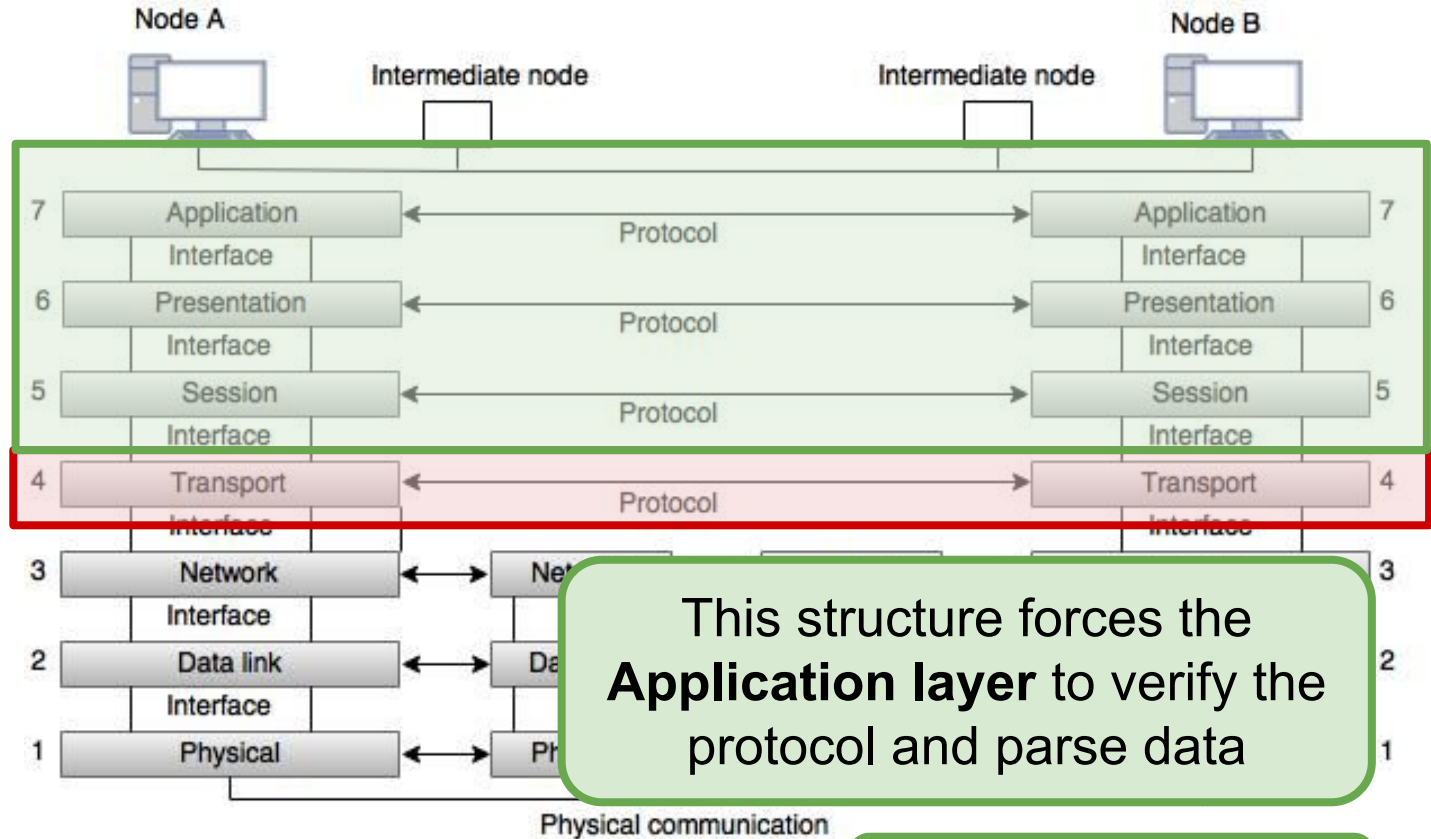
Serializing  
Messages

Sending  
Messages

Parsing  
Messages

Handle  
Messages

*Check out the recording for the discussion!*



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

Fig: OSI Model

Not really its job...

*Check out the recording for the discussion!*

# SER 321

## Middleware

With Middleware:

Serializing  
Messages

Sending  
Messages

Parsing  
Messages

Handle  
Messages

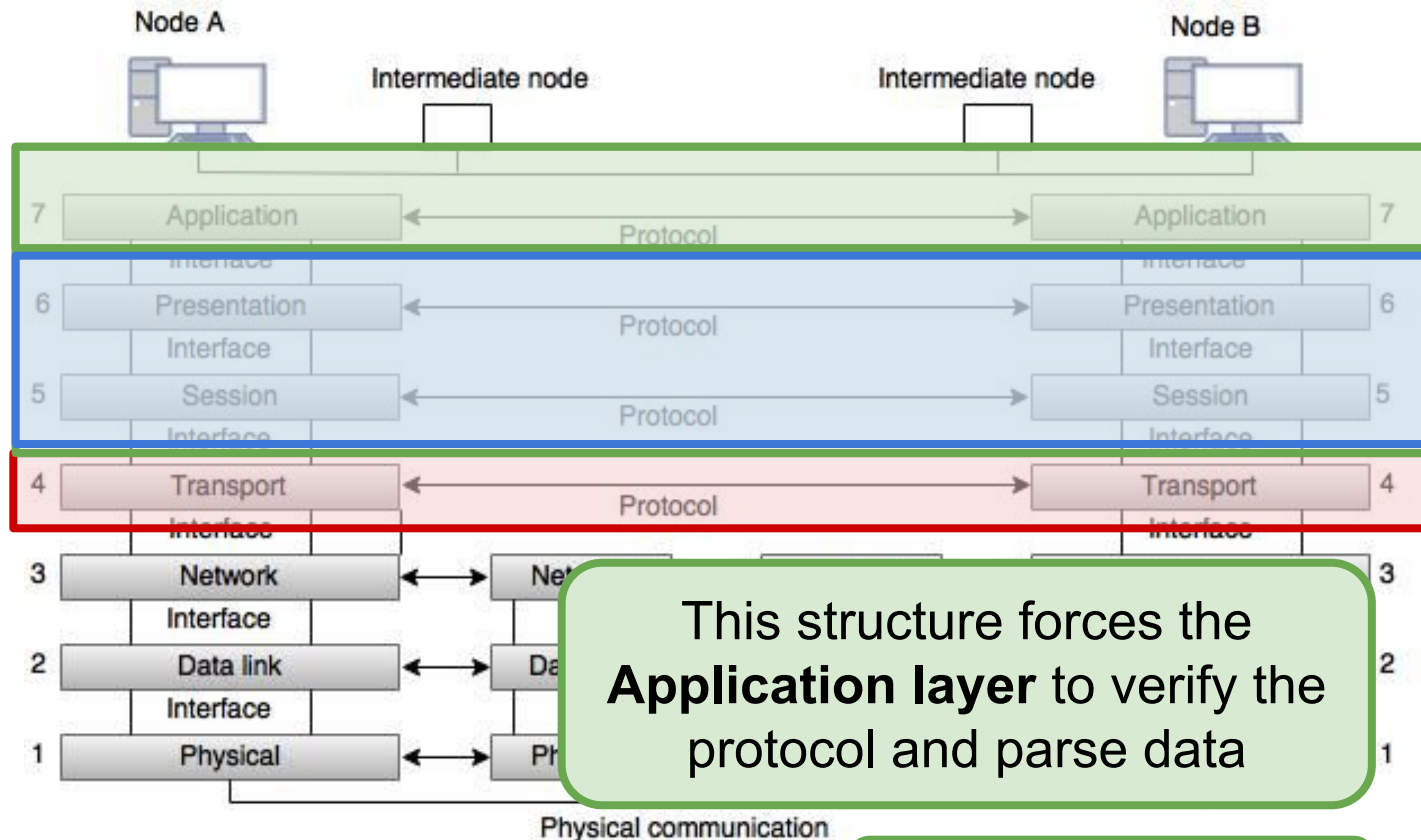


Fig: OSI Model

Not really its job...

# SER 321

## Middleware

With Middleware:

Serializing  
Messages

Sending  
Messages

Parsing  
Messages

Handle  
Messages

*Check out the recording for the discussion!*

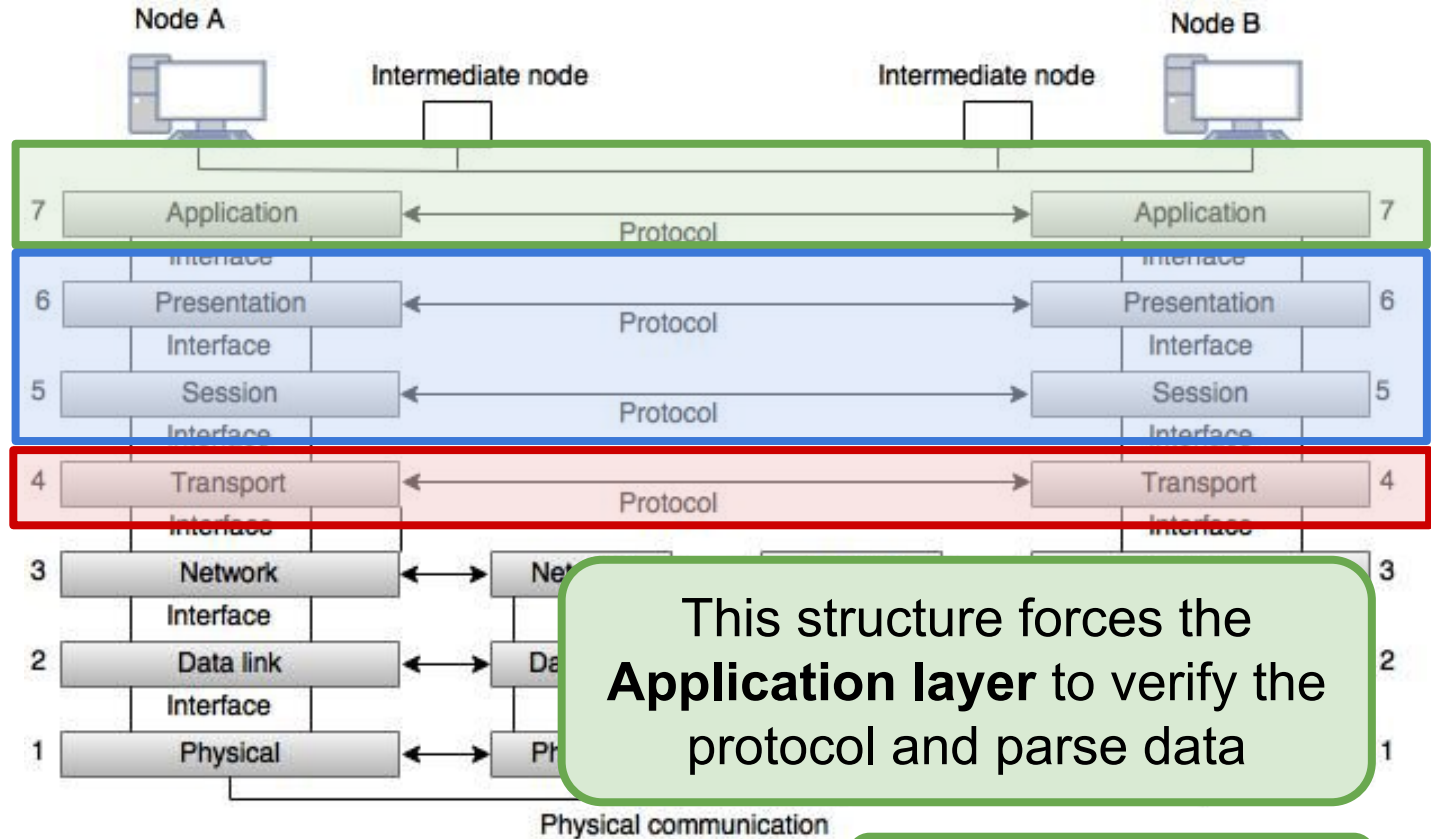


Fig: OSI Model

Not really its job...



# Middleware

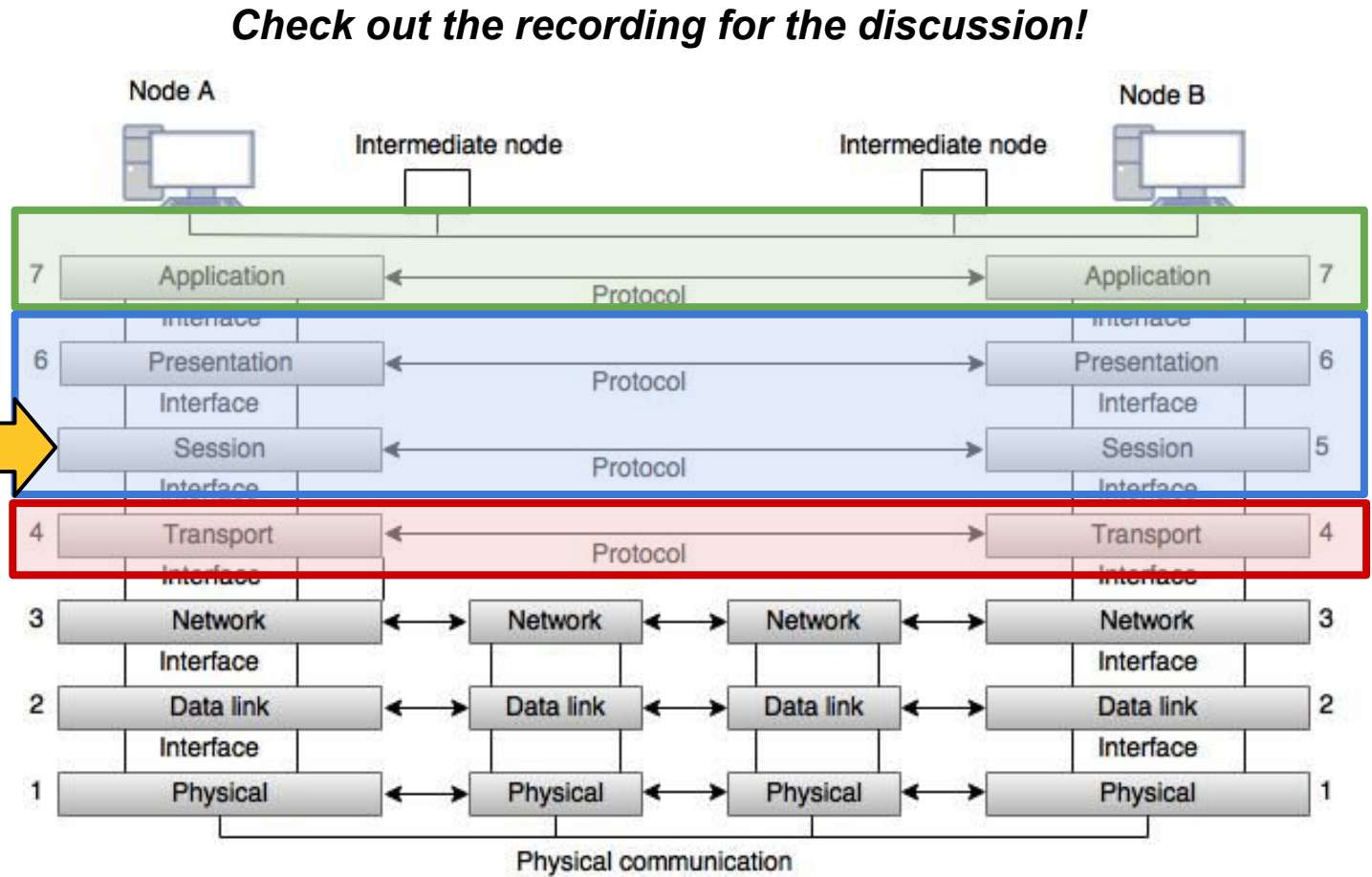
## Middleware:

## Session Layer Responsibilities:

# Authentication

# Authorization

# Session Management



**Fig: OSI Model**

*Check out the recording for the discussion!*

## SER 321 Middleware

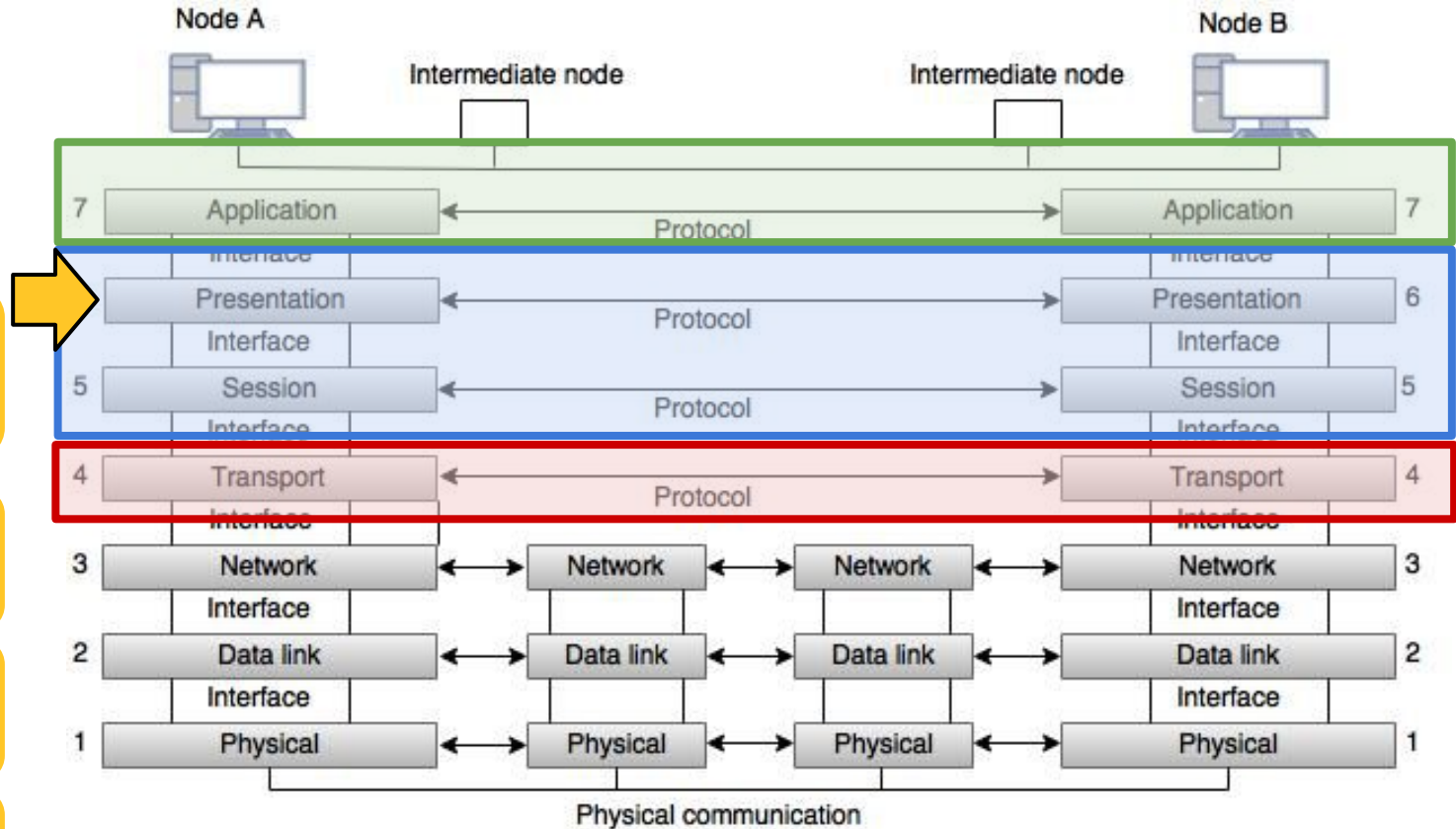
Middleware:

*Presentation  
Layer  
Responsibilities:*

Translation

Compression

Encryption



**Fig: OSI Model**



**SER 321**

**Middleware Benefits**

Why do we care?

Agility

Reusability

Efficiency

Cost  
Effectiveness

Portability

**SER 321**

**Systems**

**Parallel**



A Venn diagram with two overlapping circles. The left circle is light blue with a blue outline and is labeled 'Parallel'. The right circle is light red with a red outline and is labeled 'Distributed'. The intersection of the two circles is shaded with a mix of blue and red. The text 'SER 321' is in a yellow box at the top left, and 'Systems' is in a black box below it.

**Distributed**

### Parallel

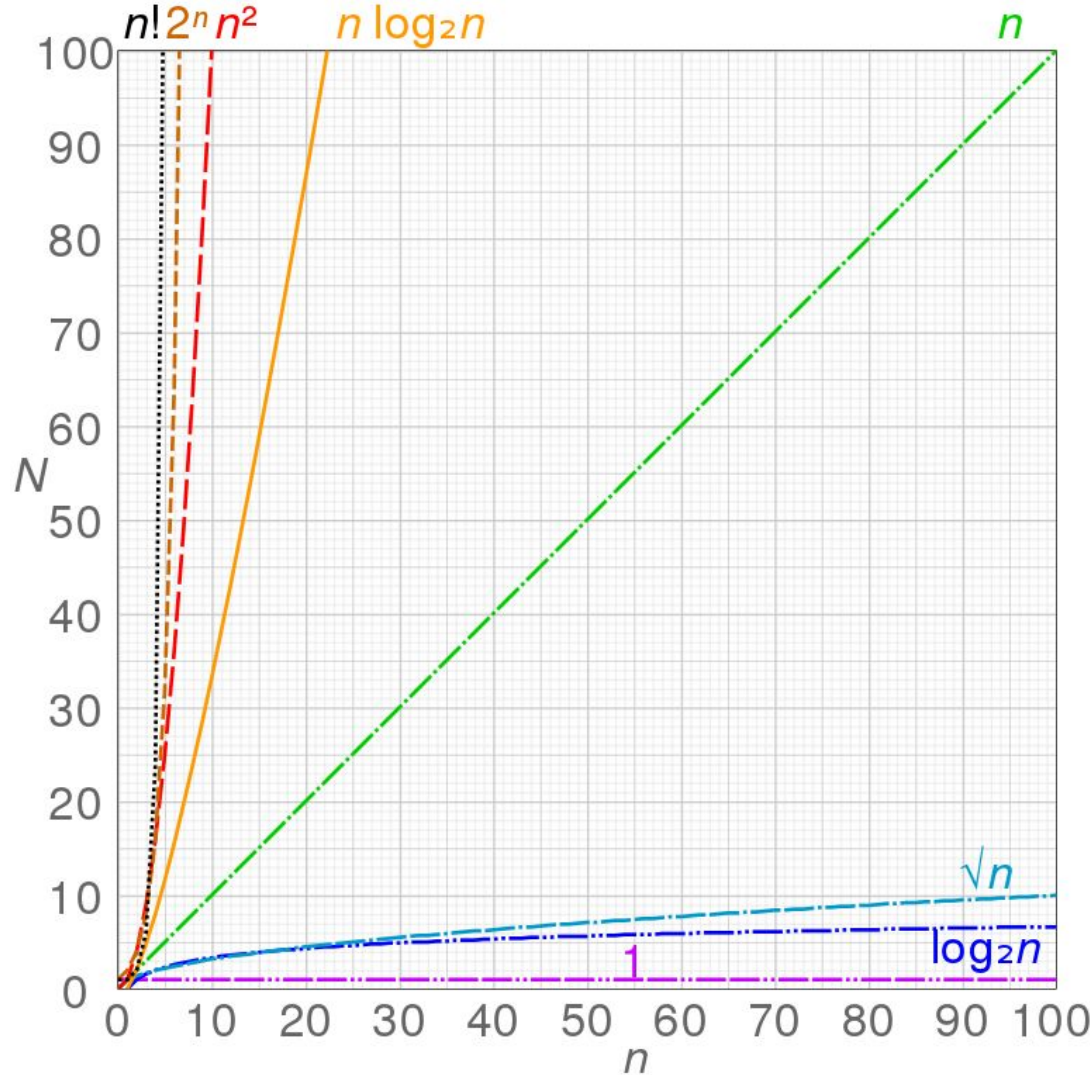
- Single computer
- Work split among different *processors*
- Memory is shared **or** distributed
- Communicate through *bus*

### Distributed

- Work is partitioned
- Partitions processed individually
- **Can** improve performance
- **Can** improve speed
- Many computers
- Work split among different *locations*
- Memory is distributed
- Communicate through *message passing*

## When to Distribute

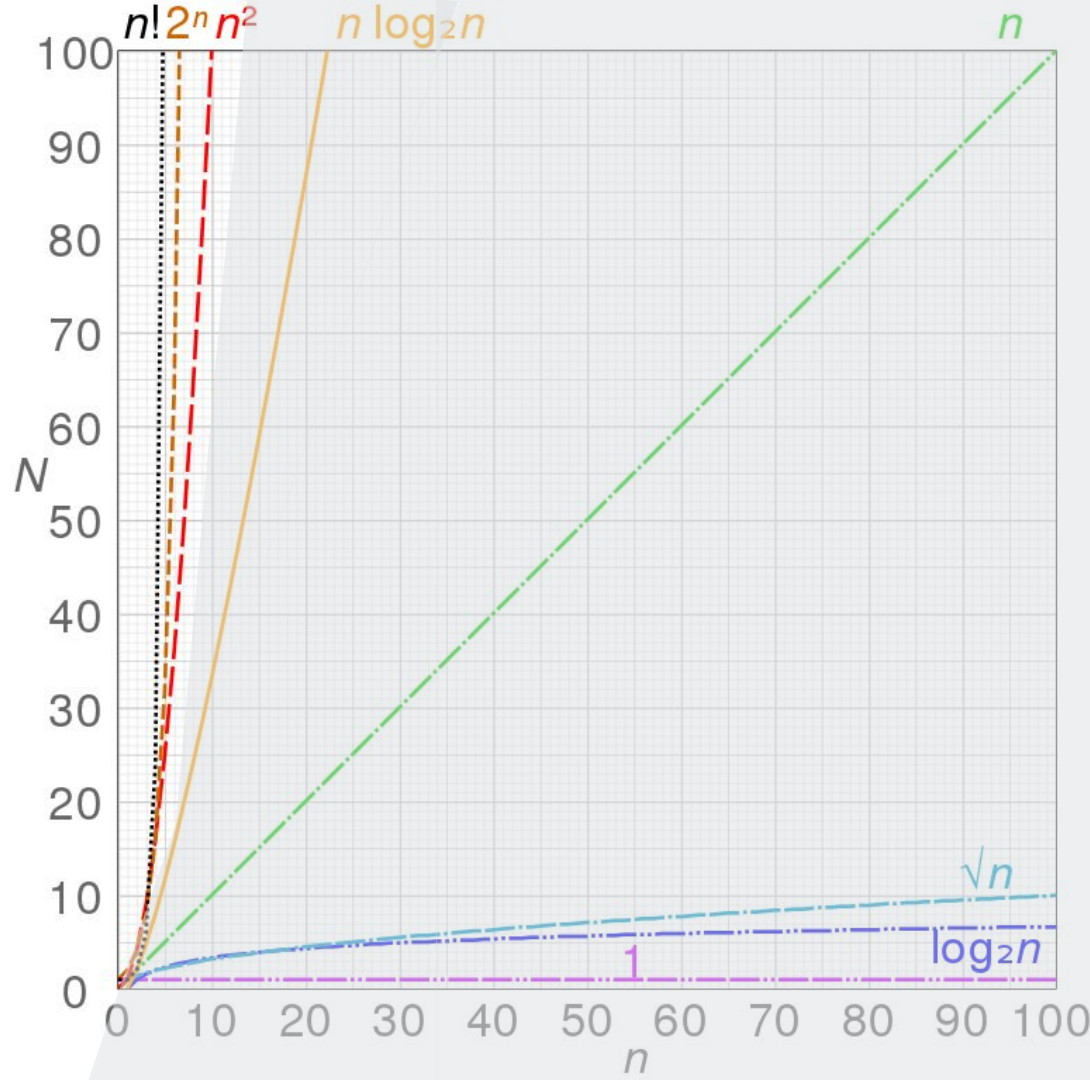
# When should we *consider* distributing?



## When to Distribute

When should  
we *consider*  
distributing?

Super Duper Extra Extra  
Large Orders of Magnitude!

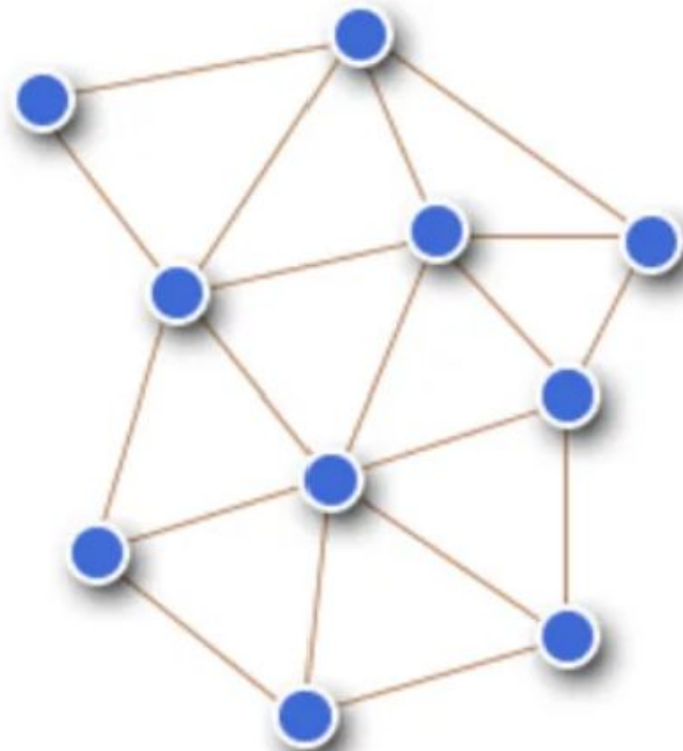


# SER 321

## Distributed Issues

Remember that we are operating in *reality*

- Nodes *will* fail
- Web of nodes *will constantly* change
- Network is not *always* reliable
- Latency is *always present*
- The path traversed *changes*
- Some resources *must be shared*
- *You* need to prevent the pitfalls!
  - No deadlocks
  - No starvation
  - No error states



# SER 321

## Consensus

“General agreement or trust amongst a group”

## What is Consensus?

Who's in charge or keeping the beat



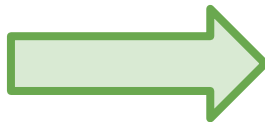
Leader Election

Check your work with a neighbor



Result Verification

Verify and maintain my copy of the data



Log Replication

Do I want to let you into my network

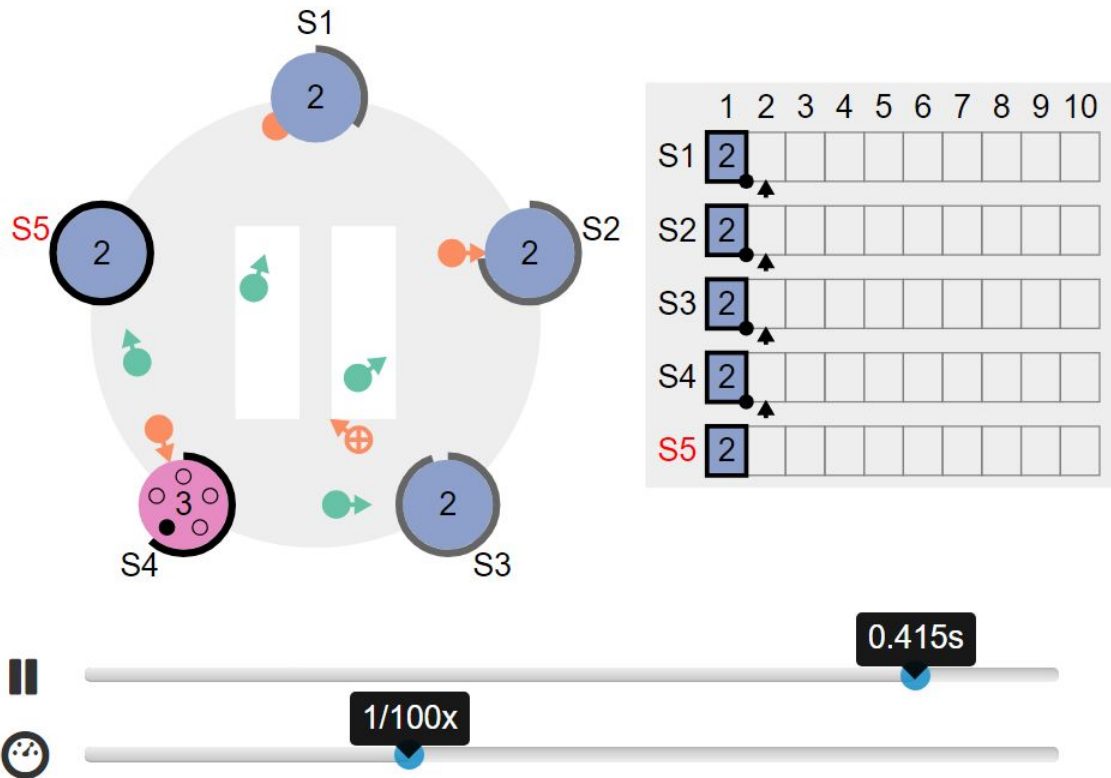


Validate Nodes

**RAFT**

# How do we feel about Consensus?

## RAFT





**SER 321**

**Scratch Space**

# Questions?



## Survey:

<http://bit.ly/ASN2324>



## Upcoming Events

### SI Sessions:

- N/A - Good luck on the final, you've got this! 🙌

### Review Sessions:

- Sunday, December 1st at 7:00 pm MST - **2 hour Review Session**
- Tuesday, December 3rd at 10:00 am MST - **Q&A Session**

# 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

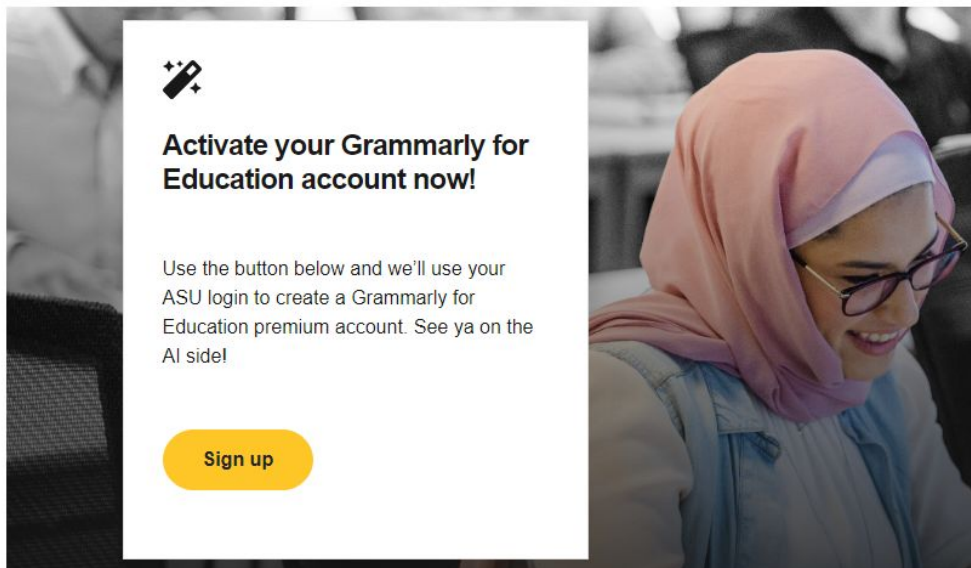
Computer Applications and Information Technology

 [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!



[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](#)