

SER 321 B Session

Exam Review Session

Sunday, April 27th 2025

6:00 pm - 8:00 pm MST

Agenda



Exam Information

Study Guide PSA

General 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
- 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 30th
@ 12:30 AM

Closes: Friday
May 2nd
@ 11:59:59 PM

Make sure to look at
the Study Guide!

Front and Back!

MUST BE Handwritten

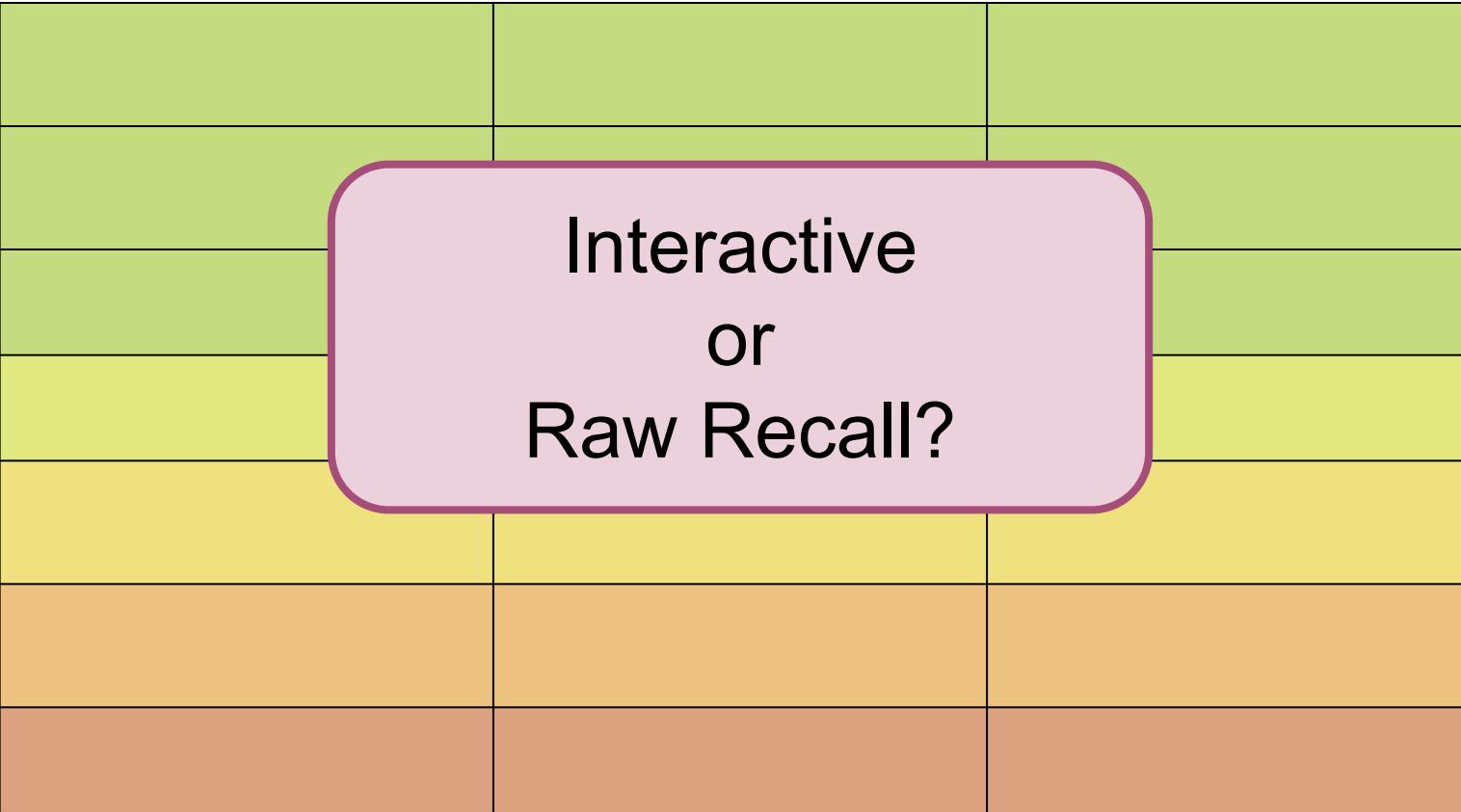


SER 321

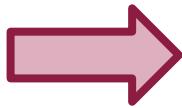
Review Requests

It's *not* too late to make a topic request!

Drop a concept in the chat
and we can cover it next!



Interactive
or
Raw Recall?



SER 321

OSI Model

Unit

Layer

What we are *really*
talking about

Bits	Physical	Signal, Binary transmission



SER 321

OSI Model

Unit

Layer

What we are *really*
talking about

Frame	Data Link	LLC, MAC, data transmission in LAN
Bits	Physical	Signal, Binary transmission



SER 321

OSI Model

Unit

Layer

What we are *really*
talking about

Packet	Network	IP address, routing and delivery
Frame	Data Link	LLC, MAC, data transmission in LAN
Bits	Physical	Signal, Binary transmission



Segment	Transport	TCP/UDP
Packet	Network	IP address, routing and delivery
Frame	Data Link	LLC, MAC, data transmission in LAN
Bits	Physical	Signal, Binary transmission



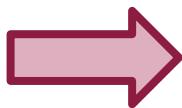


Data	Session	AuthN, authZ, session mgmt
Segment	Transport	TCP/UDP
Packet	Network	IP address, routing and delivery
Frame	Data Link	LLC, MAC, data transmission in LAN
Bits	Physical	Signal, Binary transmission

SER 321**OSI Model**

Unit

Layer

What we are *really*
talking about

Data	Presentation	Translation, compression, encryption
Data	Session	AuthN, authZ, session mgmt
Segment	Transport	TCP/UDP
Packet	Network	IP address, routing and delivery
Frame	Data Link	LLC, MAC, data transmission in LAN
Bits	Physical	Signal, Binary transmission

SER 321

Network Layer - IP

Given the following IP address, identify the...

Port

128.148.32.110 **8080**

SER 321

Network Layer - IP

Given the following IP address, identify the...

Subnet

128.148.32.110:8080

SER 321

Network Layer - IP

Given the following IP address, identify the...

Network

128.148.32.110:8080

SER 321

Network Layer - IP

Given the following IP address, identify the...

Host

128.148.32.110:8080

SER 321

TCP & UDP

What are the main differences?

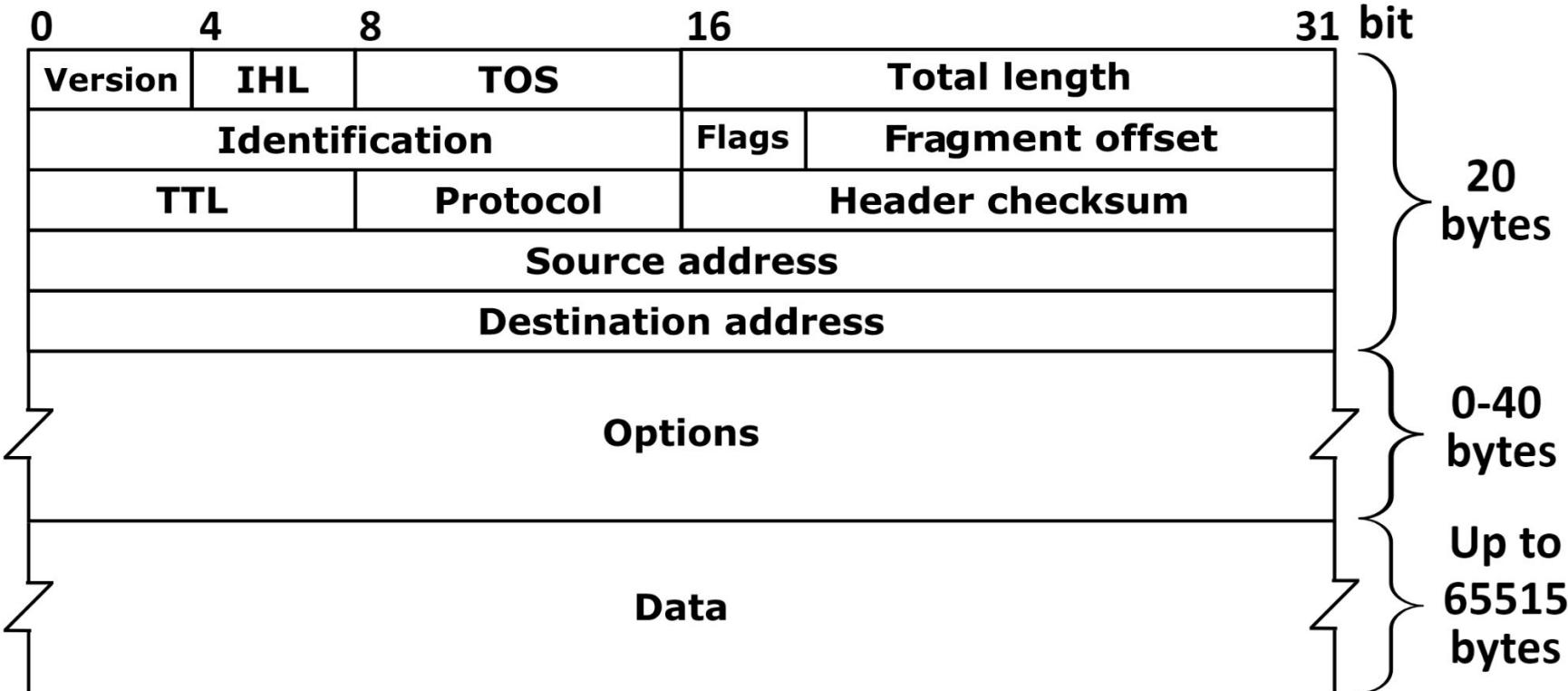
TCP	UDP

SER 321

Headers

What type of header is this?

IP Header



TCP or UDP Header?

TCP or UDP Header?

SER 321

Network Concept Checks

What is an ARP table?

Where does it fit in the OSI model?

Interface: 192.168.0.68	---	0x8
Internet Address	Physical Address	
192.168.0.1	ce:83:f	
192.168.0.21	f7:f5:1	
192.168.0.255	ff-ff-f	
224.0.0.22	04-00-5	
224.0.0.251	04-00-5	
224.0.0.252	04-00-5	
239.255.255.250	04-00-5	
255.255.255.255	...	

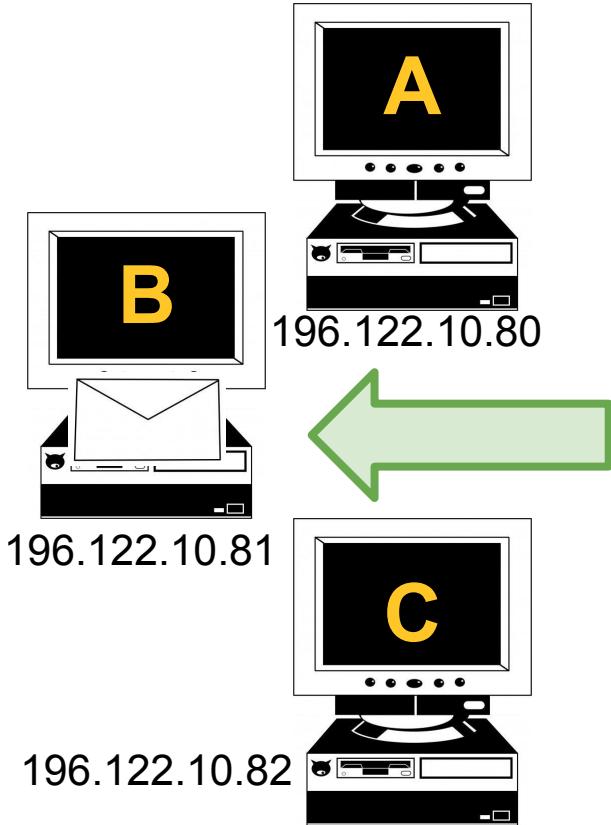


Note - numbers have been modified

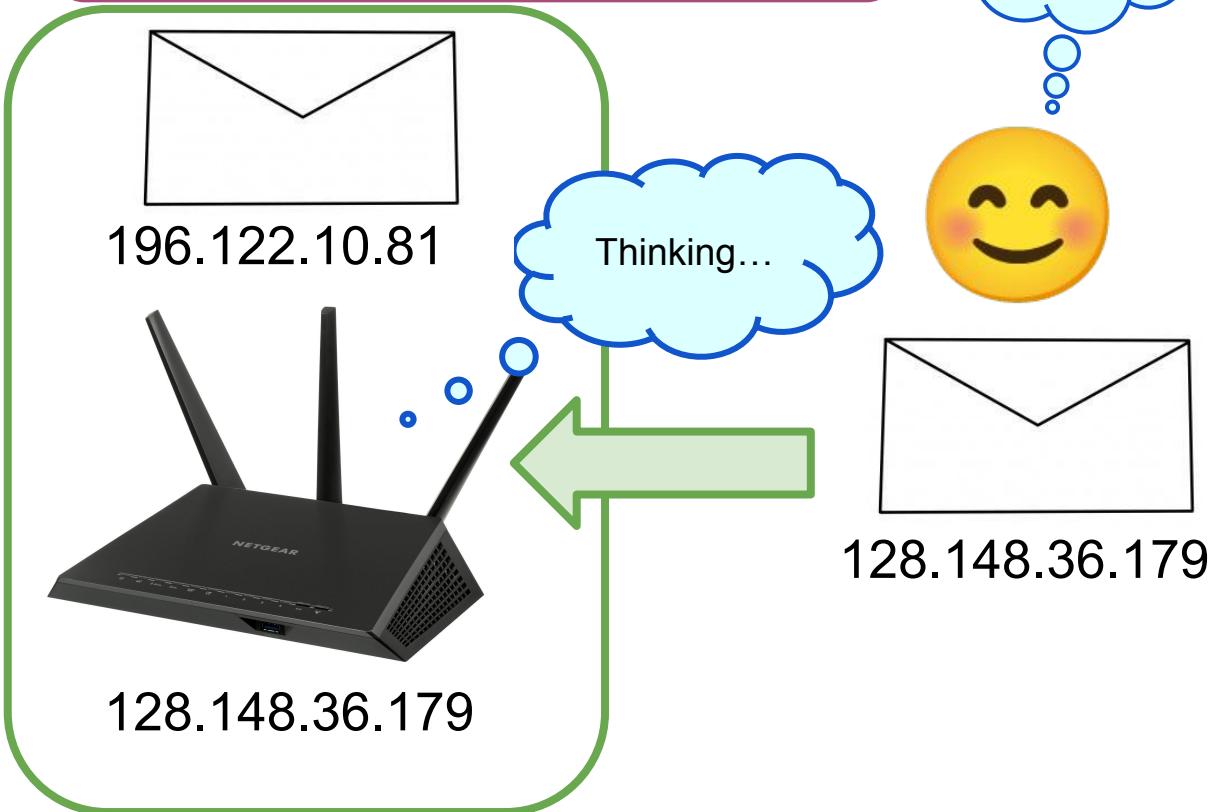
Data	Application	HTTP(s), SMTP, FTP, IMAP, POP, etc.
Data	Presentation	Translation, compression, encryption
Data	Session	AuthN, authZ, session mgmt
Segment	Transport	TCP/UDP
Packet	Network	IP address, routing and delivery
Frame	Data Link	LLC, MAC, data transmission in LAN
Bits	Physical	Signal, Binary transmission

SER 321

Network Concept Checks



What is NAT?



TO:
Device
B

SER 321

URLs

Given the following URL, identify the...

Query

<https://www.google.com/search?q=asu>

SER 321

URLs

Given the following URL, identify the...

Protocol

`https://www.google.com/search?q=asu`

SER 321

URLs

Given the following URL, identify the...

Path

`https://www.google.com/search?q=asu`

SER 321

URLs

Given the following URL, identify the...

Host

`https://www.google.com/search?q=asu`

SER 321

HTTP(s)



Stateful

OR

Stateless

Synchronous

OR

Asynchronous

What's the difference?

1. GET

2. POST

3. PUT

4. DELETE

SER 321

HTTP Responses

Status Codes →

1XX

2XX

3XX

4XX

5XX

SER 321

Lower Layer Matching

Let's do a quick
matching game!

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

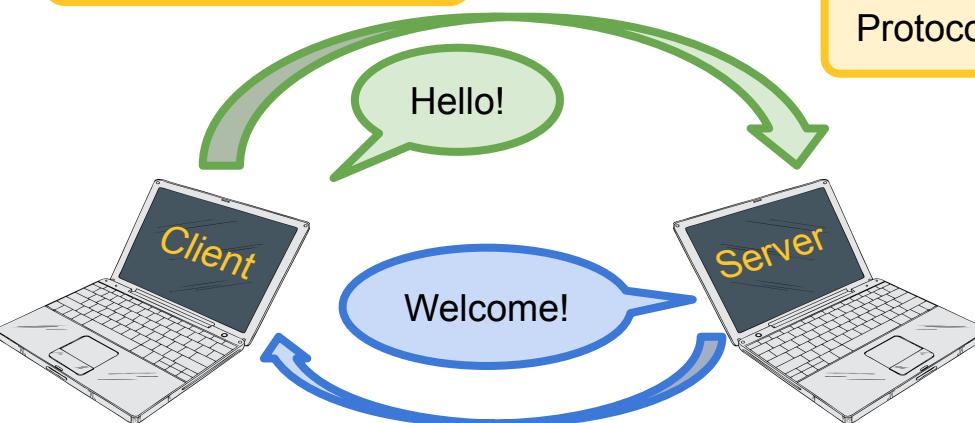
Stateful

Binary

Text

Headers

No Headers



SER 321

Sockets!

Sockets allow our client and server to communicate!

Person

Conversation Flow

Conversation Content

I to define **3 properties** before usage

IP or DNS

TCP or UDP

Protocol Specs

142.251.46.206

Connection Oriented

Synchronous

Stateless

Binary

Headers

www.google.com

Connectionless

Asynchronous

Stateful

Text

No Headers



SER 321

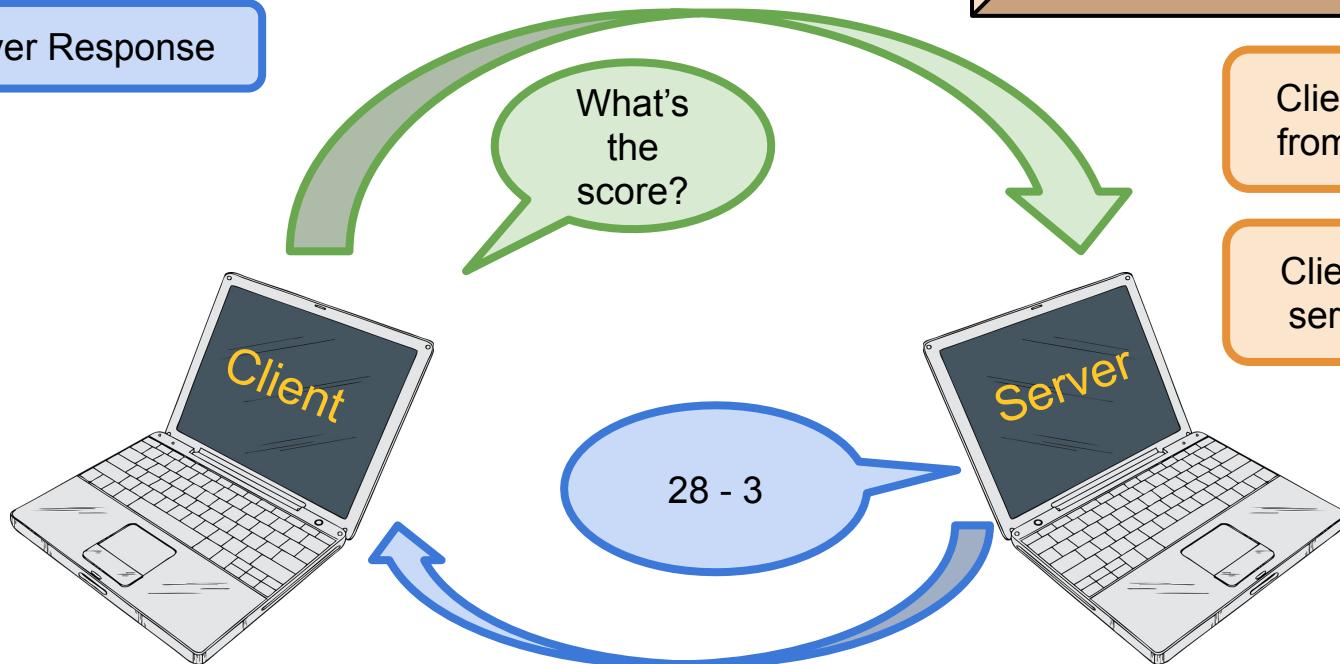
Socket Protocol Types

Two Main Conversation Models

1. Client Request

Pull/Polling Model

2. Server Response



Client *pulls* info from the server

Client *polls* the server for info

SER 321

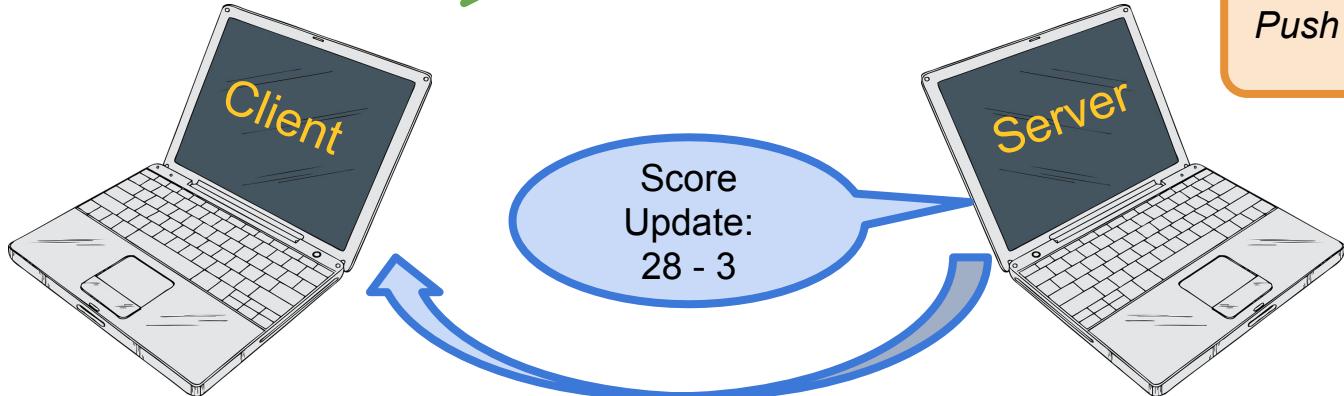
Socket Protocol Types

Two Main Conversation Models

1. Server sends update

2. Client acknowledges

Push Model



*Server pushes
info to client*

Push notifications

SER 321

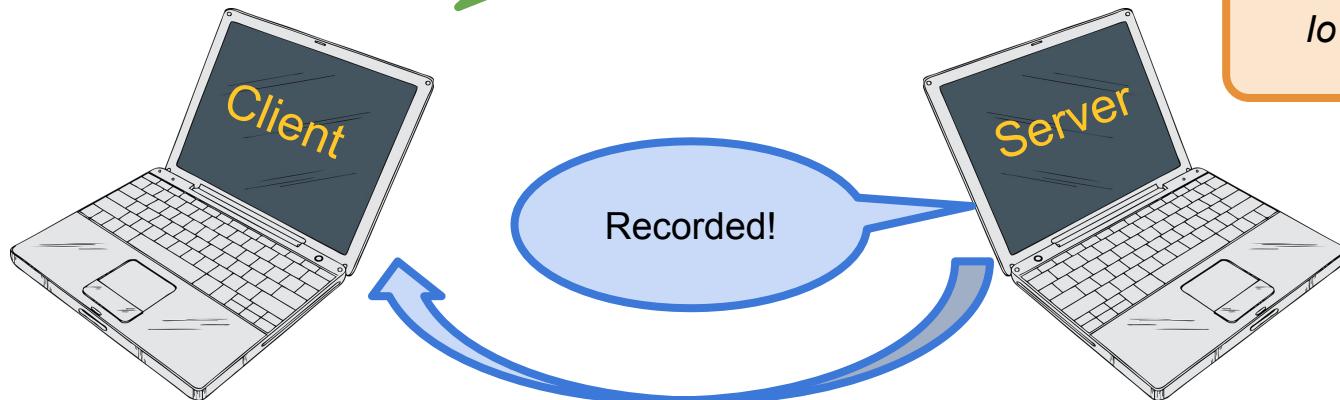
Socket Protocol Types

Two Main Conversation Models

1. Client sends update

2. Server acknowledges

Client Push Model



Client pushes info to Server

IoT sensors

SER 321

Client Socket

Steps for the **Client Socket**

1.

2.

3.

4.

5.

6.

7.

8.

Interactive
or
Raw Recall?

SER 321

Server Socket

Steps for the **Server Socket**

1.

2.

3.

4.

5.

6.

7.

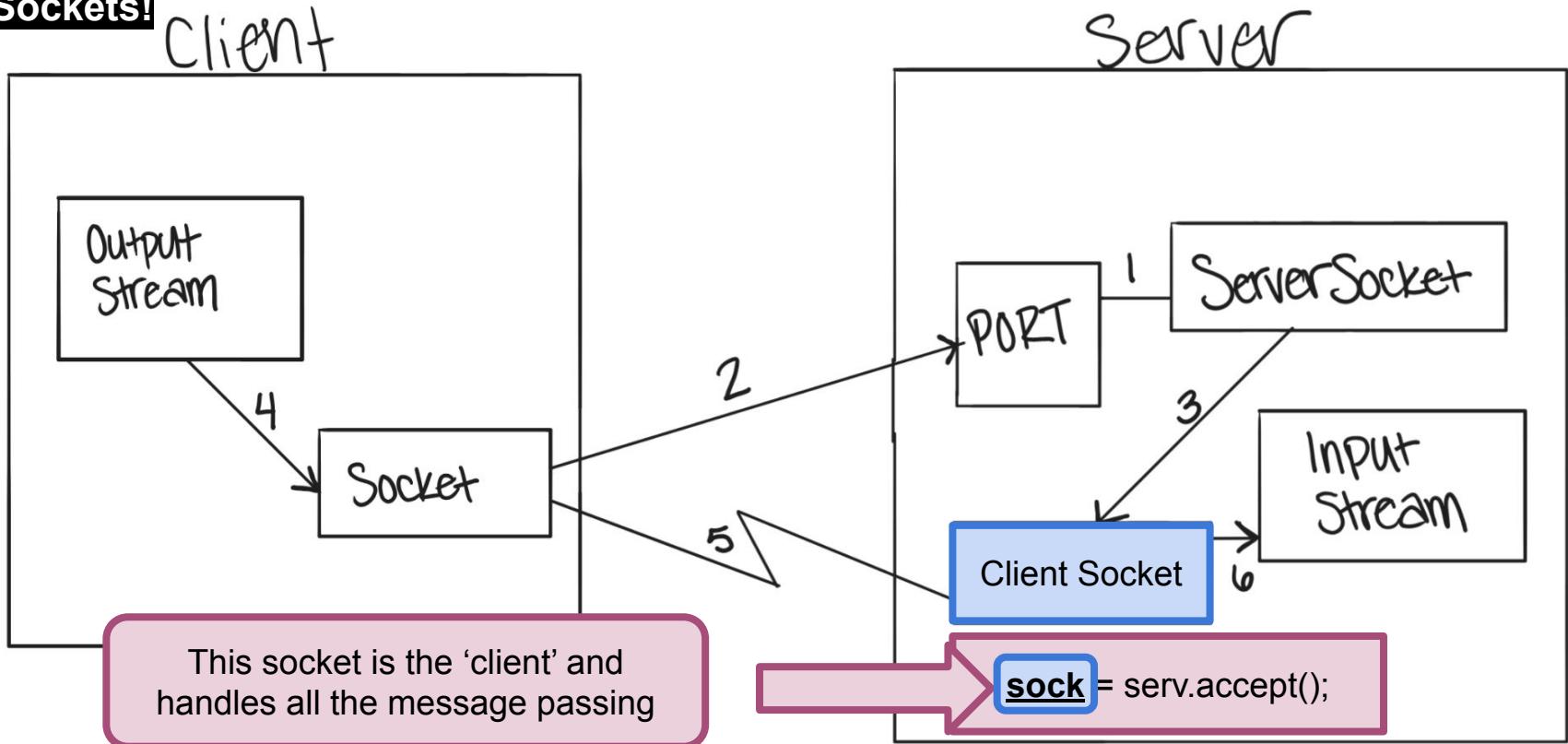
8.

9.

SER 321

Sockets!

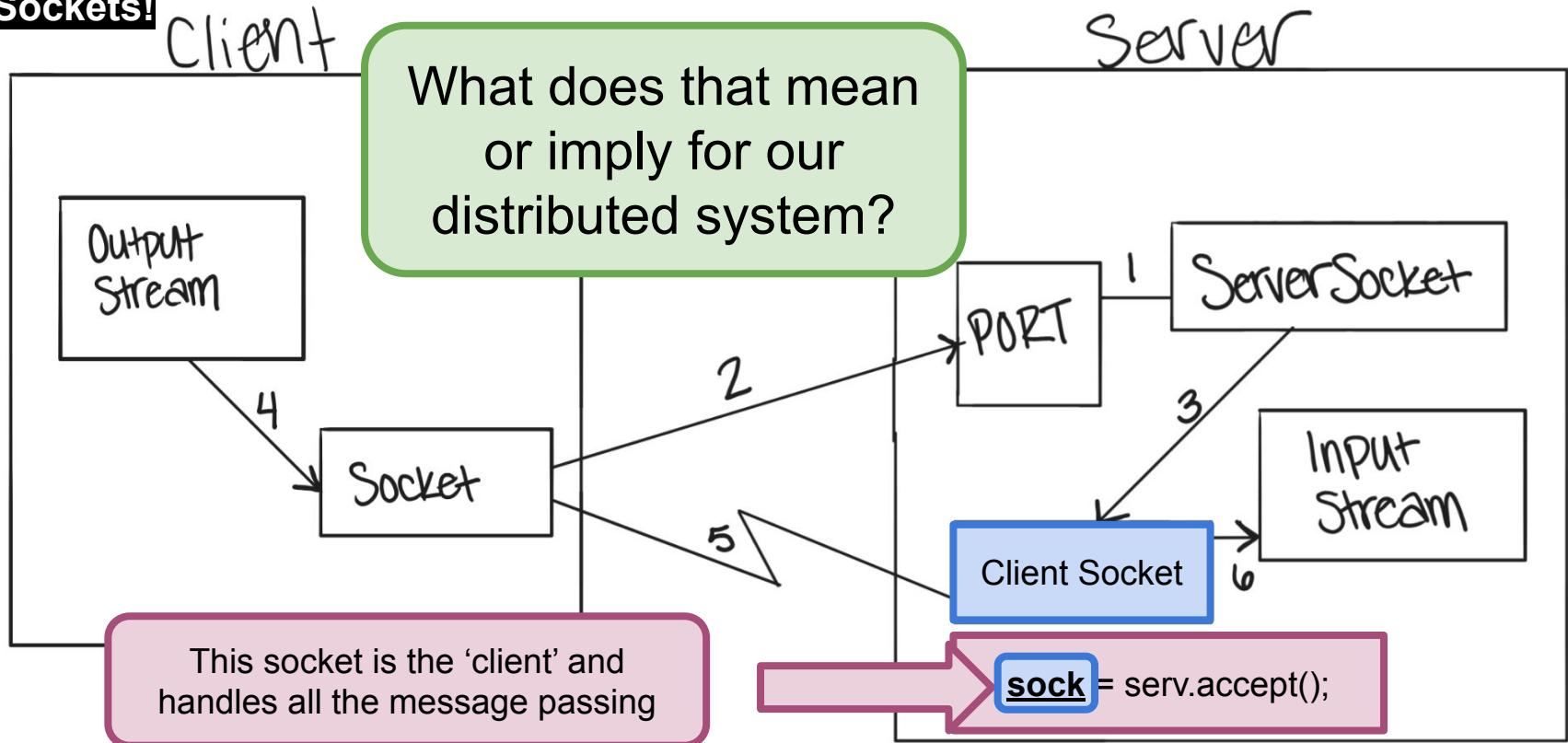
Why do we keep reviewing this?



SER 321

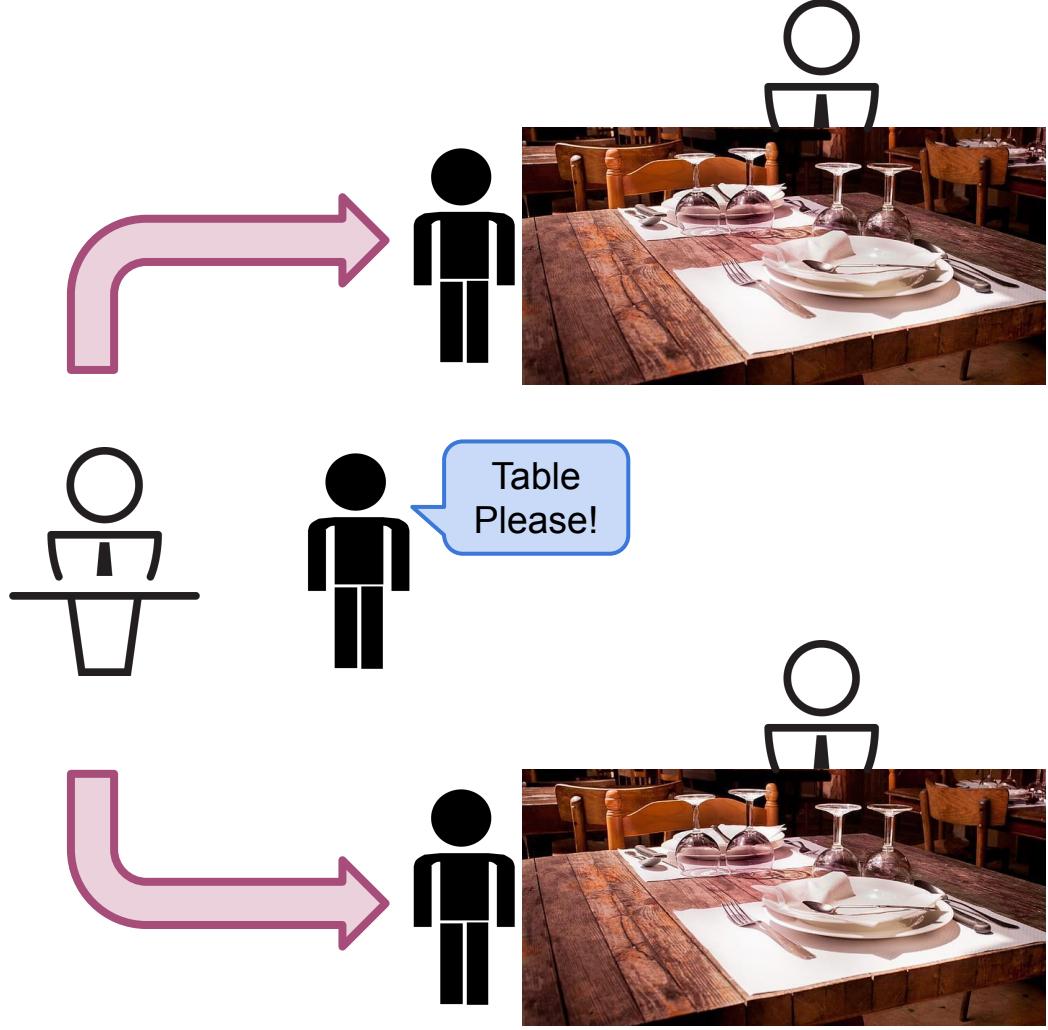
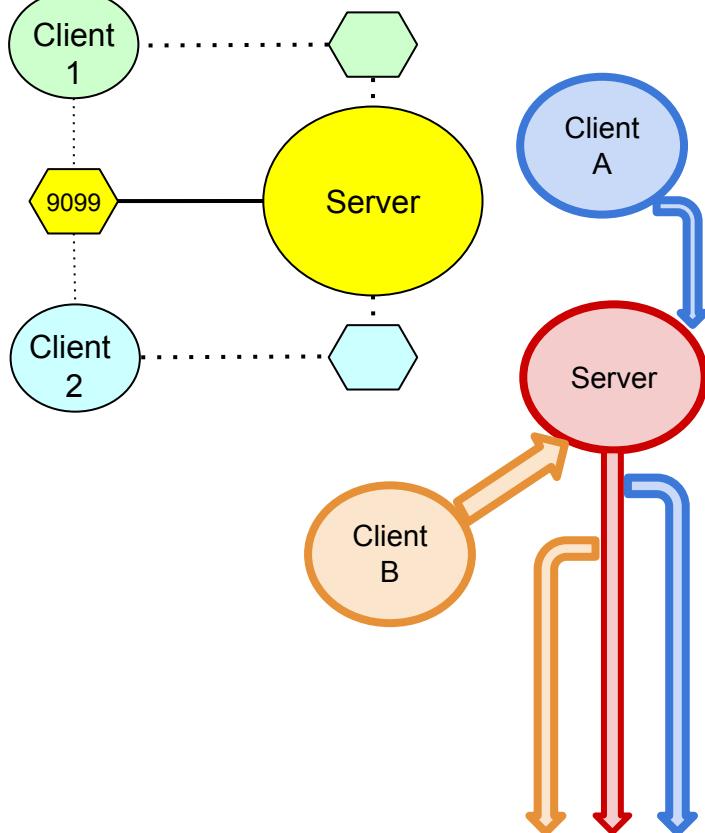
Sockets!

Why do we keep reviewing this?

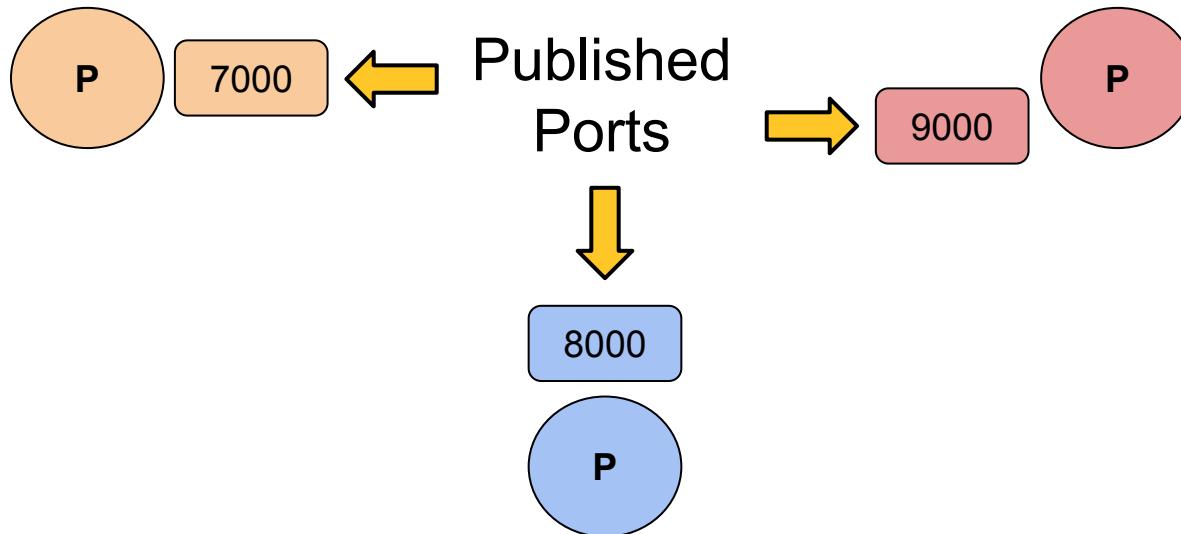


SER 321

Port Examination



Remember that the OS allocates a new port for the client socket!

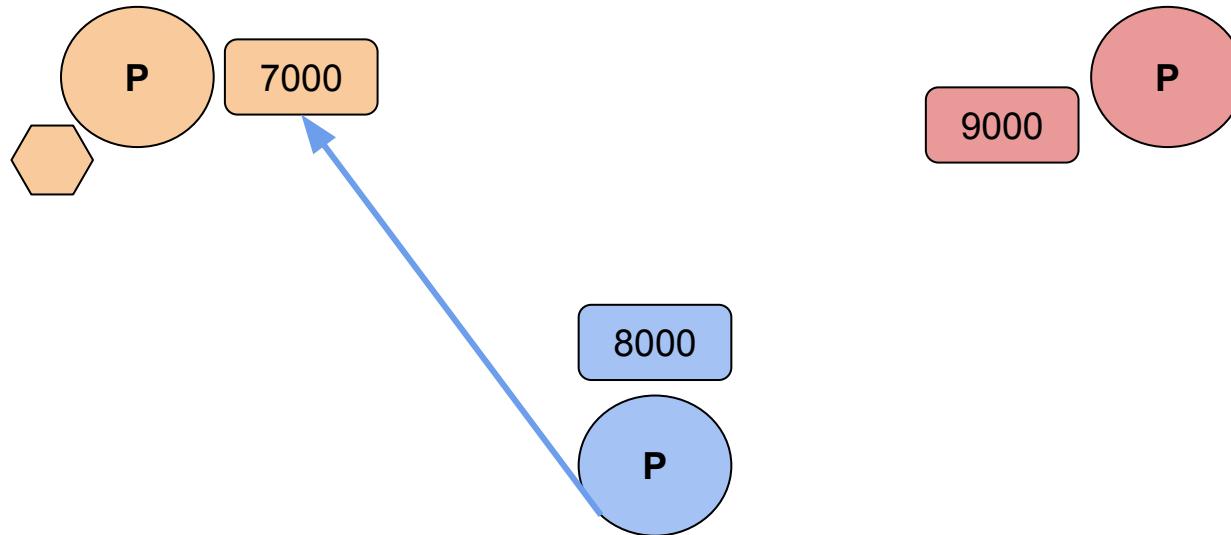


SER 321

P2P Communication

```
sock = server.accept()
```

Remember that the OS allocates a new port for the client socket!

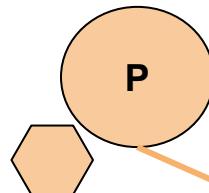
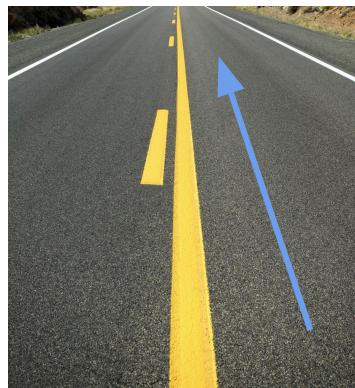


SER 321

P2P Communication

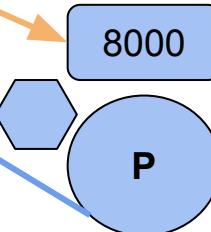
```
sock = server.accept()
```

Remember that the OS allocates a new port for the client socket!

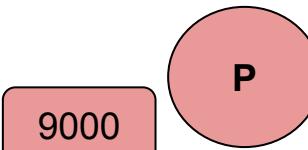


7000

Is that it?



8000

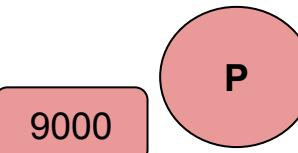
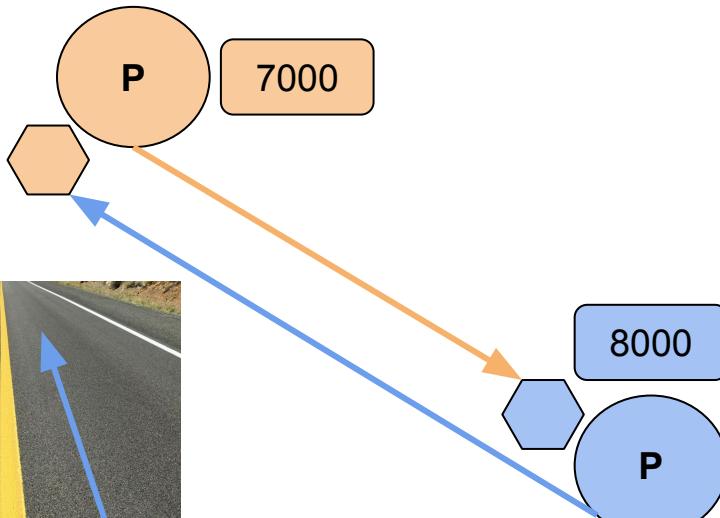
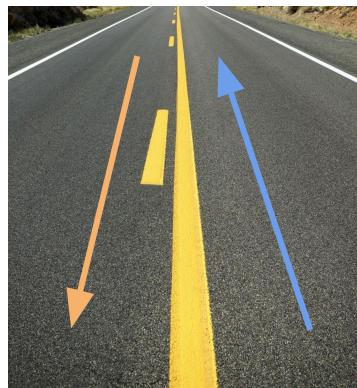


9000

SER 321

P2P Communication

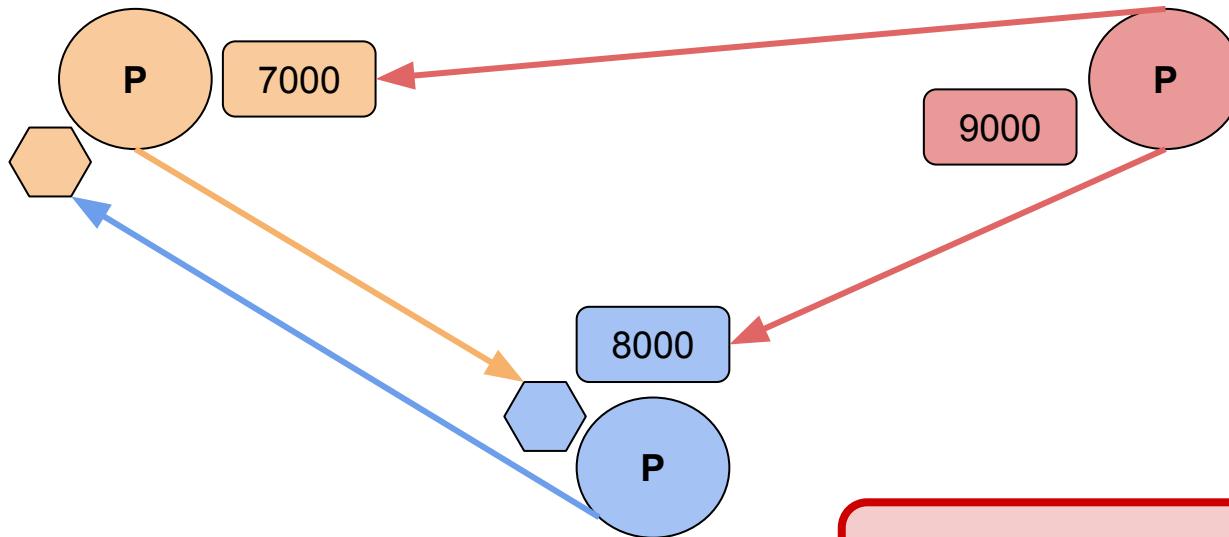
Remember that the OS allocates a new port for the client socket!



SER 321

P2P Communication

Remember that the OS allocates a new port for the client socket!

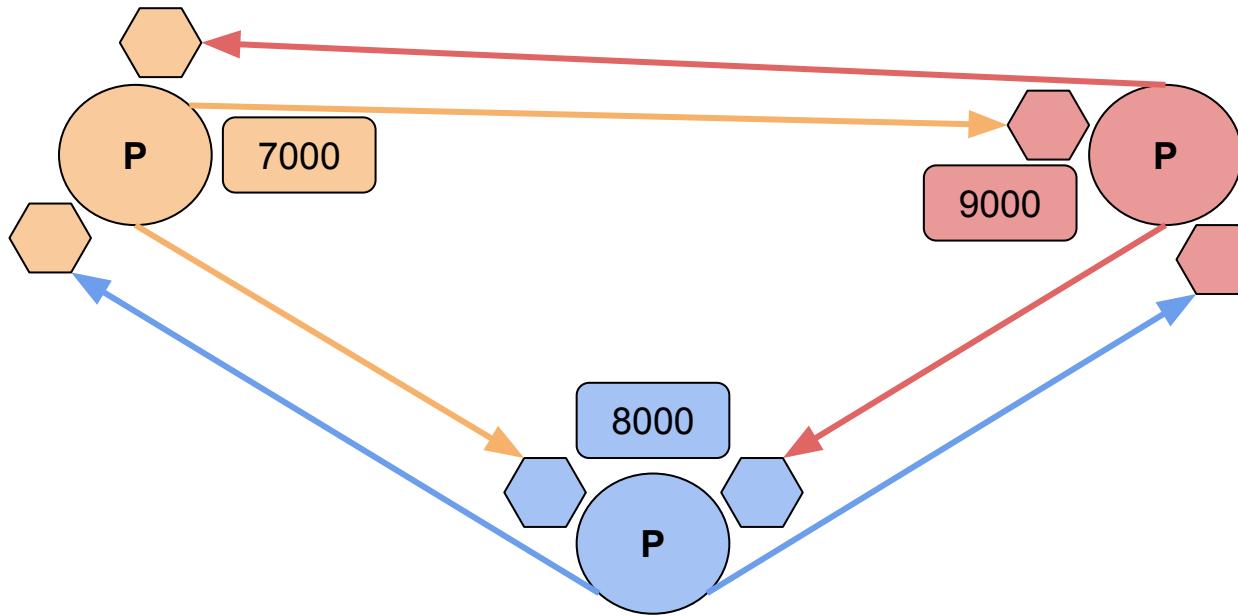


What about Peer 9000?

SER 321

P2P Communication

Remember that the OS allocates a new port for the client socket!



SER 321

Threaded Pitfalls

Starvation

More than one thread accesses a single resource at the same time

Deadlock

A thread never gains access to the resource it needs

Race Condition

A thread is only able to acquire some of the resources it needs

SER 321

Threaded Pitfalls

Starvation

More than one thread accesses a single resource at the same time

Deadlock

A thread never gains access to the resource it needs

Race Condition

A thread is only able to acquire some of the resources it needs

What's the difference?

Starvation

vs.

Deadlock

A thread never gains access to the resource it needs

A thread is only able to acquire some of the resources it needs

Waiting to access the **CPU**

Waiting to access another **resource**

Ready to go; never gets a chance

Not ready to go

SER 321

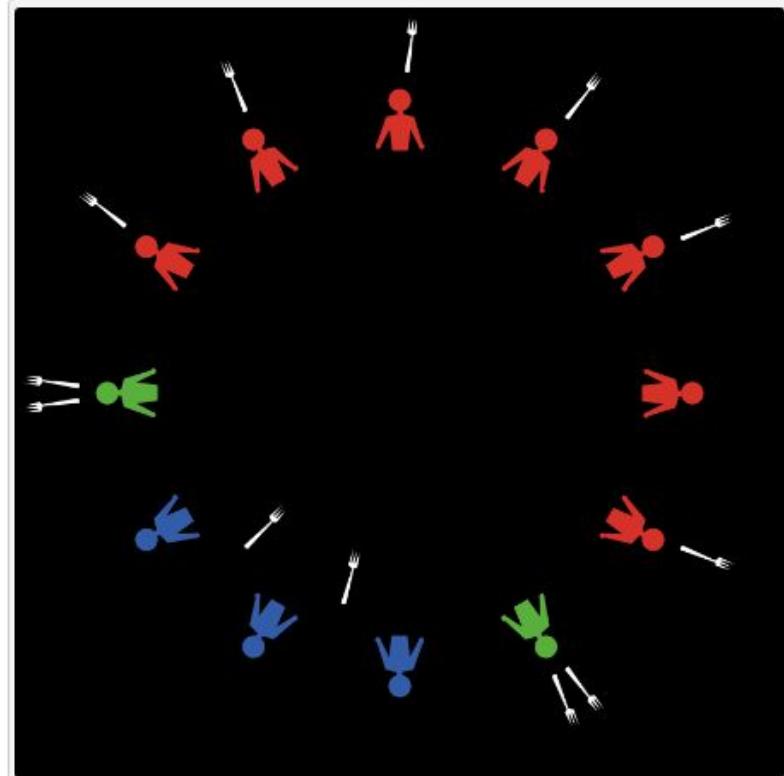
Dining Philosophers

Can we take a guess at what
is happening here?

What are the **BLUE** people doing?

What are the **GREEN** people doing?

What are the **RED** people doing?



SER 321

JSON Structure

Data is stored in...

Name:Value pairs



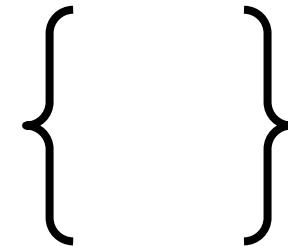
Members

"Katie"

"student" : "Katie"

What uses curly braces?

Objects



What do Objects contain?

Members

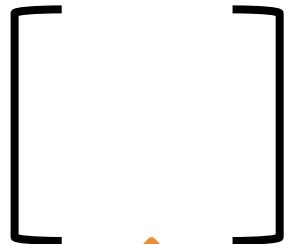


SER 321

JSON Structure

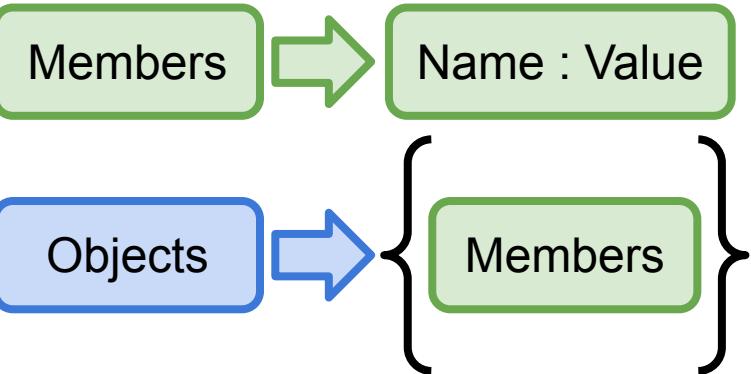
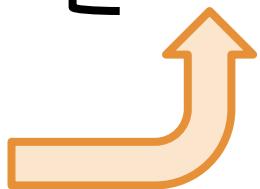
What uses brackets?

Arrays



What do Arrays contain?

Any **Valid** Value



SER 321

JSON Structure

What is a valid value?

Strings

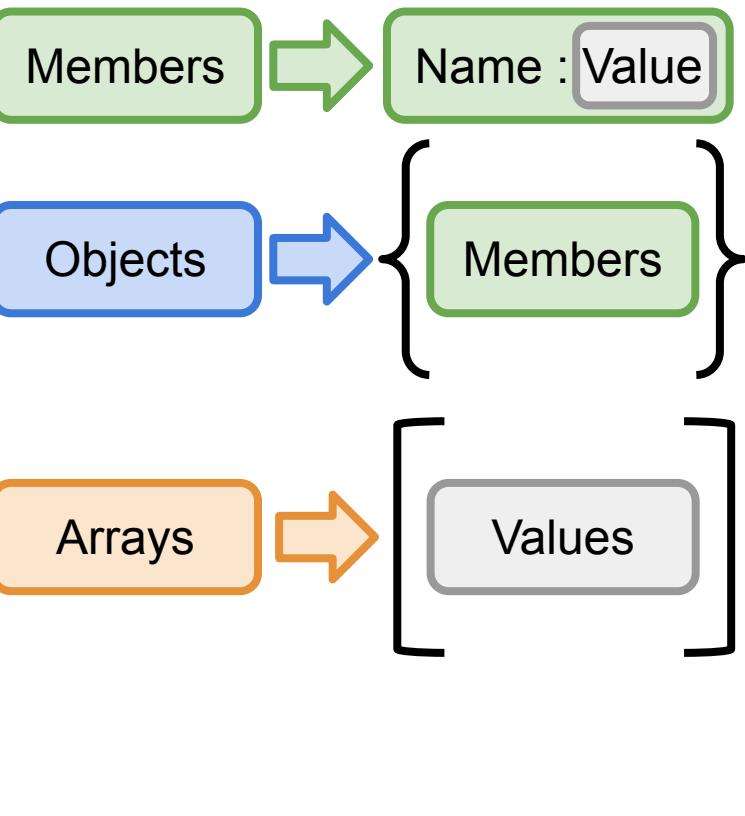
Booleans

Numbers

NULL

Objects

Arrays



Can we recall some of the formats?

JSON

Java Object
Serialization

Protocol Buffers

XML

Binary

Text

Two main
approaches for
storing the
content...

What about the data format?

JSON

Java Object
Serialization

Protocol Buffers

XML

Binary

Text

Who uses *TEXT*?

Text

JSON

Java Object
Serialization

Protocol Buffers

Text

XML

SER 321

Serialization

Binary

Text

What does
this imply?

Who uses **BINARY**?

Text

JSON

Binary

Java Object
Serialization

Binary

Protocol Buffers

Text

XML

SER 321

JSON Recognition

[org.json Docs](#)

[JSON Guide](#)

How many Objects?

How many Arrays?

How many Members?

```
{  
    "lat": 42.3434,  
    "lon": -88.0412,  
    "timezone": "America/Chicago",  
    "timezone_offset": -21600,  
    "current": {  
        "dt": 1733070576,  
        "sunrise": 1733058144,  
        "sunset": 1733091649,  
        "temp": 18.57,  
        "feels_like": 5.97,  
        "pressure": 1025,  
        "humidity": 63,  
        "dew_point": 9.21,  
        "uvi": 0.79,  
        "clouds": 0,  
        "visibility": 10000,  
        "wind_speed": 14.97,  
        "wind_deg": 280,  
        "wind_gust": 21.85,  
        "weather": [  
            {  
                "id": 800,  
                "main": "Clear",  
                "description": "clear sky",  
                "icon": "01d"  
            }  
        ]  
    }  
}
```

SER 321

JSON Practice

[org.json Docs](#)

[JSON Guide](#)

```
JSONObject json =
```

How would we...

Check for the timezone member?

```
boolean hasTimezone =
```

Get the timezone?

```
String timezone =
```

```
{  
    "lat": 42.3434,  
    "lon": -88.0412,  
    "timezone": "America/Chicago",  
    "timezone_offset": -21600,  
    "current": {  
        "dt": 1733070576,  
        "sunrise": 1733058144,  
        "sunset": 1733091649,  
        "temp": 18.57,  
        "feels_like": 5.97,  
        "pressure": 1025,  
        "humidity": 63,  
        "dew_point": 9.21,  
        "uvi": 0.79,  
        "clouds": 0,  
        "visibility": 10000,  
        "wind_speed": 14.97,  
        "wind_deg": 280,  
        "wind_gust": 21.85,  
        "weather": [  
            {  
                "id": 800,  
                "main": "Clear",  
                "description": "clear sky",  
                "icon": "01d"  
            }  
        ]  
    }  
}
```

SER 321

JSON Practice

How would we...

Obtain the temp value?

~~String temp = json.getString("temp");~~

JSONObject json = {

```
"lat": 42.3434,  
"lon": -88.0412,  
"timezone": "America/Chicago",  
"timezone_offset": -21600,  
"current": {  
    "dt": 1733070576,  
    "sunrise": 1733058144,  
    "sunset": 1733091649,  
    "temp": 18.57,  
    "feels_like": 5.97,  
    "pressure": 1025,  
    "humidity": 63,  
    "dew_point": 9.21,  
    "uvi": 0.79,
```

Recall that
nested
members
require
multiple steps!

Step 1:

Step 2:

Step 3:

Step 4:

SER 321

JSON Practice

```
JSONObject json = {
```

How would we...

Obtain the temp value?

```
String temp = json.getString("temp");
```

```
if (json.has("current")) {
```

```
{  
    "lat": 42.3434,  
    "lon": -88.0412,  
    "timezone": "America/Chicago",  
    "timezone_offset": -21600,  
    "current": {  
        "dt": 1733070576,  
        "sunrise": 1733058144,  
        "sunset": 1733091649,  
        "temp": 18.57,  
        "feels_like": 5.97,  
        "pressure": 1025,  
        "humidity": 63,  
        "dew_point": 9.21,  
        "uvi": 0.79,  
    },  
}
```

Recall that nested members require multiple steps!

Step 1: Check for parent object

Step 2:

Step 3:

Step 4:

SER 321

JSON Practice

How would we...

Obtain the temp value?

~~String temp = json.getString("temp");~~

if (json.has("current")) {

 JSONObject current =

 json.getJSONObject("current");

}

JSONObject json =

```
{  
    "lat": 42.3434,  
    "lon": -88.0412,  
    "timezone": "America/Chicago",  
    "timezone_offset": -21600,  
    "current": {  
        "dt": 1733070576,  
        "sunrise": 1733058144,  
        "sunset": 1733091649,  
        "temp": 18.57,  
        "feels_like": 5.97,  
        "pressure": 1025,  
        "humidity": 63,  
        "dew_point": 9.21,  
        "uvi": 0.79,  
    },  
}
```



Recall that nested members require multiple steps!

Step 1: Check for parent object

Step 2: Obtain parent object

Step 3:

Step 4:

SER 321

JSON Practice

```
JSONObject json = {
```

How would we...

Obtain the temp value?

```
String temp = json.getString("temp");
```

```
if (json.has("current")) {
```

```
    JSONObject current =
```

```
        json.getJSONObject("current");
```

```
    if (current.has("temp")) {
```

```
        temp = current.getString("temp");
```

```
}
```

```
{  
    "lat": 42.3434,  
    "lon": -88.0412,  
    "timezone": "America/Chicago",  
    "timezone_offset": -21600,  
    "current": {  
        "dt": 1733070576,  
        "sunrise": 1733058144,  
        "sunset": 1733091649,  
        "temp": 18.57,  
        "feels_like": 5.97,  
        "pressure": 1025,  
        "humidity": 63,  
        "dew_point": 9.21,  
        "uvi": 0.79,  
    },  
}
```

Recall that
nested
members
require
multiple steps!

Step 1: Check for parent object

Step 2: Obtain parent object

Step 3: Check for nested member

Step 4:

SER 321

Serialization

Generic
Superclass

Streams and their types

```
OutputStream out = sock.getOutputStream();
```

Buffered Stream

Bytes

Data Stream

Primitive DATA Types

Object Stream

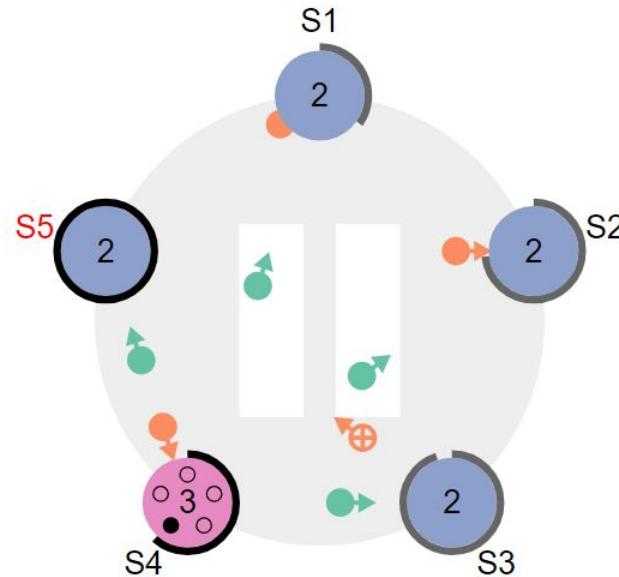
Java Objects

SER 321

RAFT

How do we
feel about
Consensus?

Do we want to
look at RAFT
again?



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



Parallel

Distributed

Parallel

- Single computer
- Work split among different *processors*
- Memory is shared *or* distributed
- Communicate through *bus*
- Latency while waiting for resources

Distributed

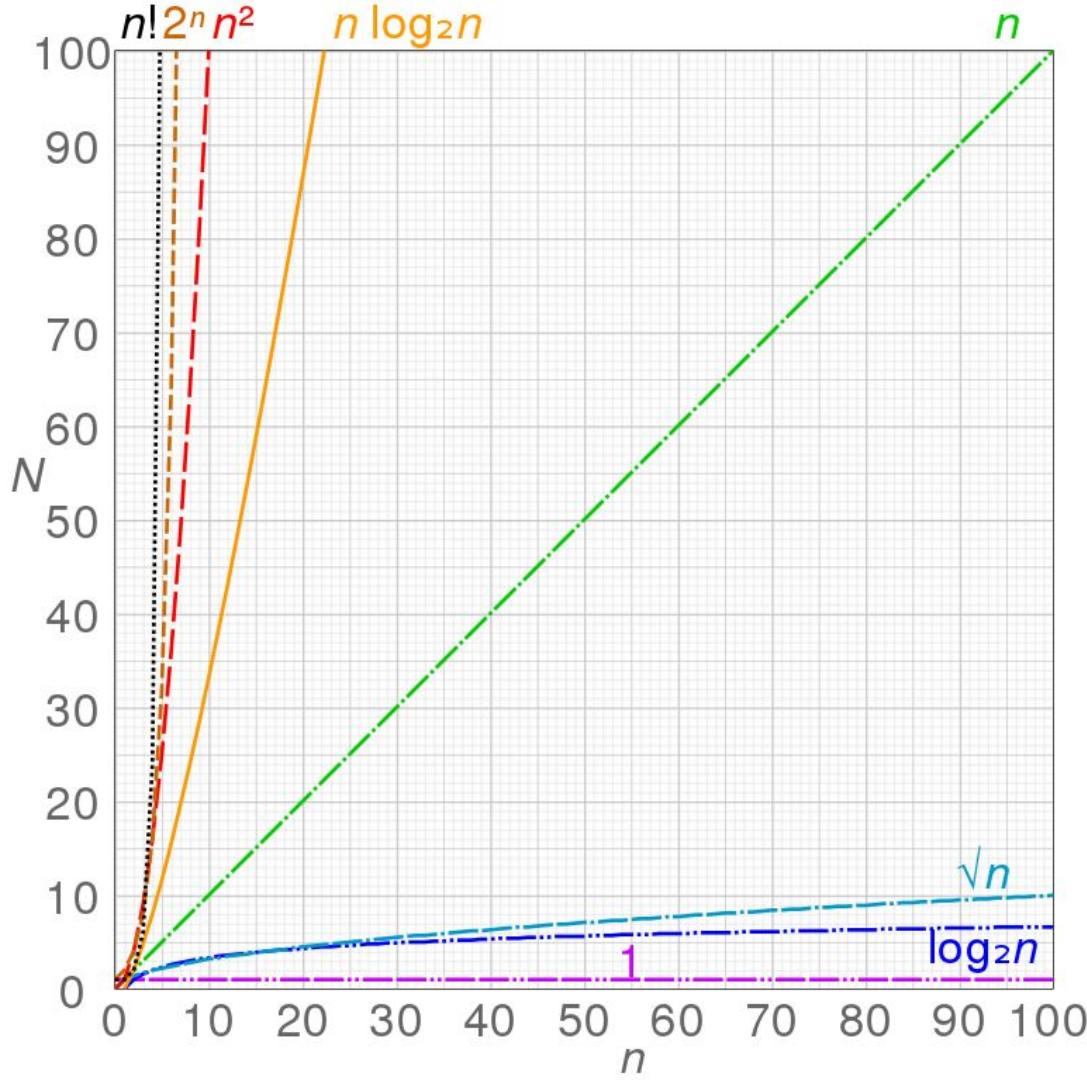
- Many computers
- Work split among different *locations*
- Memory is distributed
- Communicate through *message passing*
- Experience latency both between nodes and within nodes

- Work is partitioned
- Partitions processed individually
- **Can** improve performance
- **Can** improve speed
- Experience Latency

SER 321

When to Distribute

When should
we *consider*
distributing?

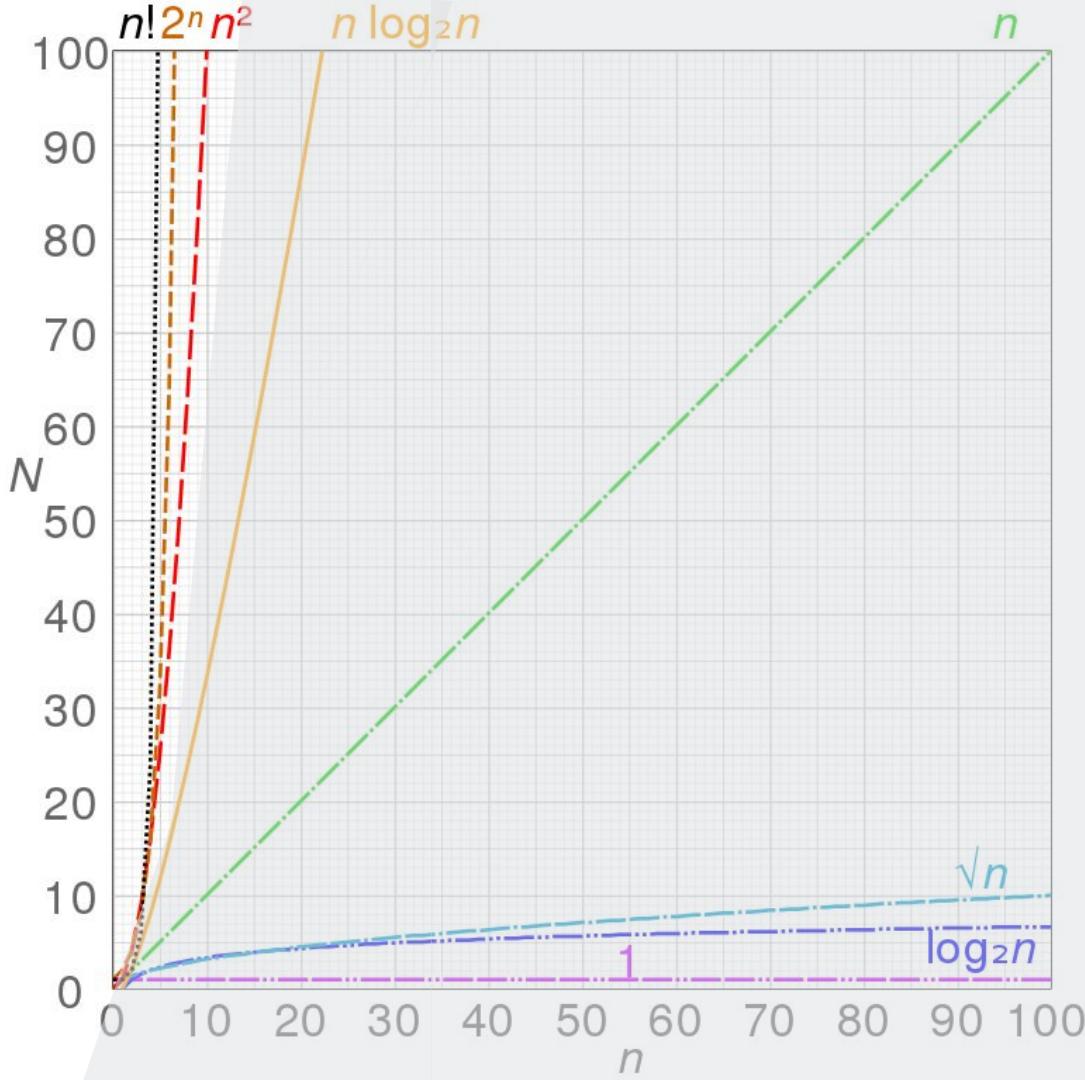


SER 321

When to Distribute

When should
we *consider*
distributing?

Super Duper Extra Extra
Large Orders of Magnitude!



SER 321

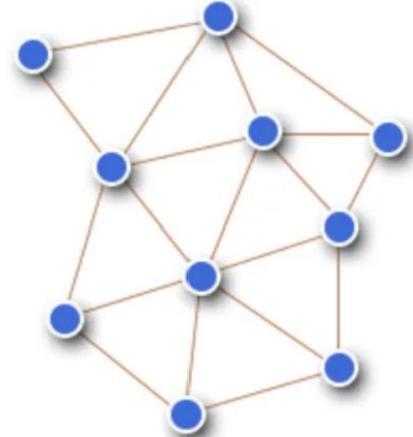
Distributed Systems

Distributed System Properties

Global Clock

No! 

Yes! 



SER 321

Distributed Systems

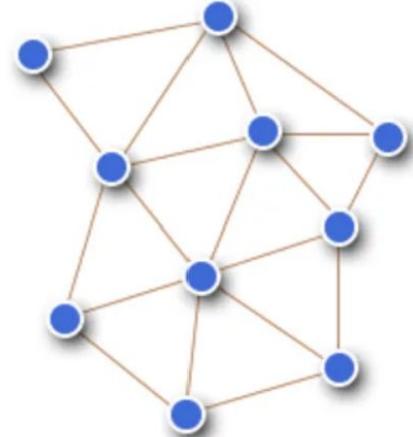
Distributed System Properties

Nodes Fail

No! 

Global Clock

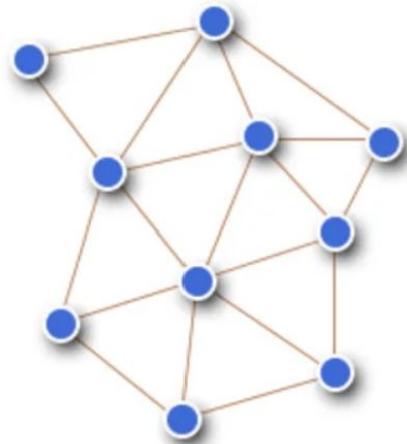
Yes! 



SER 321

Distributed Systems

Distributed System Properties



Cluster Changes

No! 

Global Clock

Yes! 

Nodes Fail

SER 321

Distributed Systems

Distributed System Properties

Network is Reliable

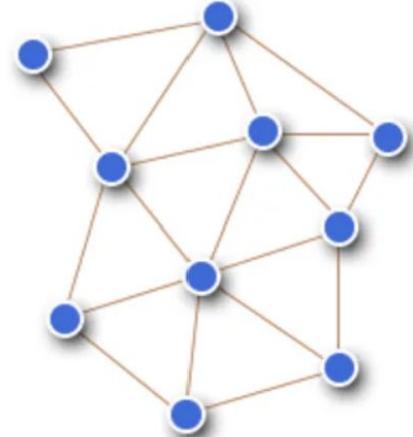
No! 

Global Clock

Yes! 

Nodes Fail

Cluster Changes



SER 321

Distributed Systems

Distributed System Properties

Latency Never Exists

No! 

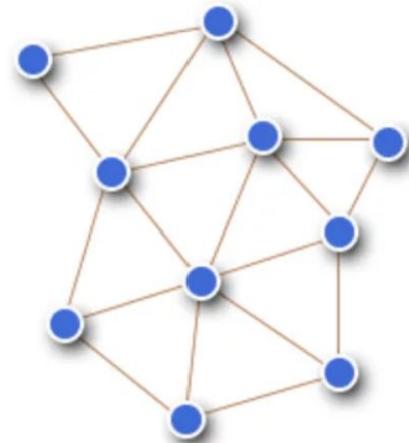
Global Clock

Network is Reliable

Yes! 

Nodes Fail

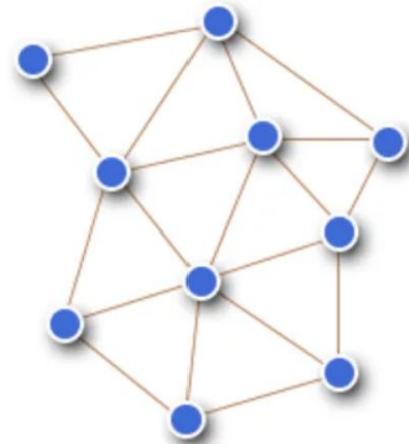
Cluster Changes



SER 321

Distributed Systems

Distributed System Properties



Path taken Changes

No!

Global Clock

Network is Reliable

Latency Never Exists

Yes!

Nodes Fail

Cluster Changes

Distributed System Properties

Share Common Resources

No! 

Global Clock

Network is Reliable

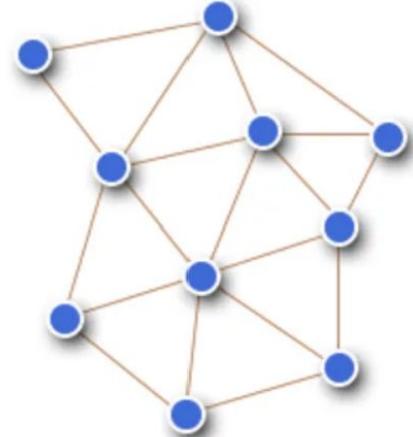
Latency Never Exists

Yes! 

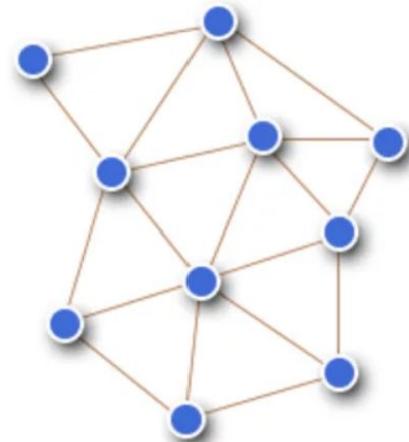
Nodes Fail

Cluster Changes

Path taken Changes



Distributed System Properties



Pitfalls handled inherently

No!

Global Clock

Network is Reliable

Latency Never Exists

Yes!

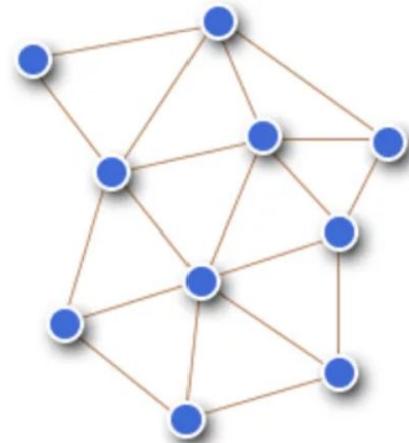
Nodes Fail

Cluster Changes

Path taken Changes

Share Common Resources

Distributed System Properties



No!

Global Clock

Network is Reliable

Latency Never Exists

Pitfalls handled inherently

Yes!

Nodes Fail

Cluster Changes

Path taken Changes

Share Common Resources

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



Node Validation

SER 321

Middleware

We have been:

Serializing
Messages

Sending
Messages

Parsing
Messages

Handle
Messages

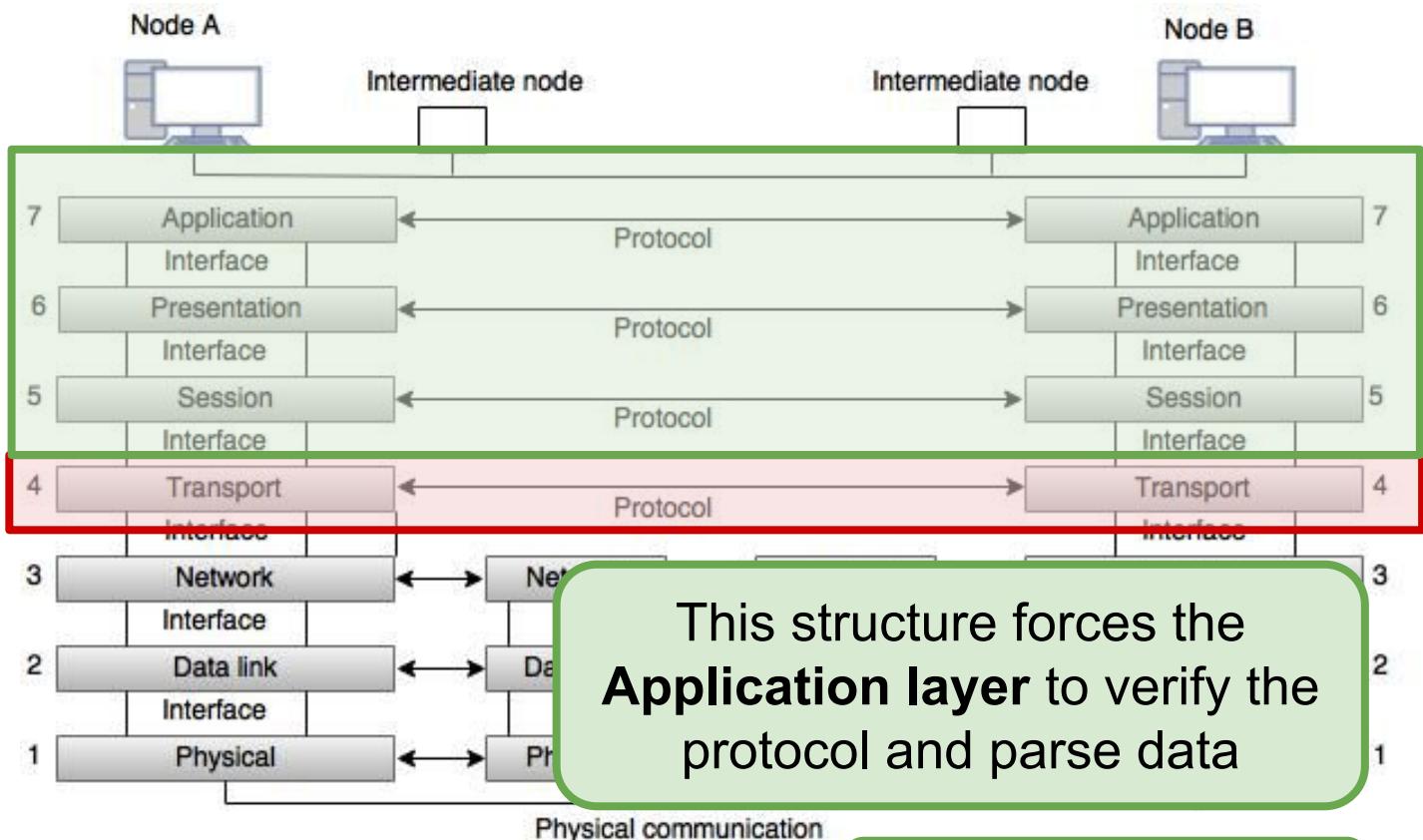


Fig: OSI Model

Not really its job...

SER 321

Middleware

With Middleware:

Serializing
Messages

Sending
Messages

Parsing
Messages

Handle
Messages

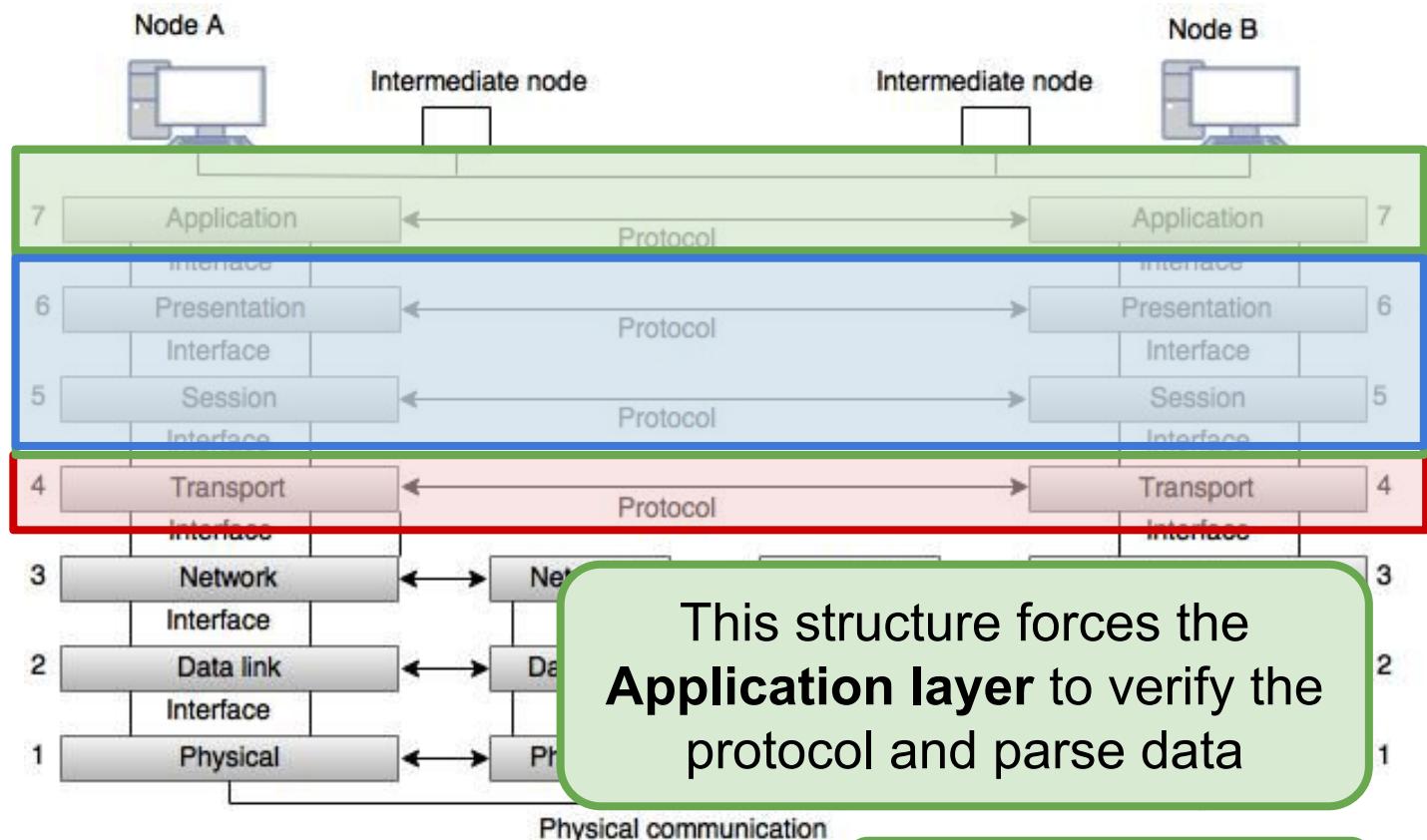


Fig: OSI Model

Not really its job...

SER 321

Middleware

With Middleware:

Serializing
Messages

Sending
Messages

Parsing
Messages

Handle
Messages

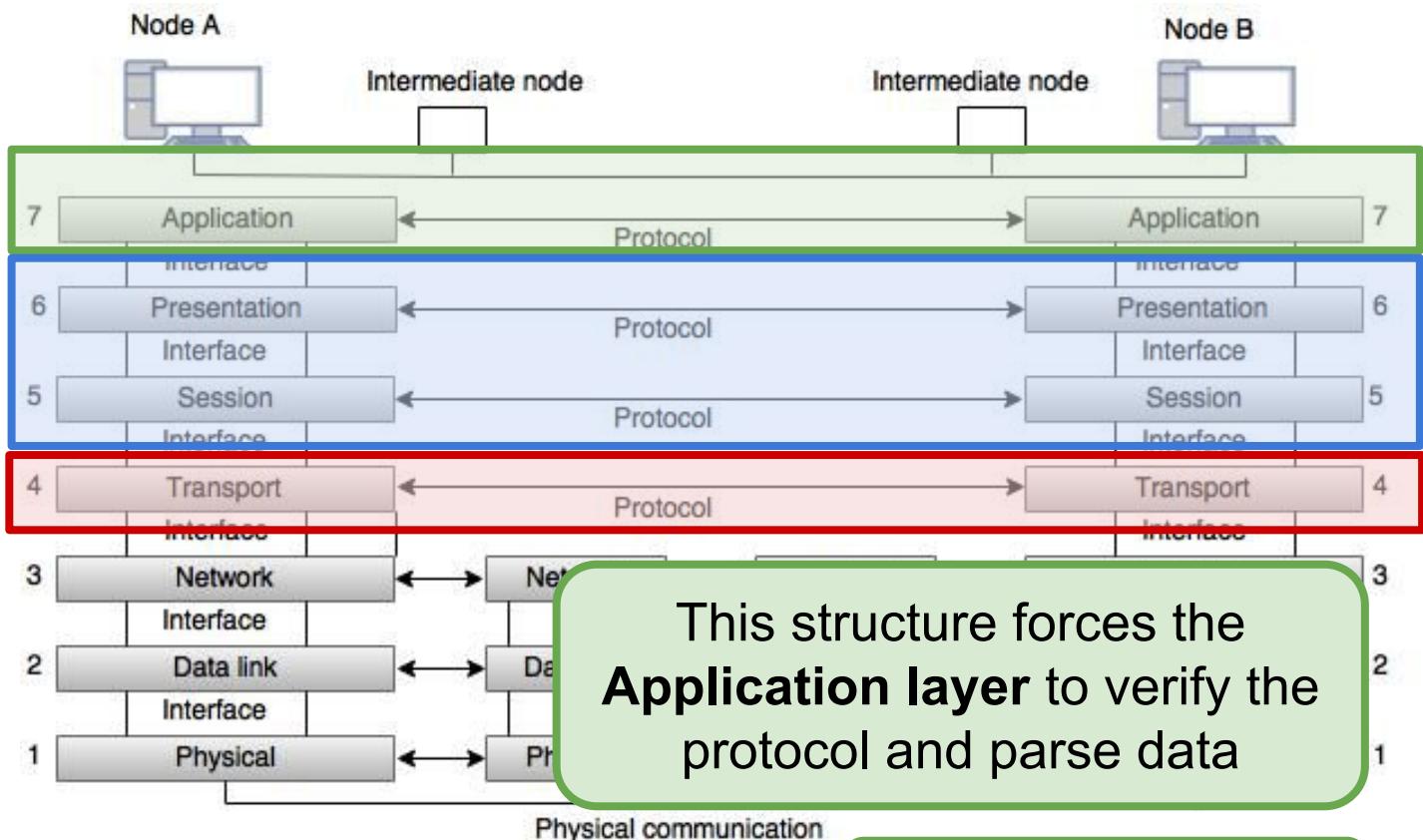


Fig: OSI Model

Not really its job...

SER 321

Middleware

Middleware:

Session Layer Responsibilities:

Authentication

Authorization

Session Management

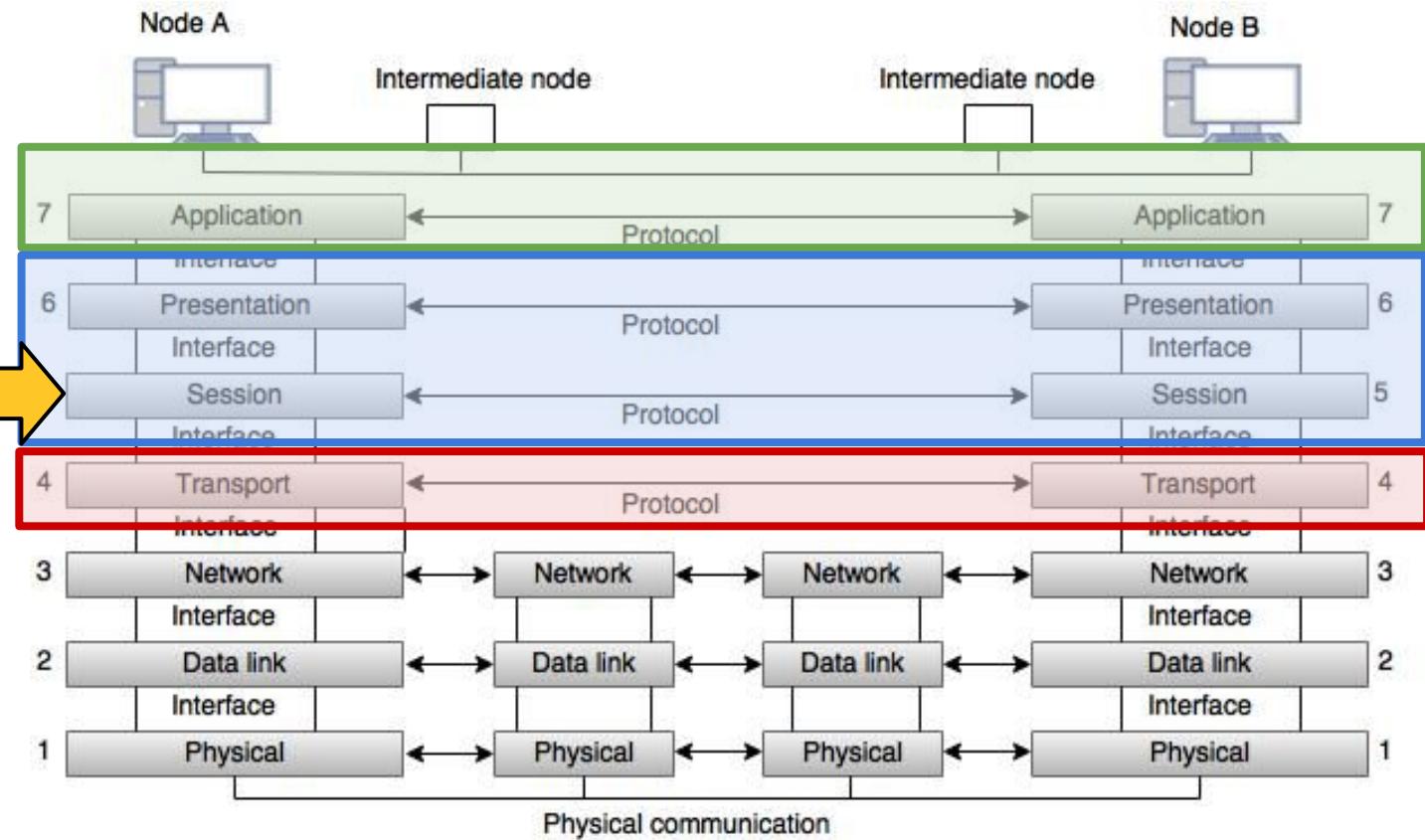


Fig: OSI Model

SER 321

Middleware

Middleware:

Presentation Layer Responsibilities:

Translation

Compression

Encryption

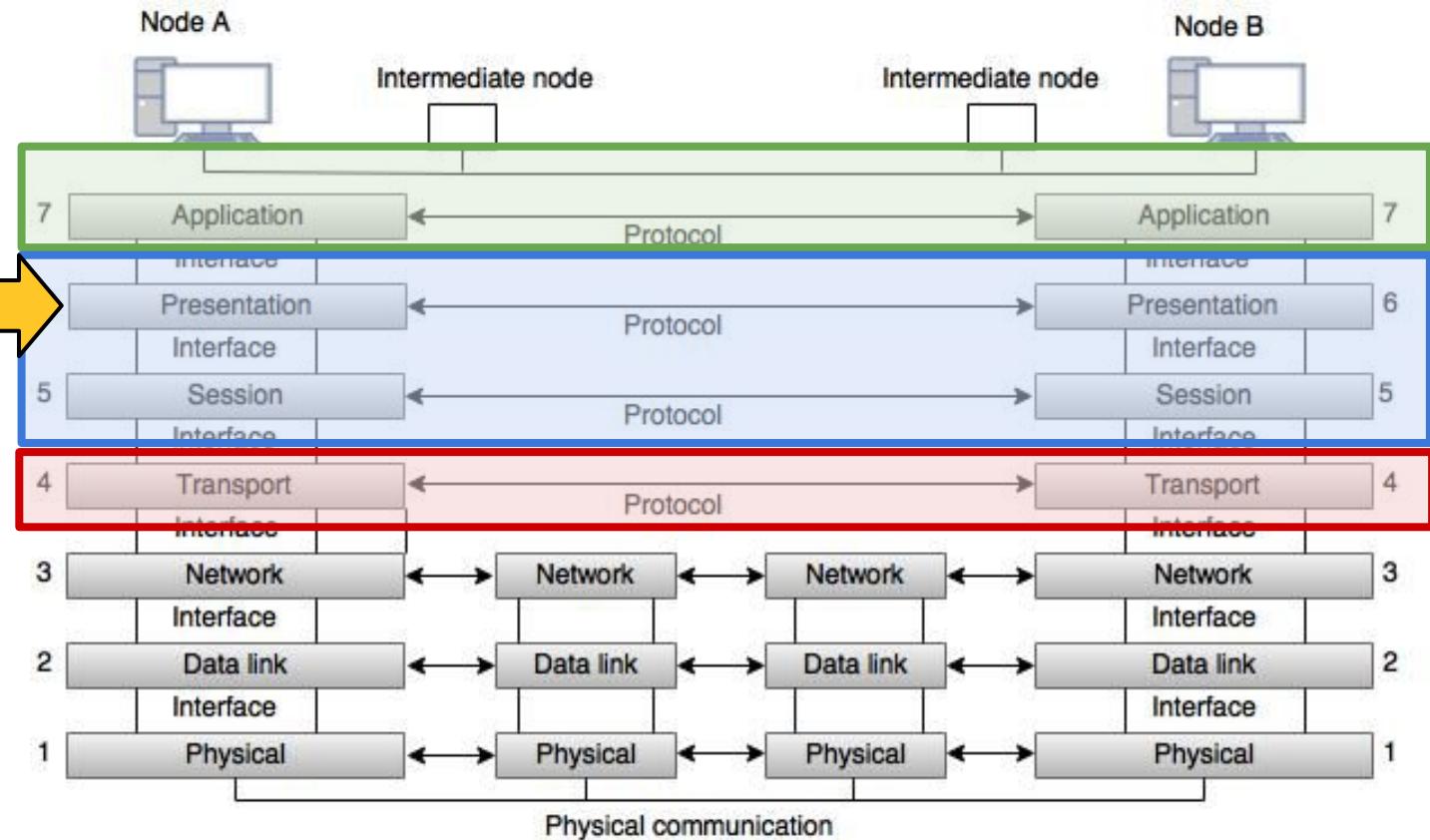


Fig: OSI Model

Examples?

Message Oriented Middleware (MOM)

Web Frameworks

Remote Procedure Calls (RPC)



App. Programming Interface (API)



Why do we care?

Agility

Reusability

Efficiency

Cost
Effectiveness

Portability

TL;DR →

It's the “glue” between
the client and server

Why do we care?



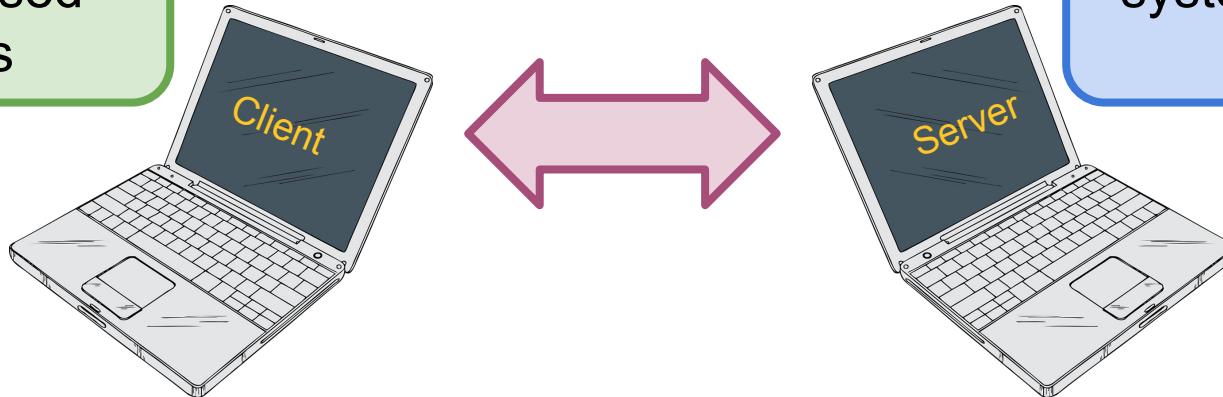
Separation
of Concerns!

Sort of like publishing a contract

“If you follow these rules, I will handle your request.”

Why do we care?

Client handles
user-focused
actions

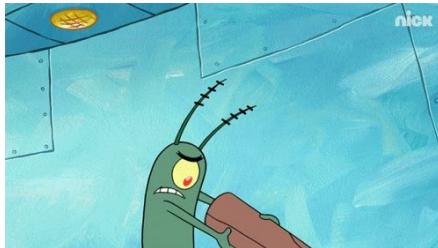


Server handles
system-focused
actions

Middleware handles
the in-between!

SER 321

Middleware



```
{ }
```

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

Add User



- Get data from user
- Validate data
- Determine Request Format
- Construct Valid Request
- Establish Connection
- Send Request
- Wait for Response
 - Read Response from Stream
 - Parse Response
 - Display Response to User

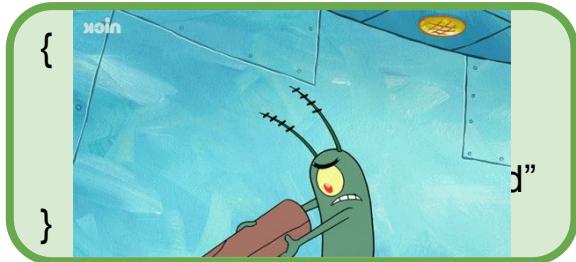
Critical Step!



SER 321

Middleware

- Get data from user
- Validate data
- Determine Request Format
- Construct Valid Request
- Establish Connection
- Send Request
- Wait for Response
 - Read Response from Stream
 - Parse Response
 - Display Response to User



How do we know?

Add User



- Read data from Stream
- Parse Data
- Validate Request Format
- Determine Request Type
- *Perform Requested Action*
- Determine Response Format
- Construct Valid Response
- [Re-establish connection]
- Send Response

Done!

Wants to Add User

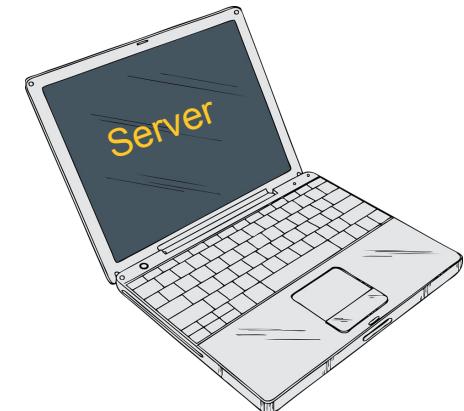
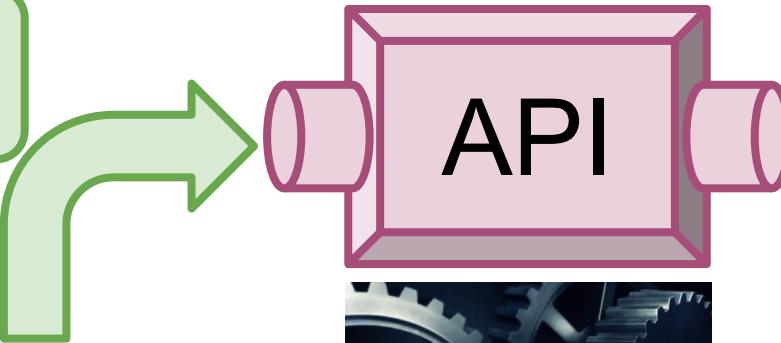


SER 321

Middleware

Add User:
Name = Katie
Password = password

With Middleware:



SER 321

Middleware

Add User:
Name = Katie
Password = password

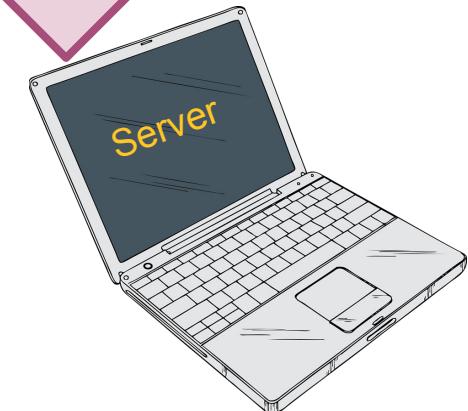
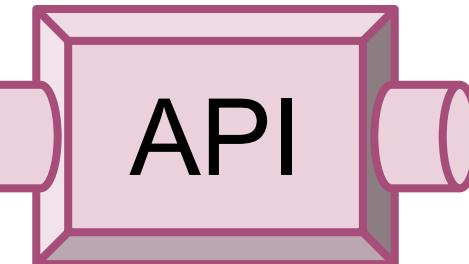
With Middleware:



{

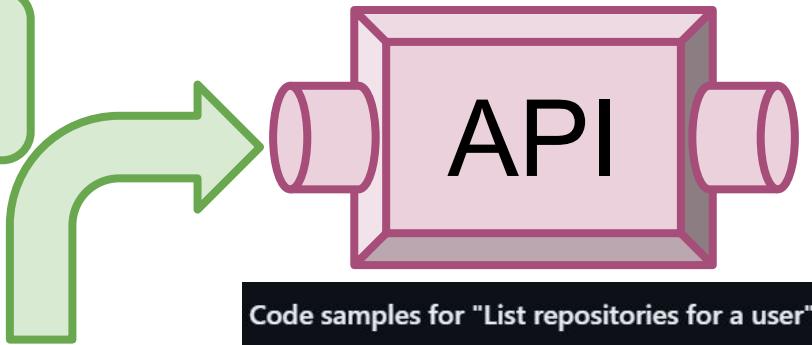
“type” : “addUser”,
“name” : “katie”,
“password” : “password”

}



With Middleware:

Get repositories for a specific user



Code samples for "List repositories for a user"

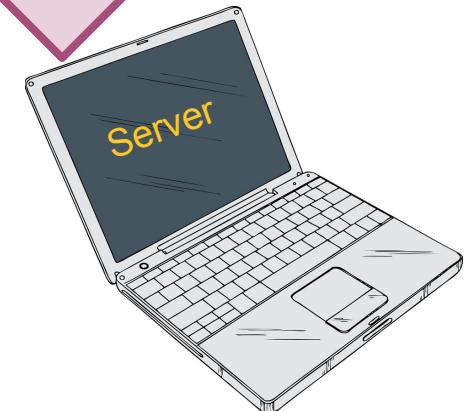
Request example

GET /users/{username}/repos

cURL JavaScript GitHub CLI

```
curl -L \
-H "Accept: application/vnd.github+json" \
-H "Authorization: Bearer <YOUR-TOKEN>" \
-H "X-GitHub-Api-Version: 2022-11-28" \
https://api.github.com/users/USERNAME/repos
```

GitHub REST API

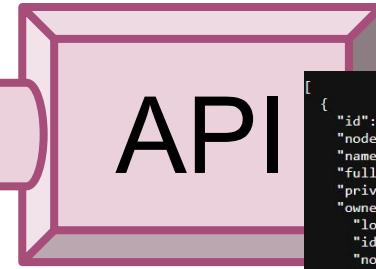


SER 321

Middleware

With Middleware:

Get repositories for a specific user



Code samples for "List repos"

Request example

GET /users/{username}/repos

cURL JavaScript GitHub CL

```
curl -L \
-H "Accept: application/vnd.github+json"
-H "Authorization: Bearer <YOUR-TOKEN>"
-Api-Version: 2022-11-28
github.com/users/kgrinne3/repos
```

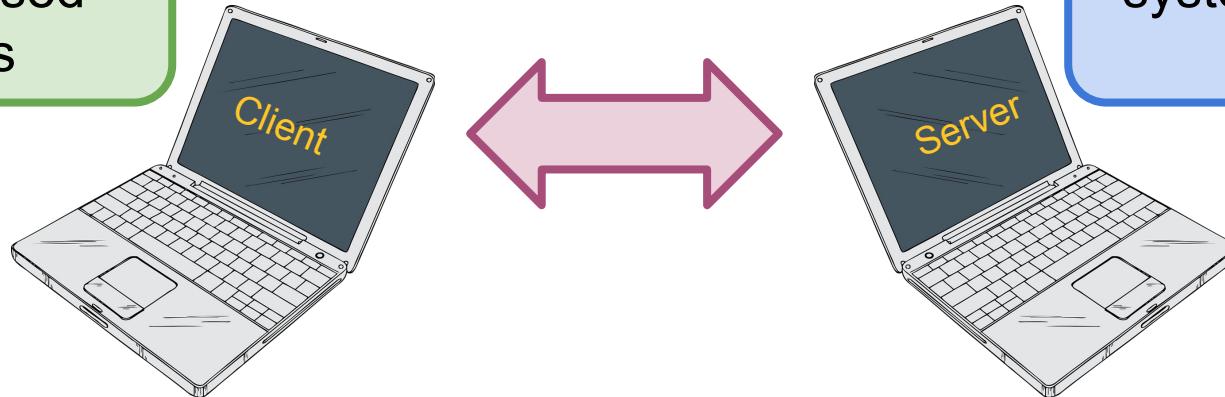
<https://api.github.com/users/kgrinne3/repos>

GitHub REST API

```
[{"id": 550568457,
"node_id": "R_kgDOINECCQ",
"name": "assign1git",
"full_name": "kgrinne3/assign1git",
"private": false,
"owner": {
  "login": "kgrinne3",
  "id": 115493885,
  "node_id": "U_kgDOBujL_Q",
  "avatar_url": "https://avatars.githubusercontent.com/u/115493885?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/kgrinne3",
  "html_url": "https://github.com/kgrinne3",
  "followers_url": "https://api.github.com/users/kgrinne3/followers",
  "following_url": "https://api.github.com/users/kgrinne3/following{/other_user}",
  "gists_url": "https://api.github.com/users/kgrinne3/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/kgrinne3/starred{/owner}{/repo}",
  "subscriptions_url": "https://api.github.com/users/kgrinne3/subscriptions",
  "organizations_url": "https://api.github.com/users/kgrinne3/orgs",
  "repos_url": "https://api.github.com/users/kgrinne3/repos",
  "events_url": "https://api.github.com/users/kgrinne3/events{/privacy}",
  "received_events_url": "https://api.github.com/users/kgrinne3/received_events",
  "type": "User",
  "site_admin": false
},
"html_url": "https://github.com/kgrinne3/assign1git",
"description": "Katie Grinnell",
"fork": false,
"url": "https://api.github.com/repos/kgrinne3/assign1git",
"forks_url": "https://api.github.com/repos/kgrinne3/assign1git/forks",
"keys_url": "https://api.github.com/repos/kgrinne3/assign1git/keys{/key_id}",
"collaborators_url": "https://api.github.com/repos/kgrinne3/assign1git/collaborators{/collaborator}",
"teams_url": "https://api.github.com/repos/kgrinne3/assign1git/teams",
"hooks_url": "https://api.github.com/repos/kgrinne3/assign1git/hooks",
"issue_events_url": "https://api.github.com/repos/kgrinne3/assign1git/issues/events{/number}",
"events_url": "https://api.github.com/repos/kgrinne3/assign1git/events",
"assignees_url": "https://api.github.com/repos/kgrinne3/assign1git/assignees{/user}",
"branches_url": "https://api.github.com/repos/kgrinne3/assign1git/branches{/branch}",
"tags_url": "https://api.github.com/repos/kgrinne3/assign1git/tags",
"blobs_url": "https://api.github.com/repos/kgrinne3/assign1git/git/blobs{/sha}",
"git_tags_url": "https://api.github.com/repos/kgrinne3/assign1git/git/tags{/sha}"},
```

Why do we care?

Client handles
user-focused
actions

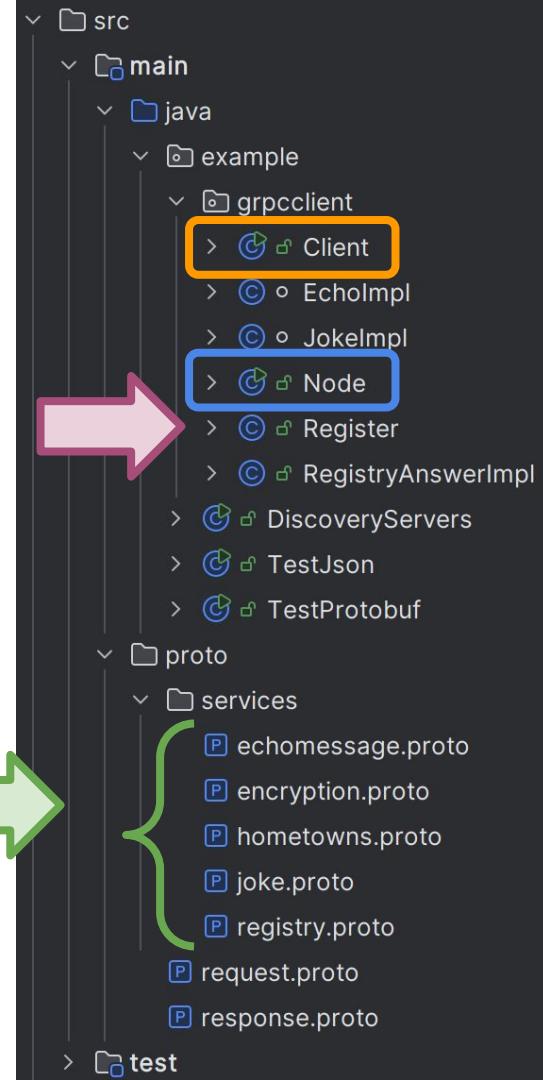
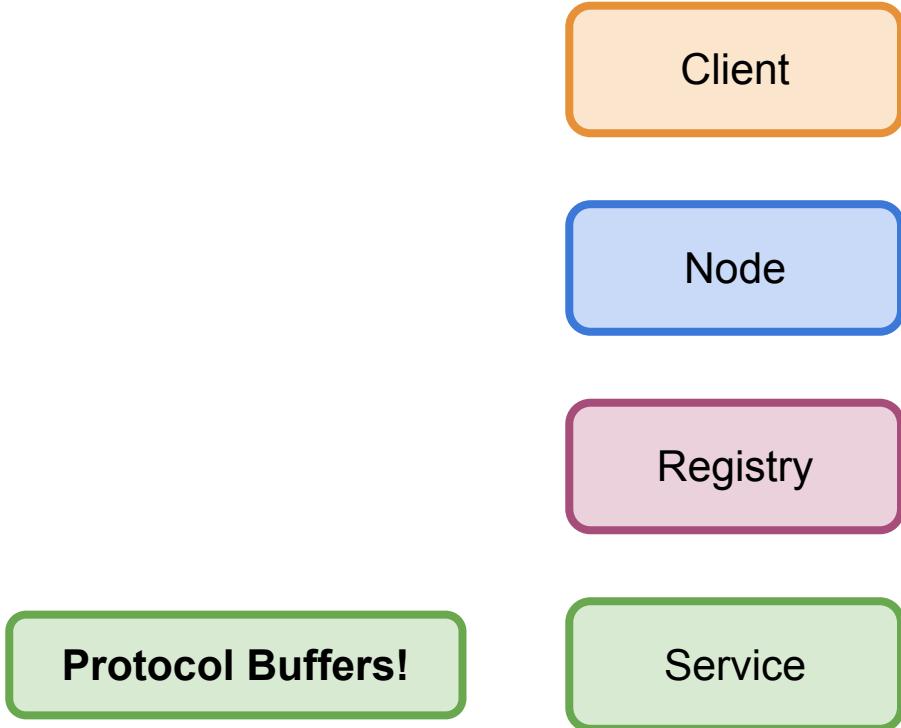


Server handles
system-focused
actions

Middleware handles
the in-between!

SER 321

Assign 6



SER 321

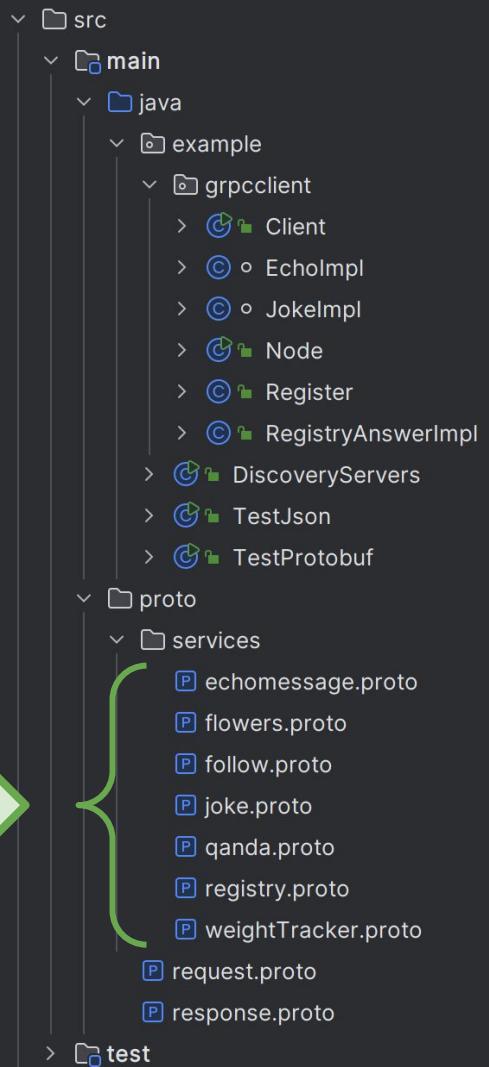
Protobuf Review

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 1usage
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();
}
```

```
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

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 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();
}
```

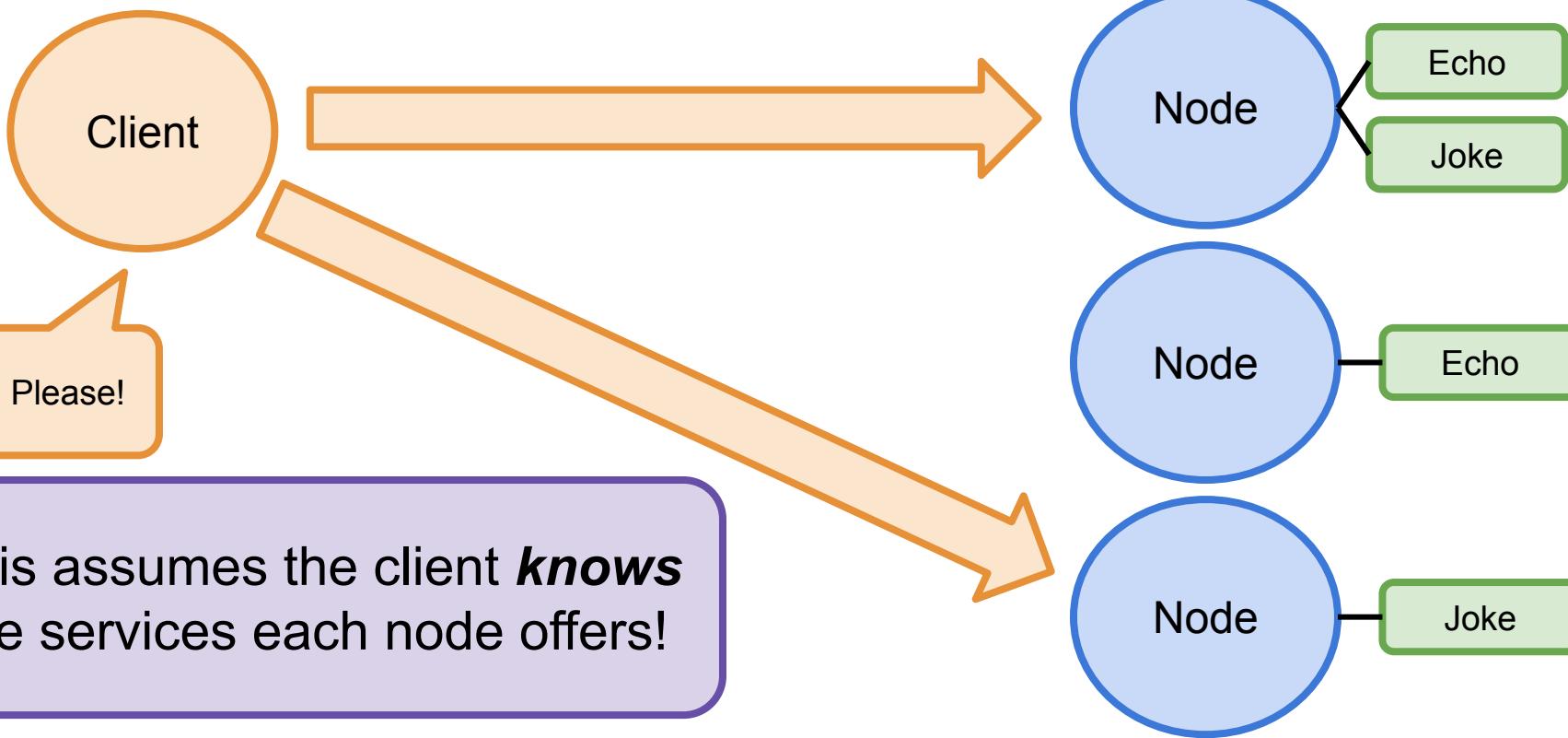
Two new concepts!

Registry

RPC

Previously...

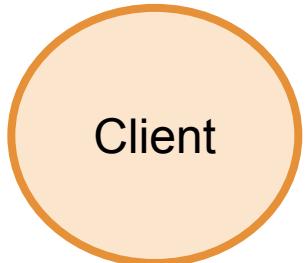
Registry



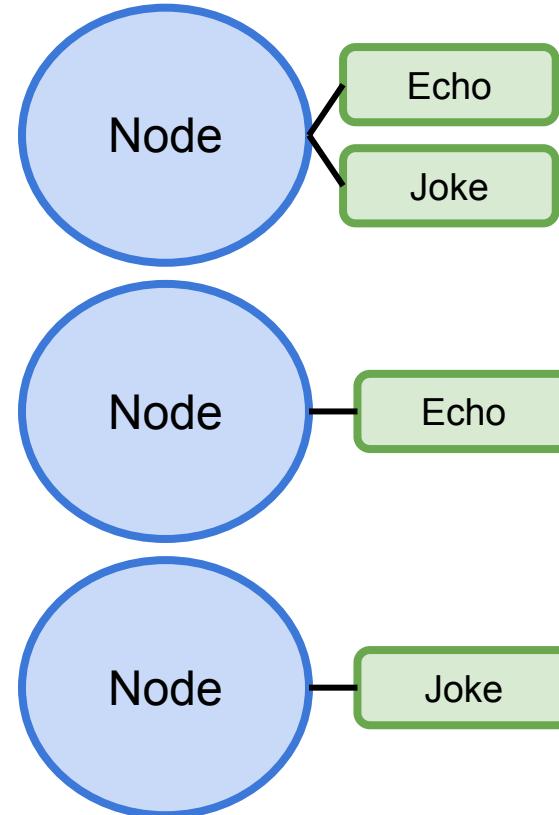
This assumes the client ***knows*** the services each node offers!

With the Registry...

Registry

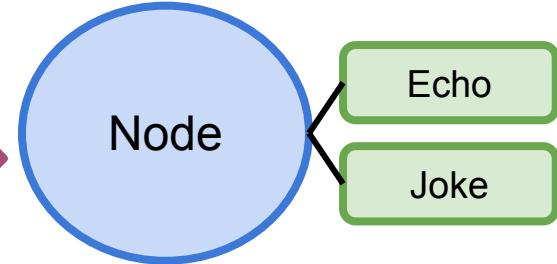
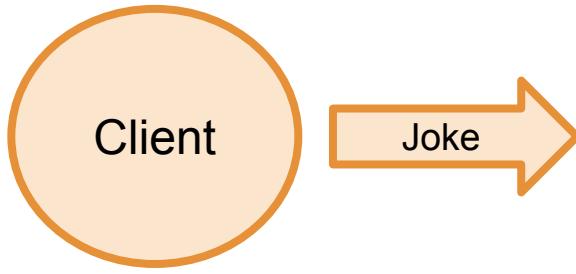


Joke Please!



With the Registry...

Registry



Joke Please!

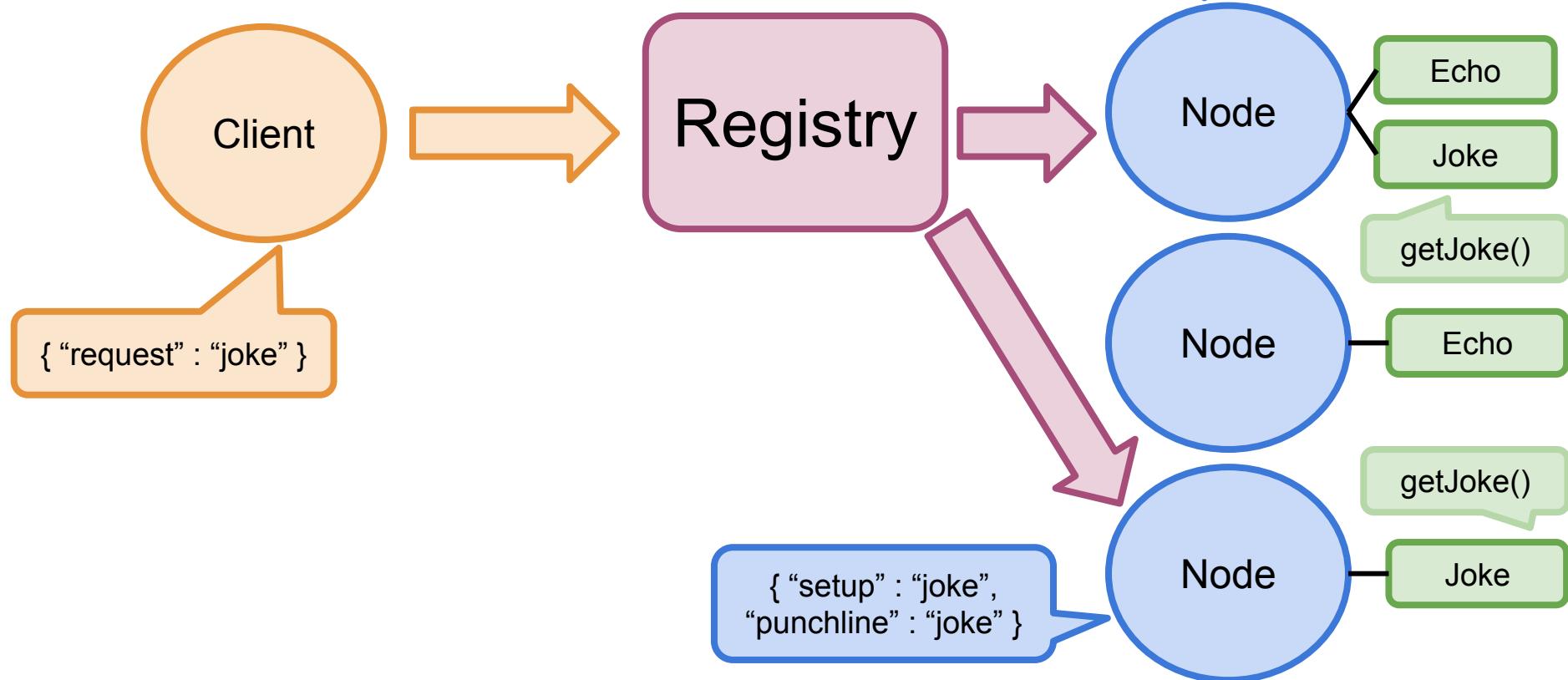
Registry directs us to a node that can handle our request!



Before RPC...

RPC

{ "setup" : "joke",
"punchline" : "joke" }



Before RPC...

RPC

{ "setup" : "joke",
"punchline" : "joke" }



HAHAHA! 😂

Registry

Node

Echo

Joke

getJoke()

Node

Echo

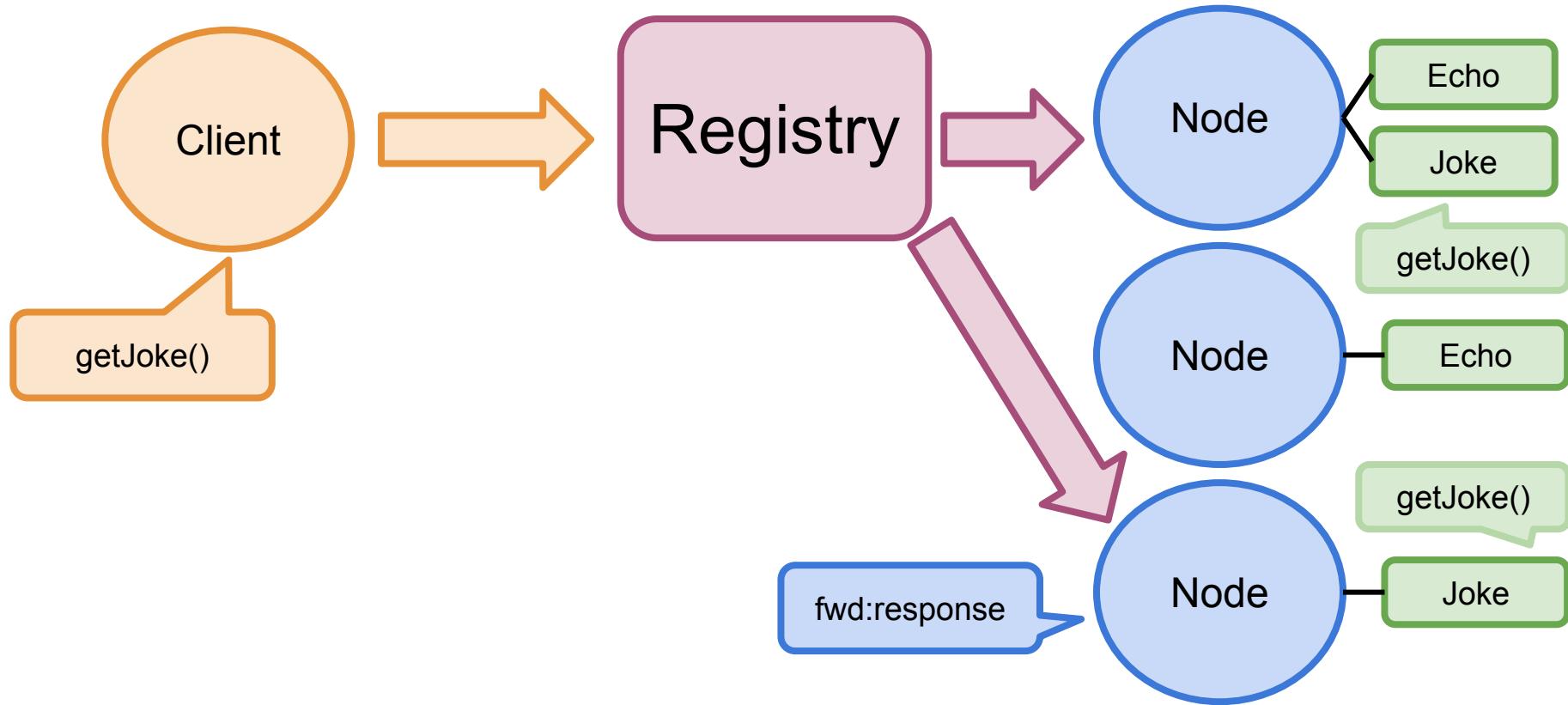
getJoke()

Node

Joke

{ "setup" : "joke",
"punchline" : "joke" }

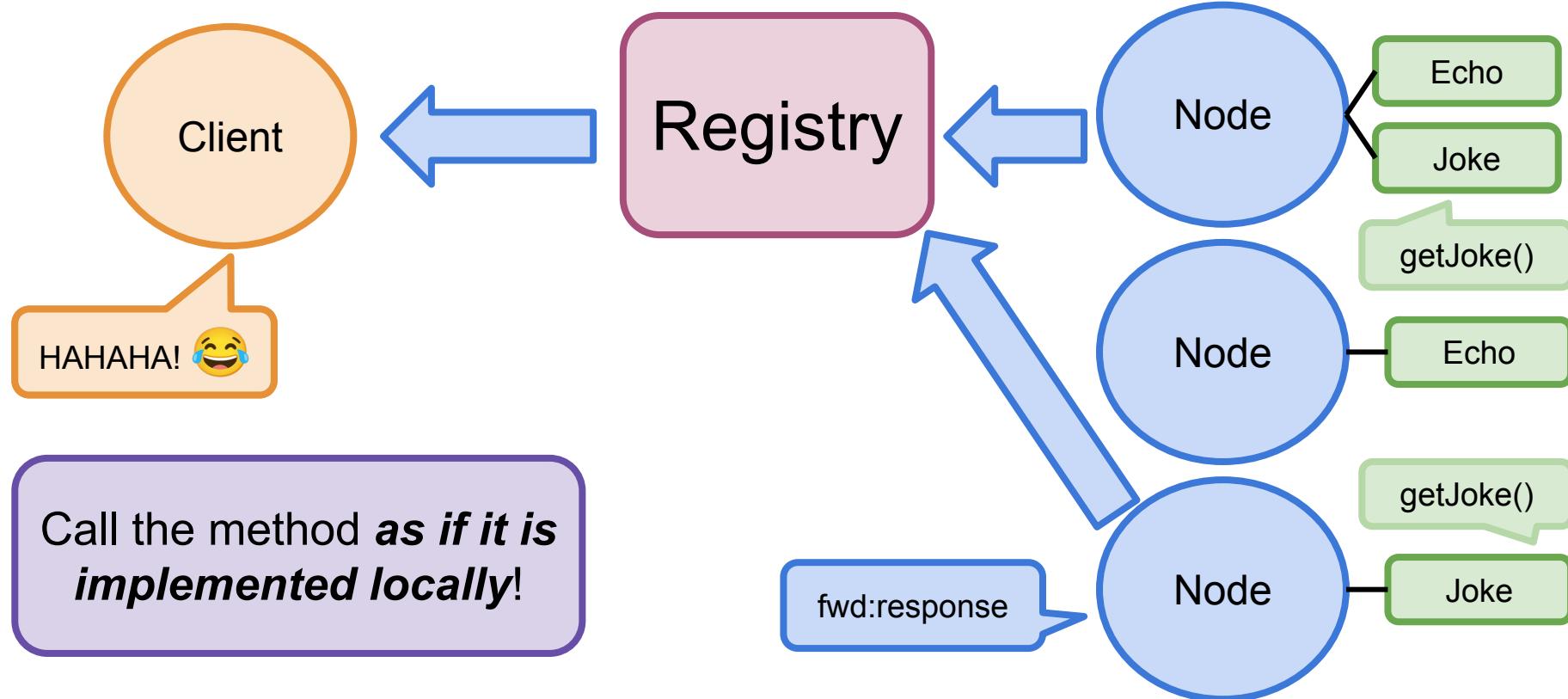
Using RPC...



Using RPC...

RPC

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 jokes  
client.askForJokes(Integer.valueOf(6));  
  
@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);  
    }  
  
    JokeSetRes resp = response.build();  
    responseObserver.onNext(resp);  
    responseObserver.onCompleted();  
}
```

Jokelmpl.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 creates request by calling the method

Client provides the info

Everything else we have had to do is done 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 {  
  
    // having a global set of jokes  
    Stack<String> jokes = new Stack<String>();  
  
    public JokeImpl(){  
        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!");  
    }  
  
    // We implement the setJoke service  
    @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);  
        }  
  
        JokeSetRes resp = response.build();  
        responseObserver.onNext(resp);  
        responseObserver.onCompleted();  
    }  
}
```

JokeImpl.java

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;  
  
    try {  
        response = blockingStub2.setJoke(request);  
    } catch (Exception e) {  
        System.out.println("Error: " + e.getMessage());  
    } finally {  
        System.out.println("Response: " + response);  
    }  
}
```

Client.java

Client Class acts like a
middleman!

*Everything else we have
had to do is done 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 joker 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 jokes  
client.askForJokes(Integer.valueOf(6));  
  
@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);  
    }  
  
    JokeSetRes respo  
    respo  
    respo  
}  
}
```

JokerImpl.java

Implementations need to
be robust and thorough!

SER 321

Scratch Space

Upcoming Events

SI Sessions:

- Tuesday, April 29th, at 10:00 am MST - Q&A Session

Review Sessions:

- ~~Sunday, April 27th at 6:00 pm MST - 2 hour Exam Review Session~~
- Tuesday, April 29th, at 10:00 am MST - Q&A Session

Questions?

Survey:

<https://asuasn.info/ASNSurvey>



More Questions?

Check out our other resources!

tutoring.asu.edu

The screenshot shows the ASU Academic Support Network homepage. At the top, there's a yellow header with the ASU logo and the text "Academic Support Network". Below the header, there's a navigation bar with links for "Services", "Faculty and Staff Resources", and "About Us". A red button labeled "University College" is visible. The main section features a large yellow banner with the text "Academic Support". Below the banner, there's a brief description of the service: "Academic Support Network (ASN) provides a variety of free services in-person and online to help currently enrolled ASU students succeed academically." There are also sections for "Services" and "Online Study Hub" with images and descriptions.

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](#)

[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.



Academic Support Network

University



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!

Select a subject

- Any -

Apply

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](#)

*Available slots for this pilot are limited



tutoring.asu.edu/expanded-writing-support

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](#)