

SER 321 B Session

SI Session

Thursday November 9th 2023

7:00 - 8:00 pm MST

Agenda



Threading Pitfalls & Concurrency

Distributed Systems

Different Structures

Consensus

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

Review Session PSA

Scheduling Poll is live in [#si_channel](#)

We have roughly 5 sessions remaining, followed by the Review Session

If the Review Session is on Monday (11/27), we can have our regularly scheduled session the day before on Sunday, November 26th

Thursday, November 23rd (Thanksgiving!)
Session is Cancelled for the Holiday



SER 321

Threading Pitfalls

- Race Condition



More than one thread accesses a single resource at one time

Crash

- Starvation



One thread never gets access to the resource it needs

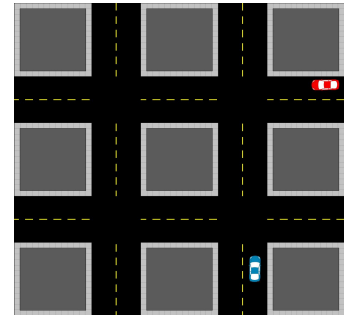
Cross Traffic

- Deadlock



A thread is only able to acquire access to part of its resources

Gridlock




Handling Threaded Pitfalls

Need to prevent threads from stepping on each other!

Mutual Exclusion

Like using a talking stick!

- Locks and semaphores
 - aka **Mutex**
 - Different types with different capabilities
- Atomic Variables
- Monitor



Synchronized

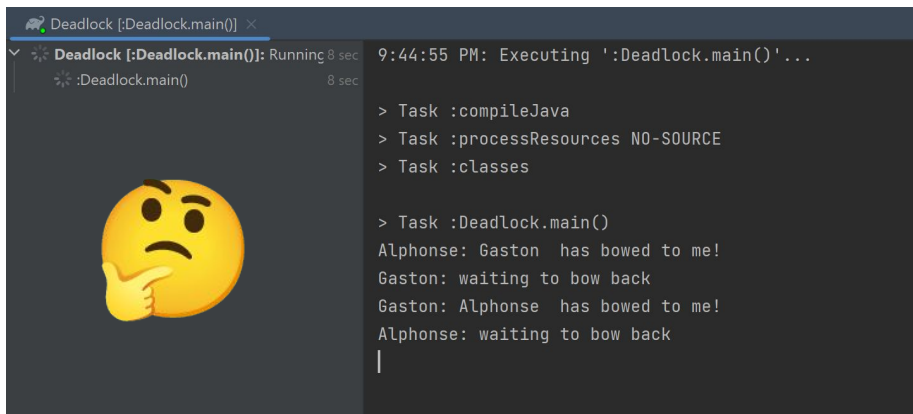
SER 321

Handling Concurrency

Bow and BowBack are both synchronized

The Synchronized keyword marks that method as a **critical section**


So we should be fine, right?



```
Deadlock [Deadlock.main()] x
Deadlock [Deadlock.main()]: Running 8 sec
Deadlock [Deadlock.main()]: Running 8 sec

> Task :compileJava
> Task :processResources NO-SOURCE
> Task :classes

> Task :Deadlock.main()
Alphonse: Gaston has bowed to me!
Gaston: waiting to bow back
Gaston: Alphonse has bowed to me!
Alphonse: waiting to bow back
|
```



```
public class Deadlock {
    6 usages amehlhase316 *
    static class Friend {
        5 usages
        private final String name;
        2 usages amehlhase316
        public Friend(String name) { this.name = name; }
        amehlhase316
        public String getName() { return this.name; }
        /* See the README.md for a reference on 'synchronized' methods */
        2 usages amehlhase316 *
        public synchronized void bow(Friend bower) {
            System.out.format("%s: %s" + " has bowed to me!\n", this.name, bower.getName());
            System.out.format("%s: waiting to bow back\n", bower.getName());
            bower.bowBack( bower: this);
        }
        1 usage amehlhase316 *
        public synchronized void bowBack(Friend bower) {
            System.out.format("%s: waiting", this.name);
            System.out.format("%s: %s" + " has bowed back to me!\n", this.name, bower.getName());
        }
    }

    amehlhase316 *
    public static void main(String[] args) {
        final Friend alphonse = new Friend( name: "Alphonse");
        final Friend gaston = new Friend( name: "Gaston");
        /* start two threads - both operating on the same objects */
        amehlhase316
        new Thread(new Runnable() {
            amehlhase316
            public void run() { alphonse.bow(gaston); }
        }).start();
        amehlhase316
        new Thread(new Runnable() {
            amehlhase316
            public void run() { gaston.bow(alphonse); }
        }).start();
    }
}
```

SAME OBJECT
Cannot both be called
at once

Deadlock from the examples repo

SER 321

Handling Concurrency

How do we fix this?

```
Deadlock [:Deadlock.main()] x
✓ Deadlock [:Deadlock.main()]: success 626 ms
> Task :Deadlock.main()
Alphonse: Gaston has bowed to me!
Gaston: waiting to bow back
Gaston: Alphonse has bowed to me!
Alphonse: waiting to bow back
Alphonse: waiting
Alphonse: Gaston has bowed back to me!
Gaston: waiting
Gaston: Alphonse has bowed back to me!
```

We can synchronize the run method of both threads

or

```
public class Deadlock {
    6 usages  amehlhase316 *
    static class Friend {
        5 usages
        private final String name;
        2 usages  amehlhase316
        public Friend(String name) { this.name = name; }
        amehlhase316
        public String getName() { return this.name; }
        /* See the README.md for a reference on 'synchronized' methods */
        2 usages  amehlhase316 *
        public synchronized void bow(Friend bower) {
            System.out.format("%s: %s" + " has bowed to me!\n", this.name, bower.getName());
            System.out.format("%s: waiting to bow back\n", bower.getName());
            bower.bowBack( bower: this);
        }
        1 usage  amehlhase316 *
        public synchronized void bowBack(Friend bower) {
            System.out.format("%s: waiting", this.name);
            System.out.format("%s: %s" + " has bowed back to me!\n", this.name, bower.getName());
        }
    }
    amehlhase316 *
    public static void main(String[] args) {
        final Friend alphonse = new Friend( name: "Alphonse");
        final Friend gaston = new Friend( name: "Gaston");
        /* start two threads - both operating on the same objects */
        amehlhase316 *
        new Thread(new Runnable() {
            new *
            public synchronized void run() { alphonse.bow(gaston); }
        }).start();
        amehlhase316 *
        new Thread(new Runnable() {
            new *
            public synchronized void run() { gaston.bow(alphonse); }
        }).start();
    }
}
```

Deadlock from the examples repo

SER 321

Handling Concurrency

How do we fix this?

```
Deadlock [:Deadlock.main()] X
✓ Deadlock [:Deadlock.main()]: success 480 ms
> Task :Deadlock.main()
Alphonse: Gaston has bowed to me!
Gaston: waiting to bow back
Gaston: waiting
Gaston: Alphonse has bowed back to me!
Gaston: Alphonse has bowed to me!
Alphonse: waiting to bow back
Alphonse: waiting
Alphonse: Gaston has bowed back to me!
```

or

Or we can synchronize the bowBack method call

Deadlock from the examples repo

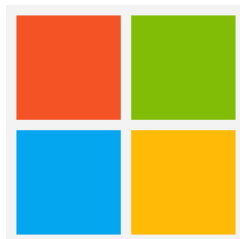
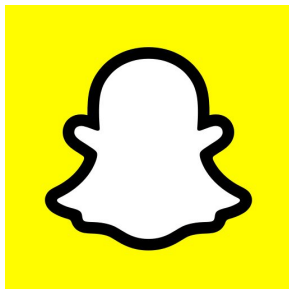
```
public class Deadlock {
    6 usages amehlhase316 *
    static class Friend {
        5 usages
        private final String name;
        2 usages amehlhase316
        public Friend(String name) { this.name = name; }
        amehlhase316
        public String getName() { return this.name; }
        /* See the README.md for a reference on 'synchronized' methods */
        2 usages amehlhase316 *
        public synchronized void bow(Friend bower) {
            System.out.format("%s: %s" + " has bowed to me!\n", this.name, bower.getName());
            System.out.format("%s: waiting to bow back\n", bower.getName());
            synchronized(this) {
                bower.bowBack( bower: this);
            }
        }
        1 usage amehlhase316 *
        public synchronized void bowBack(Friend bower) {
            System.out.format("%s: waiting\n", this.name);
            System.out.format("%s: %s" + " has bowed back to me!\n", this.name, bower.getName());
        }
    }
    amehlhase316 *
    public static void main(String[] args) {
        final Friend alphonse = new Friend( name: "Alphonse");
        final Friend gaston = new Friend( name: "Gaston");
        /* start two threads - both operating on the same objects */
        amehlhase316
        new Thread(new Runnable() {
            amehlhase316
            public void run() { alphonse.bow(gaston); }
        }).start();
        amehlhase316
        new Thread(new Runnable() {
            amehlhase316
            public void run() { gaston.bow(alphonse); }
        }).start();
    }
}
```

SER 321

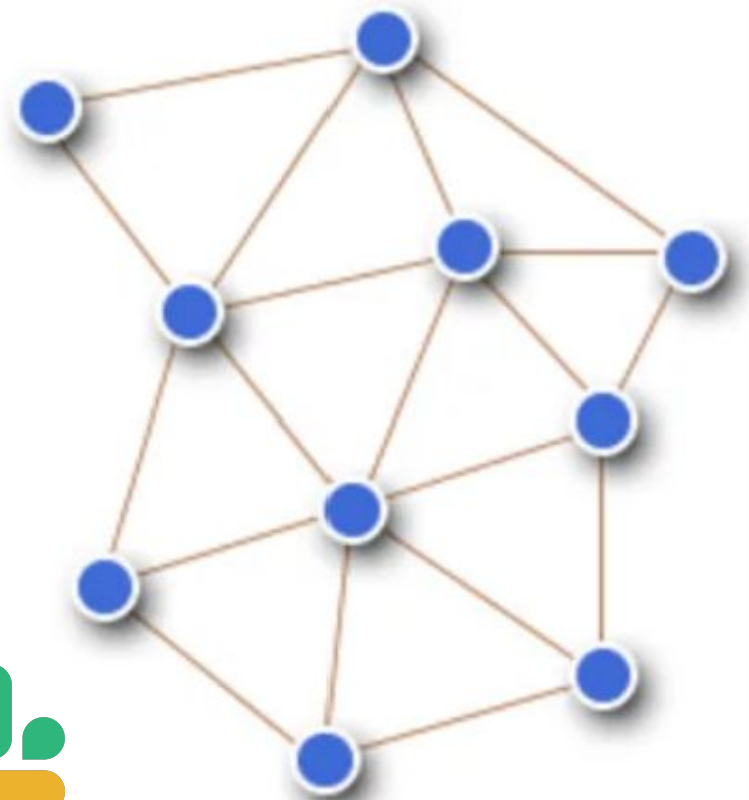
Distributed Algorithms

What's a distributed system again?

Many *nodes* working together that *appear* to be a single system from the outside



zoom

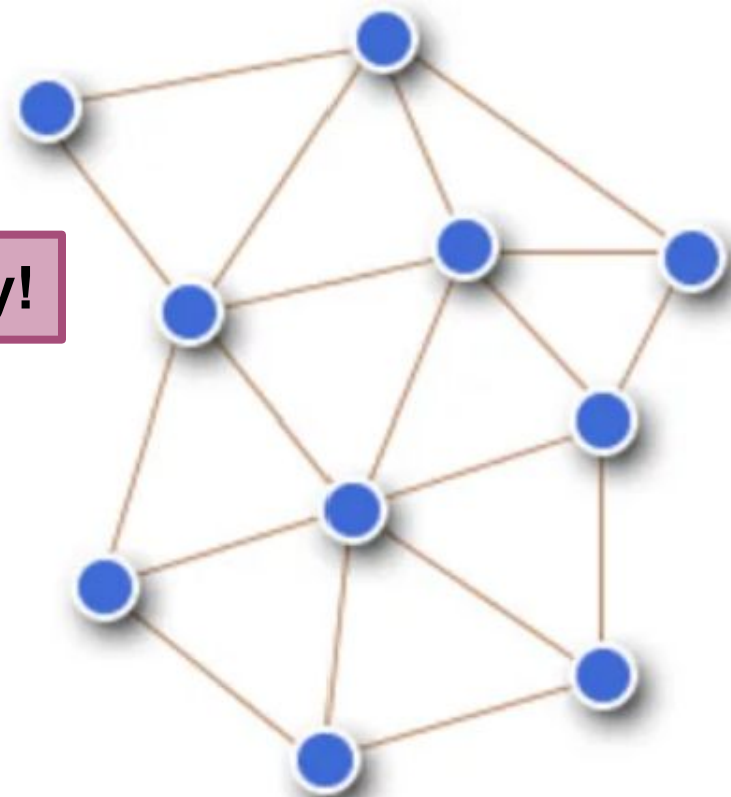


SER 321

Distributed Algorithms

Common Issues: aka working in reality!

- Handle Node failures
- Account for latency
- Account for network failures
- Protection of shared resources
- Prevention of deadlocks
- Execution safety - no errors or gross, bad stuff
- Ensuring liveness - everyone goes eventually



SER 321

Distributed Algorithms

We look at two main structure forms:

Peer to Peer

Main and Worker

SER 321

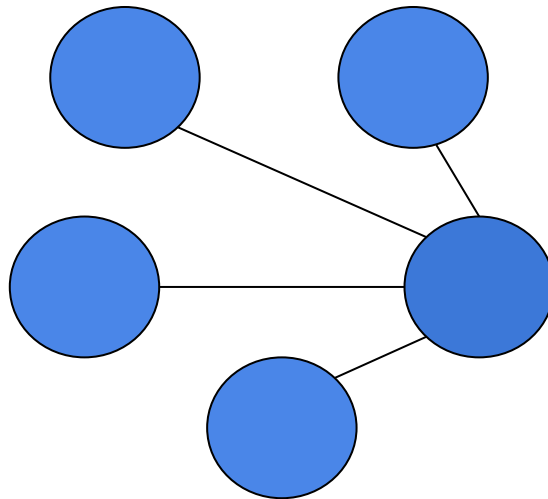
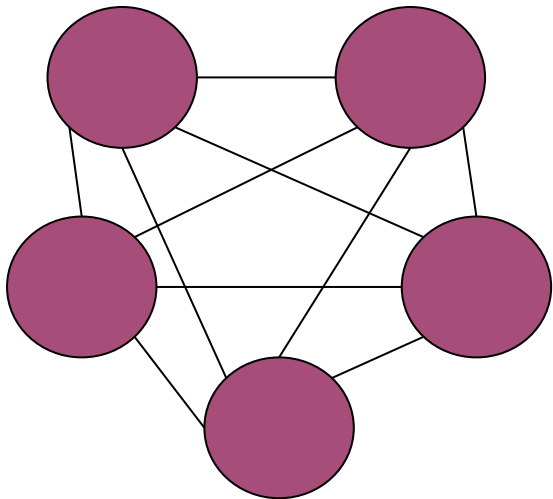
Distributed Algorithms

Which is which?

Main and Worker

Peer to Peer

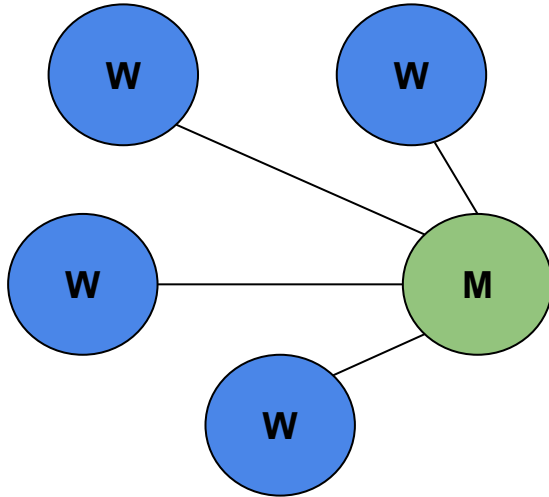
Check out the recording for the solution!



SER 321

Distributed Algorithms

Pros and Cons



PRO

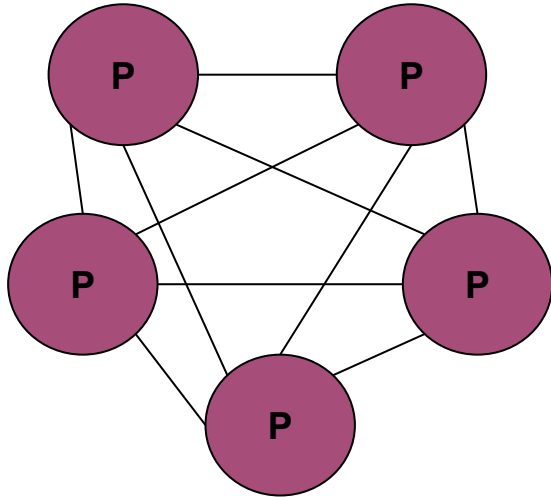
CON

Check out the recording for the solution!

SER 321

Distributed Algorithms

Pros and Cons



PRO

CON

Check out the recording for the solution!

SER 321

Consensus

What is Consensus?

- A.** Systematic calculating and recording of information about a given population
- B.** General agreement or trust amongst a group
- C.** Controversial; causing or likely to cause an argument
- D.** No Idea...

Check out the recording for the solution!

Questions?

Survey:

https://bit.ly/asn_survey



Upcoming Events

SI Sessions:

- Sunday, November 12th 2023 at 7:00 pm MST
- Monday, November 13th 2023 at 4:00 pm MST
- Thursday, November 16th 2023 at 7:00 pm MST
- Sunday, November 19th 2023 at 7:00 pm MST
- Monday, November 20th 2023 at 4:00 pm MST

Review Sessions:

- Survey is LIVE in the [#si_channel](#)
- Sunday, November 26th or Monday November 27th

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

 **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!

Additional Resources

[CoureRepo](#)

[Dining Philosophers Interactive](#)

[Austin Walter's Traffic Comparison](#)

[RAFT](#)