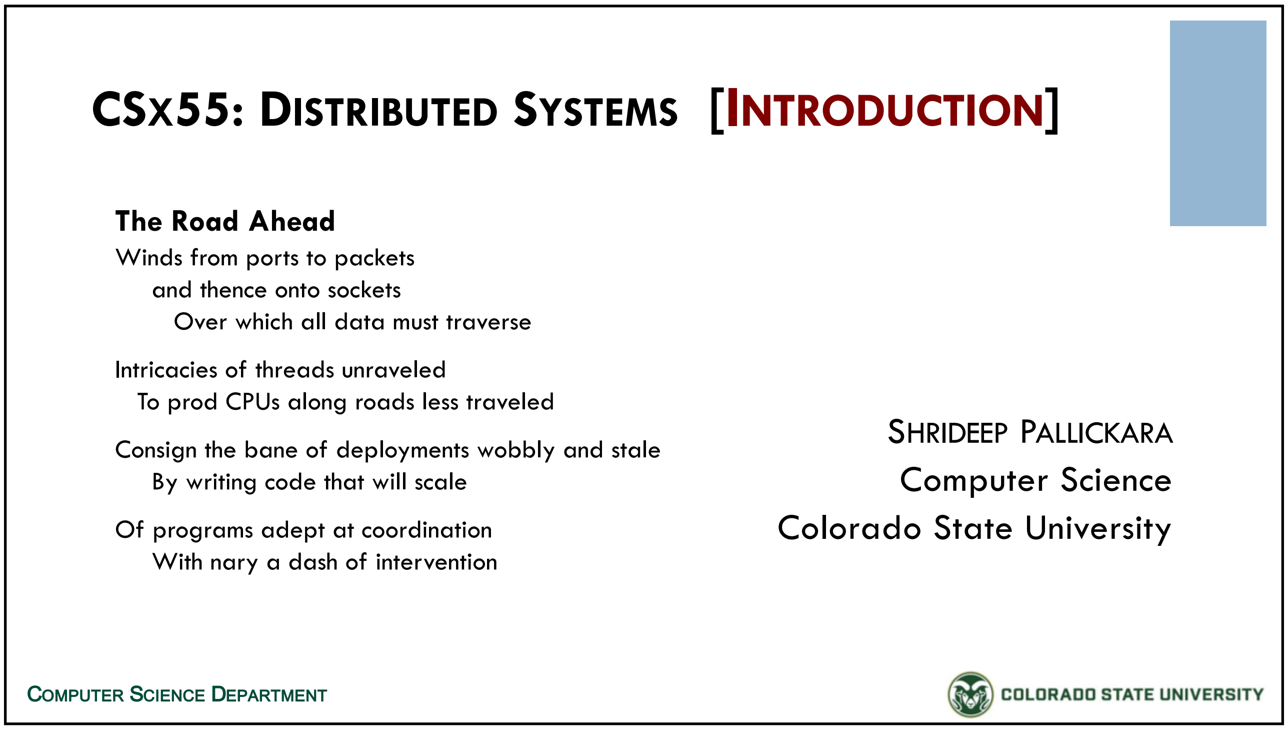


CSx55: Distributed Systems [Introduction]

The Road Ahead
Winds from ports to packets
and thence onto sockets
Over which all data must traverse

Intricacies of threads unraveled
To prod CPUs along roads less traveled

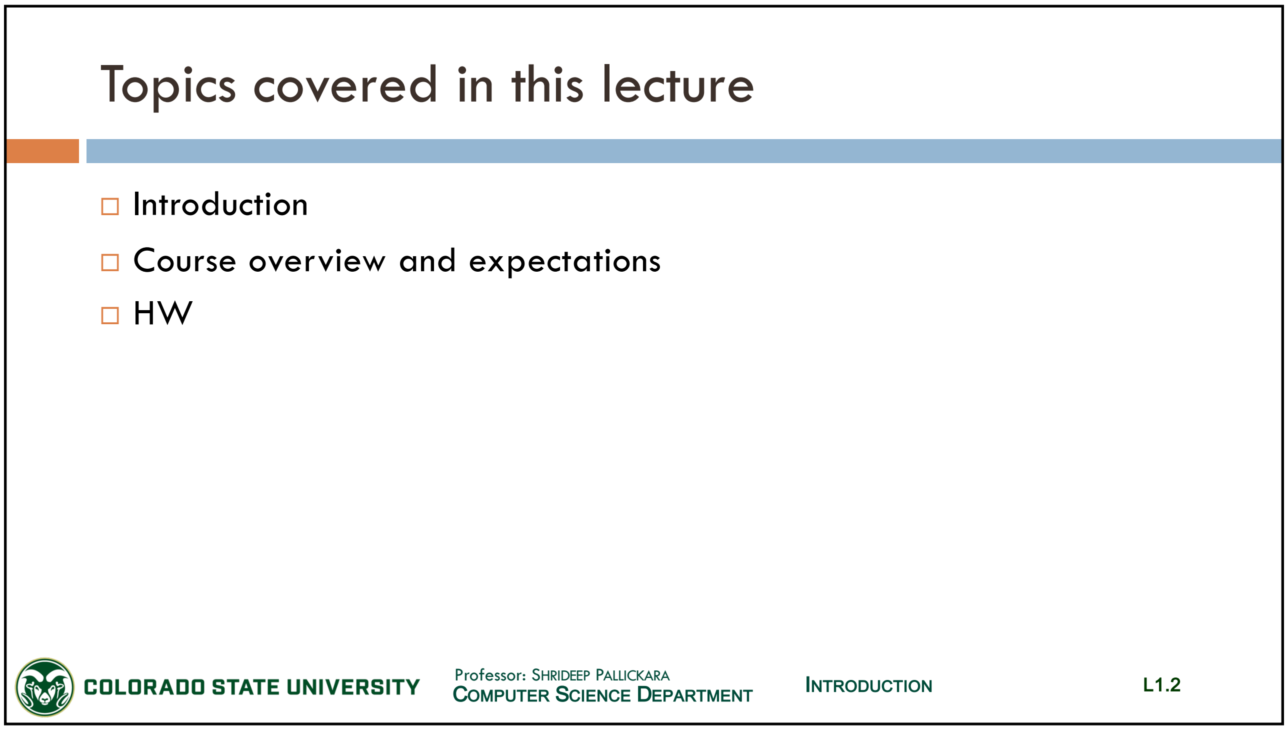
Consign the bane of deployments wobbly and stale
By writing code that will scale

Of programs adept at coordination
With nary a dash of intervention

Shrideep Pallickara
Computer Science
Colorado State University

Computer Science Department
Colorado State University

1

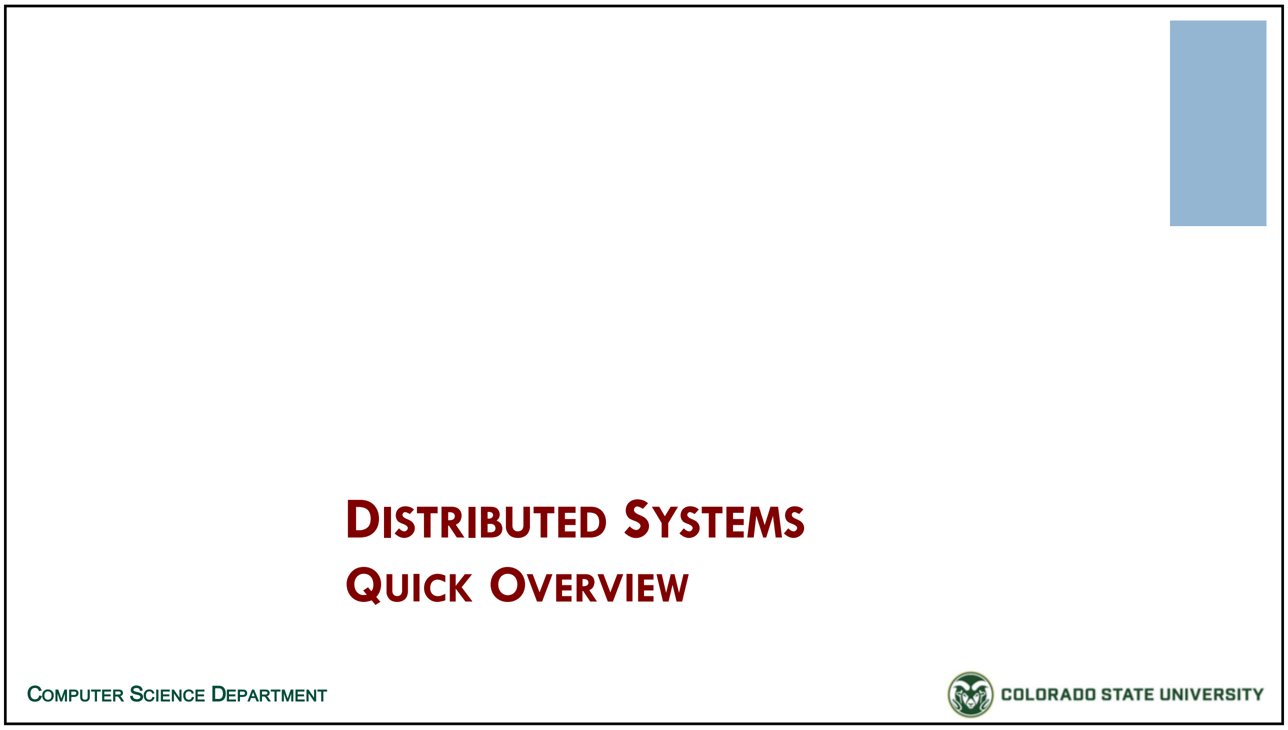


Topics covered in this lecture

- Introduction
- Course overview and expectations
- HW

Colorado State University   Professor: Shrideep Pallickara Computer Science Department   Introduction   L1.2

2

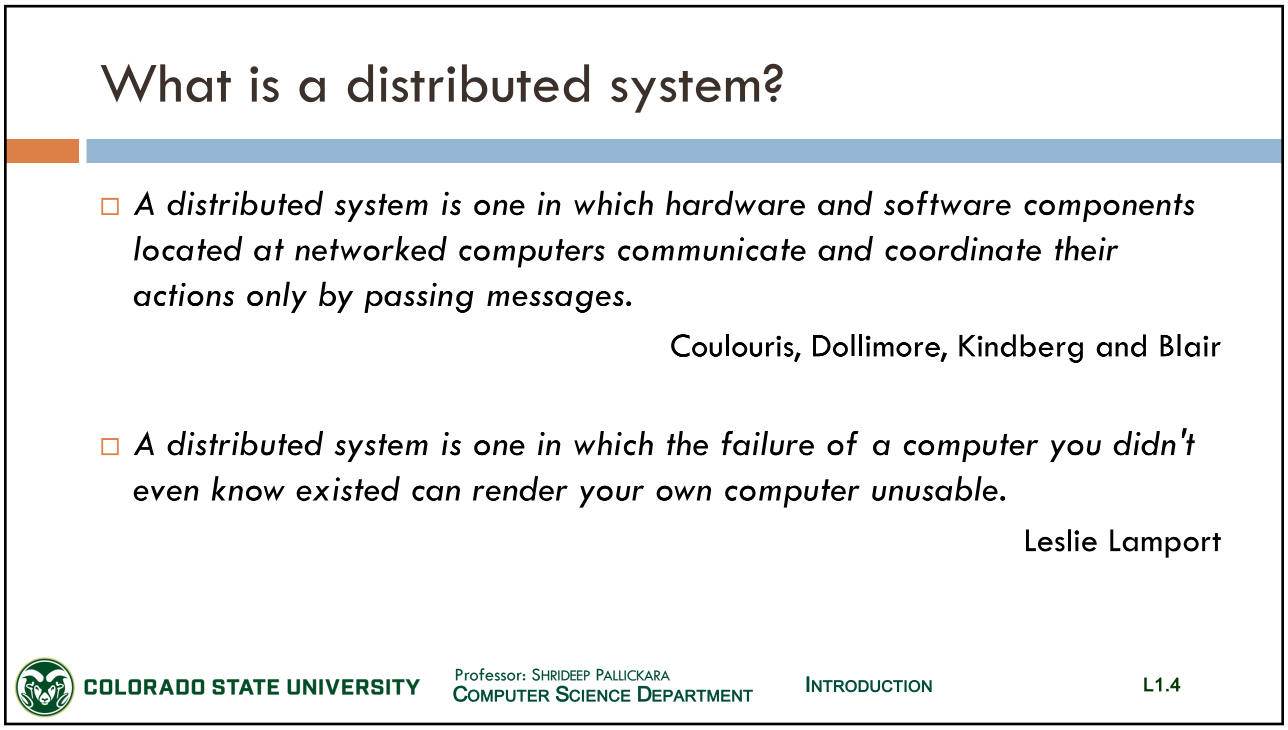


DISTRIBUTED SYSTEMS
QUICK OVERVIEW

COMPUTER SCIENCE DEPARTMENT

COLORADO STATE UNIVERSITY

3



What is a distributed system?

- *A distributed system is one in which hardware and software components located at networked computers communicate and coordinate their actions only by passing messages.*

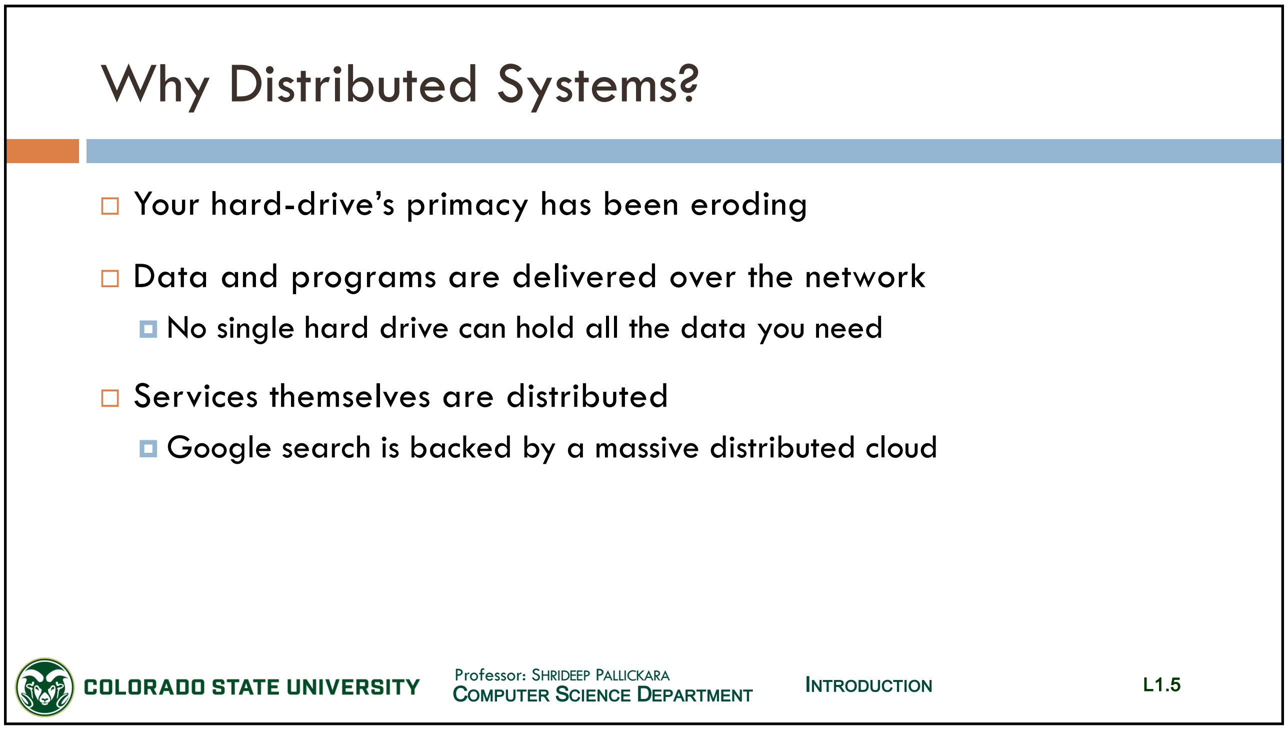
  Coulouris, Dollimore, Kindberg and Blair

- *A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.*

  Leslie Lamport

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.4

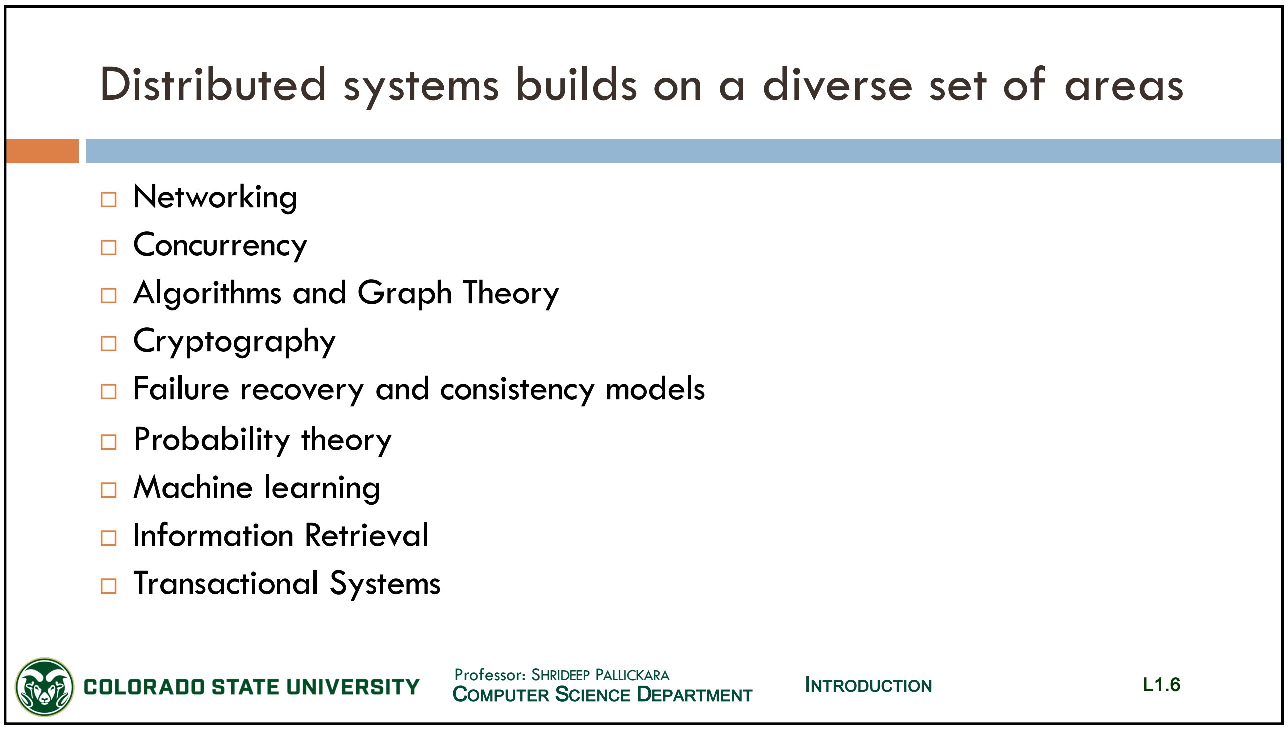
4



Why Distributed Systems?

- Your hard-drive's primacy has been eroding
- Data and programs are delivered over the network
  - No single hard drive can hold all the data you need
- Services themselves are distributed
  - Google search is backed by a massive distributed cloud

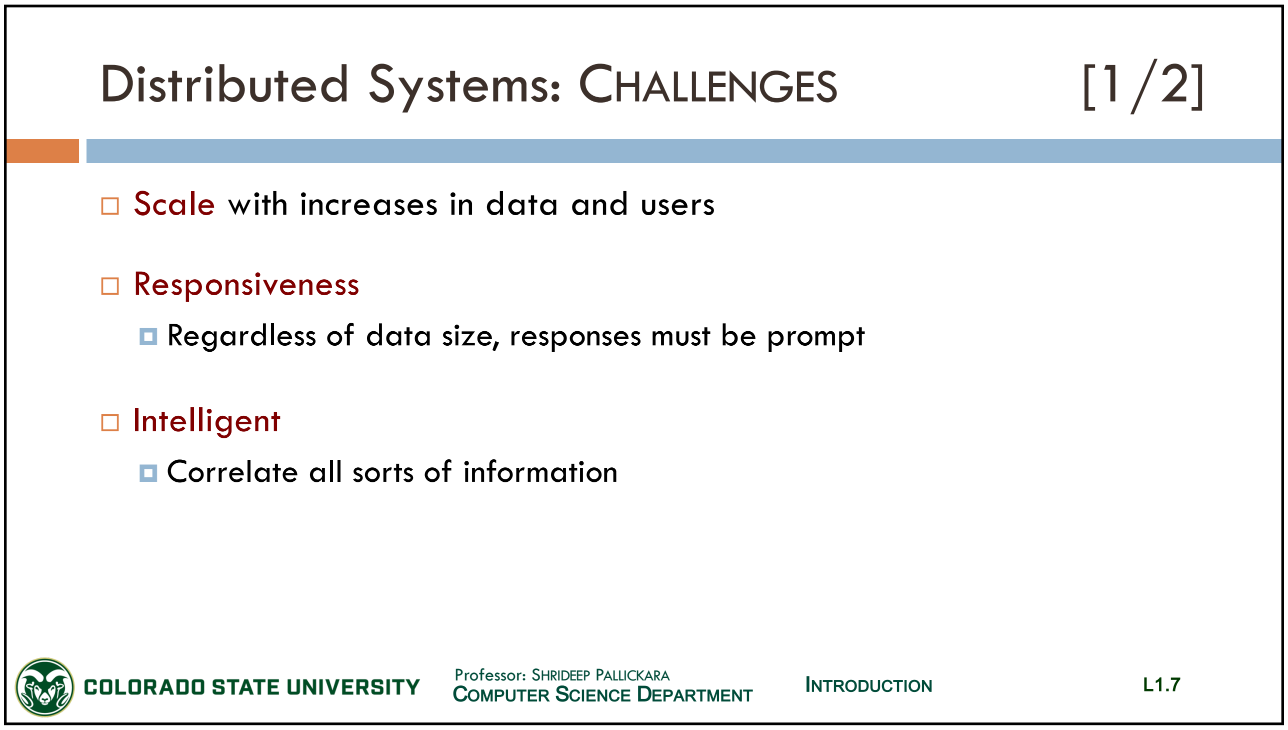
5



Distributed systems builds on a diverse set of areas

- Networking
- Concurrency
- Algorithms and Graph Theory
- Cryptography
- Failure recovery and consistency models
- Probability theory
- Machine learning
- Information Retrieval
- Transactional Systems

6

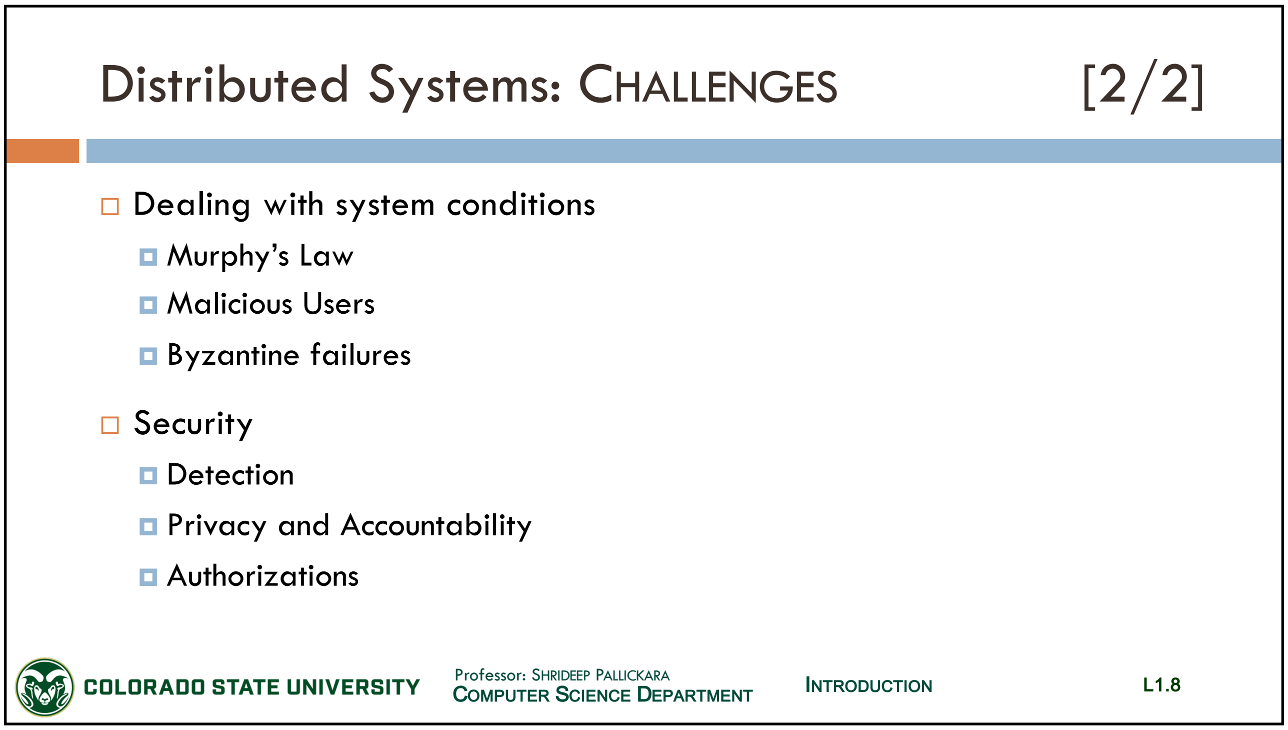


Distributed Systems: CHALLENGES [1/2]

- Scale with increases in data and users
- Responsiveness
  - Regardless of data size, responses must be prompt
- Intelligent
  - Correlate all sorts of information

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.7

7



Distributed Systems: CHALLENGES [2/2]

- Dealing with system conditions
  - Murphy's Law
  - Malicious Users
  - Byzantine failures
- Security
  - Detection
  - Privacy and Accountability
  - Authorizations

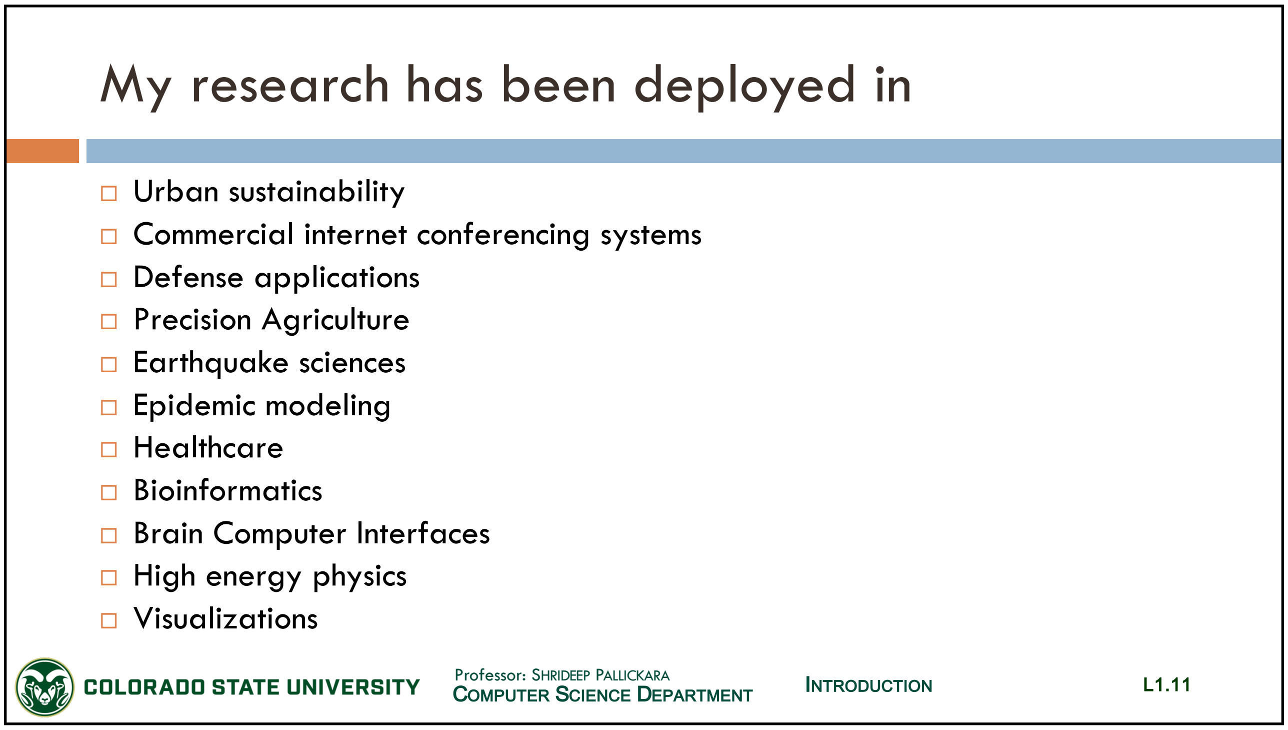
COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.8

8

ABOUT ME

About me

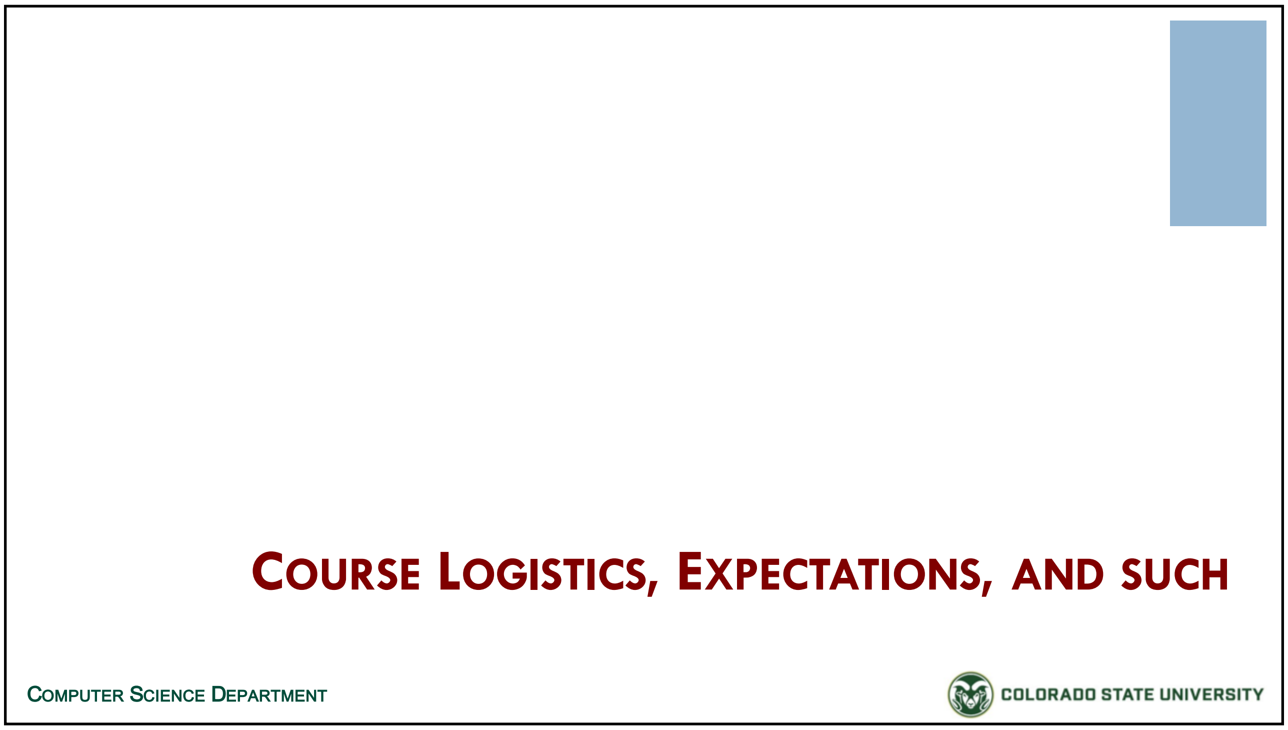
- I do research in the area of large-scale computing systems, Big Data, and GeoAI
- My research has been funded by agencies in the United States and the United Kingdom
  - These include the National Science Foundation, the Department of Homeland Security (including the *Long Range* program), the Environmental Protection Agency, the Department of Agriculture, the National Institute of Food & Agriculture, the National Endowment for the Humanities/Teagle and the U.K's e-Science program
  - Recipient of the National Science Foundation's CAREER Award
  - I direct the Center for eXascale Spatial Data Analytics and Computing (XSD) @ CSU [**https://spatial.colostate.edu**]



My research has been deployed in

- Urban sustainability
- Commercial internet conferencing systems
- Defense applications
- Precision Agriculture
- Earthquake sciences
- Epidemic modeling
- Healthcare
- Bioinformatics
- Brain Computer Interfaces
- High energy physics
- Visualizations

11



COURSE LOGISTICS, EXPECTATIONS, AND SUCH

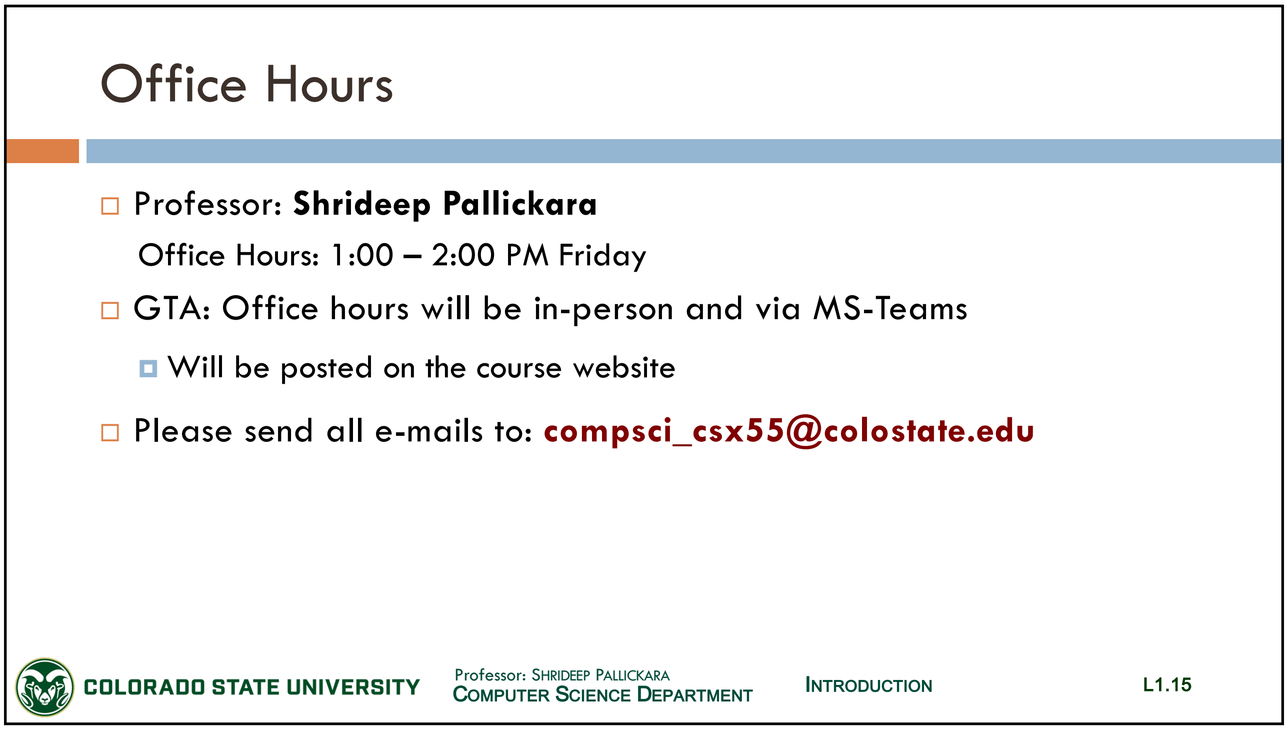
12

You are not allowed to take away learning opportunities from other students

- If you must use a laptop, you will have to sit in the last row
  - Turn off wireless and use it only for taking notes
  - Students not using a laptop should avoid sitting in the last row
- Tablet use with a stylus/pencil is allowed
  - Tablet use in keyboard mode will be treated the same way as a laptop
- When the class is in session, put away your cell-phones!
- Also, please no cross talking when the class is in-session

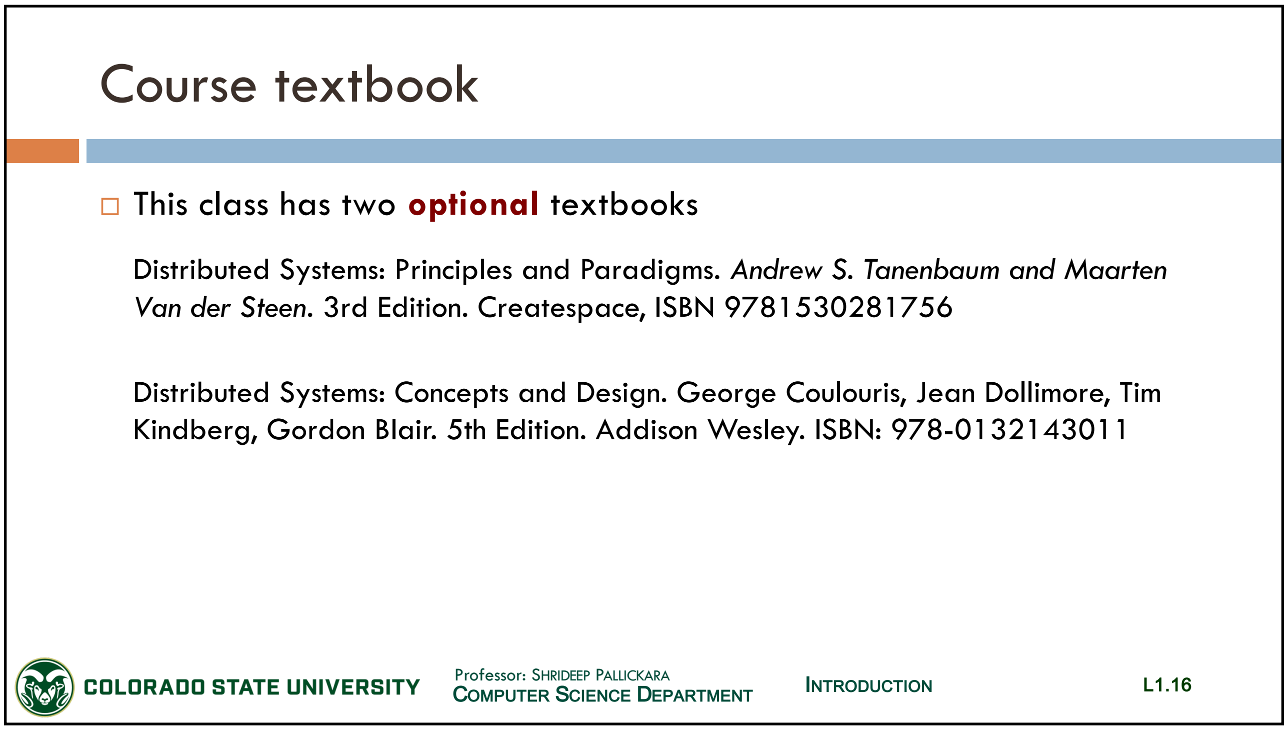
Course webpage

- All course materials will be on the course webpage **http://www.cs.colostate.edu/~cs455**
  - Schedule
  - Lectures
  - Assignments
  - Infospaces, Syllabus
- We will use Canvas for submission of assignments and grade reporting
- For the 801 sections of CS455 and CS555
  - Canvas will be used to administer exams and release lecture recordings
    - Lecture recordings are in a special Canvas section: COM-NS-CSx55-Distributed-Systems-PallickaraS
    - If you are in the 801 section, and haven't been added to the COM-NS-CSx55-Distributed-Systems-PallickaraS section please send an e-mail to **compsci_csx55@colostate.edu**



Office Hours

- Professor: **Shrideep Pallickara**
  Office Hours: 1:00 – 2:00 PM Friday
- GTA: Office hours will be in-person and via MS-Teams
  - Will be posted on the course website
- Please send all e-mails to: **compsci_csx55@colostate.edu**



Course textbook

- This class has two **optional** textbooks

  Distributed Systems: Principles and Paradigms. *Andrew S. Tanenbaum and Maarten Van der Steen*. 3rd Edition. Createspace, ISBN 9781530281756

  Distributed Systems: Concepts and Design. George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair. 5th Edition. Addison Wesley. ISBN: 978-0132143011

When I make slides …

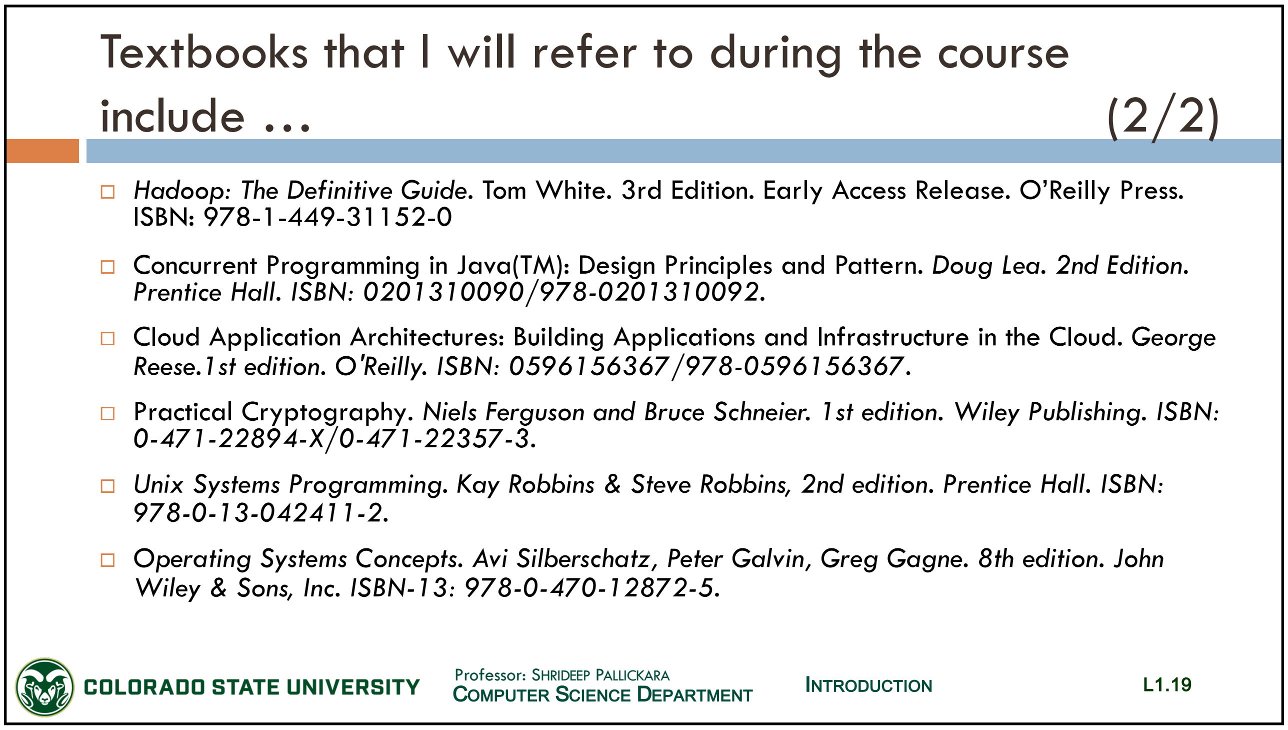
- I usually refer to several texts
  - And technical papers and articles (with URLs)
- I always list my references at the end of every slide set

17

Textbooks that I will refer to during the course include … (1/2)

- Distributed Systems: Principles and Paradigms. *Andrew S. Tanenbaum and Maarten Van der Steen. 2nd Edition. Prentice Hall. ISBN: 0132392275/978-0132392273.*
- Distributed Systems: Concepts and Design. George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair. 5th Edition. Addison Wesley. ISBN: 978-0132143011
- Distributed Computing: Principles, Algorithms, and Systems. *Ajay Kshemkalyani and Mukesh Singhal. 1st edition. Cambridge University Press. ISBN: 0521876346/ 978-0521876346*
- Computer Networks: A Systems Approach. *Larry Peterson and Bruce Davie. 4th edition. Morgan Kaufmann. ISBN: 978-0-12-370548-8.*
- *Java Concurrency in Practice*. Brian Goetz, Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes, and Doug Lea. Addison-Wesley Professional. ISBN: 0321349601/978-0321349606.
- *Java Threads*. Scott Oaks and Henry Wong. . 3rd Edition. O'Reilly Press. ISBN: 0-596-00782-5/978-0-596-00782-9

18

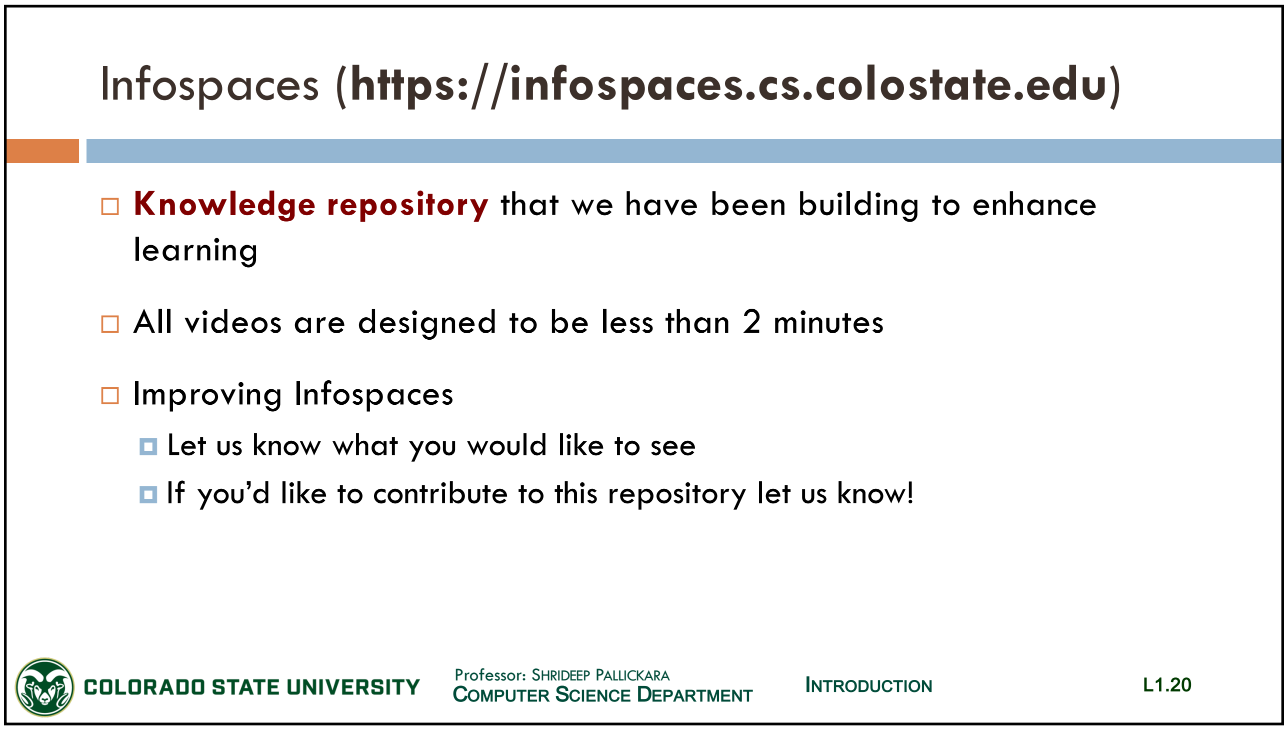


Textbooks that I will refer to during the course include … (2/2)

- *Hadoop: The Definitive Guide*. Tom White. 3rd Edition. Early Access Release. O'Reilly Press. ISBN: 978-1-449-31152-0
- Concurrent Programming in Java(TM): Design Principles and Pattern. *Doug Lea. 2nd Edition. Prentice Hall. ISBN: 0201310090/978-0201310092.*
- Cloud Application Architectures: Building Applications and Infrastructure in the Cloud. *George Reese.1st edition. O'Reilly. ISBN: 0596156367/978-0596156367.*
- Practical Cryptography. *Niels Ferguson and Bruce Schneier. 1st edition. Wiley Publishing. ISBN: 0-471-22894-X/0-471-22357-3.*
- *Unix Systems Programming. Kay Robbins & Steve Robbins, 2nd edition. Prentice Hall. ISBN: 978-0-13-042411-2.*
- *Operating Systems Concepts. Avi Silberschatz, Peter Galvin, Greg Gagne. 8th edition. John Wiley & Sons, Inc. ISBN-13: 978-0-470-12872-5.*

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.19

19



Infospaces (**https://infospaces.cs.colostate.edu**)

- **Knowledge repository** that we have been building to enhance learning
- All videos are designed to be less than 2 minutes
- Improving Infospaces
  - Let us know what you would like to see
  - If you'd like to contribute to this repository let us know!

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.20

20

GRADING

COMPUTER SCIENCE DEPARTMENT — COLORADO STATE UNIVERSITY

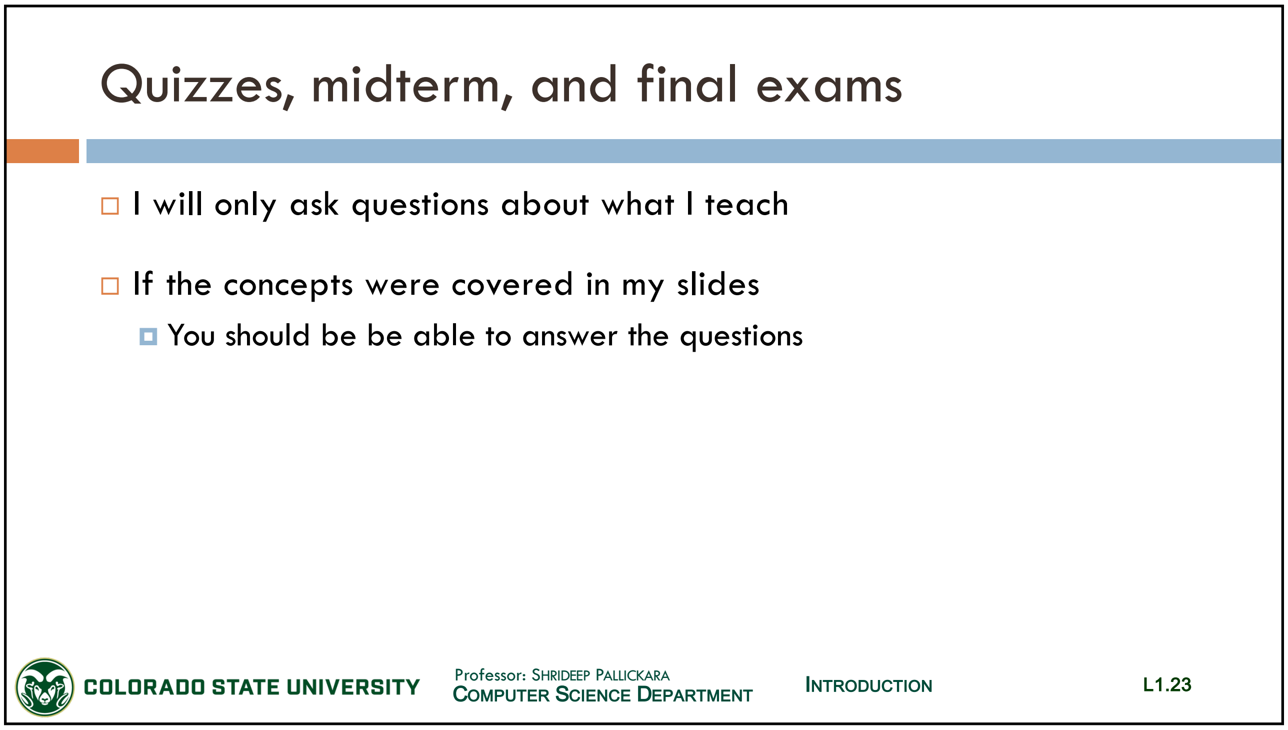
21

Grading breakdown

- Assignments: 60%
  - HW1: 15%; HW2: 15%; HW3: 15%; HW4: 15%
- Term project [report and presentation]: 10%
- Quizzes (10 best) : 10%
- Midterm: 10%
- Comprehensive final exam: 10%

COLORADO STATE UNIVERSITY — Professor: SHRIDEEP PALLICKARA, COMPUTER SCIENCE DEPARTMENT — INTRODUCTION — L1.22

22

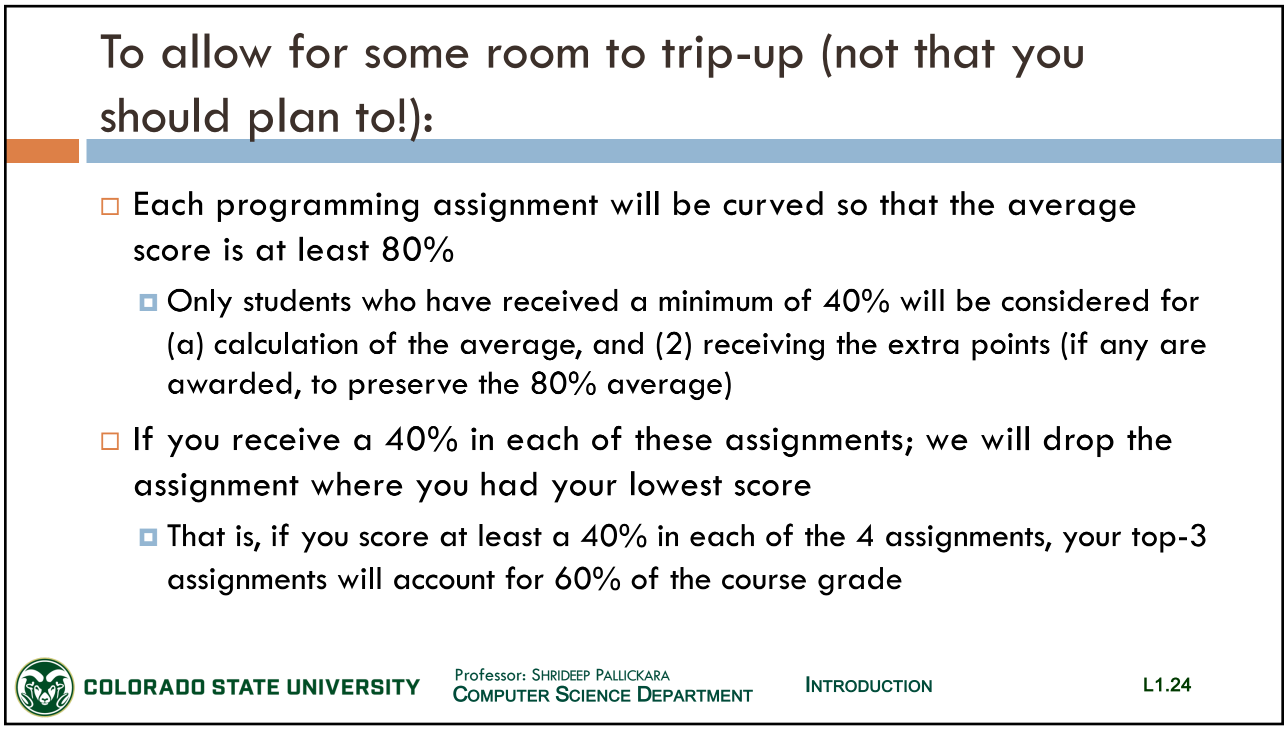


Quizzes, midterm, and final exams

- I will only ask questions about what I teach
- If the concepts were covered in my slides
  - You should be be able to answer the questions

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.23

23

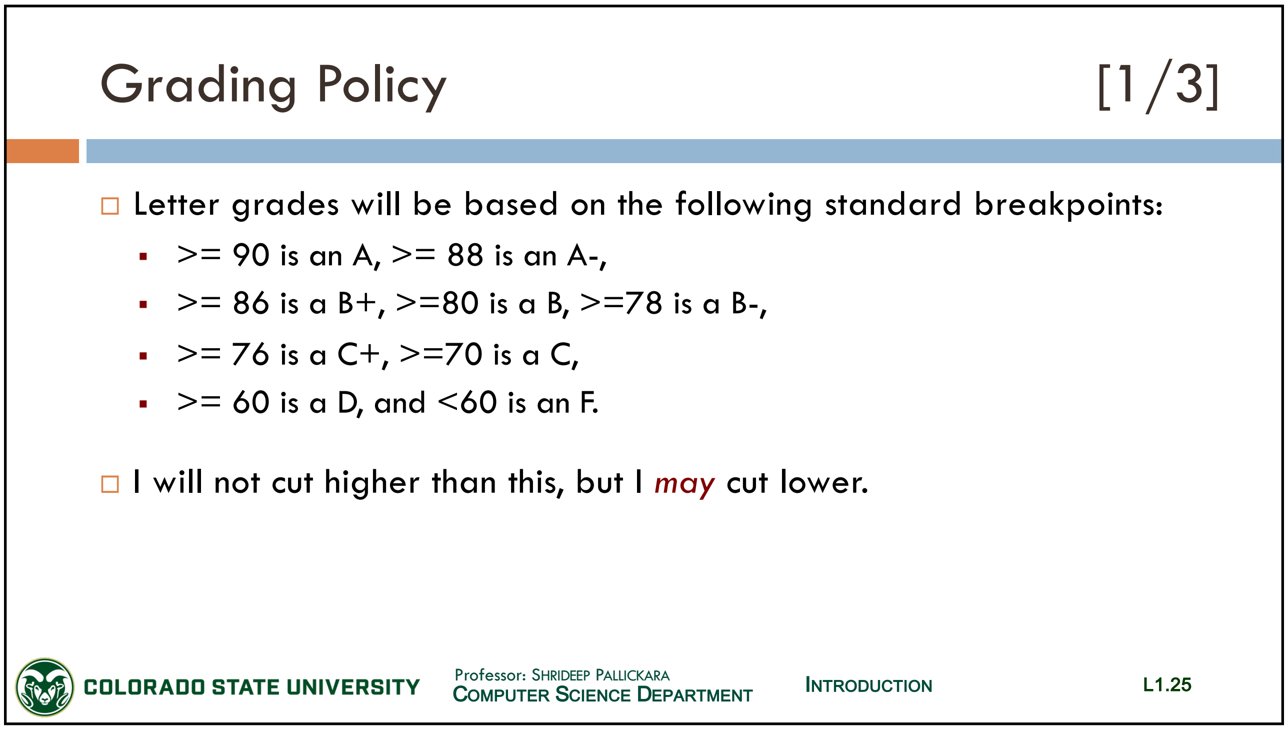


To allow for some room to trip-up (not that you should plan to!):

- Each programming assignment will be curved so that the average score is at least 80%
  - Only students who have received a minimum of 40% will be considered for (a) calculation of the average, and (2) receiving the extra points (if any are awarded, to preserve the 80% average)
- If you receive a 40% in each of these assignments; we will drop the assignment where you had your lowest score
  - That is, if you score at least a 40% in each of the 4 assignments, your top-3 assignments will account for 60% of the course grade

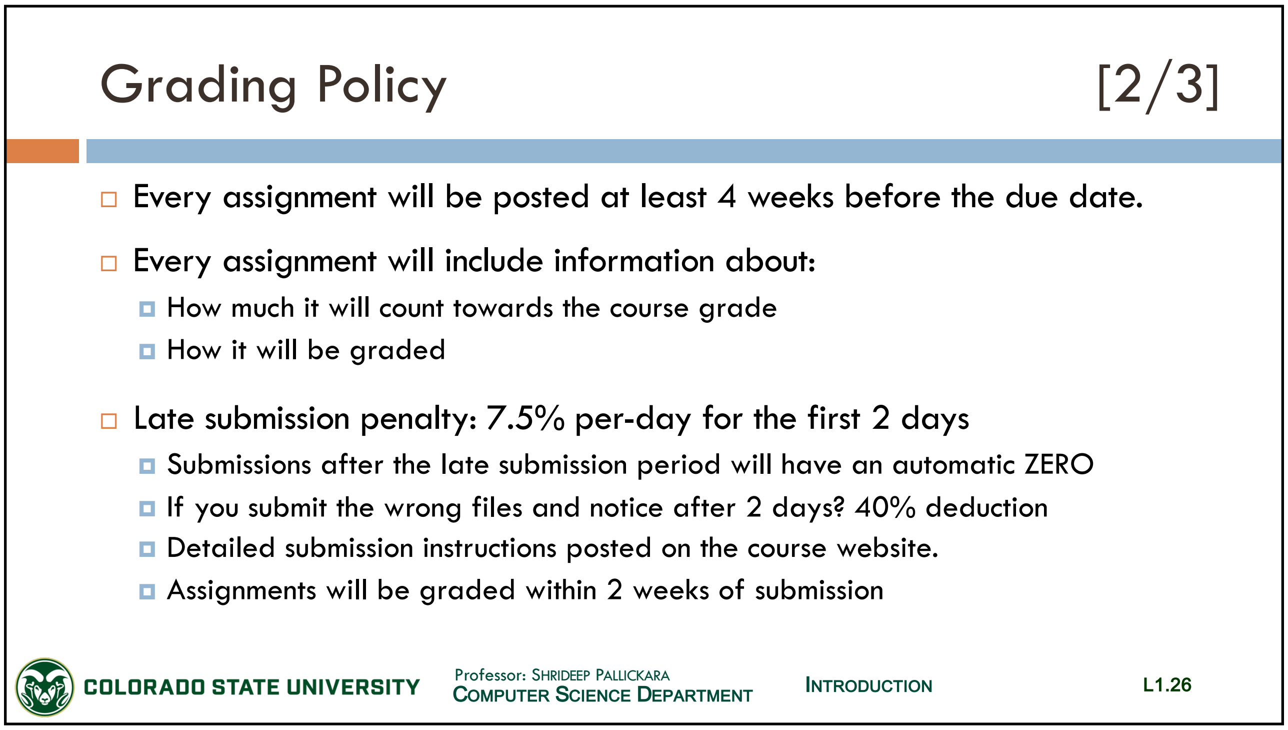
COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.24

24



Grading Policy [1/3]

- Letter grades will be based on the following standard breakpoints:
  - >= 90 is an A, >= 88 is an A-,
  - >= 86 is a B+, >=80 is a B, >=78 is a B-,
  - >= 76 is a C+, >=70 is a C,
  - >= 60 is a D, and <60 is an F.
- I will not cut higher than this, but I *may* cut lower.



Grading Policy [2/3]

- Every assignment will be posted at least 4 weeks before the due date.
- Every assignment will include information about:
  - How much it will count towards the course grade
  - How it will be graded
- Late submission penalty: 7.5% per-day for the first 2 days
  - Submissions after the late submission period will have an automatic ZERO
  - If you submit the wrong files and notice after 2 days? 40% deduction
  - Detailed submission instructions posted on the course website.
  - Assignments will be graded within 2 weeks of submission

Grading Policy [3/3]

- If you have problems with the grading
  - Talk to the GTAs first
- The GTAs strive to ensure that the grading is consistent across the board

27

ASSIGNMENTS

28

Assignments: What to expect

- There will be no busy work
  - No GUI
- Complexity will not be through obfuscation
- You will be able to look back and feel good about them
  - Delayed gratification

Assignments

- You will have about **4 weeks** to complete each assignment
- The assignments will include **milestones** that should be achieved for each week

Term project

- The term project is a group effort
  - CS455 students can only be in a team with CS455 students
    - Similarly, CS555 students can only be in a team with other CS555 students
  - Team size = 2-3 for CS455 students and you can choose your teammate
    - CS555 students: Team size is 2
  - Please **respond** to your teammate's e-mails on time!
    - Make sure teammate has the e-mail that you check regularly
  - If you have problems finding a teammate, please let us know via e-mail

Assignments: Logistics

- Programming assignments will be due on **Wednesdays at 8:00 pm**
- You are allowed to submit up to 2 days late
  - There is a **7.5%** deduction for each day that you are late
- All programming assignments are to be done **solo**
  - Java version 11; Gradle version 8.3

WHAT IT TAKES TO SUCCEED

COMPUTER SCIENCE DEPARTMENT COLORADO STATE UNIVERSITY

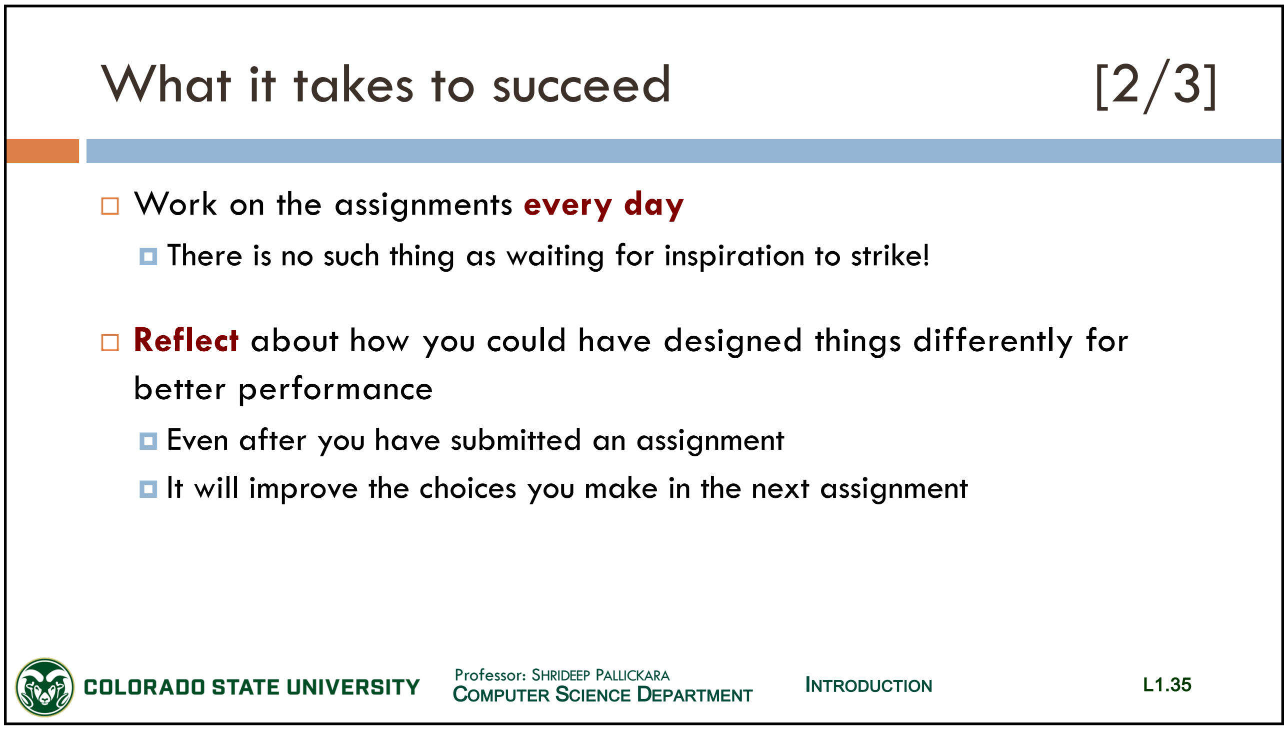
33

What it takes to succeed [1/3]

- You should plan to work **~12 hours** per-week outside of class
  - Coding and reviewing material from class
- If you miss a lecture
  - Add about 3 hours per missed lecture

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.34

34

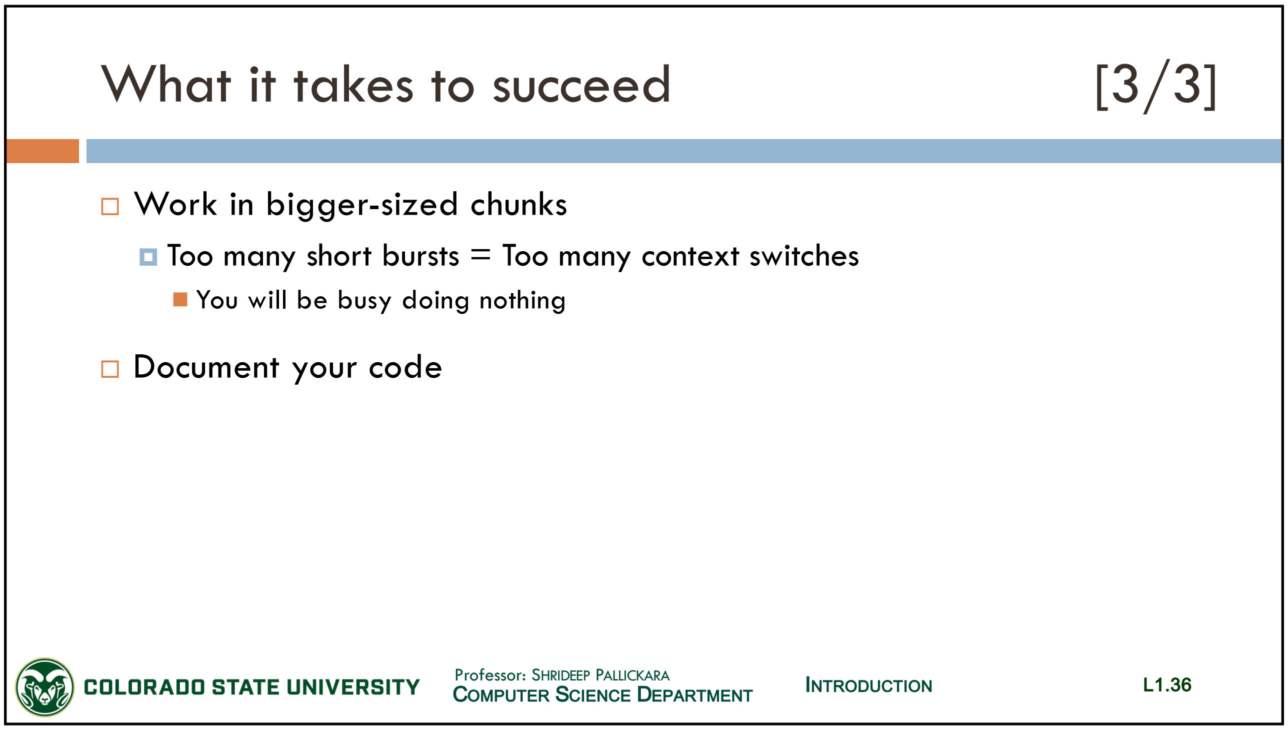


What it takes to succeed [2/3]

- Work on the assignments **every day**
  - There is no such thing as waiting for inspiration to strike!
- **Reflect** about how you could have designed things differently for better performance
  - Even after you have submitted an assignment
  - It will improve the choices you make in the next assignment

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.35

35



What it takes to succeed [3/3]

- Work in bigger-sized chunks
  - Too many short bursts = Too many context switches
    - You will be busy doing nothing
- Document your code

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.36

36

Other pitfalls

- Poor management of **course loads**
  - Plan the number and type of courses you take
  - Don't spread yourself so thin that you do not give yourself the opportunity to succeed
- Not attacking the problem and working on the fringes
  - Spend your time wisely on critical paths

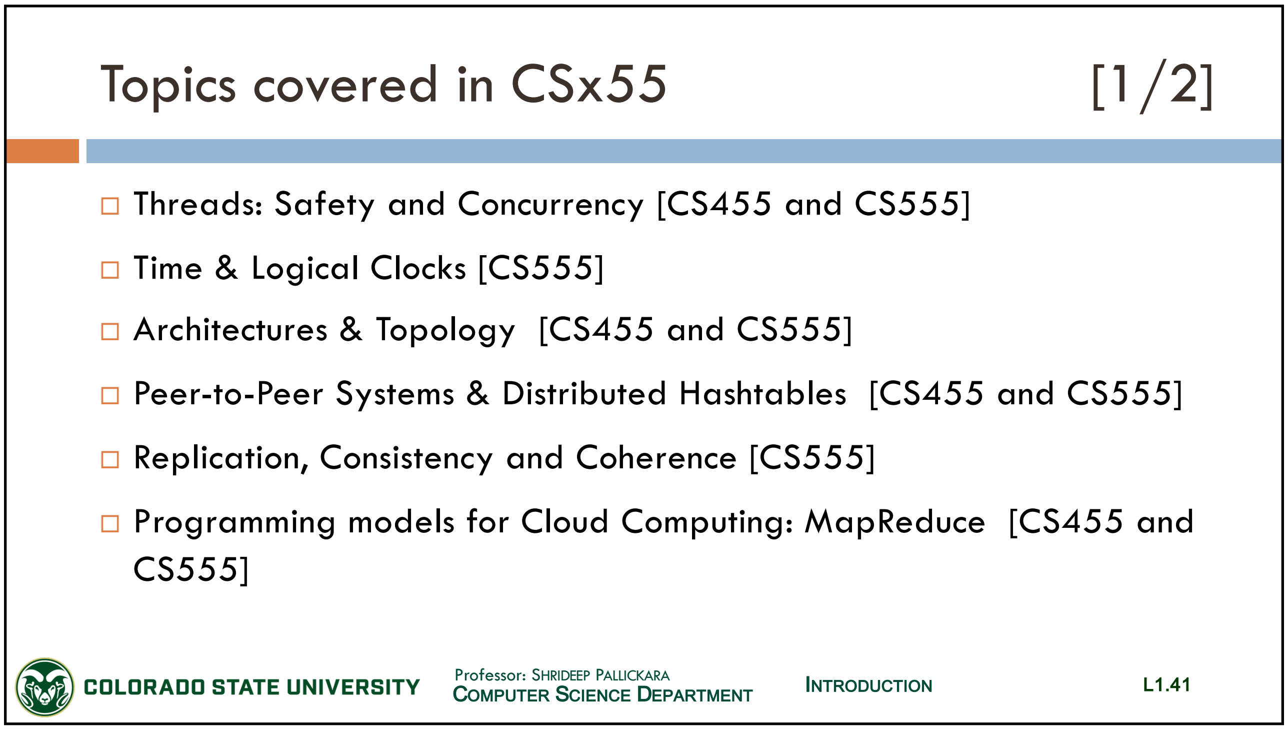
Interactions

- You can have discussions with me, the GTAs, and your peers
- There are **two constraints** to these discussions
  1. No code can be exchanged under <u>*any*</u> circumstances
  2. No one takes over someone else's keyboard
- Bumps are to be expected along the way
  - But you should get over this yourself
  - It will help you with the next problem you encounter

CSx55: Community and Brainstorming

- I have reserved CSB-130 from 5:00-7:00 pm on Fridays (except for March 1st) for the rest of the semester
- This a **student-led** activity
  - Someone should project MS-Teams (allowing students from **all x55** sections to join remotely) on the screen
  - This is a place to brainstorm with your peers about how you are approaching the problem
    - No assignment code should be shown or exchanged

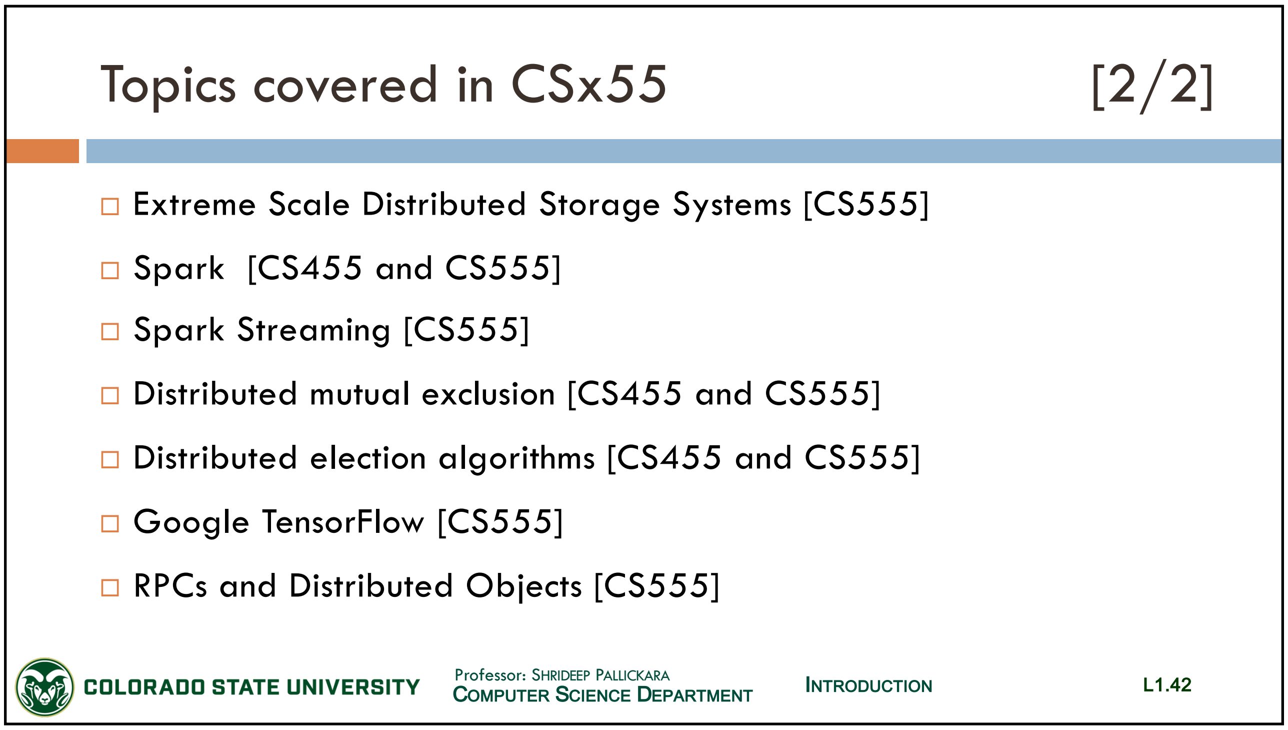
TOPICS COVERED IN CSX55



Topics covered in CSx55 [1/2]

- Threads: Safety and Concurrency [CS455 and CS555]
- Time & Logical Clocks [CS555]
- Architectures & Topology [CS455 and CS555]
- Peer-to-Peer Systems & Distributed Hashtables [CS455 and CS555]
- Replication, Consistency and Coherence [CS555]
- Programming models for Cloud Computing: MapReduce [CS455 and CS555]

41



Topics covered in CSx55 [2/2]

- Extreme Scale Distributed Storage Systems [CS555]
- Spark [CS455 and CS555]
- Spark Streaming [CS555]
- Distributed mutual exclusion [CS455 and CS555]
- Distributed election algorithms [CS455 and CS555]
- Google TensorFlow [CS555]
- RPCs and Distributed Objects [CS555]

42

SOCKET BASICS

Example:
Setting up connections to a server

- Programs open a **socket** to a server that's *listening* for connections
- To create a `Socket` you need to know the Internet host you want to connect to
- Servers don't know *who* will contact them
  - If it did, difficult to synchronize *when* this would happen

An analogy

- Server is like a person sitting by the phone
  - Doesn't know *who* will call and *when*
  - When the phone rings?
    - Talk to whoever is on the other line

Java provides a `ServerSocket` to enable writing servers

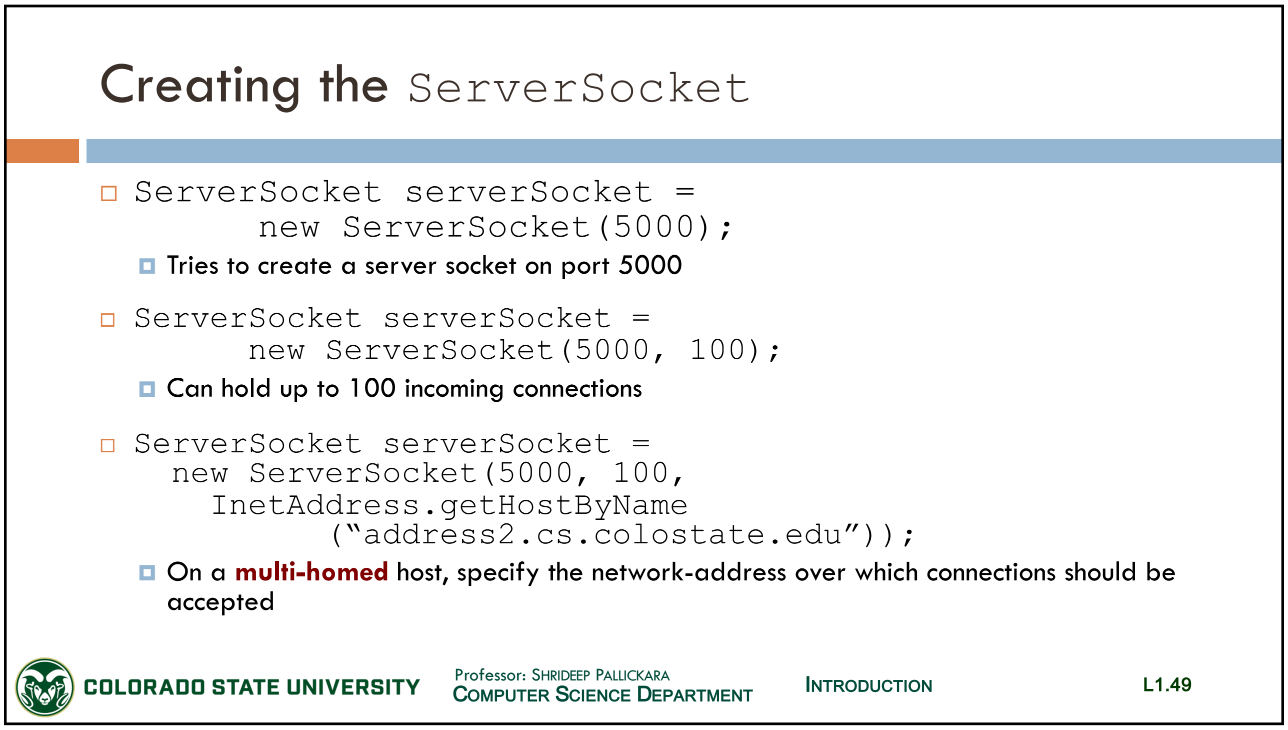
- `ServerSocket` runs on the server
  - **Listens** for *incoming* network connections on a particular **port** on the host that it runs on
- When a client socket on a remote host attempts to connect to that server port
  1. Server **wakes** up
  2. **Negotiates** a connection between the client and server
  3. **Opens** a regular `Socket` between the two hosts

Some more about the two types of sockets

- `ServerSockets` **wait** for connections
- Client `Sockets` **initiate** connections
- Once the `ServerSocket` has set up the connection?
  - **Data always travels over the regular `Socket`**

Using the `ServerSocket`

- Created on a particular **port** using the `ServerSocket(port)` constructor
- Listen for communications on that port using `accept()`
  - **Blocks** *until* a client attempts to make connection
  - Returns a `Socket` object that **connects** the client to the server
- Use the `Socket`'s `getInputStream()` and `getOutputStream()` to communicate

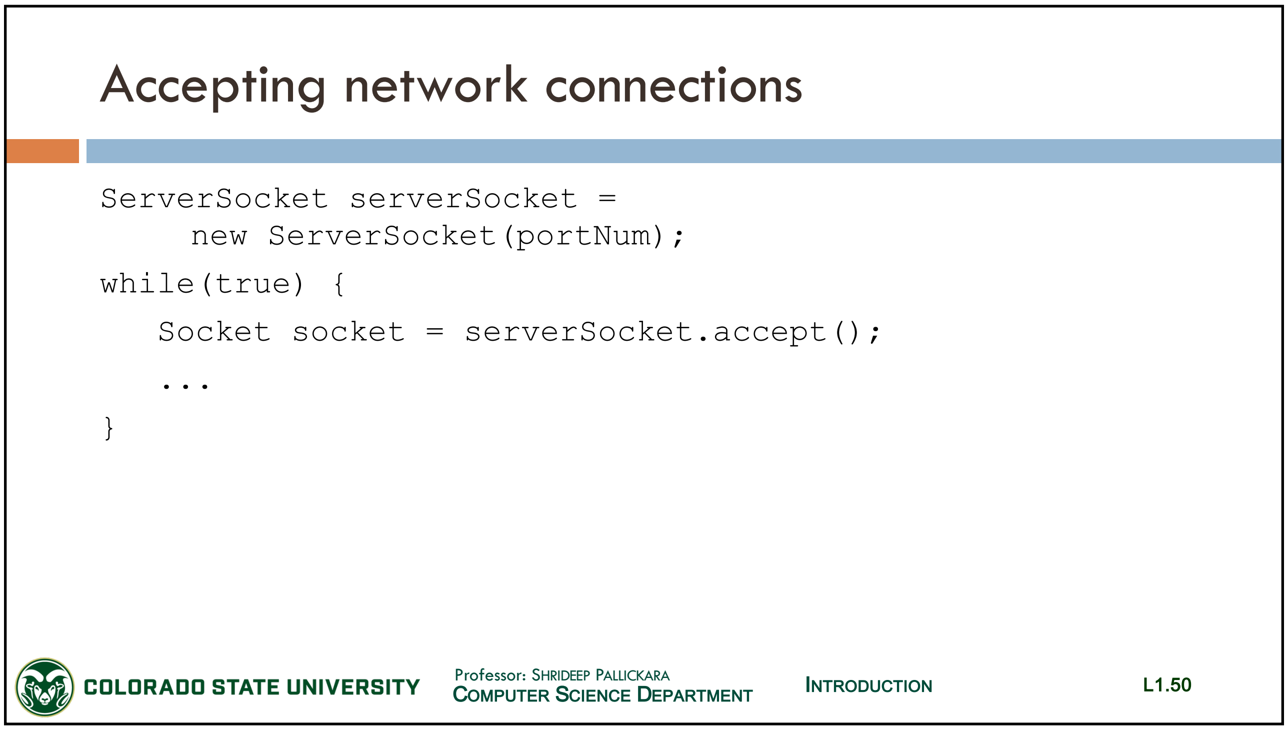


Creating the `ServerSocket`

-
  ```
  ServerSocket serverSocket =
          new ServerSocket(5000);
  ```
  - Tries to create a server socket on port 5000

-
  ```
  ServerSocket serverSocket =
         new ServerSocket(5000, 100);
  ```
  - Can hold up to 100 incoming connections

-
  ```
  ServerSocket serverSocket =
     new ServerSocket(5000, 100,
       InetAddress.getHostByName
              ("address2.cs.colostate.edu"));
  ```
  - On a **multi-homed** host, specify the network-address over which connections should be accepted



Accepting network connections

```
ServerSocket serverSocket =
     new ServerSocket(portNum);
while(true) {
   Socket socket = serverSocket.accept();
   ...
}
```

Closing the client and server sockets

- Closing a `ServerSocket` **frees** a port on the host that it runs on
- Closing a `Socket` **breaks** the connection between the local and remote hosts

We exchange byte streams over the socket

- The `java.io` package contains the `DataInputStream` and `DataOutputStream` that lets you do this elegantly

```
DataInputStream din =
  new DataInputStream(socket.getInputStream());
```

```
DataOutputStream dout =
new DataOutputStream(socket.getOutputStream());
```

HW1: Discussion

COMPUTER SCIENCE DEPARTMENT — COLORADO STATE UNIVERSITY

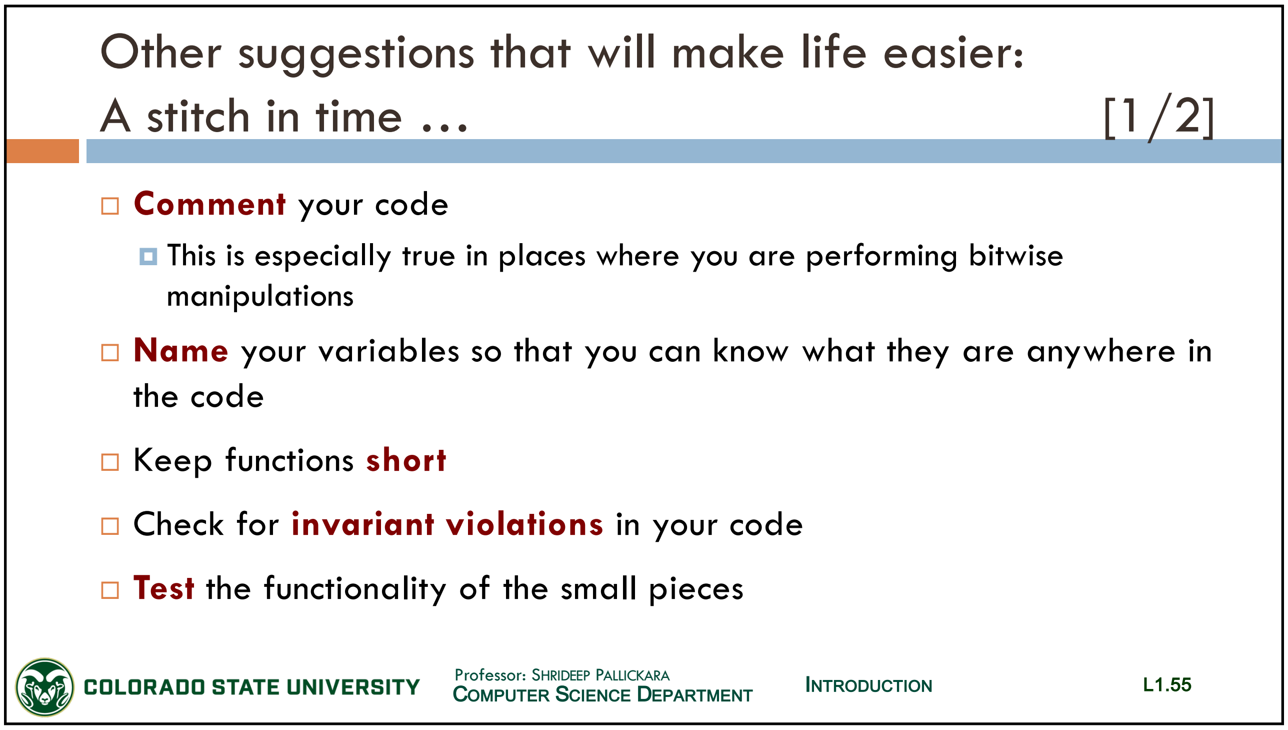
53

Why abstractions are important

- Abstraction is the key to managing **complexity**
- Good abstractions turn a difficult task into two manageable ones
  1. Defining and implementing abstractions
  2. Using abstractions to solve problem

COLORADO STATE UNIVERSITY — Professor: SHRIDEEP PALLICKARA, COMPUTER SCIENCE DEPARTMENT — INTRODUCTION — L1.54

54

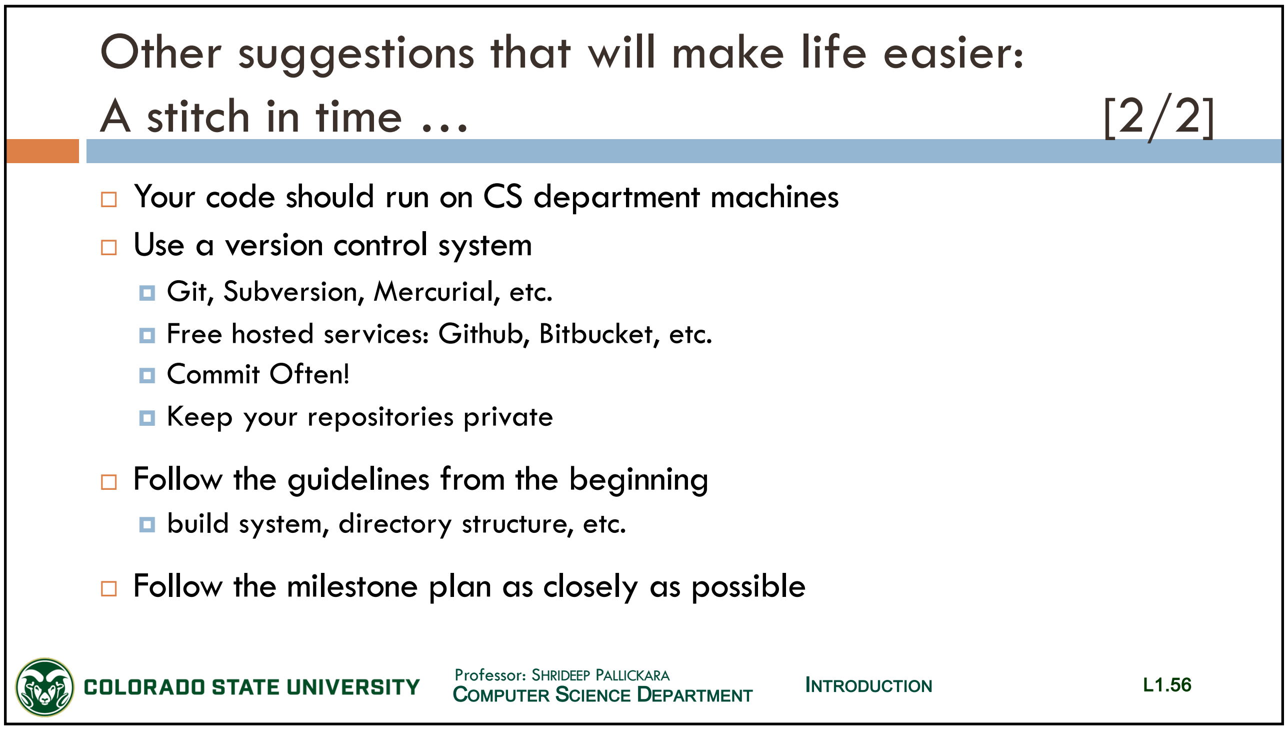


Other suggestions that will make life easier: A stitch in time … [1/2]

- **Comment** your code
  - This is especially true in places where you are performing bitwise manipulations
- **Name** your variables so that you can know what they are anywhere in the code
- Keep functions **short**
- Check for **invariant violations** in your code
- **Test** the functionality of the small pieces

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.55

55

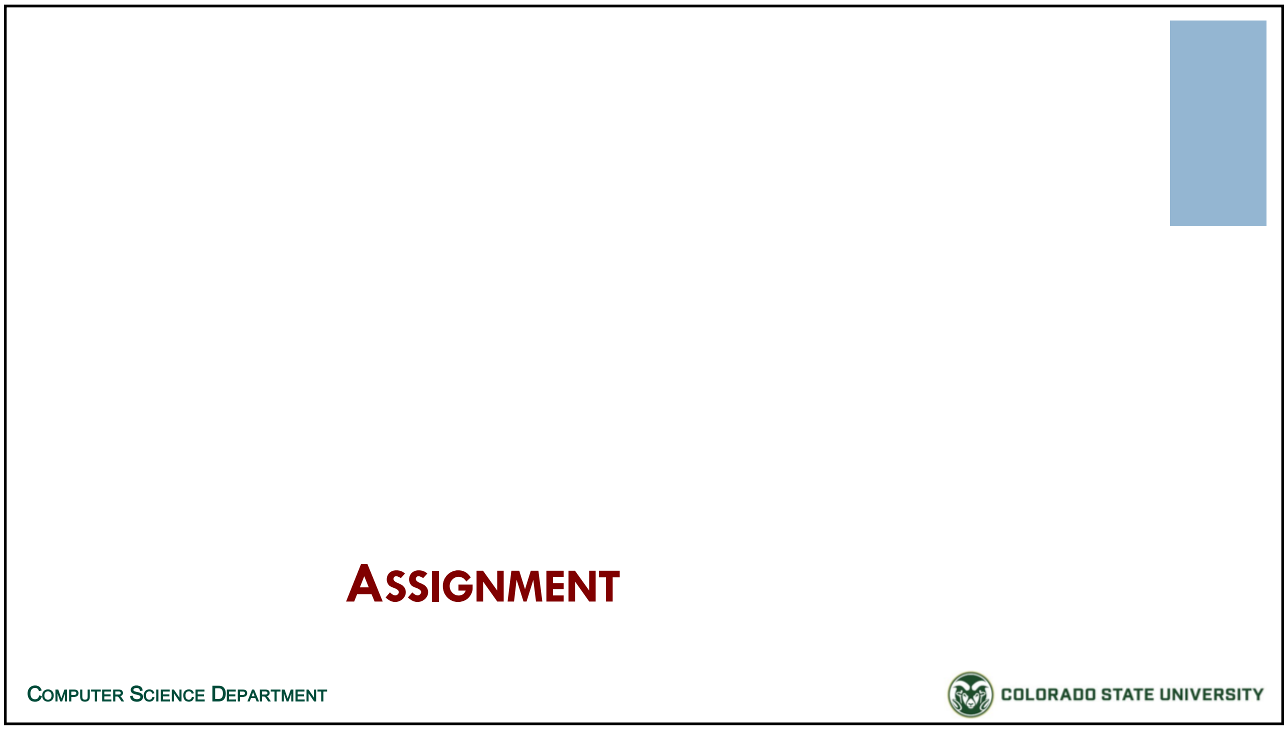


Other suggestions that will make life easier: A stitch in time … [2/2]

- Your code should run on CS department machines
- Use a version control system
  - Git, Subversion, Mercurial, etc.
  - Free hosted services: Github, Bitbucket, etc.
  - Commit Often!
  - Keep your repositories private
- Follow the guidelines from the beginning
  - build system, directory structure, etc.
- Follow the milestone plan as closely as possible

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.56

56

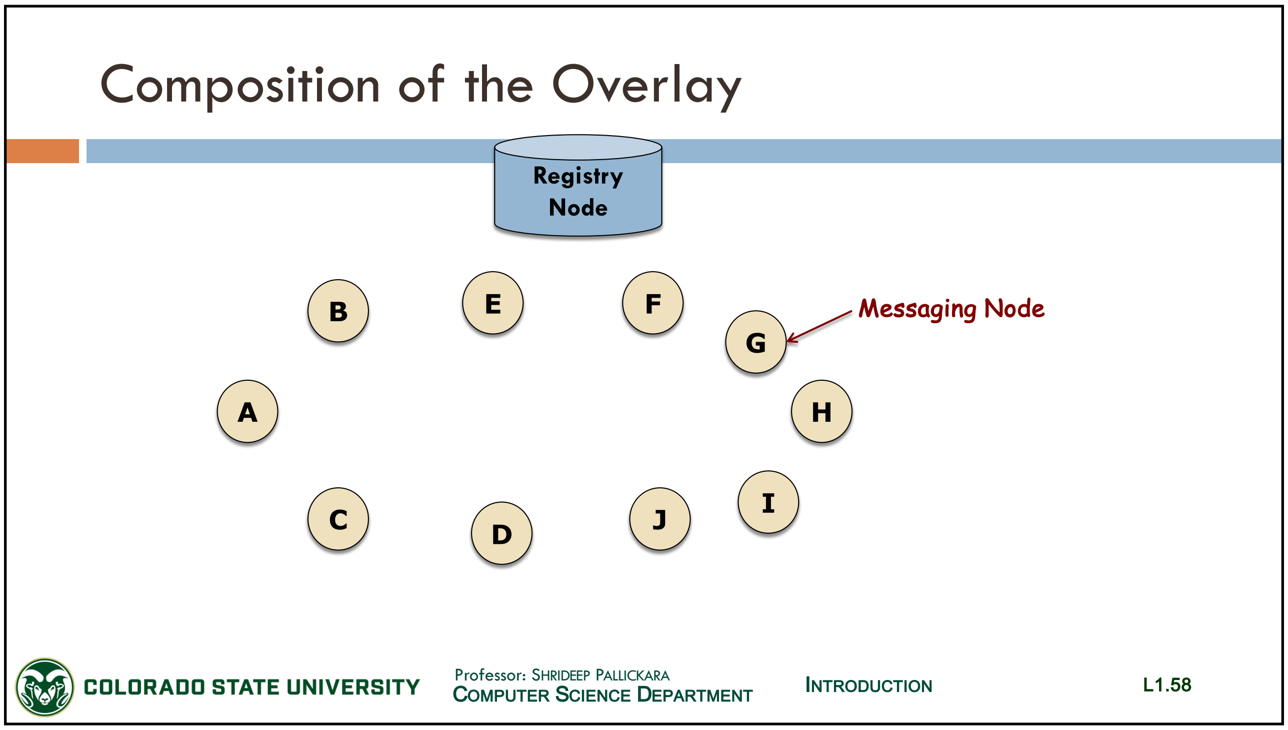


ASSIGNMENT

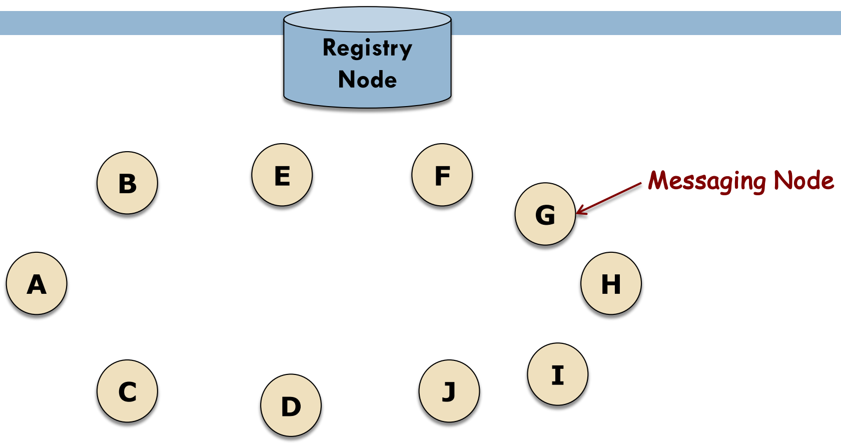
COMPUTER SCIENCE DEPARTMENT

COLORADO STATE UNIVERSITY

57

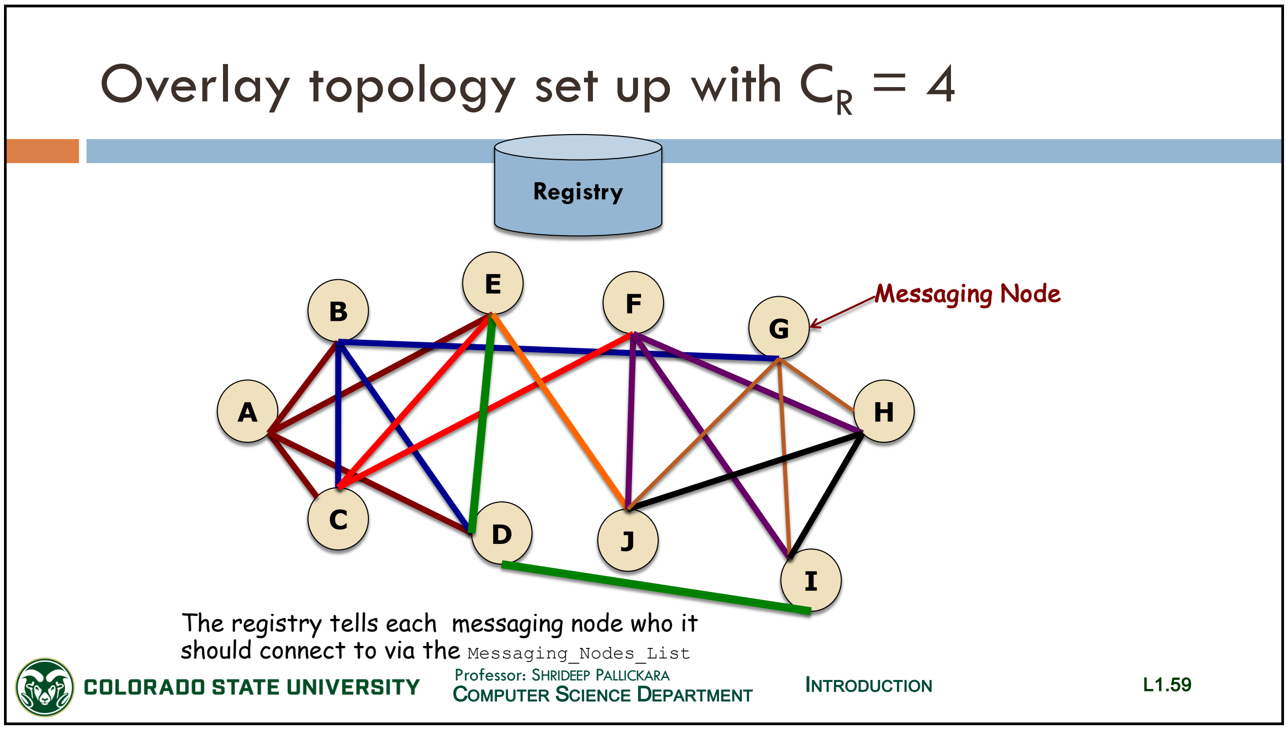


Composition of the Overlay

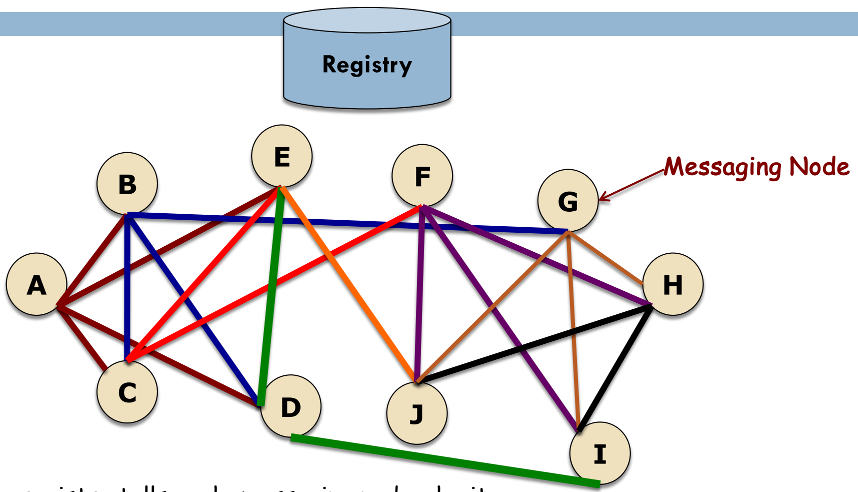


COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.58

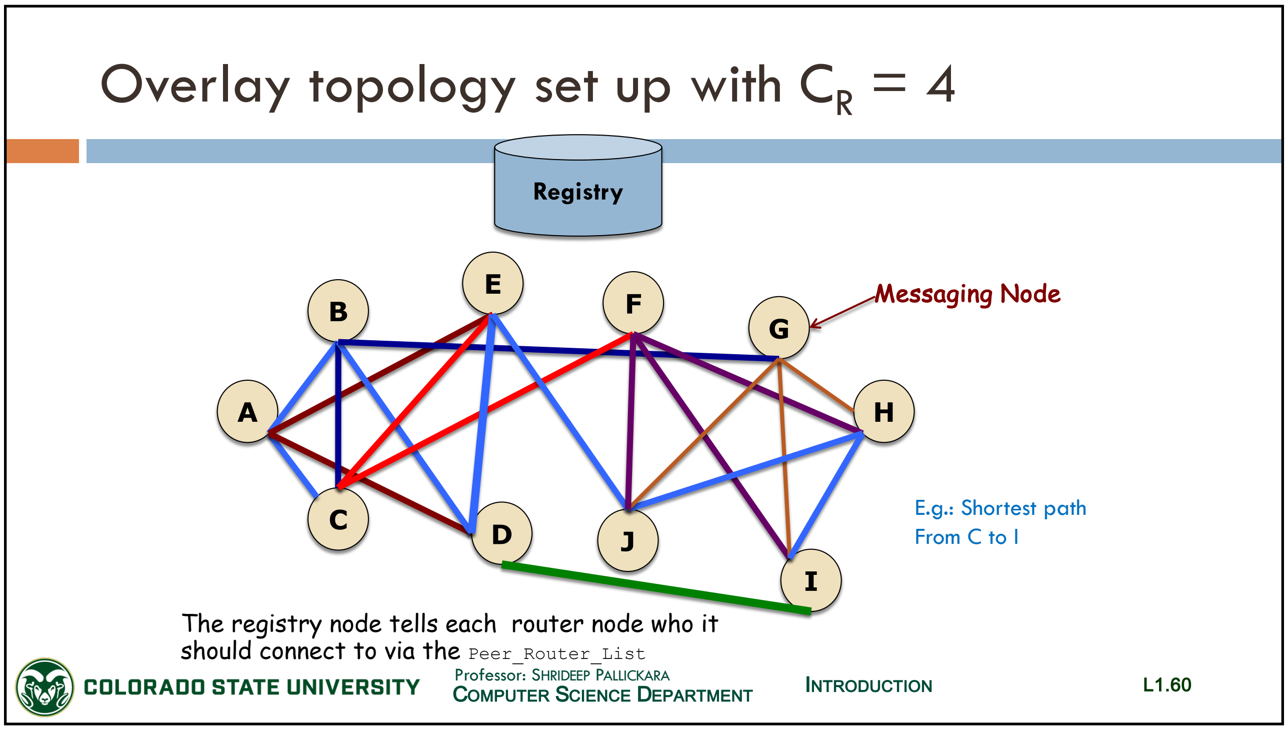
58



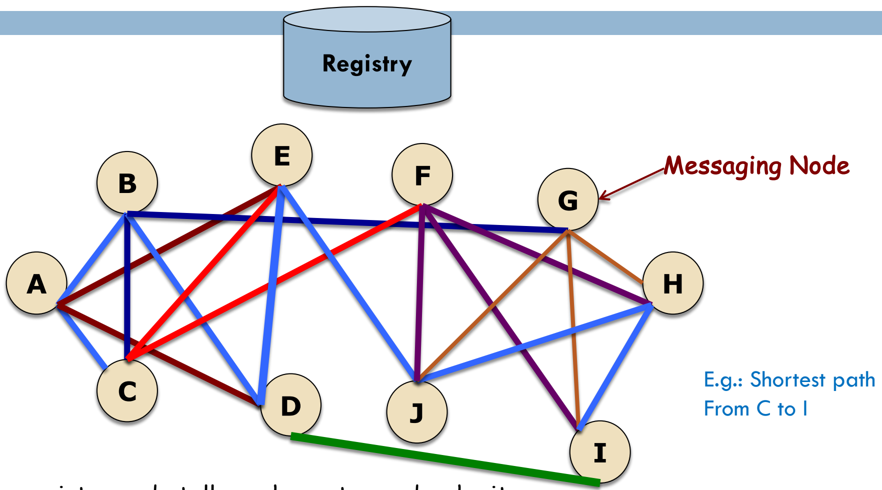
Overlay topology set up with $C_R = 4$



The registry tells each messaging node who it should connect to via the `Messaging_Nodes_List`

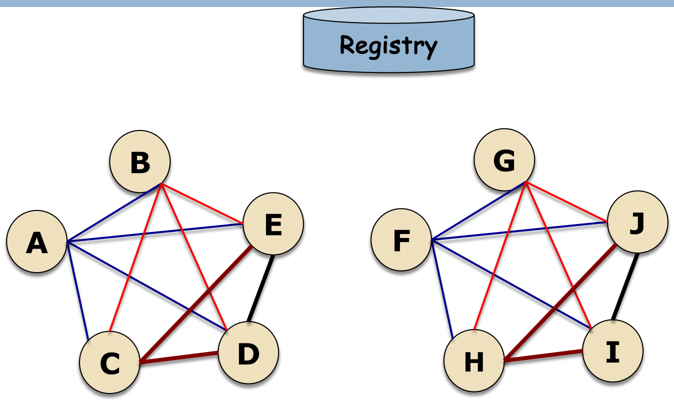


Overlay topology set up with $C_R = 4$



The registry node tells each router node who it should connect to via the `Peer_Router_List`

Looking at another overlay topology that could be set up with $C_R = 4$



This topology has a **partition**. The assignment asks you to **prevent this.**
Nodes A,B, C, D and E have no way of communicating with F,G,H,I, and J.

Avoiding network partitions

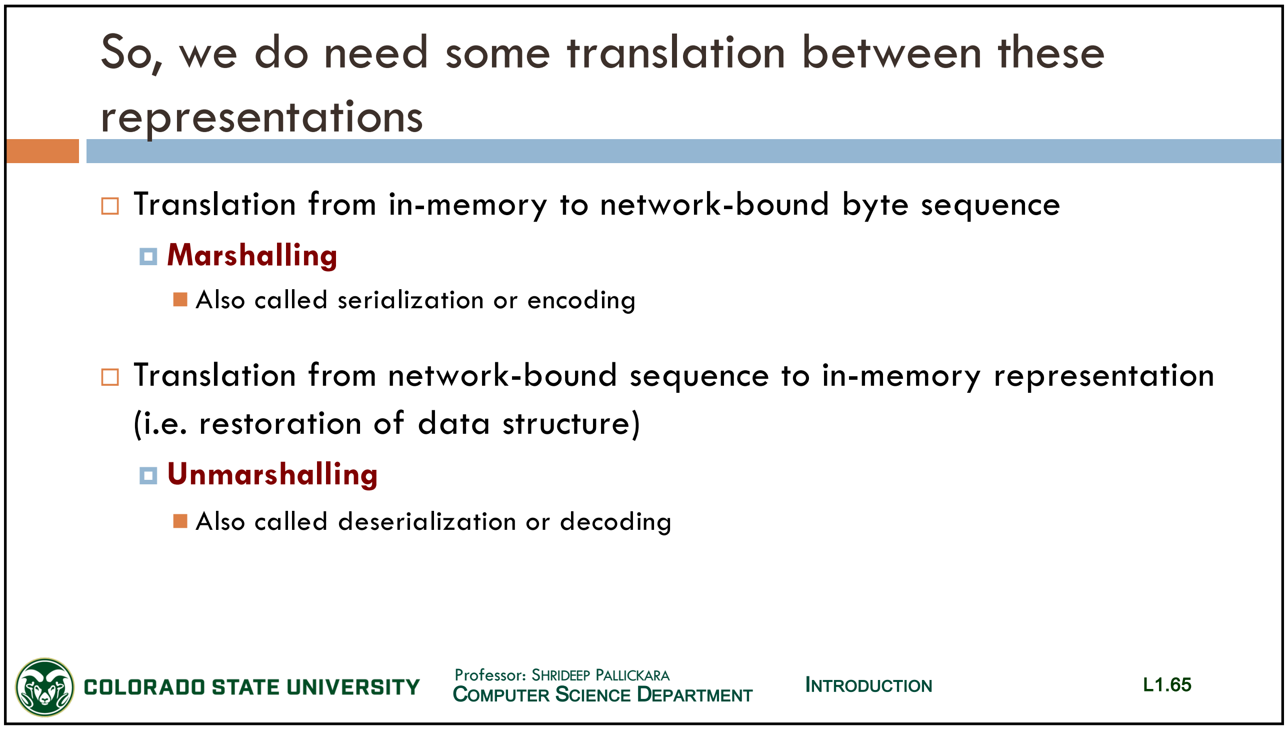
- Create a linear topology first, and then start making the required number of connections
  - A -> B -> C -> D ->E -> F -> G ->H -> I -> J
  - Starting off at the point ensures that partitions will not exist

Other topological aspects

- Necessary and sufficient conditions for a k-regular graph of order n to exist?
  - $n \geq k+1$
  - nk is even
- During testing we will only specify values where a solution exists

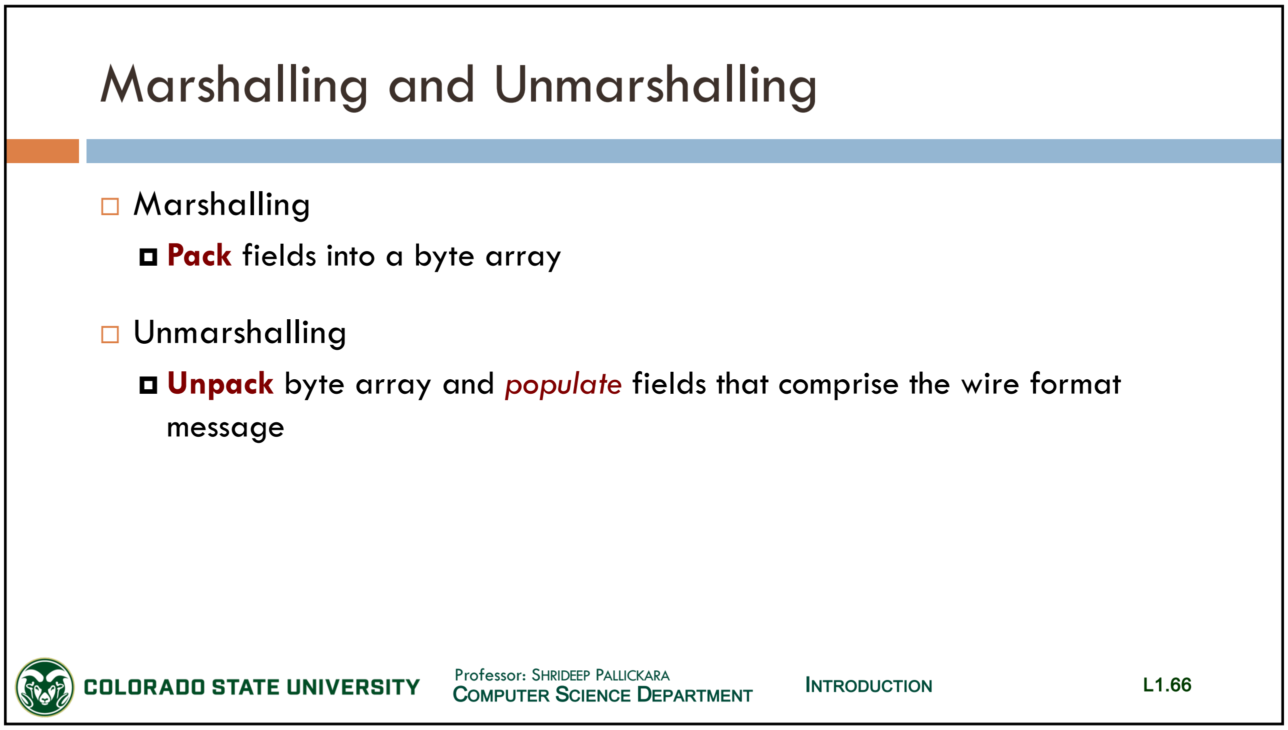
Your programs will be working with two different data representations

- In memory: This is where you have your **data structures** such as lists, arrays, hash tables, trees, etc.
- Data that you will sending over the network
  - You do this as a self-contained **sequences of bytes**
  - Do references or pointers make sense here?
    - No!
    - So, the sequence-of-bytes representation will look VERY different from data structures in memory



So, we do need some translation between these representations

- Translation from in-memory to network-bound byte sequence
  - **Marshalling**
    - Also called serialization or encoding
- Translation from network-bound sequence to in-memory representation (i.e. restoration of data structure)
  - **Unmarshalling**
    - Also called deserialization or decoding



Marshalling and Unmarshalling

- Marshalling
  - **Pack** fields into a byte array
- Unmarshalling
  - **Unpack** byte array and *populate* fields that comprise the wire format message

Example: Data Structure [1/3]

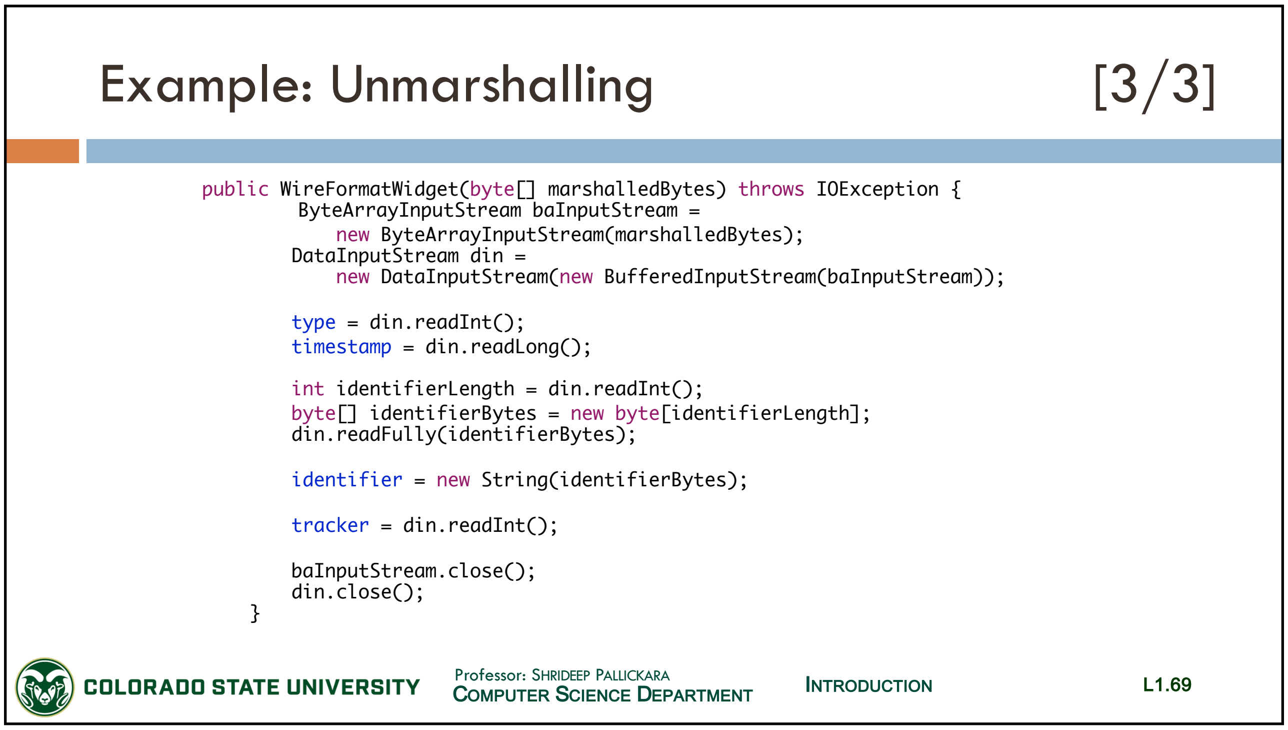
```
public class WireFormatWidget {

    private int type;
    private long timestamp;
    private String identifier;
    private int tracker;

    ...

}
```

67

Example: Marshalling [2/3]

```
public byte[] getBytes() throws IOException {
        byte[] marshalledBytes = null;
        ByteArrayOutputStream baOutputStream = new ByteArrayOutputStream();
        DataOutputStream dout =
            new DataOutputStream(new BufferedOutputStream(baOutputStream));

        dout.writeInt(type);
        dout.writeLong(timestamp);

        byte[] identifierBytes = identifier.getBytes();
        int elementLength = identifierBytes.length;
        dout.writeInt(elementLength);
        dout.write(identifierBytes);

        dout.writeInt(tracker);

        dout.flush();
        marshalledBytes = baOutputStream.toByteArray();

        baOutputStream.close();
        dout.close();
        return marshalledBytes;
    }
```

68

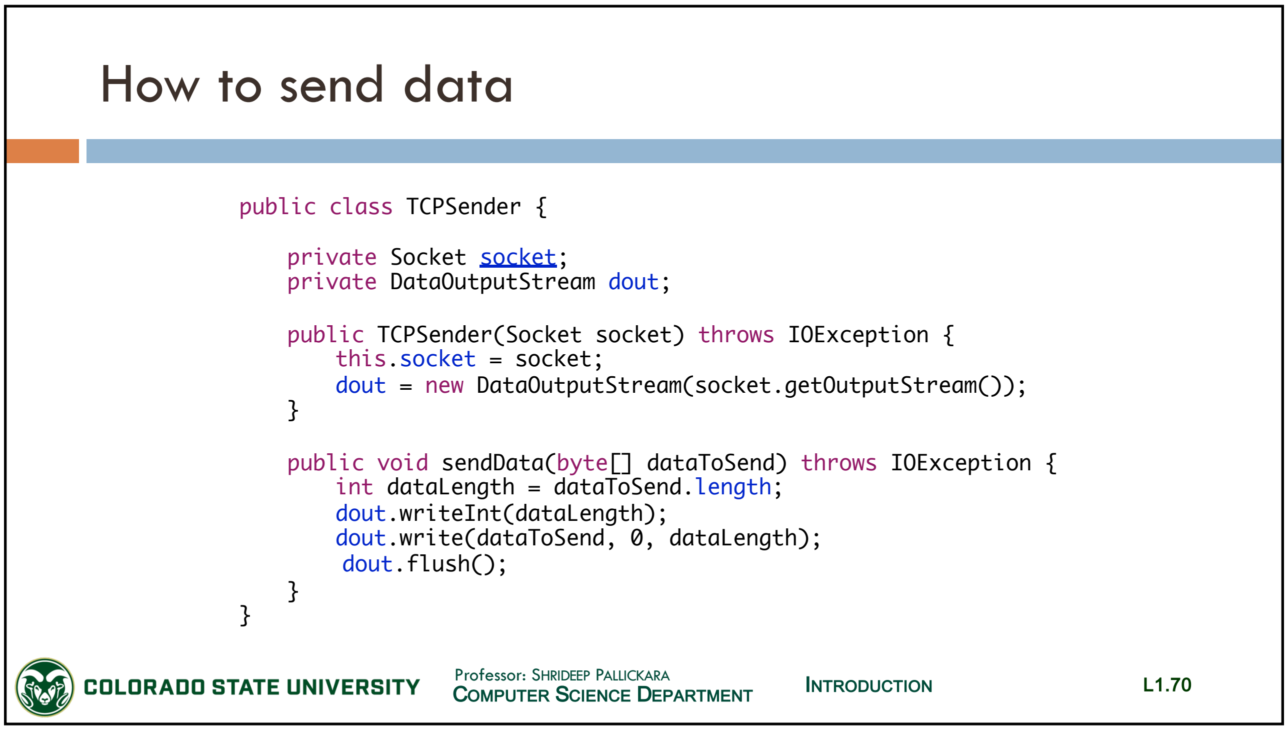


Example: Unmarshalling [3/3]

```
public WireFormatWidget(byte[] marshalledBytes) throws IOException {
        ByteArrayInputStream baInputStream =
            new ByteArrayInputStream(marshalledBytes);
        DataInputStream din =
            new DataInputStream(new BufferedInputStream(baInputStream));

        type = din.readInt();
        timestamp = din.readLong();

        int identifierLength = din.readInt();
        byte[] identifierBytes = new byte[identifierLength];
        din.readFully(identifierBytes);

        identifier = new String(identifierBytes);

        tracker = din.readInt();

        baInputStream.close();
        din.close();
    }
```

COLORADO STATE UNIVERSITY Professor: Shrideep Pallickara Computer Science Department Introduction L1.69

69

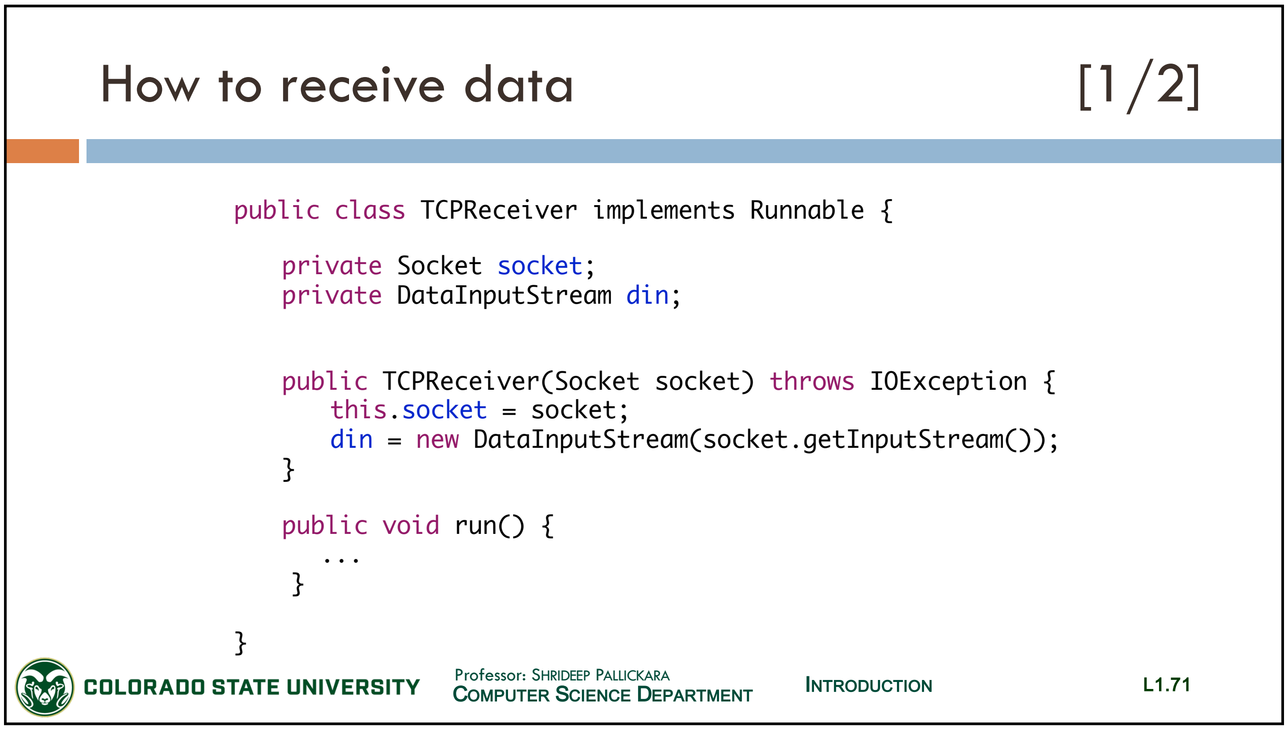


How to send data

```
public class TCPSender {

    private Socket socket;
    private DataOutputStream dout;

    public TCPSender(Socket socket) throws IOException {
        this.socket = socket;
        dout = new DataOutputStream(socket.getOutputStream());
    }

    public void sendData(byte[] dataToSend) throws IOException {
        int dataLength = dataToSend.length;
        dout.writeInt(dataLength);
        dout.write(dataToSend, 0, dataLength);
         dout.flush();
    }
}
```

COLORADO STATE UNIVERSITY Professor: Shrideep Pallickara Computer Science Department Introduction L1.70

70

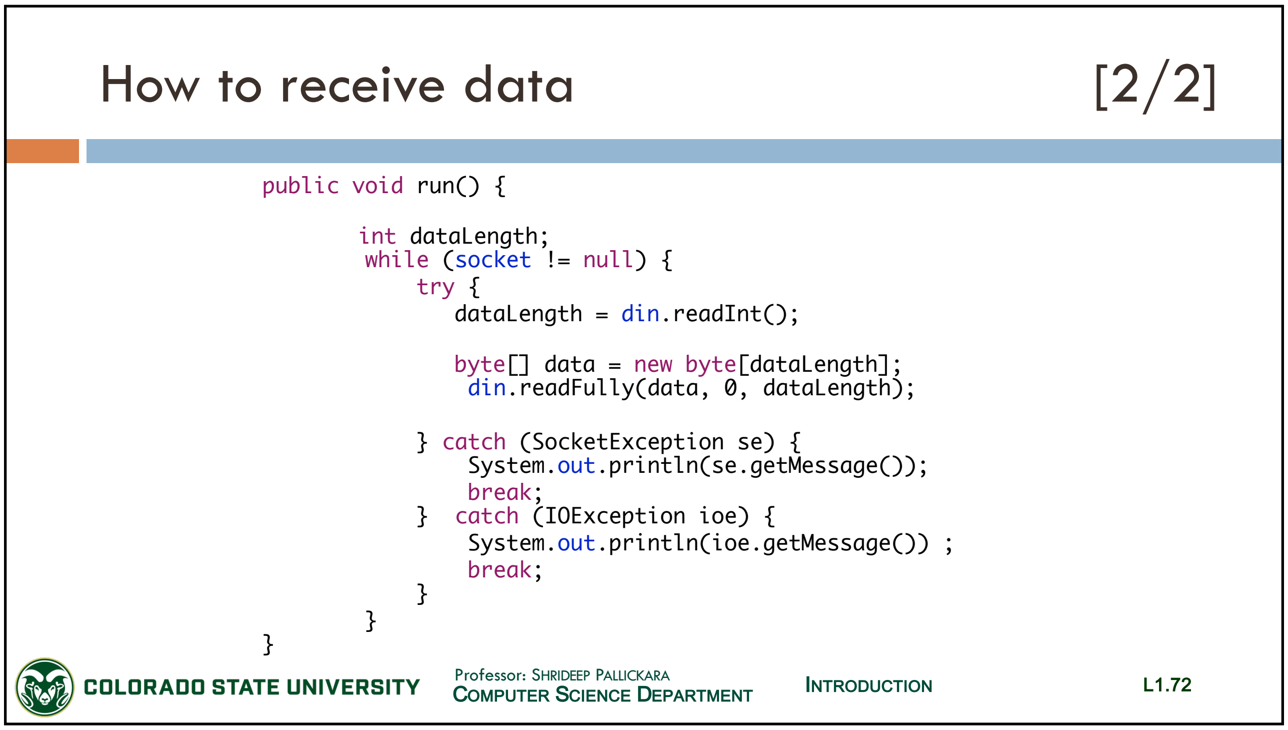


How to receive data [1/2]

```java
public class TCPReceiver implements Runnable {

    private Socket socket;
    private DataInputStream din;


    public TCPReceiver(Socket socket) throws IOException {
        this.socket = socket;
        din = new DataInputStream(socket.getInputStream());
    }

    public void run() {
        ...
     }

}
```

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.71

71



How to receive data [2/2]

```java
public void run() {

        int dataLength;
         while (socket != null) {
             try {
                dataLength = din.readInt();

                byte[] data = new byte[dataLength];
                 din.readFully(data, 0, dataLength);

             } catch (SocketException se) {
                 System.out.println(se.getMessage());
                 break;
             }  catch (IOException ioe) {
                 System.out.println(ioe.getMessage()) ;
                 break;
             }
         }
}
```

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.72

72

A simple breakdown of classes

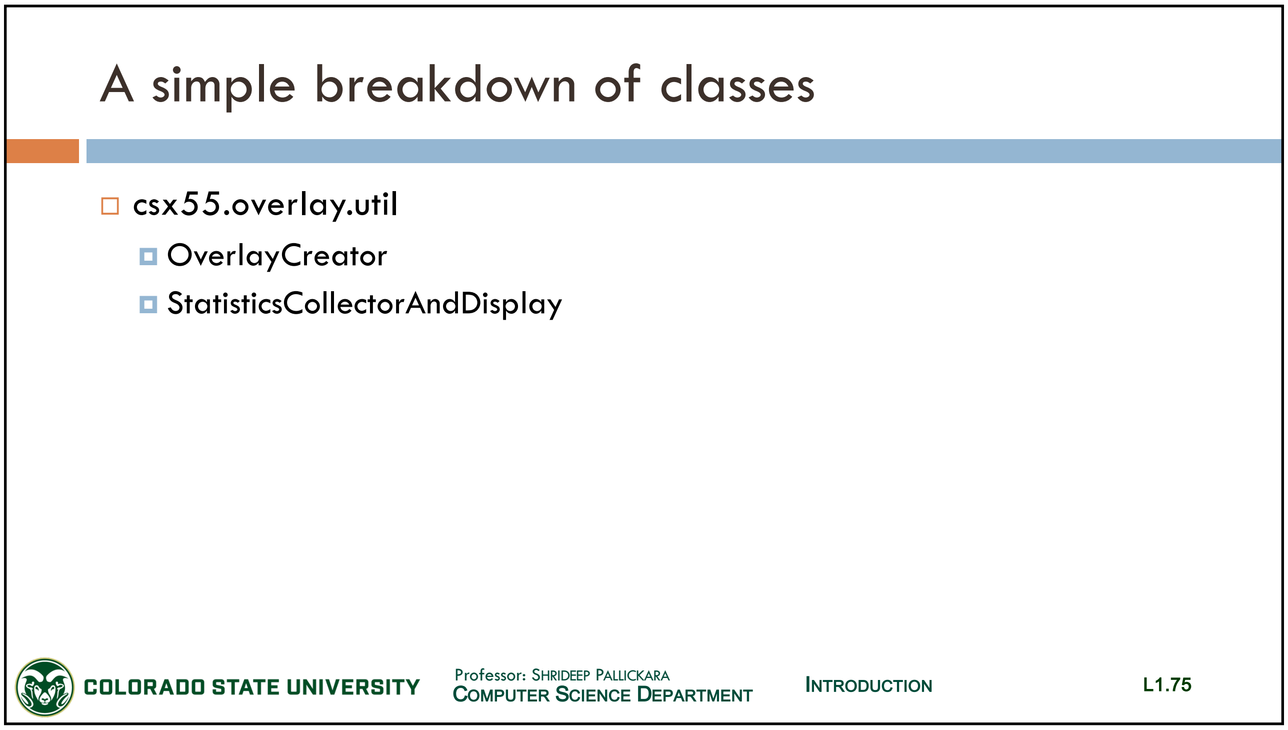
- csx55.overlay.wireformats
  - Protocol
  - Event [This is an interface with the getType() and getBytes() defined]
  - EventFactory [Singleton instance]
  - Register
  - Deregister
  - MessagingNodesList
  - LinkWeights
  - TaskInitiate
  - Message
  - TaskComplete
  - TaskSummaryRequest
  - TaskSummaryResponse

73

A simple breakdown of classes

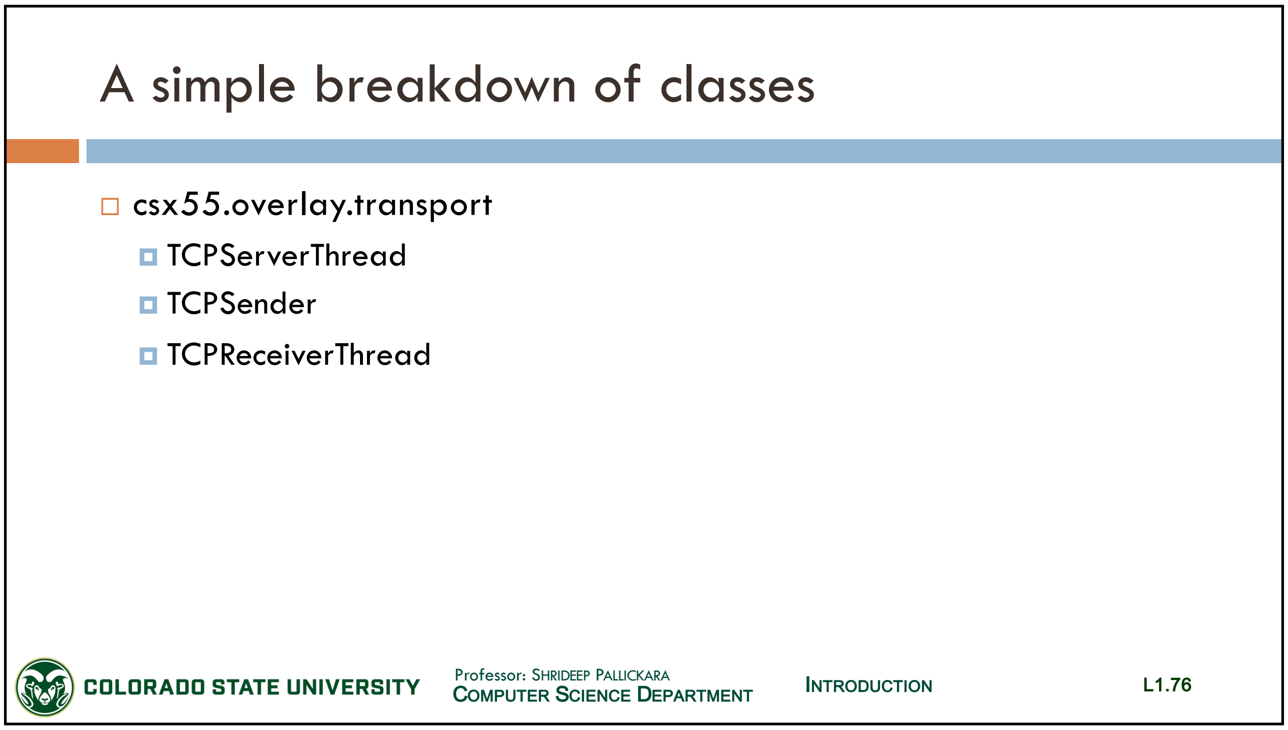
- csx55.overlay.dijkstra
  - ShortestPath
  - RoutingCache

74



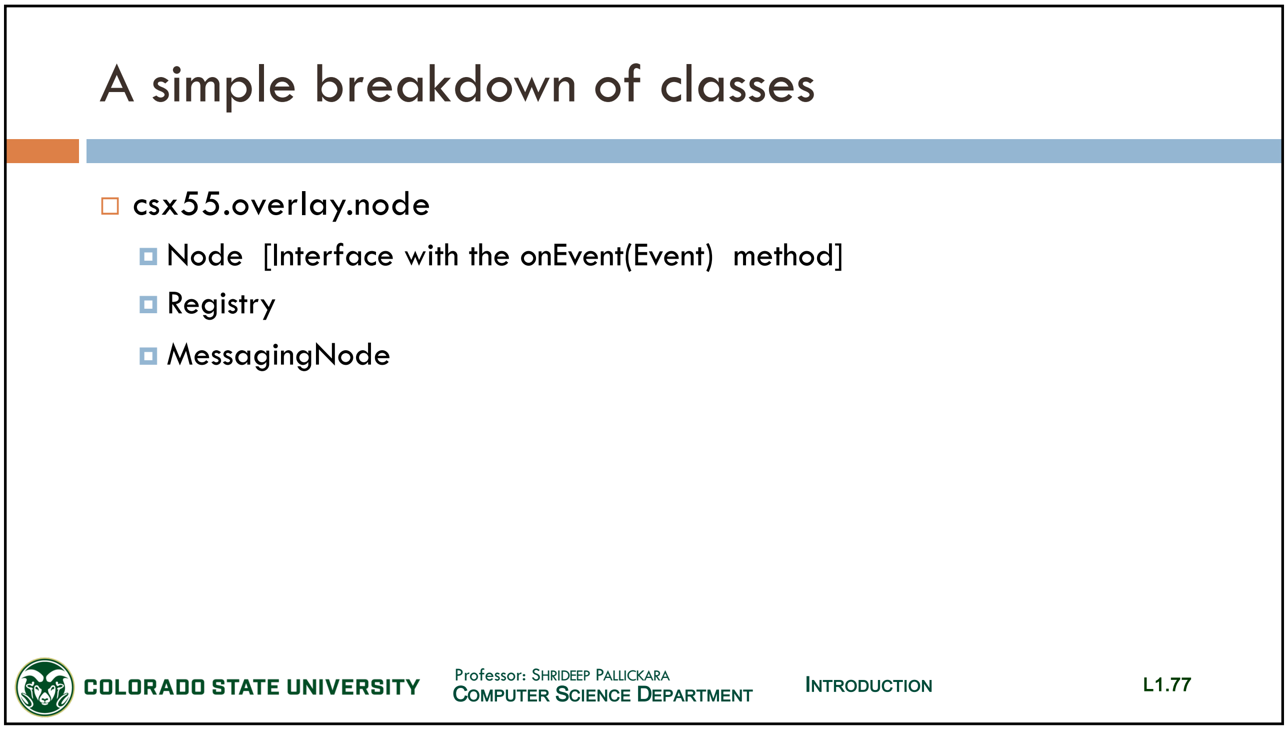
A simple breakdown of classes

- csx55.overlay.util
  - OverlayCreator
  - StatisticsCollectorAndDisplay



A simple breakdown of classes

- csx55.overlay.transport
  - TCPServerThread
  - TCPSender
  - TCPReceiverThread

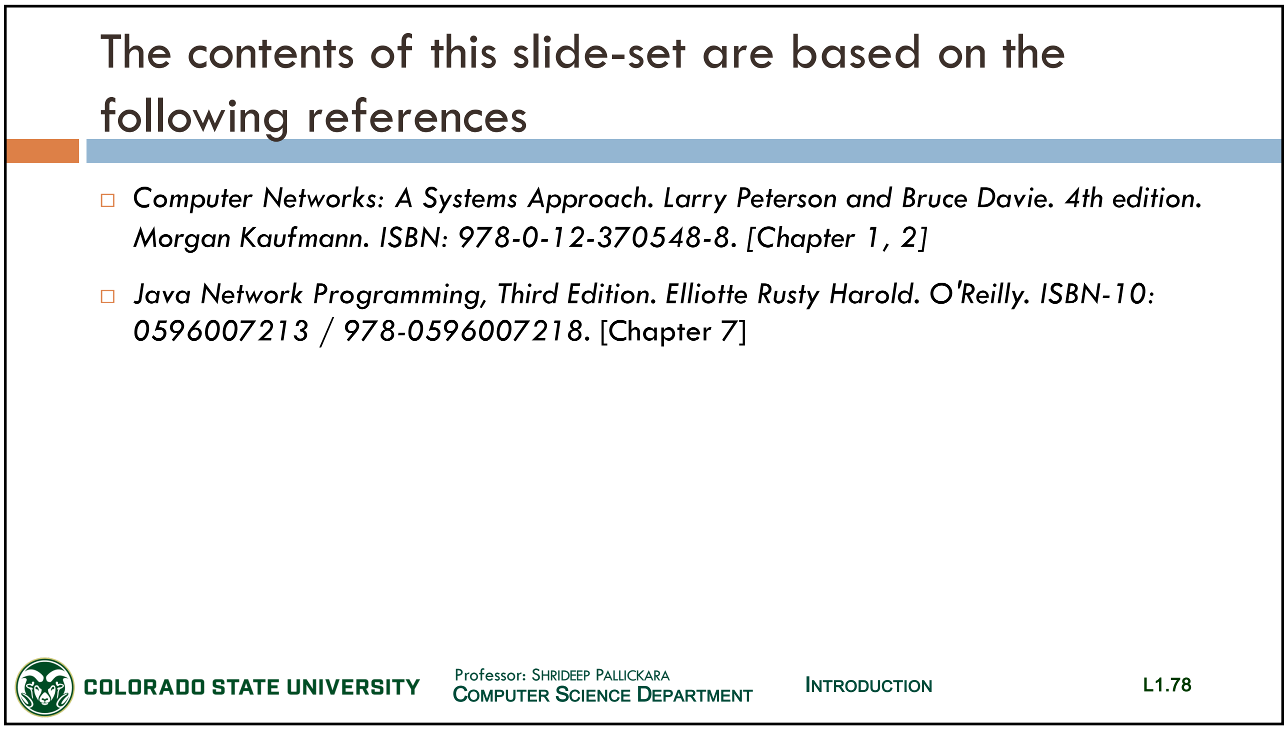


A simple breakdown of classes

- csx55.overlay.node
  - Node [Interface with the onEvent(Event) method]
  - Registry
  - MessagingNode

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.77

77



The contents of this slide-set are based on the following references

- *Computer Networks: A Systems Approach. Larry Peterson and Bruce Davie. 4th edition. Morgan Kaufmann. ISBN: 978-0-12-370548-8. [Chapter 1, 2]*
- *Java Network Programming, Third Edition. Elliotte Rusty Harold. O'Reilly. ISBN-10: 0596007213 / 978-0596007218.* [Chapter 7]

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.78

78