

CS 309: Autonomous Intelligent Robotics

FRI I

Lecture 11: Publish/Subscribe & ROS Topics

Instructor: Justin Hart

http://justinhart.net/teaching/2018_spring_cs309/

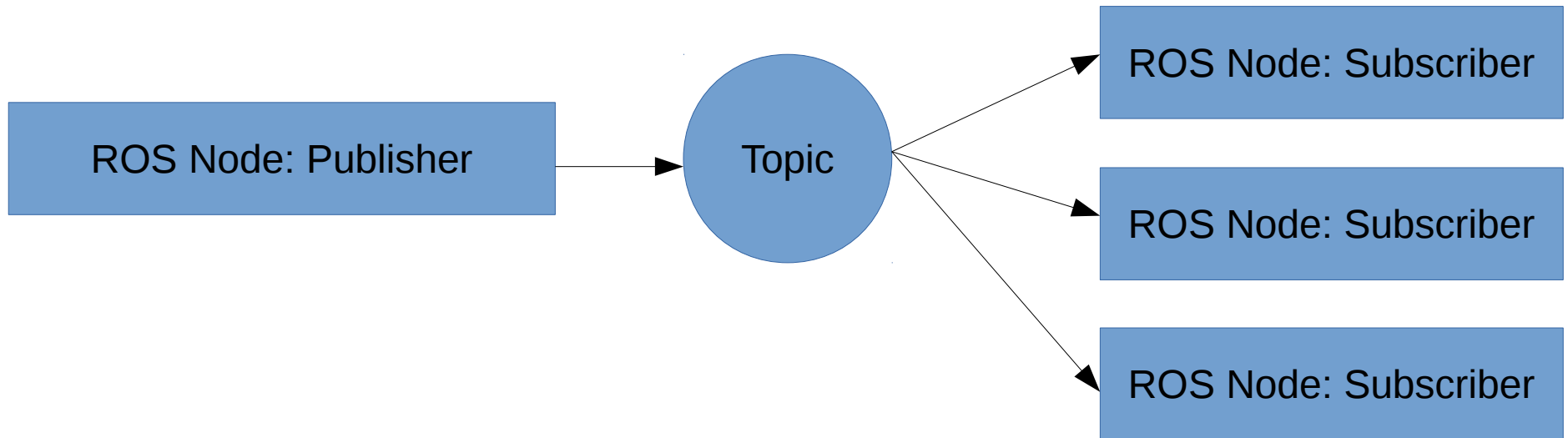
A couple of quick notes

- Homework 2: Is out. Due March 1.
 - PDDL
 - Questions on the homework build on each other.
 - You could, in theory, use only one domain file for the entire assignment
 - But multiple fact files
 - Any questions?

Publish/Subscribe

- ROS Topics use a Publish/Subscribe architecture
 - Publishers
 - Broadcast data on a “topic”
 - Subscribers
 - Multiple subscribers can read data from a topic simultaneously
 - Example
 - rviz example from Kinect v2 data

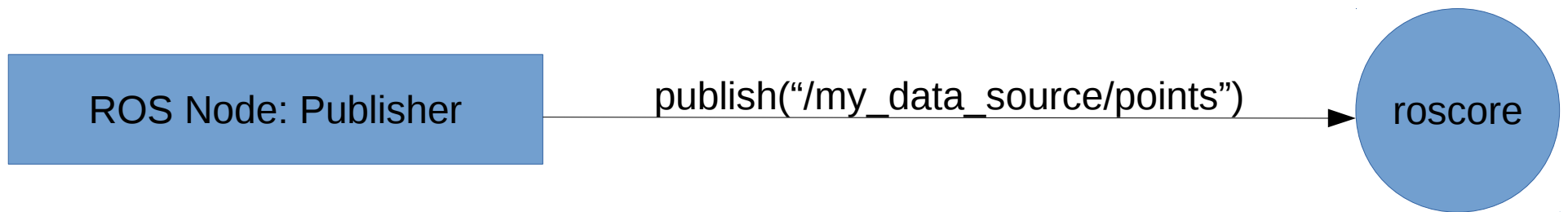
A simplified, conceptual diagram



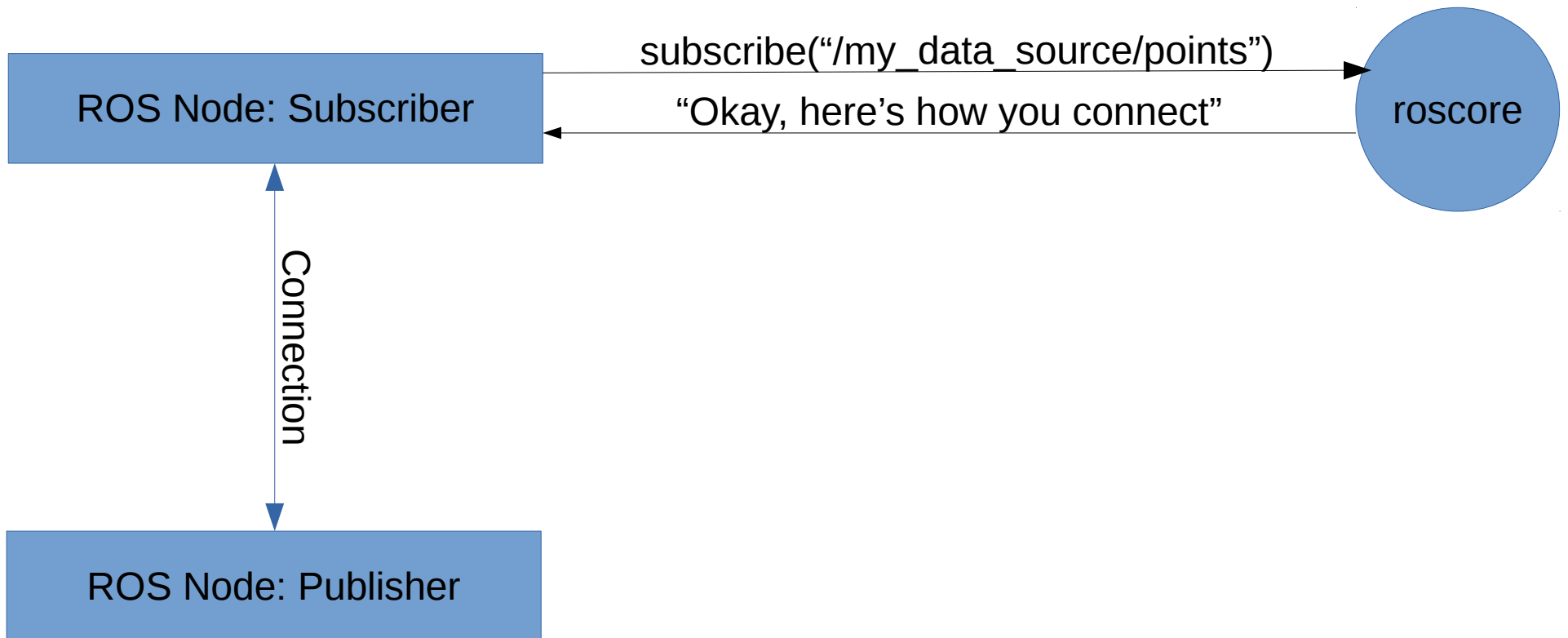
roscore

- The program roscore coordinates ROS communications
 - When a node publishes a topic, it registers this topic under a path in roscore with a name
 - Other nodes can subscribe to the topic using this name
 - These topic names can be configured in ROS launch and other configuration files, which we will later explore more in-depth

roscore, in practice



roscore, in practice



ROS Topics

- Connect over a network socket
 - Stored in `$ROS_MASTER_URI`
 - Default is `http://localhost:11311`
 - You probably won't use it in this class, but for `RoboCup@Home`, we use a network of 3 or 4 computers to drive the robot
 - To manage this, we use `$ROS_MASTER_URI` to coordinate communications

ROS Topics

- Last time, we quickly ran through a ROS Topics tutorial. Let's revisit this:
 - <http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics>

Understanding ROS Topics

- [http://wiki.ros.org/ROS/Tutorials/Understanding Topics](http://wiki.ros.org/ROS/Tutorials/Understanding%20Topics)
- Stop at part 3.1 for more explanation

ROS Message Types

- Built into ROS is a special compiler for messaging
- You can specify ROS messages that are published on topics
- ROS Topics use a **very** simplified language
- The compiler outputs code in every language supported by ROS (usually C++ or Python)

ROS Message Types

- To use a ROS topic
 - You don't **really** need to understand networking
- As we use it, each message type will be stored in a C++ class
 - The compilers in ROS write this class.
 - You do not.
 - You write a simple text file saying what goes into the message

ROS Message Types

- For most types of data used in robotics, ROS has already specified one of these message files
 - This is how all of these programs interoperate
 - If you use standard message types, your ROS nodes will work with existing ROS nodes
 - For instance, if you publish point cloud data using `sensor_msgs/PointCloud`, your point cloud data can be rendered in rviz and used by existing ROS nodes which utilize point cloud data!

Let's look at a simple ROS message

- `/opt/ros/kinetic/std_msgs/String.msg`

Catkin workspace & build tools

- We briefly looked over this two lectures ago
 - ROS comes with build tools which build on make
 - Top-level tools which work like make
 - `catkin_make`
 - `catkin build`
 - This one is a bit better, but most tutorials will use `catkin_make`
 - `catkin_init_workspace`
 - Top-level script that initializes your catkin workspace
 - `catkin_create_package`
 - Shortcut script to make a ROS package for your nodes to go into.

Writing our first publisher

- String.msg is used in the ROS Publisher/Subscriber tutorial:
 - <http://wiki.ros.org/ROS/Tutorials/CreatingPackage>
 - <http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29>

Writing our first subscriber

- Event-driven programming
 - Main loops are used in event-driven programming
 - The idea is to wait for something to happen, and then act on the thing that happened
 - The thing that happened is called an event

Writing our first subscriber

- The main loop allows the program to check if something has happened repeatedly
 - `ros::spinOnce()` does the checking
 - Subscribers check to see if a message has been published on a topic
- At the start of the program, a function called a “callback” is “registered” with the event-driven framework
 - In our case, that framework is ROS.
 - When the event happens, the callback is called.
- How this works will become more clear when we try it out

Writing our first subscriber

- Back to:

- <http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29>