**Introduction**

ROS (Robot Operating System) is a system of libraries and tools that helps developers make robot applications.

It provides:
- package management
- hardware abstraction
- libraries
- visualizers
- message passing
- other useful robot development tools

An active ROS system consists of a set of nodes.

A **Master** node provides:
- naming and registration services (for nodes to find each other)
- a Parameter Server (that nodes use as a shared dictionary)

Nodes communicate by:
- publishing / subscribing to topics
- offering / requesting services

**Problem: Single Master**

Many robotic systems require wireless communication between multiple robots. The only quick way to do this in ROS is to run a single Master on one of the robots, which all of the robots use to initialize nodes and access parameters. This presents two main problems:

- Robots disconnected from Master cannot initialize new nodes or wait for services => unable to plan path back to Master
- Robots want to communicate simple, high-level info, such as relative position. But in ROS:
  - *all* position info published as transforms on topic /tf
  - subscribers to /tf get flooded with unnecessary info, such as other robots’ joint angles
  - sharing high-bandwidth topics like /tf saturates network

**Solution: Multimaster**

Giving every robot its own Master solves both problems:

- Robots retain complete independence (can initialize new nodes and wait for services even when disconnected):
  - Robots can change tasks, retrieve data and take corrective action (plan a path back to wireless range).
- High-bandwidth, built-in topics like /tf remain local. Messages can be published on low-bandwidth remote (non-local) topics:
  - Robots can share select data (such as its position relative to another robot) without saturating the network.

**Implementation: Foreign Relay**

Using a ROS package called **WiFi_Com**m, we open up foreign relays between Masters.

- High-bandwidth, built-in topics like /tf can remain local
- Low-bandwidth messages can be published on /foreign_topic, which gets relayed to the other Master via foreign_relay
- If Masters become disconnected, foreign topics die, but robots remain functional
- Allows robots to communicate relative positions without saturating network

**Results & Future Work**

Successfully ran two-master PR2 simulations using Gazebo and RViz:

- Ran two complete PR2 simulations on two different computers, each with its own Master
- Restarting foreign relay after wireless disconnect / reconnect resumed simulation with updated robot positions
- Computed relative robot positions in real time with negligible latency (yellow X marks position of PR2 for single-master system at same time):

**Future work:**

- Test with more than two robots
- Test with real robotic systems
- Rigorous testing of communication limits and points of unreliability
- Explore alternative approaches to multimaster