# Introduction to rosjava

Damon Kohler

# What to expect and what not to expect

Expect

- Developer oriented
- High level concepts
- Abbreviated code samples

Not

- Tutorial
- ROS overview
- Java overview
- Android overview

# rosjava: One Year

# Why use rosjava?

# Asynchronous

What does that mean?

- Methods take listener objects (aka callbacks)
- Netty (http://netty.io)

Why?

- Better performance
- Better coupling/cohesion

Example

```
Subscriber.addMessageListener(MessageListener)
```

# Node vs. NodeMain

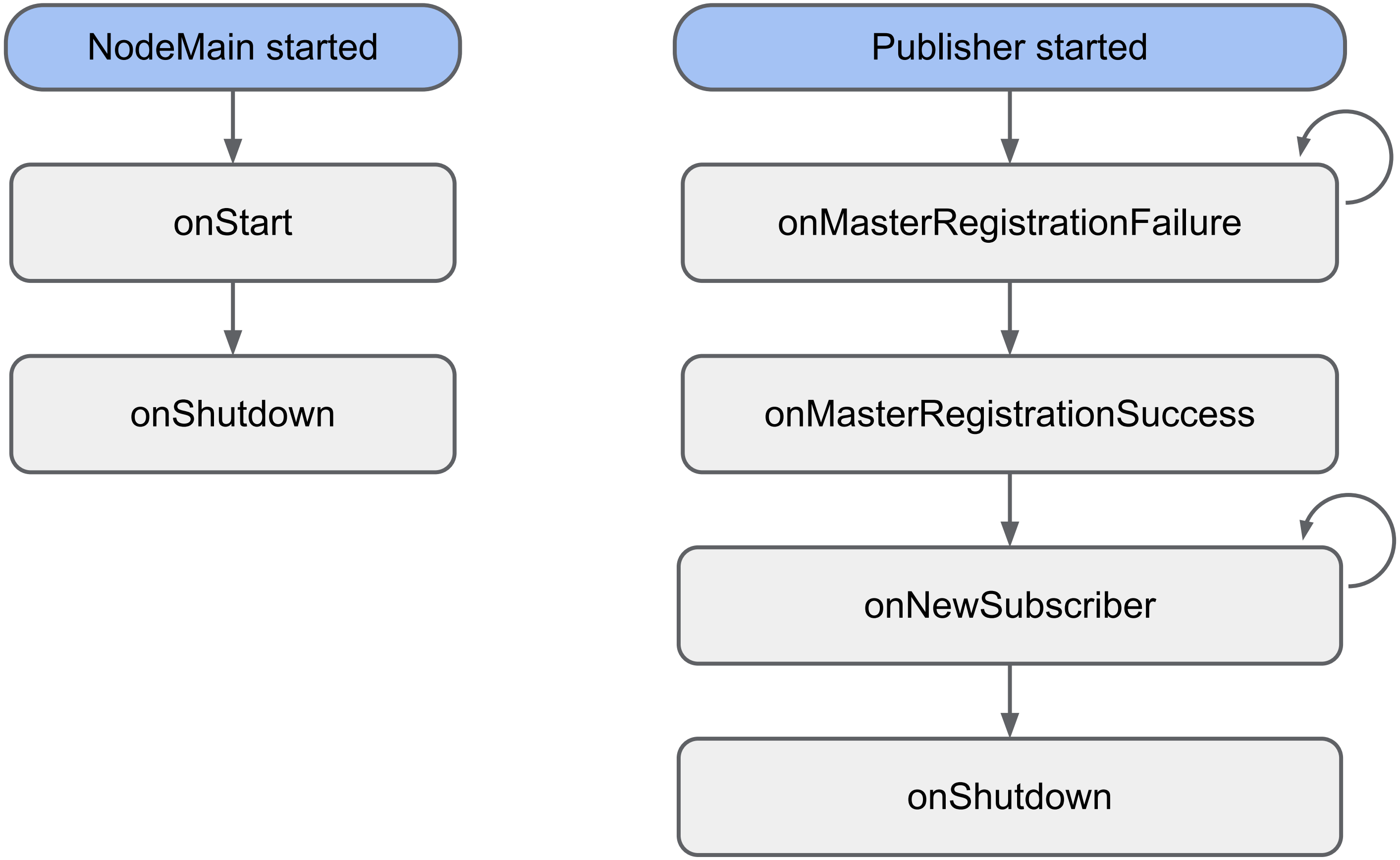
Node encapsulates

- NodeConfiguration
- factories (e.g. `newPublisher()`, `newSubscriber()`)
- parameter client
- master client
- slave server

NodeMain

- encapsulates business logic
- is a NodeListener

# Lifecycle events



# NodeMain

```
public class MyNode implements NodeMain {

  @Override
  public GraphName getDefaultNodeName() {
    return new GraphName("my_node");
  }

  @Override
  public void onStart(Node node) {
    // TODO: Get down to business.
    ...
  }
  ...
}
```

# More threads, fewer processes

NodeMainExecutor encapsulates

- NodeFactory
- ExecutorService (thread pool)

```
public static void main(String[] argv) {
  NodeMain nodeMain;
  NodeConfiguration nodeConfiguration;
  NodeMainExecutor nodeMainExecutor;
  ...
  nodeMainExecutor.execute(nodeMain, nodeConfiguration);
  ...
}
```

# No spin

```
node.executeCancellableLoop(new CancellableLoop() {
  @Override
  protected void loop() {
    System.out.println("Work, work, work.");
  }
}
```

## Not

```
while (node.isOk()) {
  System.out.println("Work, work, work.");
  node.spin();
}
```

# Publishers and Subscribers

```
Node node;
std_msgs.String message;
...

Publisher<std_msgs.String> p =
    node.newPublisher("chatter", std_msgs.String._TYPE);
p.publish(message);

Subscriber<std_msgs.String> subscriber =
    node.newSubscriber("chat", std_msgs.String._TYPE);
subscriber.addMessageListener(
    new MessageListener<std_msgs.String>() {
  @Override
  public void onNewMessage(std_msgs.String message) {
    ...
  }
});
```

# Service server

```
node.newServiceServer(
    "add_two_ints",
    test_ros.AddTwoInts._TYPE,
    new ServiceResponseBuilder<
        test_ros.AddTwoInts.Request,
        test_ros.AddTwoInts.Response>() {
      @Override
      public void build(
          test_ros.AddTwoInts.Request request,
          test_ros.AddTwoInts.Response response) {
        response.setSum(request.getA() + request.getB());
      }
    });
```

# Service client

```
ServiceClient<
    test_ros.AddTwoInts.Request,
    test_ros.AddTwoInts.Response> client =
    node.newServiceClient(
        "add_two_ints", test_ros.AddTwoInts._TYPE);
    test_ros.AddTwoInts.Request request =
        client.newMessage();
    client.call(
        request,
        new ServiceResponseListener<
            test_ros.AddTwoInts.Response>() {
      @Override
      public void onSuccess(
          test_ros.AddTwoInts.Response response) {
        ...
      }
    });
```

# Messages

- Empty interfaces backed by dynamic proxy
- rosjava_messages contains the world
- Definitions parsed at runtime

```
Node node;
...
MessageFactory factory = node.getTopicMessageFactory();
std_msgs.String m =
    factory.newFromType(std_msgs.String._TYPE);

Publisher<std_msgs.String> publisher;
...
std_msgs.String m = publisher.newMessage();
m.setData("Hello, world!");
```

# Build system

- Java dependency management, *not* debs
- Gradle, *not* rosmake

Gradle (http://www.gradle.org)

- Flexible
- Build-by-convention
- Dependency management

# Package directory layout

```
my_stack
    stack.xml
    my_package
        manifest.xml
        build.gradle
        src
            main
                java
                resources
            test
                java
                resources
        build
```

# Basic build.gradle

```
apply plugin: 'java'

sourceCompatibility = 1.6
targetCompatibility = 1.6

version = '0.0.0-SNAPSHOT'
group = 'ros.my_stack'

dependencies {
  compile 'ros.rosjava_core:rosjava:0.0.0-SNAPSHOT'
}

repositories {
  mavenLocal()
  mavenCentral()
}
```

# Running nodes

## build.gradle

```
apply plugin: 'application'

mainClassName = 'org.ros.RosRun'
installApp.into project.file('dist')
distZip.destinationDir project.file('dist')
```

## Build and execute

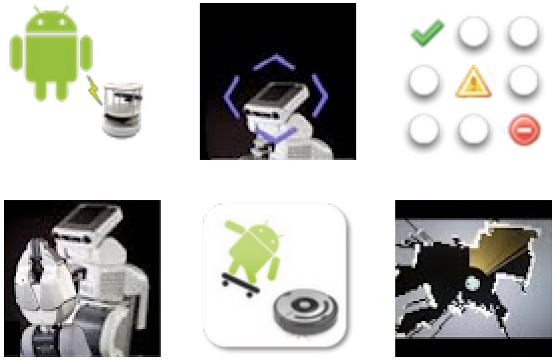
```
./gradlew installApp
rosrun my_package my_package com.example.MyNodeMain
```

# What about Android?

# Play today

- Sensor driver
- Robot monitor
- Teleop
- Map maker
- Map navigation
- PR2 props
- PR2 pan/tilt

Google play



# Adding more than what's on board

CDC ACM (android_acm_serial)

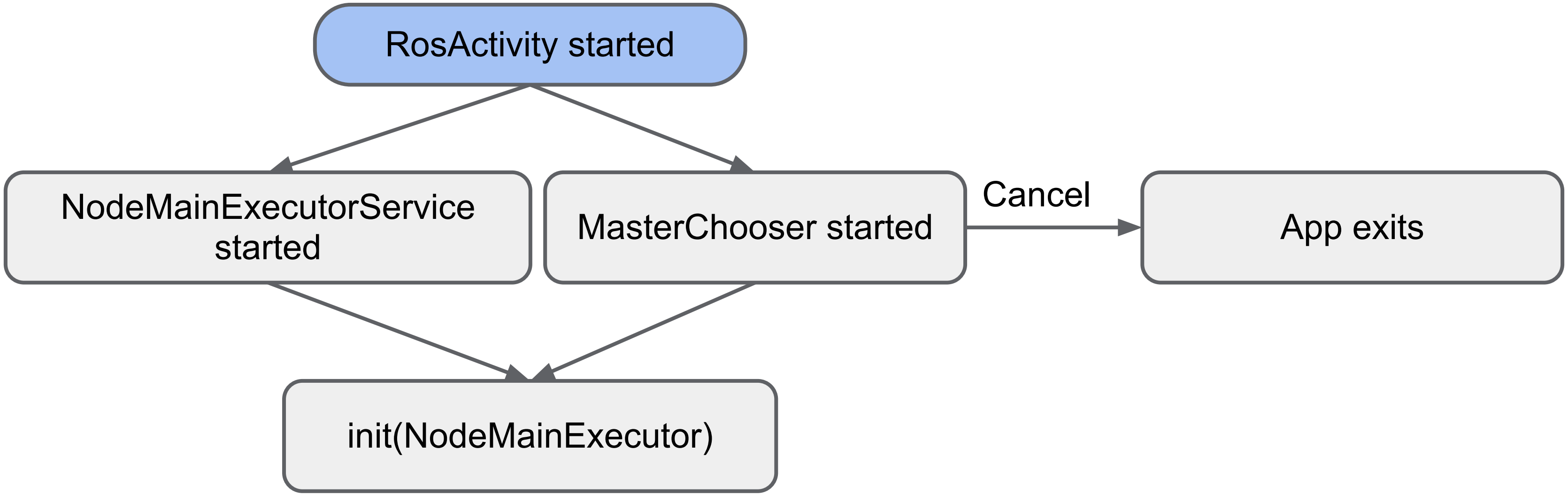
# Android plus lasers

# Android apps

```
public class MainActivity extends RosActivity {

  @Override
  protected void init(NodeMainExecutor
      nodeMainExecutor) {
    NodeMain nodeMain;
    ...
    NodeConfiguration cfg =
        NodeConfiguration.newPublic();
    cfg.setMasterUri(getMasterUri());
    nodeMainExecutor.execute(nodeMain,
        nodeConfiguration);
  }
  ...
}
```

# RosActivity lifecycle



- Wake and WiFi locks
- Persistent background service
- Notification

# Android patterns

- Views as NodeMains
- Data driven UIs

```
public class RosTextView<T> extends TextView
    implements NodeMain {

  @Override
  public void onStart(Node node) {
    Subscriber<T> s = node.newSubscriber(...);
  }
  ...
}
```

# Android components

- RosTextView
- RosImageView
- RosCameraPreviewView
- OrientationPublisher
- VirtualJoystick
- VisualizationView
- DistanceView
- ...

# Best practices

Java packages

- Domain name should be followed by the ROS package name
- Only core packages should begin with org.ros

Messages

- Use fully qualified class name (e.g. std_msgs.String)
- Prefer newMessage() helpers to factories

Asynchronous

- Prefer callbacks to blocking

# Current focus

- Feature parity with roscpp/rospy
- Stable API
- Comprehensive documentation
- 1.0 release

# Future directions

- OSGi
- Groovy deployment scripts
- Java CLI tools (e.g. rostopic, rosservice)
- Topic multiplexing
- Topic-based services
- Topic-based master, multi-master
- ...

http://www.cloudrobotics.com/

damonkohler@google.com