ROS.NET
(known aliases: ROS#, ROSSharp)

ROS for .NET 3.5+ Applications

Eric McCann
Umass Lowell Robotics Lab
October 16, 2014
Outline

• Introduction
  o What is ROS.NET
  o Background
  o Development

• Design of ROS.NET client core
  o Message handling
  o Toolchain
  o Implemented ROS capabilities

• Samples
  o Juxtaposition of ROS.NET’s and ROSCPP’s “talker” sample
  o Overview of our UI from the 2013 RASC-AL RoboOps sample collection challenge
# What is ROS.NET?

<table>
<thead>
<tr>
<th></th>
<th>ROS.NET</th>
<th>ros-win</th>
<th>roscs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Windows only</td>
<td>Windows only (cross compiled or built on)</td>
<td>Linux only (uses mono)</td>
</tr>
<tr>
<td>Language</td>
<td>C#</td>
<td>C++</td>
<td>C#</td>
</tr>
<tr>
<td>.NET support</td>
<td>3.5 or higher</td>
<td>None (win32/COM)</td>
<td>N/A</td>
</tr>
<tr>
<td>Toolchain</td>
<td>Msbuild</td>
<td>Cmake/nmake/catkin</td>
<td>Rosmake</td>
</tr>
<tr>
<td>Tools and utilities</td>
<td>Enough of them?</td>
<td>All of them</td>
<td>All of them</td>
</tr>
</tbody>
</table>
Development

• Oldest commit in git history: June 24, 2011
• Borrows one 3\textsuperscript{rd} party dependency from ROS Diamondback (XmlRpc++)
  o XmlRpc++ had project metadata to compile it with Sisual Studio – Just needed adjustment to compile as a DLL and some function exports for P/Invoke
    ▪ (.NET $\leftrightarrow$ Win32 interop)
  o Wrapped with a C# wrapper
Development

https://www.youtube.com/watch?v=S30OznDBj44
Message Handling

ROS.NET generates C# code for every message type, calculates messages’ MD5Sums and builds a DLL that applications can reference.

All internal ROS.NET message handling is introspective.

std_msgs/String.msg

string data

Messages/std_msgs/String.cs
Toolchain

• MSBuild

• Pros:
  o Reasonably good at dependency management

• Cons:
  o Tight coupling between the ROS.NET client library and the messages being used by one application
  o “super set” of messages always built.
    ▪ Messages.dll has every message in it, and ROS_Csharp.dll refers to it
What ROS *things* does ROS.NET have?

- Publishers
- Subscribers
- Services
- Parameters
- Dynamic reconfigure (*parameters* + *publishers* + *subscribers*)
- Rosout (*publisher* + *service server*)
Talker sample

Talker.cpp

```cpp
int main(int argc, char** argv) {
  ros::init(argc, argv, "talker");
  ros::nodehandle nh;
  ros::Publisher chatter_pub =
    nh.advertise< std_msgs::String >
    ("chatter", 1000);
  ros::rate loop_rate(10); //10 Hz
  int count=0;
  while (ros::ok()) {
    std_msgs::String msg;
    std::string_stream ss;
    ss << "hello world " << count;
    msg.data = ss.str();
    ROS_INFO("%s", msg.data.c_str());
    chatter_pub.publish(msg);
    loop_rate.sleep();
    ++count;
  }
}
```

Talker.cs

```cs
private static void main(string[] args) {
  ROS.Init(args, "talker");
  NodeHandle nh = new NodeHandle();
  Publisher<Messages.std_msgs.String> chatter_pub =
    nh.advertise<Messages.std_msgs.String>
    ("chatter", 1000);
  int count=0;
  while (ROS.ok) {
    Messages.std_msgs.String msg =
      new Messages.std_msgs.String();
    msg.data = "hello world "+count;
    chatter_pub.publish(msg);
    ROS.Info(msg.data);
    Thread.Sleep(100); //10Hz
    ++count;
  }
}
```
Talker sample

Talker.cpp

```cpp
int main(int argc, char** argv) {
}
```

Talker.cs

```cs
private static void main(string[] args) {
}
```
Talker sample

Talker.cpp

ros::init(argc, argv, "talker");

Talker.cs

ROS.Init(args, "talker");
Talker sample

Talker.cpp

ros::nodehandle nh;

Talker.cs

NodeHandle nh = new NodeHandle();
Talker sample

**Talker.cpp**

```cpp
ros::Publisher chatter_pub =
    nh.advertise<std_msgs::String>("chatter", 1000);
```

**Talker.cs**

```cs
Publisher<Messages.std_msgs.String> chatter_pub =
    new Publisher<Messages.std_msgs.String>("chatter", 1000);
```
Talker sample

Talker.cpp

```cpp
int count=0;
while (ros::ok()) {
    //NEXT SLIDE
    loop_rate.sleep();
    ros::spinOnce();
    ++count;
}
```

Talker.cs

```cs
int count=0;
while (ROS.ok) {
    //NEXT SLIDE
    Thread.Sleep(100); //10Hz
    ++count;
}
```
Talker sample

Talker.cpp

```cpp
std_msgs::String msg;
std::stringstream ss;
ss << "hello world " << count;
msg.data = ss.str();
ROS_INFO("%s", msg.data.c_str());
chatter_pub.publish(msg);
```

Talker.cs

```cs
Messages.std_msgs.String msg =
    new Messages.std_msgs.String();
msg.data = "hello world " + count;
ROS.Info(msg.data);
chatter_pub.publish(msg);
```
Talker sample

**Talker.cpp**

```cpp
int main(int argc, char** argv) {
    ros::init(argc, argv, "talker");
    ros::nodehandle nh;
    ros::Publisher chatter_pub = 
        nh.advertise<std_msgs::String>  
            ("chatter", 1000);
    ros::rate loop_rate(10); //10Hz
    int count=0;
    while (ros::ok()) {
        std_msgs::String msg;
        std::stringstream ss;
        ss << "hello world " << count;
        msg.data = ss.str();
        ROS_INFO("%s", msg.data.c_str());
        chatter_pub.publish(msg);
        loop_rate.sleep();
        ++count;
    }
}
```

**Talker.cs**

```cs
private static void main(string[] args) {
    ROS.Init(args, "talker");
    NodeHandle nh = new NodeHandle();
    Publisher<Messages.std_msgs.String> chatter_pub = 
        nh.advertise<Messages.std_msgs.String>  
            ("chatter", 1000);
    int count=0;
    while (ROS.ok) {
        Messages.std_msgs.String msg = 
            Messages.std_msgs.String();
        msg.data = "hello world "+count;
        ROS_INFO(msg.data);
        chatter_pub.publish(msg);
        Thread.Sleep(100); //10Hz
        ++count;
    }
}
```
2013 RASC-AL RoboOps

About the competition

• 8 Universities
• Collect rock samples hidden around Johnson Space Center rock yard
• Robot controlled from University
• Our UI used ROS.NET
  o … and has components written by a half dozen undergraduates
2013 RASC-AL RoboOps

Hand-fabricated rover-style robot

Teleoperated in Houston, TX from downstairs

… in this building
2013 RASC-AL RoboOps

About the robot

• Onboard Mini ITX i7 w/ 16GB RAM
• 3 Logitech C910 webcams
• 1 DOF camera mast (3 ft*lb servo lifts mast once, then shuts off)
• 3 DOF arm (including gripper)
• 4 Windshield Wiper motors
• 4 U.S. digital optical encoders
• 2 Roboteq MDC2250s
• 3 large lithium ion batteries
• 4 Missile Switches
• ~3 yards of NASA-style wire wrap
• 1 undergraduate Robotics 1 class
• 3 graduate students
2013 RASC-AL RoboOps

“Comms spec”
In action

https://www.youtube.com/watch?v=9zs_K9zwtVs
Thank you

https://github.com/uml-robotics/ROS.NET

Web: http://robotics.cs.uml.edu
Youtube: http://www.youtube.com/umlrobotics

Eric McCann
emccann@cs.uml.edu
http://www.cs.uml.edu/~emccann

Time permitting, a glimpse at the rover UI will follow