Sistemas Operacionais Embarcados - ROS

Ícaro Gonçalves Siqueira

UNIVERSIDADE CATÓLICA DE PELOTAS

Engenharia de Computação - 2021/1

Histórico da distribuição

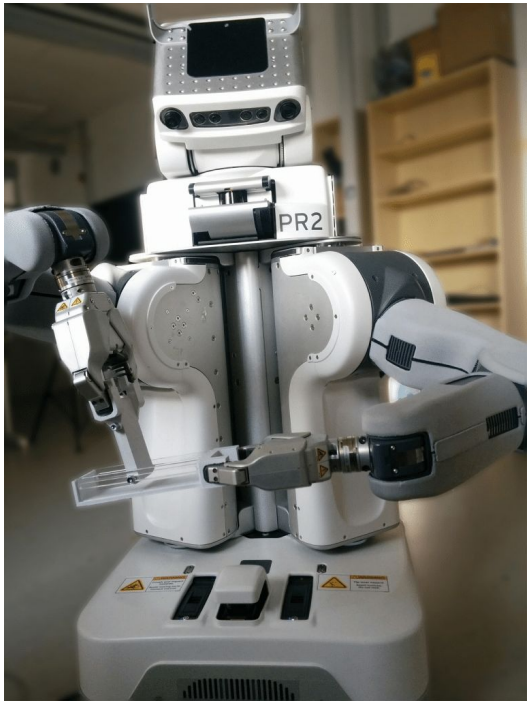
- Stanford University no laboratório de robótica do Prof. Kenneth Sailsbury;
- Doutorandos Eric Berger e Keenan Wyrobek;
- Scott Hassa, o fundador da Willow Garage;
- Compartilhava a mesma visão de criar um "Linux para a robótica";
- Primeira versão do código fonte **ROS foi adicionada ao SourceForge em novembro de 2007.**

Histórico da distribuição

- Doar 11 robôs PR2 para instituições acadêmicas ajudou a espalhar a notícia sobre os ROS pelo mundo da robótica;
- O primeiro lançamento oficial de distribuição de ROS (ROS Box Turtle), foi **lançado em 2 de março de 2010**;
- Esses desenvolvimentos possibilitaram a utilização de **ROS em drones, carros autônomos** e à adaptação de ROS para **Lego Mindstorms.**

Histórico da distribuição

- Com o programa PR2 Beta em andamento, o robô **PR2 foi lançado oficialmente para compra em 9 de setembro de 2010.**



Motivações para sua criação

- Projetos com progressos atrasados pela **natureza diversa da robótica:**
  - Um desenvolvedor de software poderia não ter o conhecimento de hardware necessário;
  - Alguém que trabalha em planejamento de rotas poderia não ter conhecimentos em visão computacional.

Arquitetura e Visão geral

- ROS (Robot Operating System);
- Coleção de frameworks de software para desenvolvimento de robôs;
- ROS fornece serviços padrões de sistema operacional:
  - Abstração de hardware;
  - Controle de dispositivos de baixo nível;
  - Implementação de funcionalidades comumente usadas;
  - Passagem de mensagens entre processos;
  - Gerenciamento de pacotes.
- Processos do ROS são representados em uma arquitetura de grafos onde o processamento se realiza em nós que podem receber e enviar mensagens.

Arquitetura e Visão geral

- **Infraestrutura de comunicação:**
  - No baixo nível o ROS é comumente referido como um **middleware.**
  - **Passagem de mensagens:**
    - O sistema de mensagens integrado do ROS **economiza tempo**, gerenciando os detalhes da comunicação entre os nós por meio do mecanismo de **publish/subscribe anônimo**;
    - Outro benefício de usar um sistema de passagem de mensagens é que ele força você a implementar **interfaces claras** entre os nós em seu sistema, melhorando assim o encapsulamento e promovendo a **reutilização de código.**

Arquitetura e Visão geral

- **Infraestrutura de comunicação:**
  - **Gravação e reprodução de mensagens:**
    - O sistema de publish/subscribe é anônimo e assíncrono, ROS torna mais **fácil capturar os dados publicados** em um arquivo e **republicar esses dados do arquivo** posteriormente.
  - **Chamadas de procedimento remoto:**
    - Às vezes você deseja interações síncronas de solicitação/resposta entre os processos. O middleware ROS fornece esse recurso usando serviços.

Arquitetura e Visão geral

- **Infraestrutura de comunicação:**
  - **Sistema de Parâmetros Distribuídos:**
    - O middleware ROS também fornece uma maneira para as tarefas **compartilharem informações** de configuração por meio de um armazenamento de **valor-chave global.**

Arquitetura e Visão geral

- **Recursos específicos do robô:**
  - Além dos componentes centrais de middleware, o ROS fornece **bibliotecas e ferramentas comuns** específicas para robôs.
  - **Mensagens de robô padrão:**
    - Um conjunto de formatos de **mensagem padrão** que cobrem a maioria dos **casos de uso comuns** em robótica;
    - Definições de mensagem para:
      - **Conceitos geométricos** como poses, transformações e vetores;
      - Para **sensores** como câmeras, IMUs e lasers.

Arquitetura e Visão geral

- **Recursos específicos do robô:**
  - Além dos componentes centrais de middleware, o ROS fornece bibliotecas e ferramentas comuns específicas para robôs.
  - **Mensagens de robô padrão:**
    - Para dados de **navegação** como odometria, caminhos e mapas; entre muitos outros;
    - Usando essas mensagens padrão o código irá interoperar perfeitamente com o resto do ecossistema ROS.

Arquitetura e Visão geral

- **Ferramentas:**
  - Um dos recursos mais fortes do ROS é o poderoso conjunto de **ferramentas de desenvolvimento**;
  - Oferecem suporte à introspecção, depuração, plotagem e visualização do estado do sistema;
  - As ferramentas ROS tiram proveito dessa capacidade de introspecção por meio de uma ampla coleção de **utilitários gráficos e de linha de comando.**

Arquitetura e Visão geral

- **Ferramentas:**
  - **Ferramentas de linha de comando:**
    - ROS pode ser usado 100% sem uma GUI;
    - Todas as principais funcionalidades e ferramentas de introspecção são acessíveis através de linha de comando;
    - Comandos para lançar grupos de nós, introspecção de tópicos, serviços e ações; gravação e reprodução de dados;
    - Com as ferramentas gráficas, **rviz** e **rqt** fornecem funcionalidades semelhantes.

Arquitetura e Visão geral

- Ferramentas independentes da língua e principais bibliotecas clientes de ROS (C++, Python e Lisp) são **softwares de fonte aberta** voltados para um **sistema Unix;**
- O Ubuntu é listado como "suportado";
- Fedora, Mac OS X e Windows são designados "experimentais".

Arquitetura e Visão geral

- A biblioteca cliente **Java ROS** nativa, **rosjava**, permitiu software baseado em ROS a ser escrito para **Android**;
- A rosjava também permitiu que ROS fosse integrado em uma **caixa de ferramentas MATLAB** oficialmente suportada, que pode ser usada em **Linux, Mac OS X e Microsoft Windows;**
- **ROS não é um sistema operacional de tempo real**, embora seja possível integrar ROS com código em tempo real;
- A falta de suporte para sistemas em tempo real está sendo abordada no desenvolvimento de **ROS 2.0**.