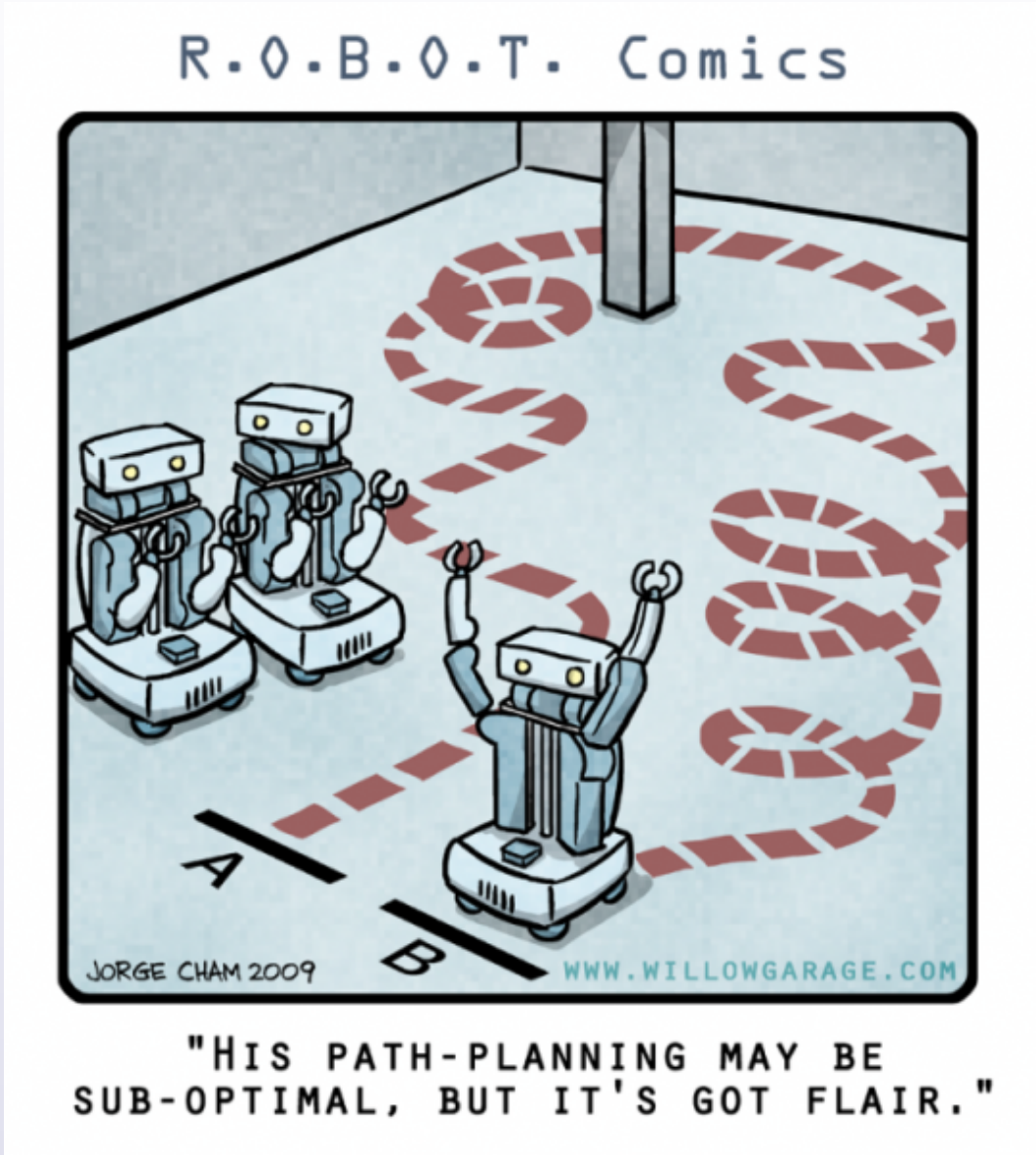

R.O.B.O.T. Comics
A
B
JORGE CHAM 2009
WWW.WILLOWGARAGE.COM
"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

ROS

Eine Einführung in das Robot Operating System

Florian Kathe

fkathe@uni-koblenz.de

Institut für Computervisualistik
Universität Koblenz-Landau

9. November 2012

Überblick

ROS.org

Was *ist* ROS?

- kein Betriebssystem
- eher ein Framework
- für Roboter
- ist Open Source
- läuft unter Linux

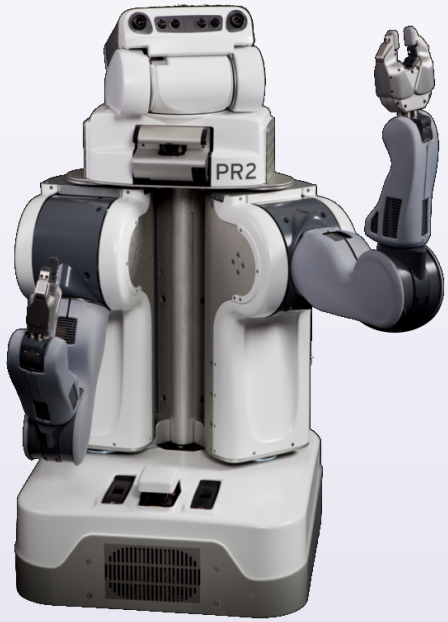
ROS.org

Was *ist* ROS?

- der Code von ROS ist *BSD* lizensiert, man kann ihn also leicht in das eigene Projekt integrieren
- ROS wird unterstüzt von *Willow Garage*
- Willow Garage entwickelt auch *OpenCV*
- entwickelt sich vielleicht zur Standartplattform für Robotik

ROS.org

Roboter



ROS.org


Volkswagen Electronics Research Lab
5
3
intel
RedBull
MDV
ibeo
Powered by Diesel.

ROS.org

Was *kann* ROS?

Stellt aber Funktionen zu Verfügung, die man von einem Betriebssystem erwarten würde:

- Hardwareabstraktion
- Gerätetreiber
- Implementierung von viel genutzten Funktionalität
- Inter-Prozess-Kommunikation
- Paket-Management

ROS.org

Was *will* ROS?

ROS will unterstützen, Code für Forschung und Entwicklung wiederzuverwenden

- loser Verbund von individuellen Programmteilen (**Nodes**)
- einzelne Programmteile können einfach geteilt und verbreitet werden (**Packages** und **Stacks**)
- ROS stellt Repositories zu Verfügung, um dort Code zu teilen (`http://www.ros.org/browse`)

ROS.org

Was *will* ROS?

Weitere Ziele:

- Einfachheit: Code für ROS kann auch für andere Roboter Frameworks verwendet werden
- Sprachunabhängigkeit: Man kann in C++, Python und Lisp programmieren (experimentell auch Java)
- Skalierbarkeit: ROS ist für große Systeme und einen großen Entwicklungsprozess entwickelt worden

ROS.org

Ebenen

ROS besteht aus 3 Ebenen:

1. ROS Filesystem
2. ROS Computation Graph
3. ROS Community

ROS.org

Computation Graph

Peer-to-peer Netzwerk von ROS Prozessen, die zusammen Daten verarbeiten

Nodes

Prozesse, die Berechnungen durchführen.
Werden in *C++* oder *Python* erstellt.
Jeder Node läuft in einem eigenen Prozess

Master

Führt die Nodes zusammen, damit sie Mitteilungen austauschen und Services nutzen können.

ROS.org

Message

Darüber kommunizieren Nodes miteinander.
Messagesfunktionieren mittels des Veröffentlichen zu und Abonnierens von Themen.
(*Post & Subscribe*)

Services

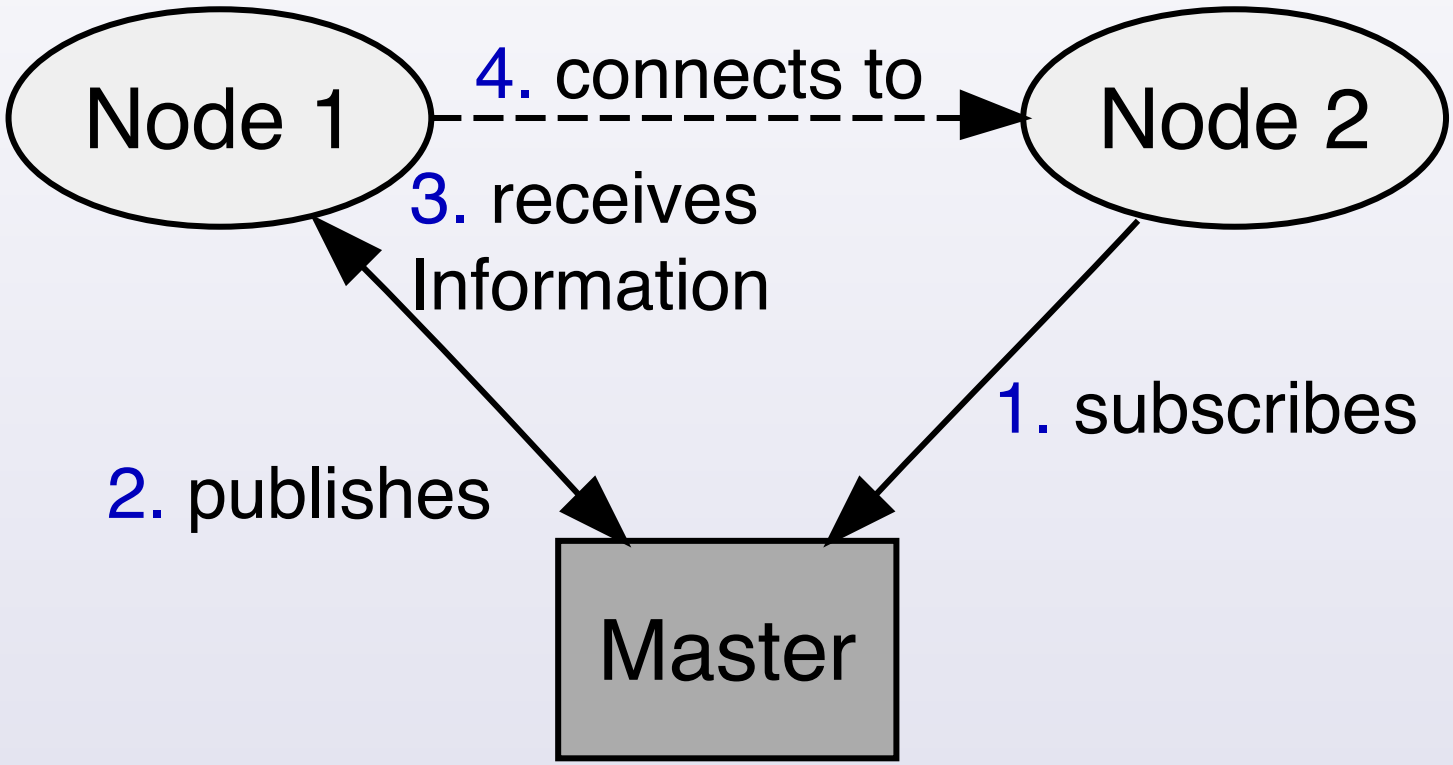
Nodes können Services bereitstellen, die von anderen Nodes genutzt werden können.
(*Request & Reply*)

Topic

Darüber können die Nessages und Services eindeutig identifiziert werden.

Message

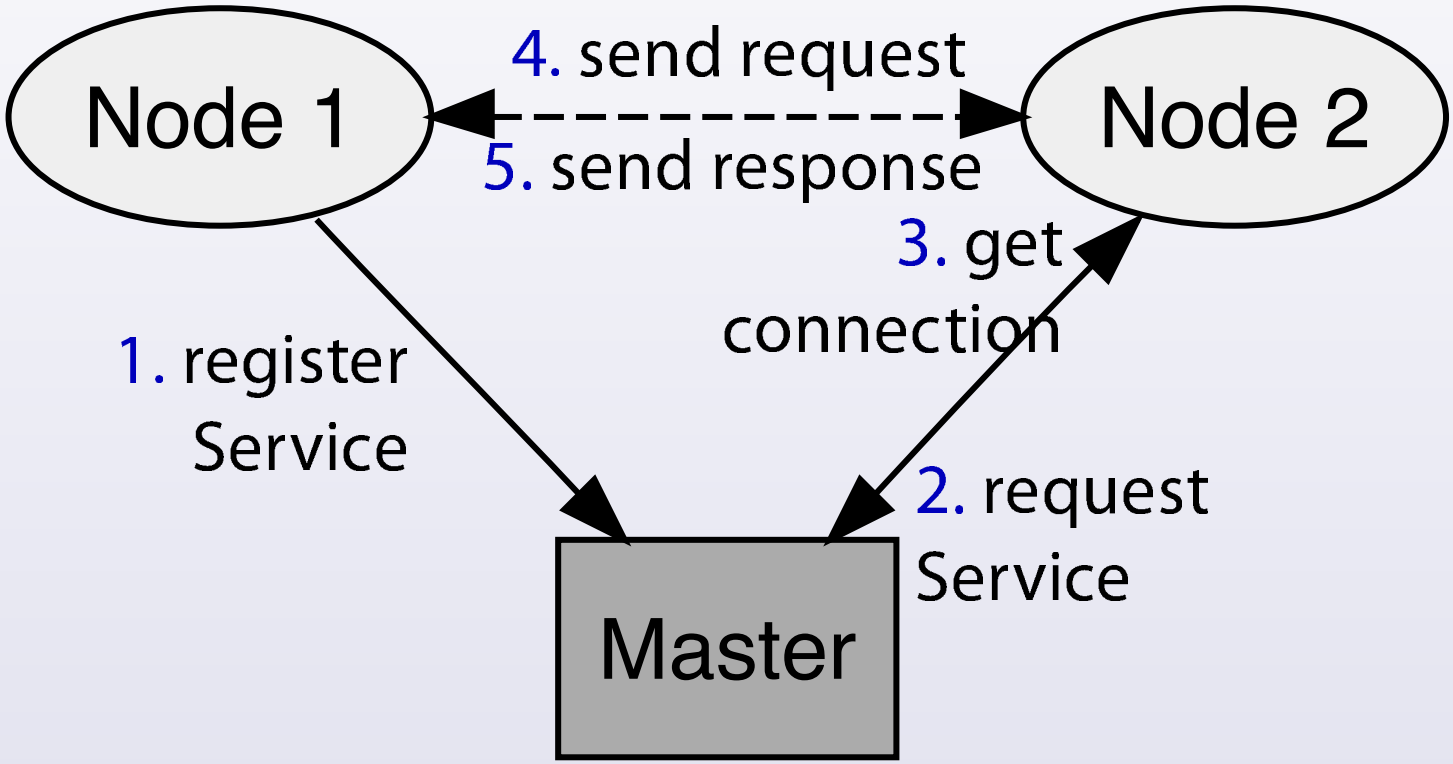
1:n Kommunikation



ROS.org

Service

1:1 Kommunikation



ROS.org

sensor_msgs/LaserScan Message

```
Header header
    uint32 seq
    time stamp
    string frame_id
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

ROS.org

Verwaltung der lokalen Daten:

Package

verwaltet die Software
Enthält: Nodes, Bibliotheken, Konfigurationen, etc

Stack

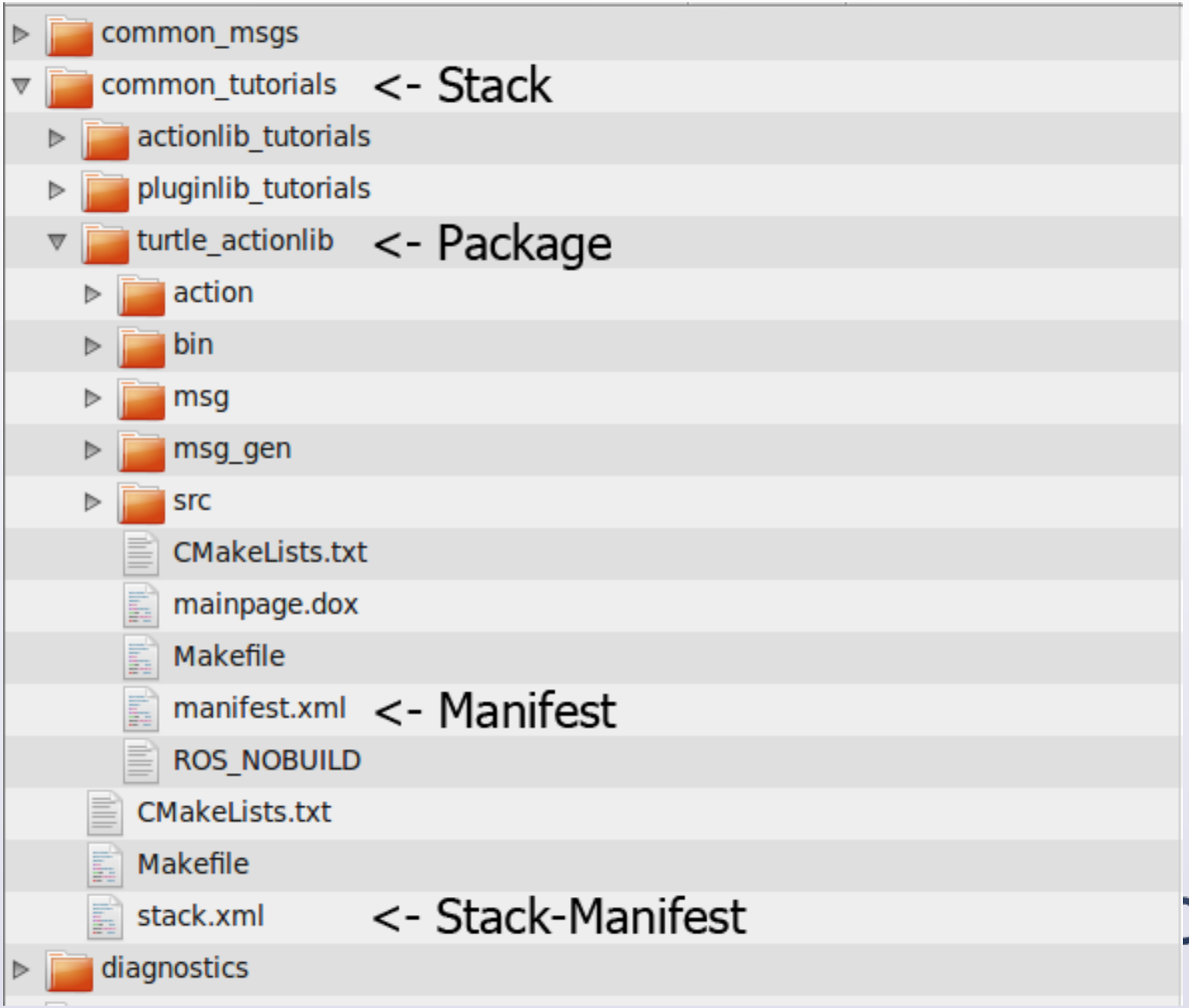
verwaltet die Packages

Manifest

erstellt die Metadaten (Lizenzen, Abhängigkeiten)
bei Package: sprachabhängige Informationen

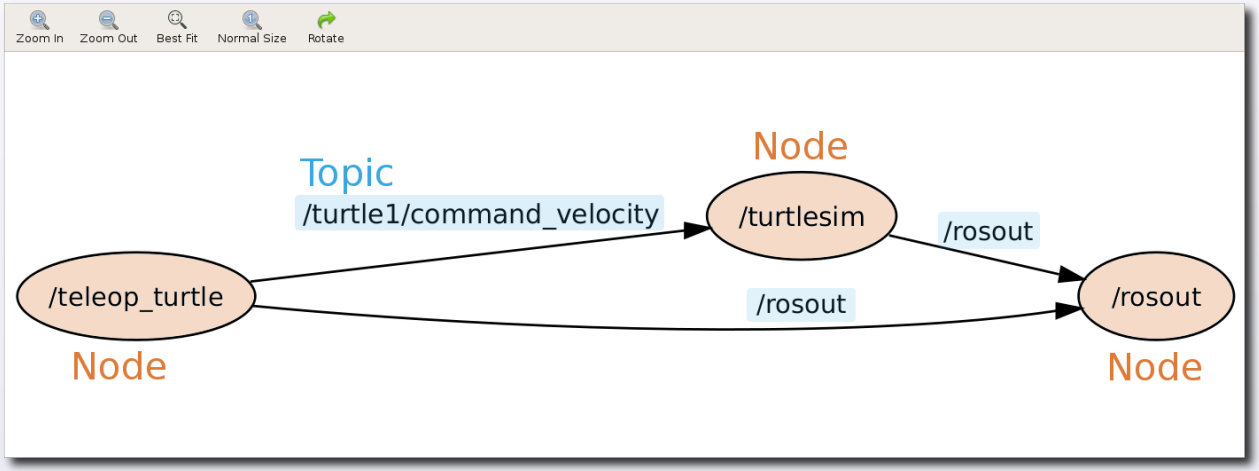
ROS.org

Beispiel


common_msgs
common_tutorials <- Stack
actionlib_tutorials
pluginlib_tutorials
turtle_actionlib <- Package
action
bin
msg
msg_gen
src
CMakeLists.txt
mainpage.dox
Makefile
manifest.xml <- Manifest
ROS_NOBUILD
CMakeLists.txt
Makefile
stack.xml <- Stack-Manifest
diagnostics

rxgraph

```
$ rxgraph
```



- rxgraph zeigt alle aktiven Nodes
- und den Nachrichtenverkehr zwischen den Nodes
- auf der rechten Seite werden Informationen zum ausgewählten Node angezeigt

(Nodes und Topics sind in echt nicht farblich markiert)

rxbag

Tool zum Aufzeichnen und Abspielen der verschickten Messages

`$ rosbag record...`

- zum Analysieren der Fehler
- zum Simulieren von Daten (z.B. Laserscanner)
- zum Testen von Algorithmen

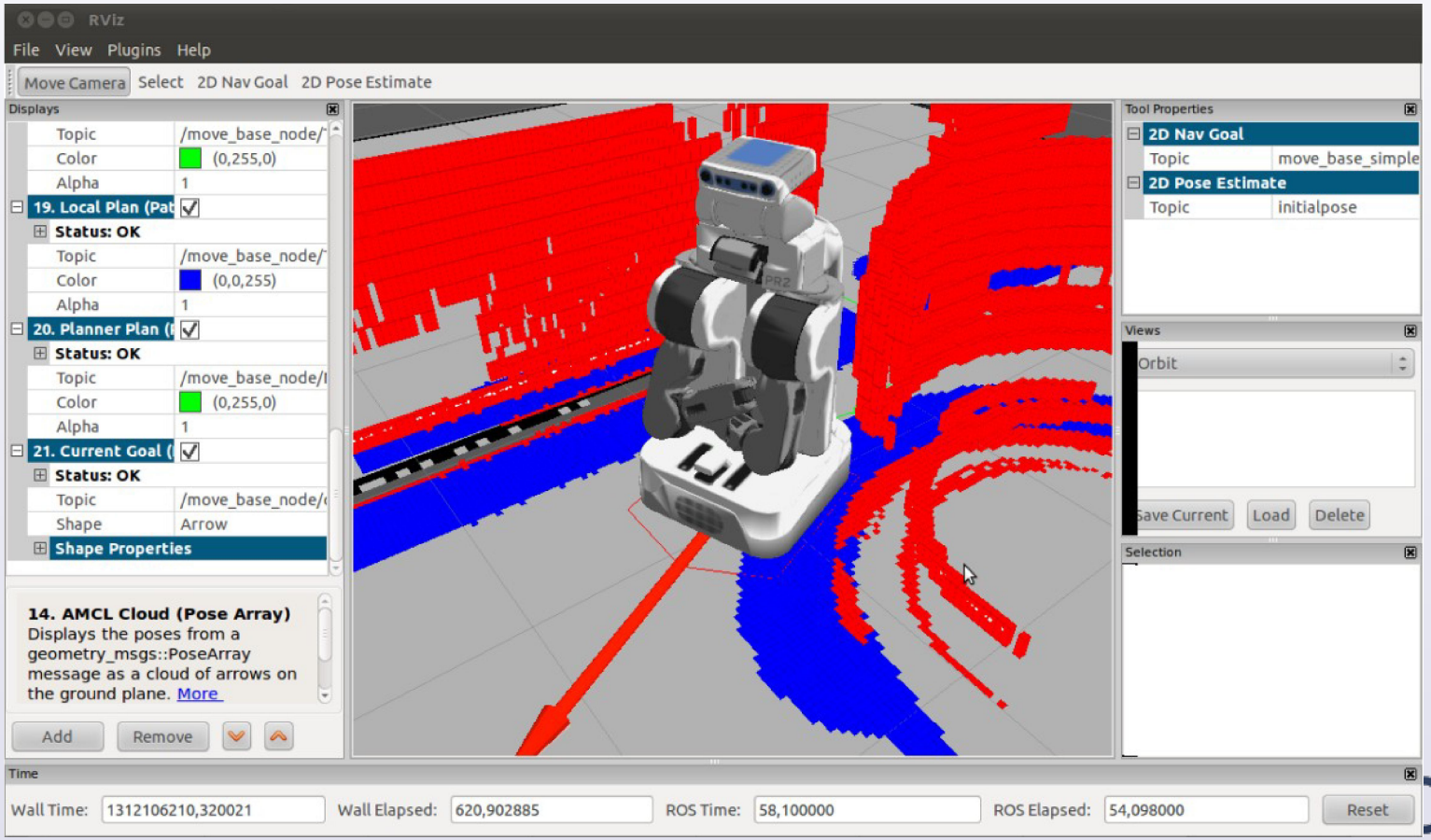
`$ rosbag play...`

ROS.org

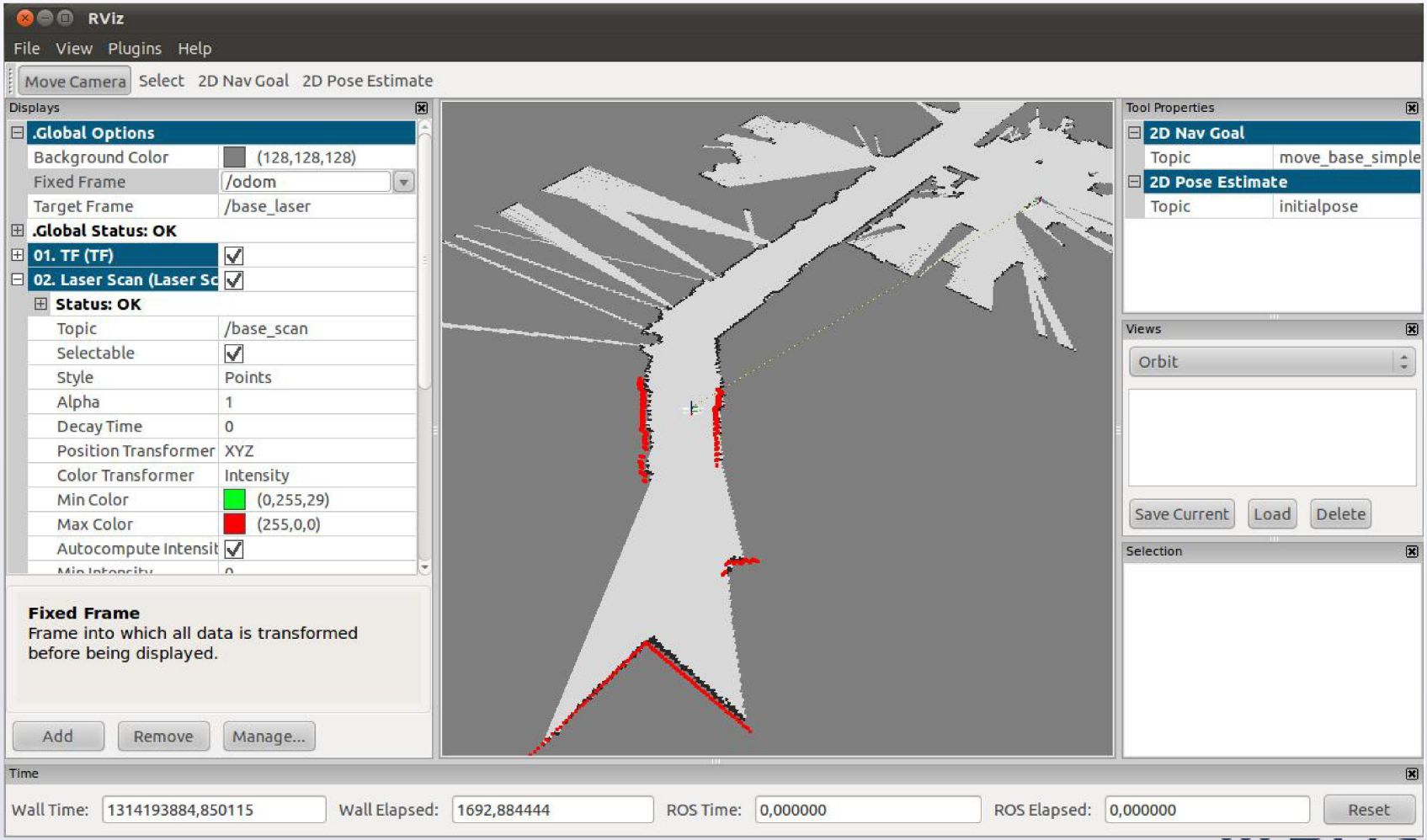
rviz

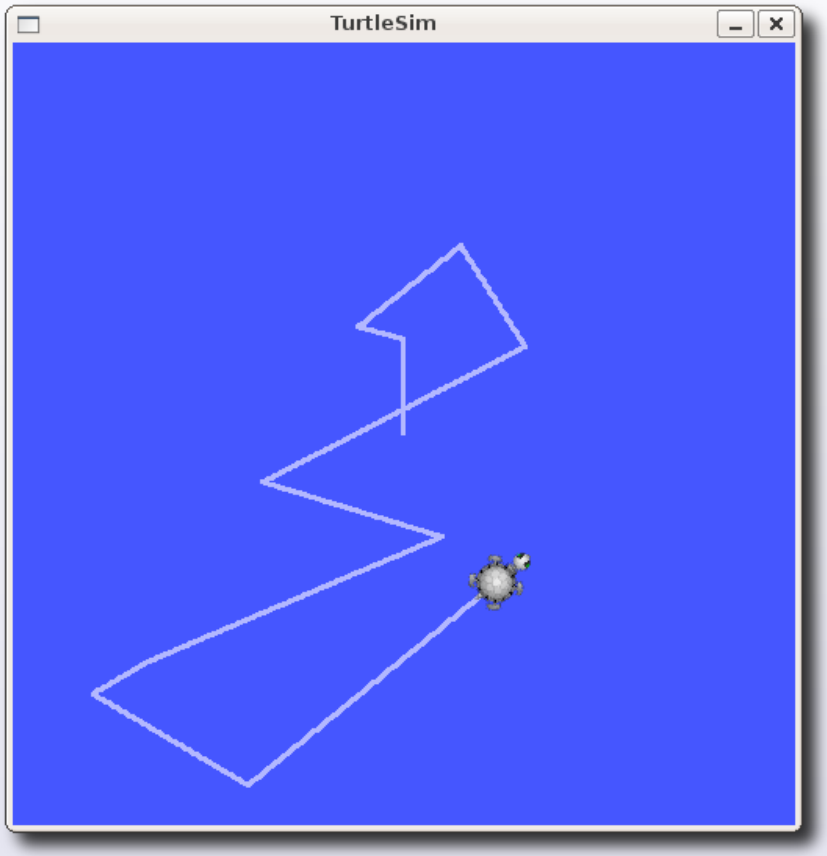
rviz ist ein mächtiges Tool zum Anzeigen von Daten

```
$ rosrun rviz rviz
```



rviz


RViz
File View Plugins Help
Move Camera Select 2D Nav Goal 2D Pose Estimate
Displays
.Global Options
Background Color (128,128,128)
Fixed Frame /odom
Target Frame /base_laser
.Global Status: OK
01. TF (TF)
02. Laser Scan (Laser Sc
Status: OK
Topic /base_scan
Selectable
Style Points
Alpha 1
Decay Time 0
Position Transformer XYZ
Color Transformer Intensity
Min Color (0,255,29)
Max Color (255,0,0)
Autocompute Intensit
Fixed Frame
Frame into which all data is transformed
before being displayed.
Add Remove Manage...
Tool Properties
2D Nav Goal
Topic move_base_simple
2D Pose Estimate
Topic initialpose
Views
Orbit
Save Current Load Delete
Selection
Time
Wall Time: 1314193884,850115
Wall Elapsed: 1692,884444
ROS Time: 0,000000
ROS Elapsed: 0,000000
Reset



(kurze) **Live-Demo**

ROS.org

Pro

- Nachrichten-basierte Software Architektur
  - verschiedene Komponenten sind unabhängig voneinander mit dem System verbunden
  - unterschiedliche Komponenten können miteinander verbunden werden, ohne jedes Mal das Programm neu zu Kompilieren
  - Netzwerkfähigkeit
- einfaches Debugging und Simulieren
- Absturz eines Nodes führt nicht zum Absturz des ganzen Systems
- für ROS lässt sich in mehreren Sprachen programmieren
- ROS hat eine große Community, die viele Daten und Programme zu Verfügung stellen

Contra

- durch Nachrichten-basierte Systemarchitektur Bottleneck bei großer Datenmenge
- Steuerung des Systems über Kommandozeile

Vielen Dank für die Aufmerksamkeit!

ROS.org