UNIVERSITÀ di VERONA | Dipartimento di INFORMATICA

LTAIR robotics lab

# Laboratorio ciberfisico (2018/2019): ROS 3

Diego Dall'Alba

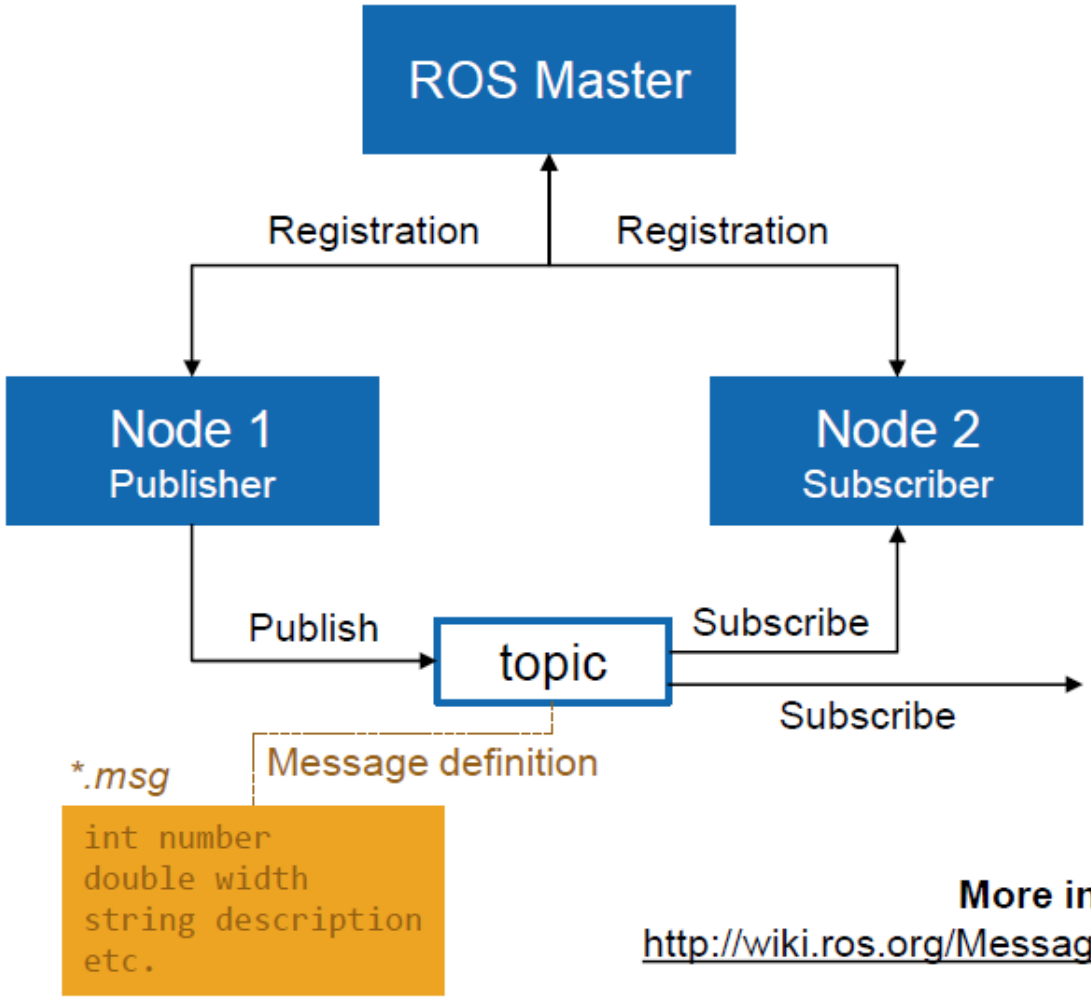
Altair robotics lab
Department of computer science – University of Verona, Italy

# Sommario di oggi

- Implementazione nodi ROS in C++
- Esempio interattivo

- Assegnazione eleborato 2.1

# Nella puntata precedente...

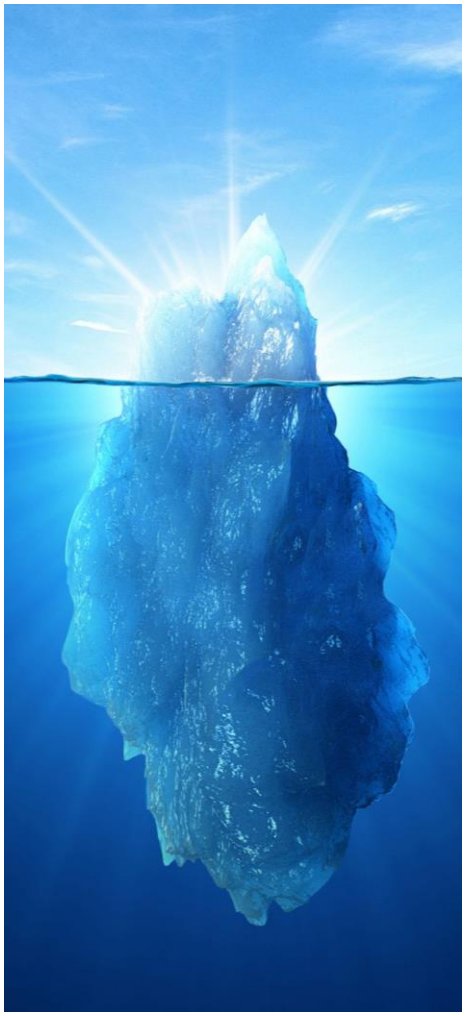


# Client Libraries

Le ROS client libraries permettono a nodi scritti in diversi linguaggi di programmazione di comunicare e sfruttare le funzionalità messe a disposizione dal middleware:

- **rospy** = python client library
- **roscpp** = c++ client library

Ci focalizzeremo sulla libreria C++, probabilmente la più utilizzata dalla comunità ROS 😐. E' una libreria molto potente, vedremo solo una parte delle funzionalità

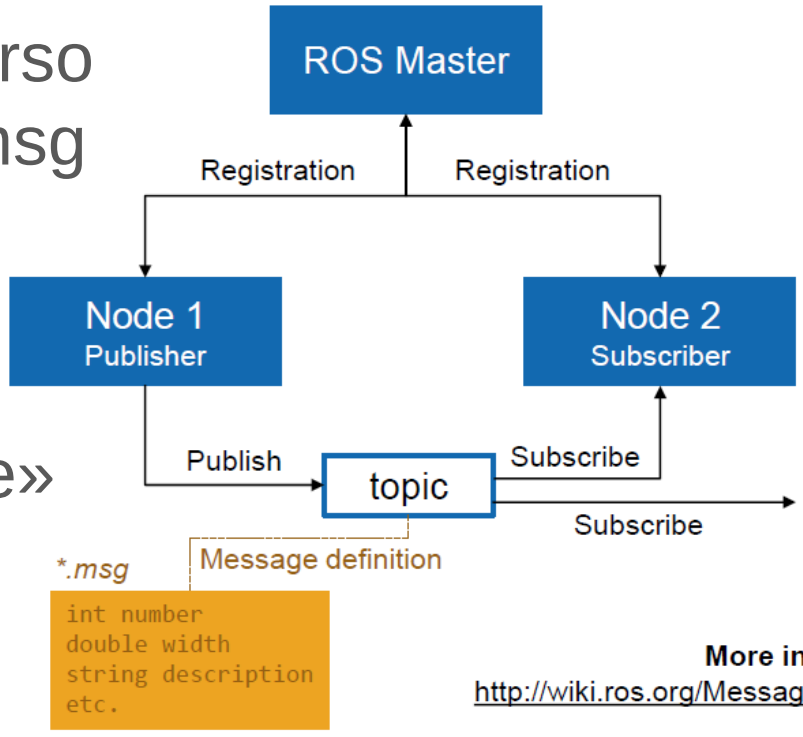


Maggiori dettagli: http://wiki.ros.org/roscpp

# Esercitazione di oggi

Implementeremo questa architettura:

- 2 nodi che comunicano attraverso un topic per lo scambio di un msg
- Un nodo «scriverà» le info pubblicandole sul topic
- L'altro nodo  si sottoscriverà (allo stesso topic) per «leggerle»
- Vedremo come i diversi passi rappresentati si «mappano» in funzioni e comandi C++



# Pre-requisiti e Passi preliminari

- Installazione funzionante (nativa o su VM) di ROS Melodic su Ubuntu Bionic (18.04)
- «Ambiente ROS» configurato correttamente
- Un editor di testo per il codice (io utilizzerò Visual studio Code visto che lo utilizzate anche per l'altro modulo del corso)

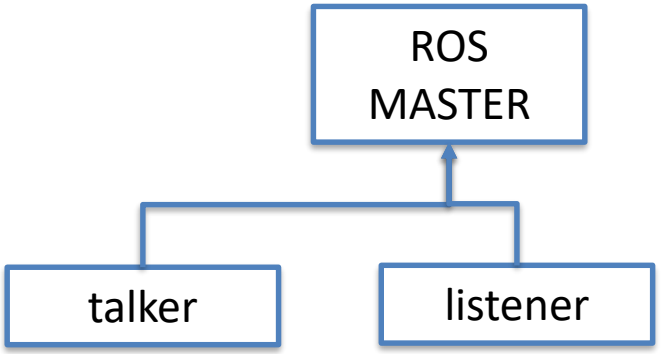
- Creare un nuovo package nel vostro catkin workspace che dipenda da: **roscpp** e **std_msgs**
- Modificare correttamente il file manifest (package.xml)
- Aprire il seguente link (utilizzeremo il codice sorgente, non seguiremo tutti i passi del tutorial):

http://wiki.ros.org/roscpp_tutorials/Tutorials/WritingPublisherSubscriber

# Passi principali (sessione interattiva)

- Scaricate il codice sorgente di talker e listener:

https://raw.github.com/ros/ros_tutorials/kinetic-devel/roscpp_tutorials/talker/talker.cpp
https://raw.github.com/ros/ros_tutorials/kinetic-devel/roscpp_tutorials/listener/listener.cpp




- Modificare il file CMakeLists.txt
- Provare a compilare il codice (con catkin)
  - in caso di errori provare a risolverli e ritentare la compilazione
- Provare a lanciare i nodi (rosrubn) e verificarne il funzionamento con i comandi visti (rosnode, rostopic, ecc…)

# Elaborato 2.1 – Parte A (CMake)

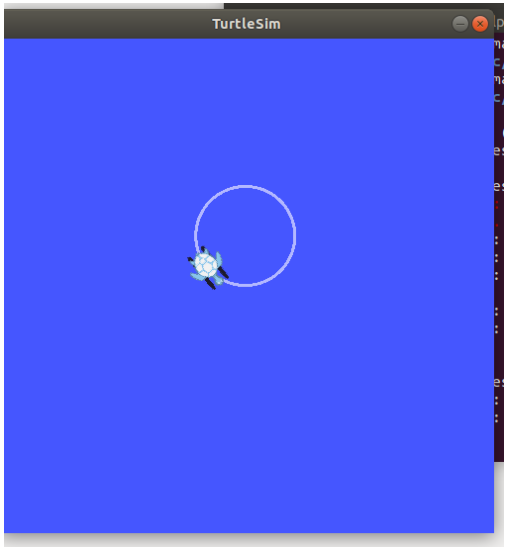
1. Configurare con CMake la compilazione del framework openIGTLink in modo da compilare gli esempi e le librerie Dinamiche, ma disabilitando tutte le funzioni relative al testing.
2. Dalla cartella contenente i sorgenti degli esempi copiare la sottocartella Status e compilarla separatamente.
3. Modificare lo StatusClient in modo che la stringa di status (StatusString) contenga la vostra matricola.
4. Provare la comunicazione :
   - In un terminale far partire l'eseguibile «ReceiveServer [num_port]»
   - In un secondo terminale far partire «StatusClient localhost [num_port]

5. Modificare il messaggio di status in modo che contenga anche un numero progressivo (volendo potete usare anche gli altri campi, non solo StatusString)

# Elaborato 2.1 – Parte B (ROS)

Scopo: Creare un nodo ROS che faccia muovere la tartaruga in turtlesim lungo una circonferenza.
Passi principali:

1. Creare un nuovo package ROS nel vostro workspace catkin, dovrà avere le giuste dipendenze
2. Creare un nuovo nodo, a partire da listener o talker visti a lezione
3. Modificare il codice in modo da pubblicare o sottoscriversi correttamente al topic per controllare il turtlesim
4. Compilare il codice e verificare il funzionamento.



# Fine lezione

Approfondimenti:
http://docs.ros.org/api/roscpp/html/