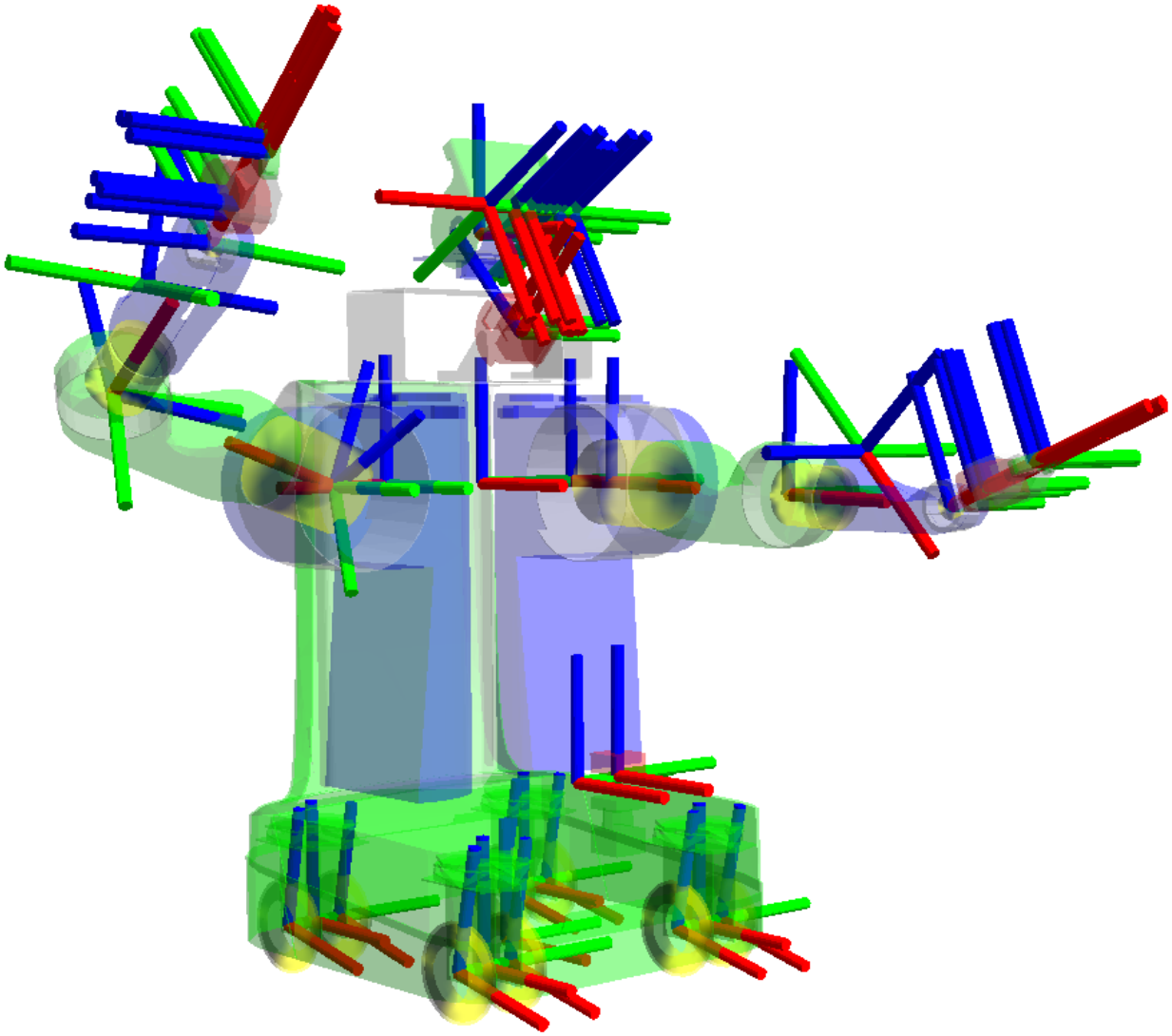
1. Theoretical background

- Stands for **"transform library"**
- A robotic system typically has many **3D coordinate frames** that change over time, such as a world frame, base frame, gripper frame, head frame, etc.
- tf2 keeps track of the **poses** of all these frames over time in a **tree structure** (parent-son)
- The pose is represented using **quaternions**
- Is used for:
  - Transform vectors relative to a frame to another
  - Keep a temporal track (schedule) of the frames **poses (transforms)**

- Get the information necesary for a kinematic analysis



2. Tf boadcasters and listeners

2.1 Definitions

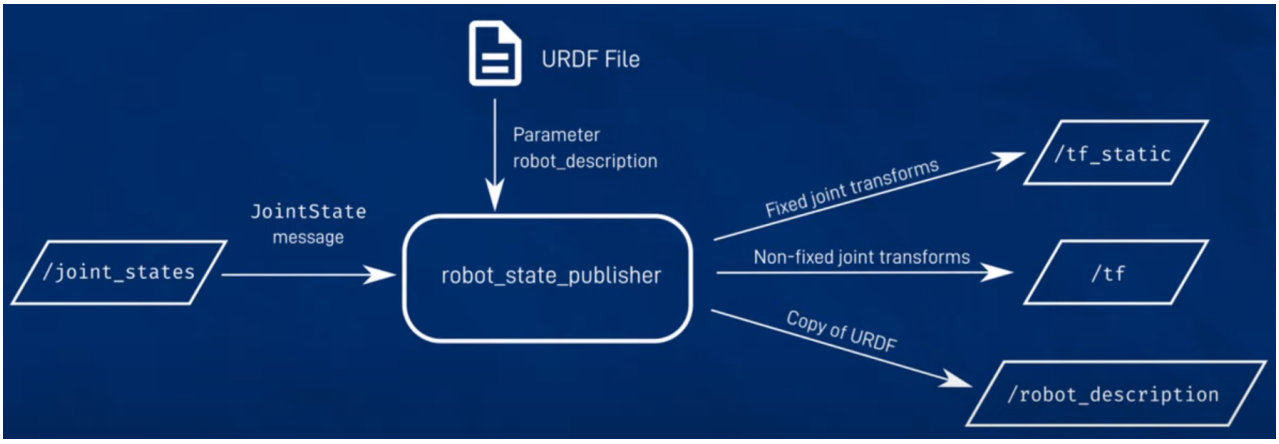
- Broadcast: To publish **all** the transforms of a robotic system
- Listen: To subscribe and read **all** the transforms of a robotic system
- Broadcaster
  - Node that **broadcast the relative time-changing poses** of the frames of a robot to the **/tf topic**

    - A single robotic system can use **several broadcasters** for different parts of its structure
- Static broadcaster:
    - Broadcast the pose of frames that **don't chage over time** to the **tf_static topic**, for instance: sensors, and non-moving parts relative to the robot base.
    - They are published once (non-periodically publishing)
    - This **saves storage and lookup time**
    - The default build-in ROS static broadcaster is **static_transform_publisher**
        - It can be used both in a node and at the CLI
- Listener:
    - Node that listen to a tf broadcaster at the **/tf and /tf_static topics**

2.2 robot_state_publisher

- The **robot_state_publisher** is a node that broadcast both the static and non-static poses of a robot:
  - It subscribes to the **/joint_states** topic to get the **joint vector** of the robot. Then it calculates the **forward kinematcs** to get the robot transformations
  - It publishes the transformations of the robot to both the **/tf** and the **/tf_static topics**
  - It also reads the **URDF file** and publish it once (non-periodically)

to the /robot_description topic

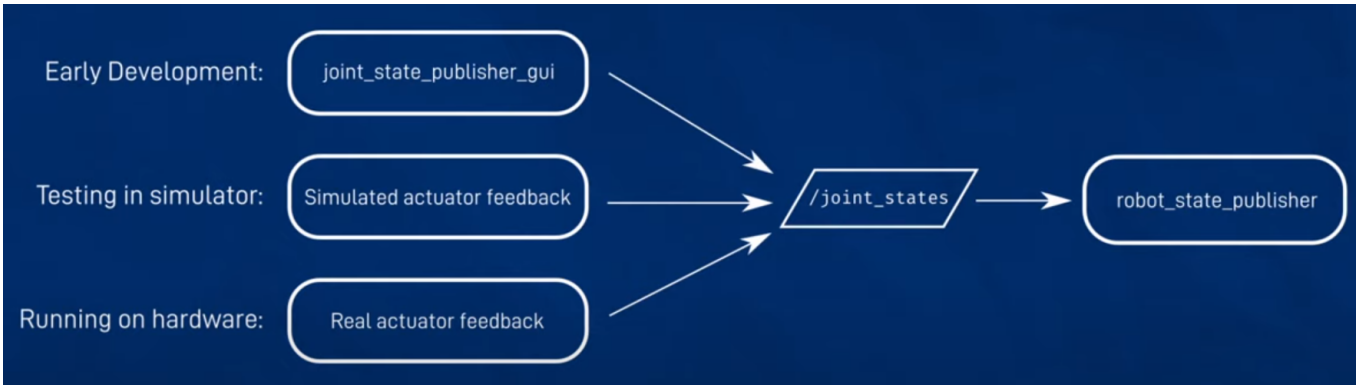


- The robot_state_publisher can be installed as:

```
        sudo apt install ros-<distro>-robot-state-
publisher
```

- The /joint_states topic contains the joint vector of the robot readed by the sensors. Nevertheless, it can be published by several sources:
  - Physical sensors feedback
  - Simulated sensors feedback: From simulators such as Gazebo
  - joint_state_publisher and joint_state_publisher_gui nodes: To facilitate the running of tests at early development stages without the necesity of sensors.
  - The joint_state_publisher can be installed as:

```
        sudo apt install ros-<distro>-joint-state-
publisher
        and/or
        sudo apt install ros-<distro>-joint-state-
publisher-gui
```



Python

C++

C++ API

3. CLI utilities

- Utilities intallation

```
sudo apt install ros-<distro>-tf2-tools ros-<distro>-tf-transformations
```

- Generate a tree-like view of the transforms published at the /tf topic(in a pdf):

```
ros2 run tf2_tools view_frames
```

- Echo the pose of a frame relative to another:

```
ros2 run tf2_ros tf2_echo <reference_frame> <target_frame>
```

Information sources

1. Paper

2. Official description
3. Official tutorials