**Future Work**

Version 0.0.2

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 08-05-2017 | 0.0.01 | Document outlining extensions required for the reconfiguration framework | Anindya Gautam |
| 08-14-2017 | 0.0.2 | Updating details of implementation of Actions | Aditya Mathur |

**Formal Approval**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Issues** | **Approved by** |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Introduction 4](#_Toc490731712)

[1.1 Purpose 4](#_Toc490731713)

[1.2 Definitions, Acronyms, Abbreviations 4](#_Toc490731714)

[2. Pending Features/Future Extensions 5](#_Toc490731715)

# **Introduction**

The Turtlebot project aims at building a framework to support configuration/reconfiguration for mobile robots at run time.

## Purpose

The purpose of this document is to detail all the pending features left to be implemented in the framework, and some of the extensions conceptualized as future additions to the framework.

## Definitions, Acronyms, Abbreviations

|  |  |
| --- | --- |
| **DAA** | **Term** |
| ROS | Robot Operating System |
| Bot | Robot |
| TBD | To Be Discussed |
| SCN | System Control Node |
| API | Application Programing Interface |

# **Pending Features/Future Extensions**

1. **Python Wrappers**

The current framework library provides wrappers for all C++ service-client and publisher-subscriber interactions. Similar wrappers need to be implemented for Python service-client and publisher-subscriber interactions for application developers developing nodes using Python and require to use the reconfiguration framework.

1. **Node Load State and Save State**

The framework library currently provides callback functions for Save State and Load state to the application node. The framework developer is responsible for implementing these functions for state management for cases when the node is killed, or when the node is restarted. In the current implementation, the responsibility for saving the state lies entire with the application developer and is not a responsibility of the framework. However, the application node has no information as to why it is entering reconfiguration mode, and if it will be killed or restarted or will just eventually exit reconfiguration mode without any actions performed on it. This may not be an ideal design for reconfiguration scenarios where the application node requires this information for state management. Thus, the framework requires some implementation logic to manage state information of all the application nodes, such as passing function pointers from the old version of the node to the new version of the node so that it can restore its state after being restarted.

1. **Switching Multiple Nodes**

At present the framework is capable of performing a one to one swap of nodes i.e. the framework can accept a node and the replacement node and perform the swap.

Future work involves accepting a list of nodes to swap out and a list of nodes to swap in. Also, the framework can be extended to replace a set of nodes by a single node.

1. **Action Servers**

The framework currently supports only service and topic interactions. It needs to be extended for actions and action servers as well.

One possible implementation of including Action Servers into the framework would be to wrap “actionlib” (Action Library) to register the nodes, using “actions”, with the System Configuration Framework. This can be done in a similar fashion as that of “Services” i.e. the nodes providing the action server would register with the SCN as a server and the nodes using the action will register with the SCN as action clients.

At the time of a reconfiguration activity, unlike the “Service” part of the framework, the SCN can cancel all goals of the action server after putting all dependent client nodes into reconfiguration mode. This would ideally pause the action server for the period of reconfiguration. At the end of the reconfiguration, it should be the client node’s responsibility to send a new goal to the action server.

Another approach to this could be storing the state of the action server at the start of the reconfiguration mode, and reloading the state at the end of the reconfiguration mode. But this would require modifications to the already existing “actionlib”.

1. **Framework Model**

The current model only depicts the interactions related to entering and exiting reconfiguration mode correctly. The model currently only has two persistent nodes that exchange services and go into reconfiguration mode. It doesn't spawn new replacement nodes as part of reconfiguration or kill existing ones, and it doesn't have any other nodes that might be using/be used by the two nodes. To make the model relevant to the realistic usage of the protocol, it needs to be extended.

Further, the model code using Promela currently depicts only the service and topic interactions. It needs to be extended to include action interactions, as per the Action model described in the model document.

1. **Usage documentation**

The current usage criterion for the framework is that all relevant nodes that require reconfiguration and all their dependent nodes need to using the reconfiguration framework to work successfully. However, some of our tests with this framework has shown that not all nodes may need to be using our framework for reconfiguration to work correctly for certain use cases. Extensive study and analysis needs to be done to find out when it may not be necessary for a node to be using the framework and yet for reconfiguration to work successfully. This analysis needs to be documented to provide sufficient reason for cases when the reconfiguration framework is necessary to be used, and when it not required, for different reconfiguration scenarios.