**Turtlebot Project**

**Framework Model Document**

Version 0.0.6

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 06-20-2017 | 0.0.1 | Software Model for Reconfiguration Framework | Aditya Mathur |
| 06-20-2017 | 0.0.2 | Software Model for Reconfiguration Framework | Aditya Mathur |
| 07-01-2017 | 0.0.3 | Updated Model | Aditya Mathur |
| 07-07-2017 | 0.04 | Added Model regarding Services, Removed Software Model | Anindya Gautam |
| 07-08-2017 | 0.05 | Updated Model regarding Topics | Anindya Gautam |
| 07-11-2017 | 0.06 | Updated Document as per Client Feedback | Anindya Gautam |

**Formal Approval**

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

Contents

[1. Introduction 4](#_Toc487567756)

[1.1 Purpose 4](#_Toc487567757)

[1.2 Definitions, Acronyms, Abbreviations 4](#_Toc487567758)

[2. Model for Services/Topics 5](#_Toc487567759)

[3. Model for Actions 17](#_Toc487567760)

[4. Model Interactions 19](#_Toc487567761)

# **Introduction**

This document presents the state model of the “Reconfiguration Framework” for mobile robots running on ROS. Model checking of the framework is documented in this document.

## Purpose

The purpose of this document is to provide an overview of the software model for the framework and to record the analysis performed on the model. It aims at documenting all possible states involved in the reconfiguration framework such that the framework handles all these states.

## Definitions, Acronyms, Abbreviations

|  |  |  |
| --- | --- | --- |
| **DAA** | **Term** | **Definition** |
| ROS | Robot Operating System |  |
| Bot | Robot |  |
| TBD | To Be Discussed |  |
| SCN | System Control Node | It is a process that acts as a moderator for reconfiguration activities in the system. |

# **Model for Services/Topics**

* 1. **Brief Description**

This model depicts the individual states of the client, server and system control node in a system in which the reconfiguration framework is present and all the nodes are part of the reconfiguration framework. The same model can also be applied to depict the individual states of the publisher, subscriber and system control node in a system in which the reconfiguration framework is present and all the nodes are part of the reconfiguration framework.

* In each case, we start with the understanding that the given node does not exist at the start, and is spawned. It then goes through its initialization phase and eventually begins its normal operation.
* We portray the states of each node during reconfiguration as well, and use different state variables to portray each individual state of each node.
* A point to note here is that the system control node is triggered to enter reconfiguration mode by the user interface, whereas the server and client (or publisher and subscriber) nodes are triggered to enter the reconfiguration mode by the system control node.
* Certain states, such as “Normal Waiting on Service” is triggered only in the service-client scenario when a service is called, and is not applicable for the publisher-subscriber scenario, as a node is blocked only when a service is called.
* We have depicted sub-states for the Init and Reconfiguration states as these sub-states will always occur and will not affect the overall functionality of the system, and hence can be abstracted for proving the LTL properties on the main model. These sub-states have only been presented to explain the actual implementation within these particular states.
* The “InitState” variable is used as the system control node may already be in reconfiguration mode when a new node is spawned. This case may arise as the reconfiguration required the launch a new node, and in such a case, the new node should start in reconfiguration mode before it starts its normal operations and interactions with other nodes in the system.
* The assumption for this model is that there is a single queue for each node for sending requests or processing requests. This ensures that for a client, all requests sent before the request to enter reconfiguration mode is received, are processed and returned, and for a server, all request received before the request to enter reconfiguration mode is received, are processed.
* For the server and client nodes, the “save state” and “load state” states are present to save the state of the node in case it needs to be restarted and put to its original state if a reconfiguration did not work out correctly and the state of the system needs to be rolled back to its original state. For doing the “save state” and “load state”, the SCN shall only call two services by these names at the required time. The implementation of these services needs to be done by the application node developer.
* For the SCN, the “Processing services to store dependencies” state is when the dependency is stored in the runtime data structure of the SCN. At runtime, the SCN stores all these dependencies in its data structure for future use during reconfiguration. The “Save SCN state to disk” state is used to store the current runtime dependency data to an external storage. This is done so that this information can later be loaded back if the SCN crashes unexpectedly or needs to be restarted due to some unexpected behavior.
* For modeling purposes, we use a Promela model which does interactions between processes using channels. We use the channels to communicate between the different processes (client, server and SCN processes), and send messages such as “enter recon mode request” over these channels.
* An important point to note here is that this framework shall not provide a service to hand off information or data related to state of the node to other nodes or the SCN.
* Only nodes whose isDependent variable is true will enter reconfiguration mode when the request to do some is received from the SCN. This variable is used to model dependency cases.
* An assumption here is that the client node or server node shall not restarted on its own by ROS, and always has its respawn flag set to false.
  1. **Model for Client/Subscriber**



Figure 1: State diagram for client node using service(s) of another node or subscribing to a topic which is published by another node

Figure 1 shows the high-level state model of any node which acts as a client to any service provided by any other node, or as a subscriber to any topic on which information is published by another node.



Figure 2: Sub-states within the “Init” state of a client/subscriber node



Figure 3: Sub-states within the “Reconfiguration” state of a client/subscriber node

* 1. **Model for Server/Publisher**



Figure 4: State diagram for server node providing service(s) to another node, or a publisher node publishing on a topic that is subscribed to by other nodes

Figure 4 shows the high-level state model of any node which acts as a server for any service provided to any other node, or published information on a topic that is subscribed to by other nodes



Figure 5: Sub-states within the “Init” state of a server/publisher node



Figure 6: Sub-states within the “Reconfiguration” state of a server/publisher node

* 1. **Model for System Control Node**



Figure 7: State diagram for System Control Node for a system which includes a client and a server, or a publisher and a subscriber

Figure 7 shows the high-level state model of the system control node for any system using the reconfiguration framework and includes a client node and a server node, or a publisher node and a subscriber node



Figure 8: Sub-states within the “Init” state of SCN node



Figure 9: Sub-states within the “Reconfiguration” state of SCN node

* 1. **LTL Properties**
     1. **Properties related to normal mode**

1) After the init state, a node shall be in normal mode until it receives a request to enter recon mode

2) A node processing a service call shall not enter recon model until it completes processing the service call

3) If SCN node is in normal mode, any other node shall be in normal mode.

* + 1. **Properties related to entering reconfiguration mode**

1) If a node is in reconfiguration mode implies the SCN is in reconfiguration mode.

2) A node providing a service to another node shall not enter recon mode until the other node is in recon mode

* + 1. **Properties related to exiting reconfiguration mode**

1) If a node enters reconfiguration mode, it will eventually either exit reconfiguration mode or not exist.

2) If the SCN is reconfiguration mode, it will eventually exit reconfiguration mode.

3) If the first node is calling a service of the second node, the second node shall always exit reconfiguration mode before the first node

* + 1. **Properties related to interaction between nodes**

1) A node making a service call always implies it is in normal mode

* + 1. **Properties related to dependencies**

1) When a recon request is received, a nodes that has dependencies with respect to the nodes present in the request shall eventually enter recon mode

# **Model for Actions**

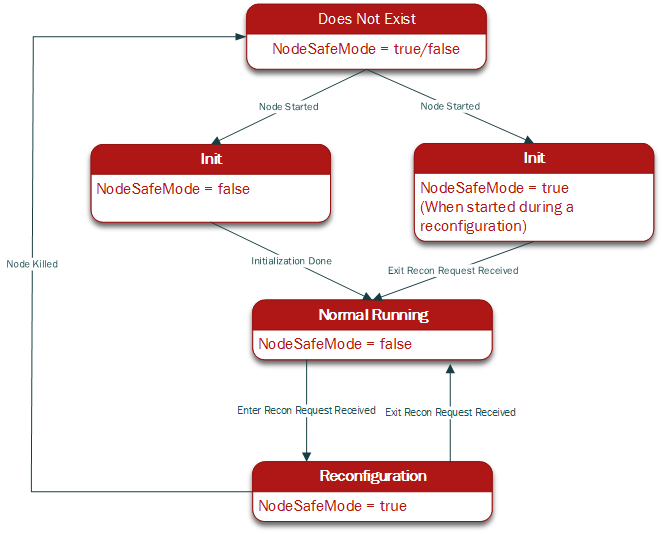


Figure 10: State Diagram of an action node (Action Server/Action Client)

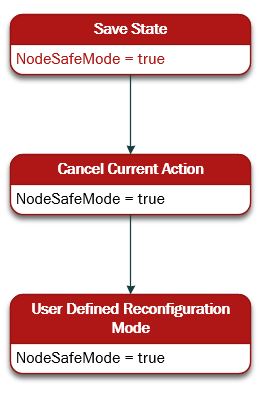


Figure 11: Sub-states within the “Reconfiguration State” for a node which is an Action Client

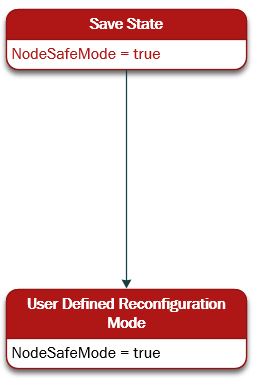


Figure 11: Sub-states within the “Reconfiguration State” for a node which is an Action Server

* 1. **LTL Properties**

1. All nodes shall be in normal mode until they receive a request to enter recon mode
2. A node must cancel action (goal) before entering user defined recon mode
3. A node shall not set action goals when in recon mode
4. If SCN node is in normal mode, all other nodes shall be in normal mode
5. Once a reconfiguration request is received, the system will eventually complete reconfiguration successfully or be rolled back to the state in which it was before the reconfiguration request was received
6. If the first node is requesting a service (action service) from the second node, the first node shall always enter recon mode before the second node
7. When a recon request is received, all nodes that have dependencies with respect to the nodes present in the request shall eventually enter recon mode
8. If the first node requesting a service (action service) from the second node, the second node shall always exit reconfiguration mode before the first node
9. If the SCN node is in reconfiguration mode, all other nodes dependent on the node(s) to be reconfigured shall eventually enter reconfiguration mode.

# **Model Interactions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sender Process | Sender State | Receiver Process | Receiver State | Message |
| Client | Normal running>Normal waiting on service | Server | Normal running>Normal processing | Service called |
| Server | Normal processing>Normal running | Client | Normal waiting on service>Normal running | Service returned |
| SCN | Reconfiguration | Client | Normal running>Reconfiguration | Enter recon mode request |
| SCN | Reconfiguration | Server | Normal running>Reconfiguration | Enter recon mode request |
| SCN | Reconfiguration | Client | Reconfiguration>Normal Running | Exit recon mode request |
| SCN | Reconfiguration | Server | Reconfiguration>Normal Running | Exit recon mode request |
| SCN | Normal running | Client | Normal Running | Set IsDependent |
| SCN | Normal running | Server | Normal Running | Set IsDependent |