Hardware implementation of Preemtable Scheduling Approaches for the Robot Operating System (ROS) via actionlib

Ziyuan Zhang

Task description

- Research state-of-the-art for similar techniques
  - (Preemptable) HW schedulers
  - Real-time HW schedulers
  - VHDL implementations of FSMs
- Become familiar with the Robot Operating System (ROS)
  - Understand the differences among topics, services and actions
  - Find and propose examples for actionlib to be used for proof of concept for one and multiple actions
- Implement Client and Server FSMs on HW (VHDL)
  - Proof of concept for one action
  - Generalize for multiple actions at the same time
    - Are multiple "actionlib" FSMs needed? If so, define the interaction among them
    - Is a general FSM needed to coordinate all "child" FSMs?

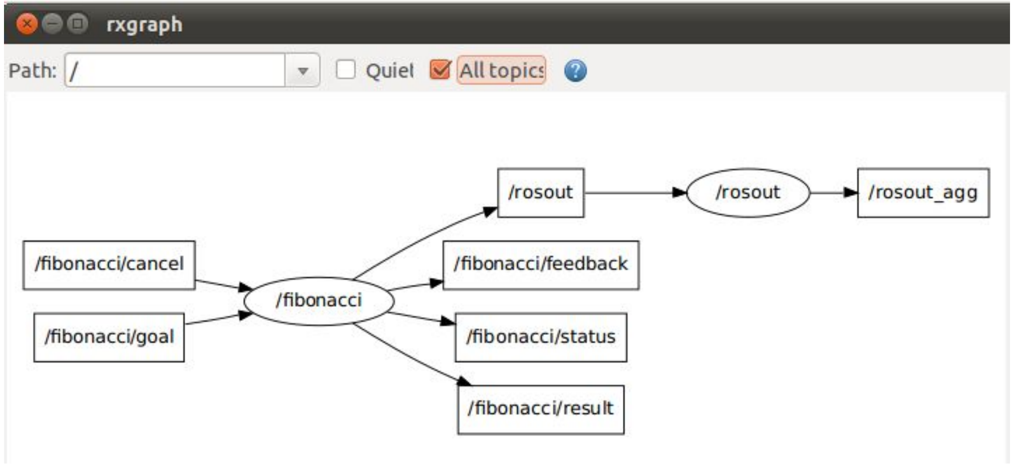
19.4 .2022   3:00 pm

Actionlib and ROS

- Actionlib is a library of Robot Operation System(ROS)
- Actionlib concepts are based on components and concepts of ROS modules.
- Actionlib uses several key concepts of ROS as basic block to build up its function.
  - Node
  - Topic
  - Service

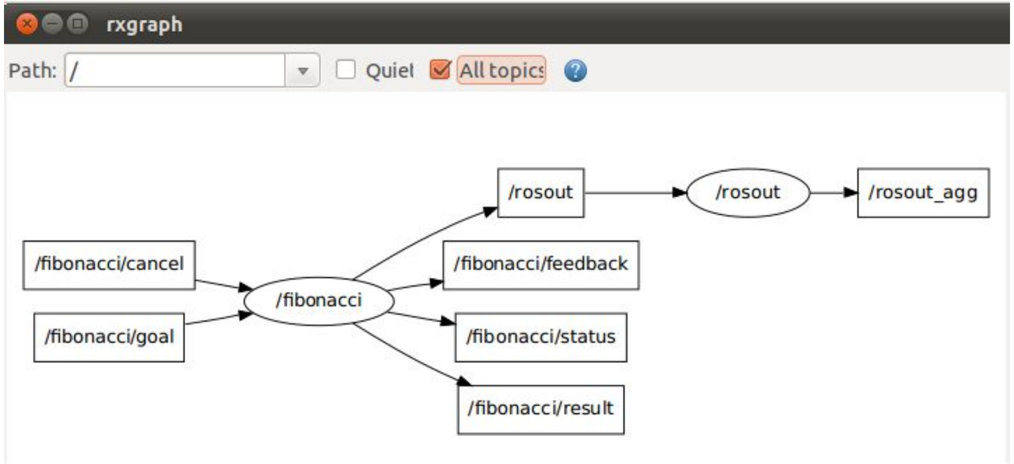
Node

- A node is an executable file within a ROS package.
- HW-Actionlib assumption: Heterogeneous distributed system



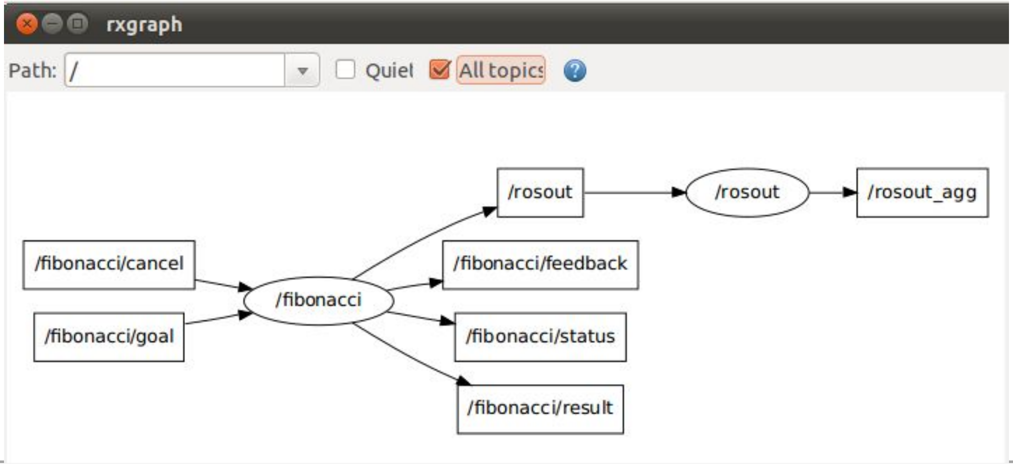
Topic

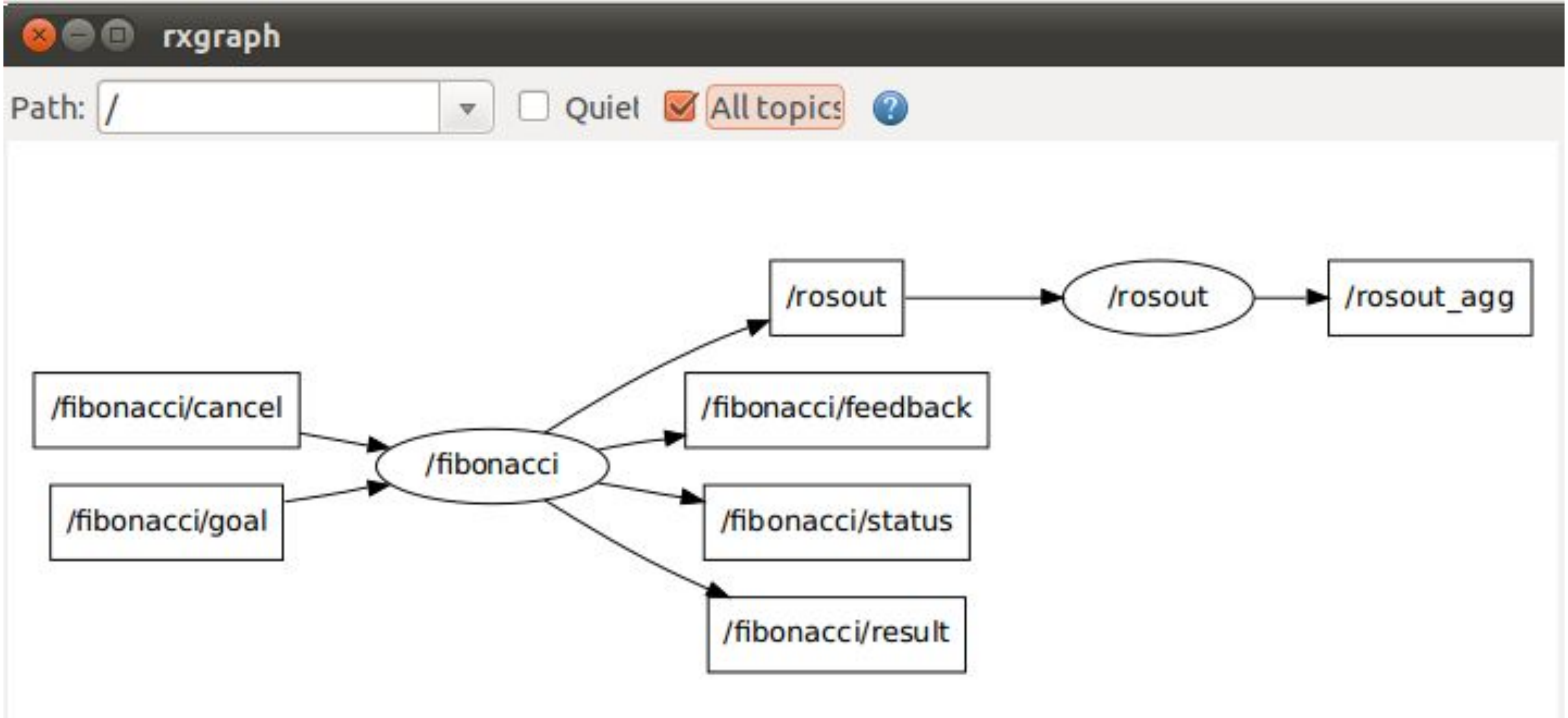
- ROS: Topics are used by nodes for many-to-many one-way communication.
- HW-Actionlib: Many-to-one one-way communication.
- ROS: use TCPROS for communication.
- HW-Actionlib: use AXIS for communication.



Service

- Service is an another mechanism provided by ROS for communication between nodes. Service is defined by two messages: one for the request and one for the reply.





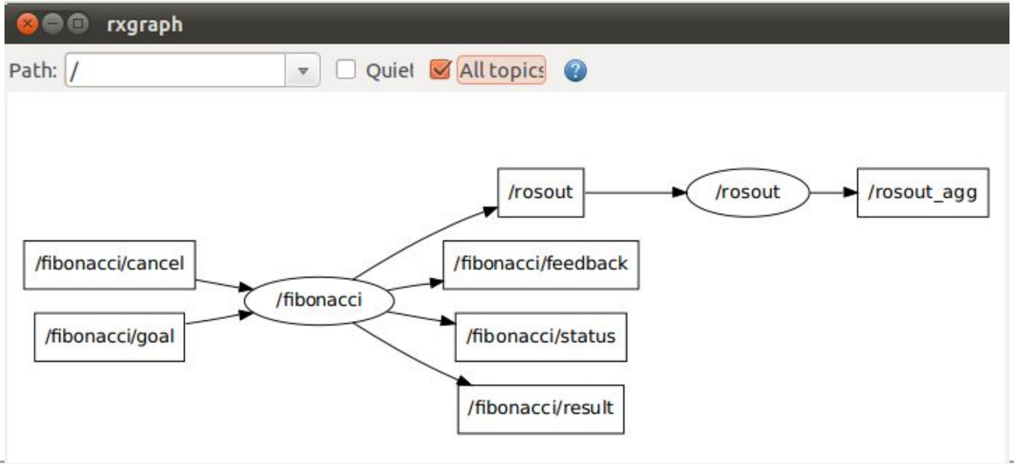
Path: /

□ Quiel

☑ All topics

Actionlib

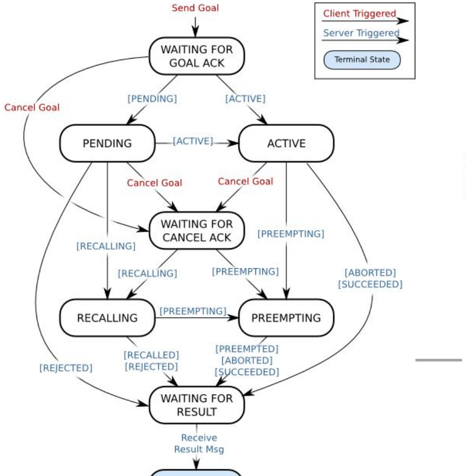
- The ability to **cancel the request** during execution or get **periodic feedback** about how the request is progressing.
- **Framework** for managing preemptable tasks.



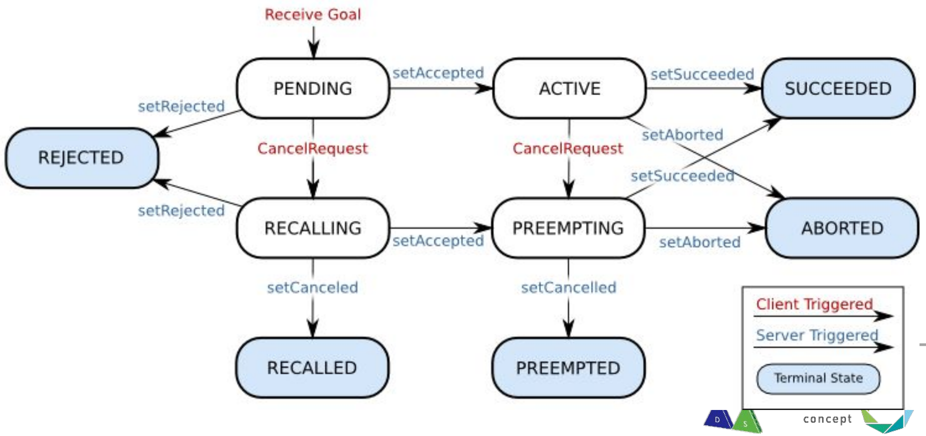
Understanding Actions in Actionlib Library

- Actionlib concepts ware designed for software, and will be slightly modified in hardware implementation.
- An Action is defined by **5 ROS Messages**(nested datatype/struct/class).
- An action progress is descripted by client and server **states machine**

Client State Transitions



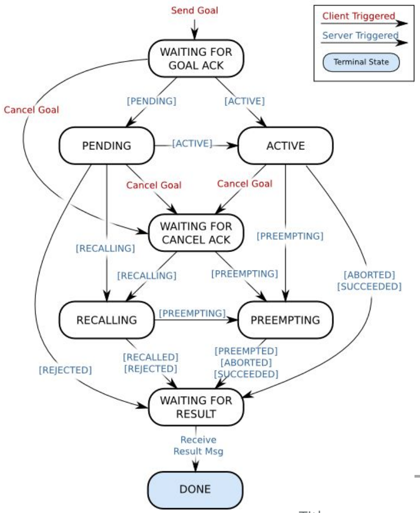
Server State Transitions



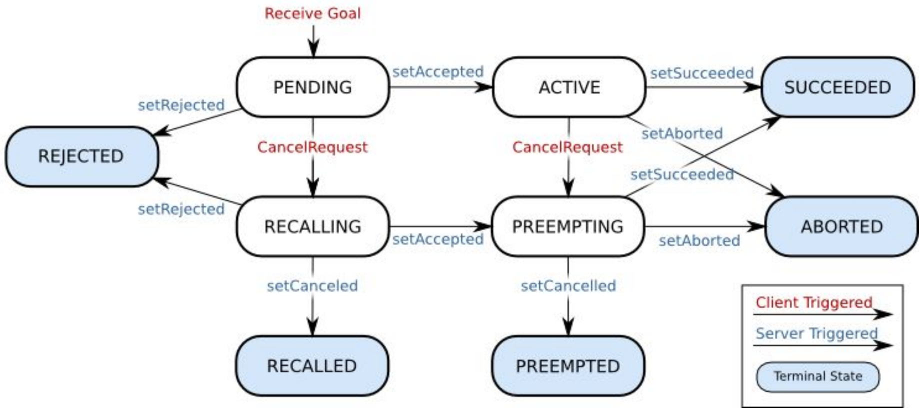
Understanding Actions in Actionlib Library

- Actionlib user should implement the client and server interface using given states to control the running action process.
- Server keeps a queue to hold all goals(tasks/requests).

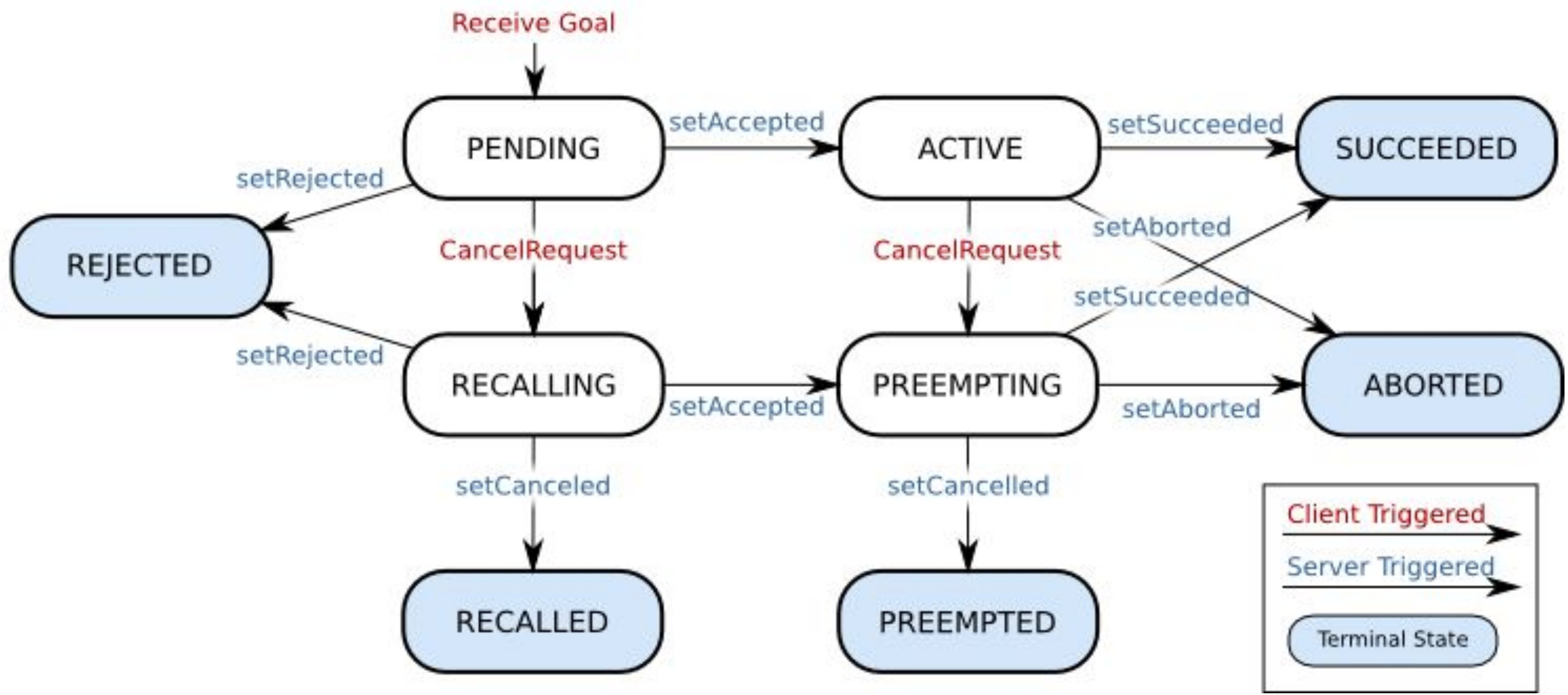
Client State Transitions



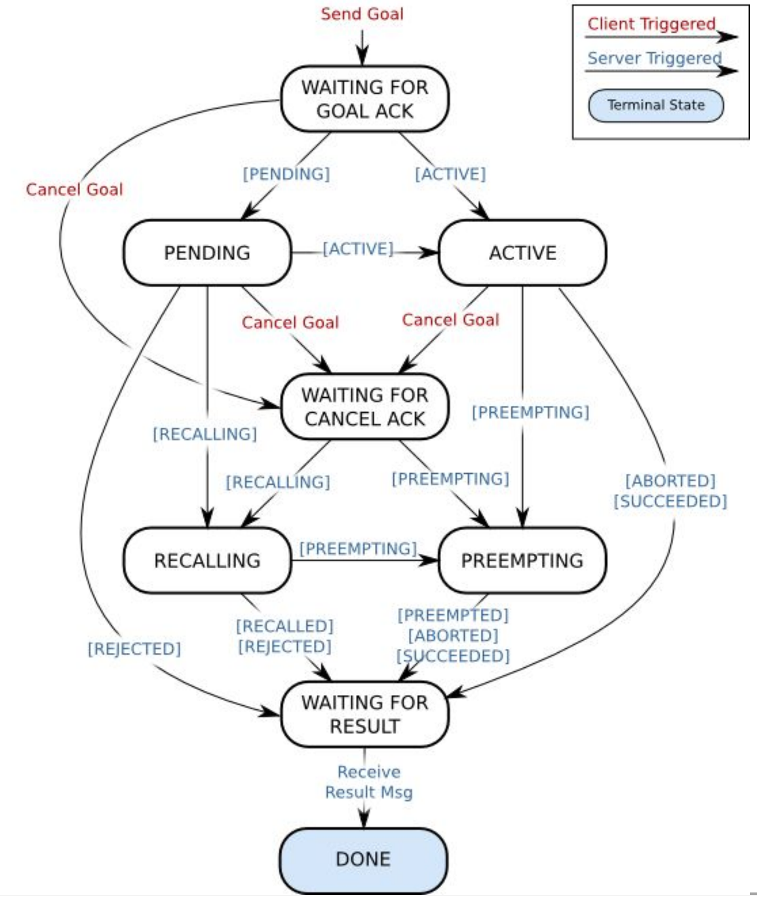
Server State Transitions



Server State Transitions



Client State Transitions



Actionlib and Difference Between Topics and Services

- Actionlib is 5 Topics + some code.
  - Without the library, user can achieve the same function using Topics/Service.
- Actionlib behaves like Service: one server and multiple clients, but provide 4 more connections for cancel and periodic feedback.
- Using Actionlib makes the management for preemptable tasks easier.
  - Easy to coordinate data structure(msg) for communication
  - Easy to place code for preemption handle according to the state machine.

Multiple Actions Extension

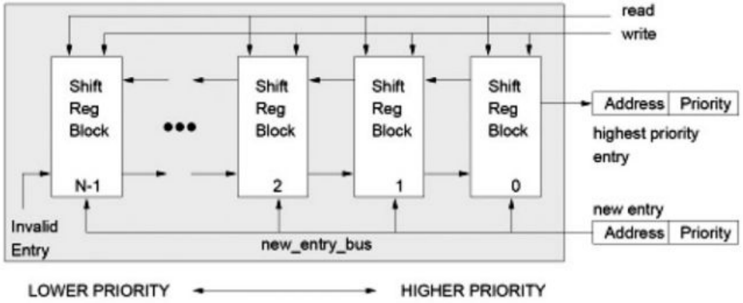
- In Service and Actionlib, server is designed to receive one data structure and **serve one purposes**.
- In order to extend the system to support multiple actions, the most easy way is to keep the one purpose assumption and
- Simply **use multiple servers**.

Multiple Actions Extension

- Multiple server bring us scheduler and **multiprocessor** problems.
- (In the Actionlib concept and software design)Actionlib user should handle the scheduler and multiprocessor problems.
- Multiprocessor scheduler design is much harder. Most of the papers are for cloud computing and almost impossible for HW implementation.

History of Hardware Scheduler

- Scheduler for Network
  - HW-scheduler are mostly designed for network.
- Real-time Processors
  - HW-scheduler can reduce interrupt and context switch
  - Increase stability for read-time tasks
- Single Processor
  - Priority queue based on Shift register.



HW-Actionlib Concept

Software

- Interface of client/server nodes.
- Goal list each action server.
- User handle all state.
- Transmission using software library: Topic, TCPROS, Mutex.

Hardware

- Centralized task scheduler.
- Global goal list.
- Scheduler handle Recalling.
- Hardware AXIS based transmission.

ROS Structure

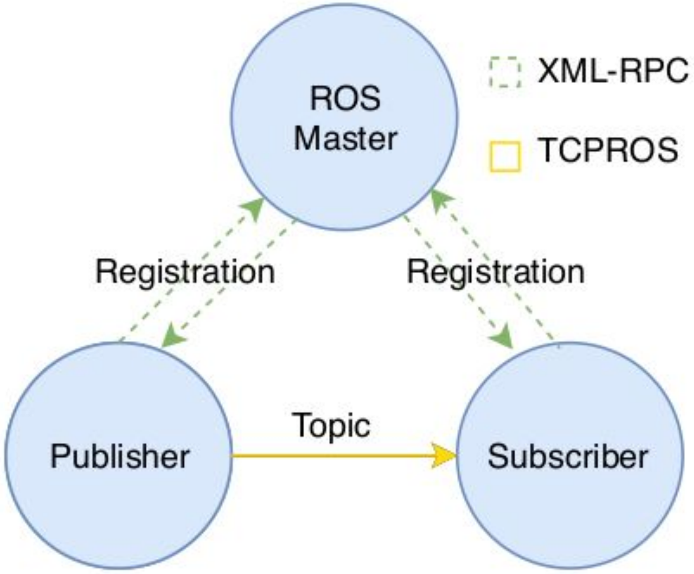
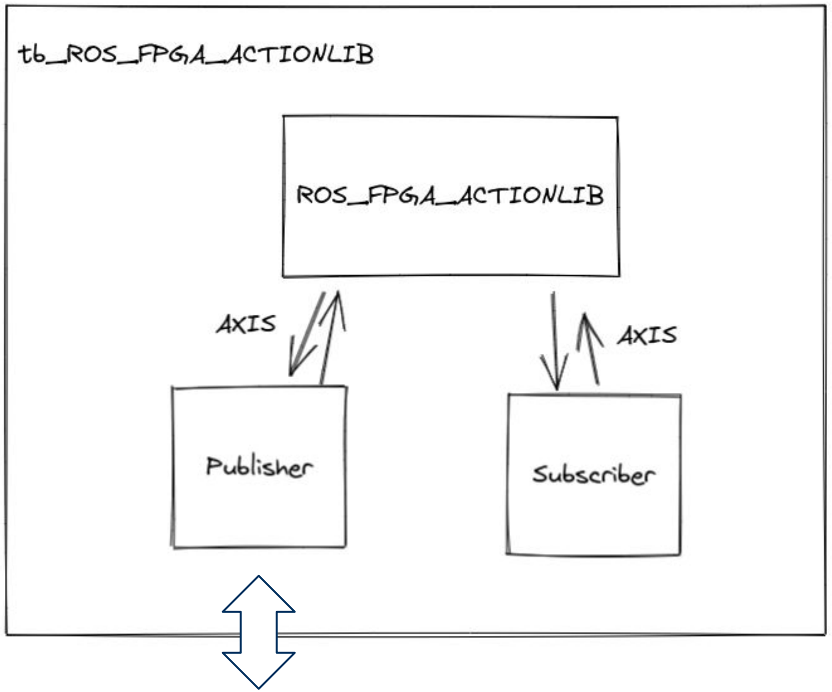


Fig. 1: Basic ROS architecture.



[1] A. Podlubne and D. Göhringer, ''FPGA-ROS: Methodology to augment the robot operating system with FPGA designs,'' in Proc. Int. Conf. ReConFigurable Comput. FPGAs (ReConFig), Dec. 2019, pp. 1–5.

Scheduler only cares if the c/s is ready for task.

- The client is interested in the difference between server states in the same group.
- HW-Actionlib only need to deliver “Active”, ”Preempting ” and ”Aborting” to clients.
- HW-Actionlib need to deal with “recalling” state.

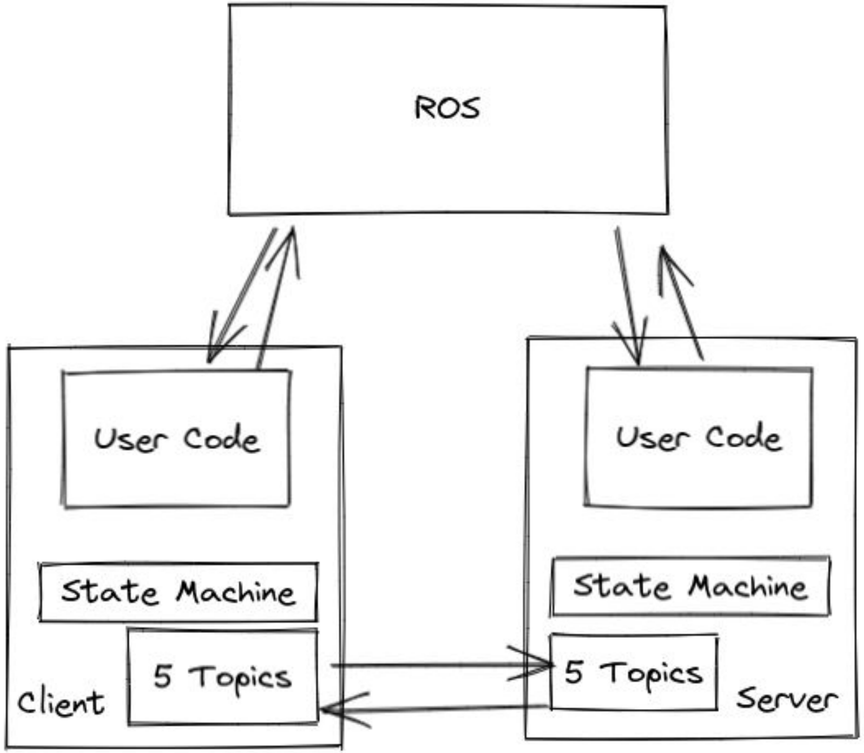
VHDL

- Once the FSM is drawn, it's easy to transform to hardware design with VHDL.

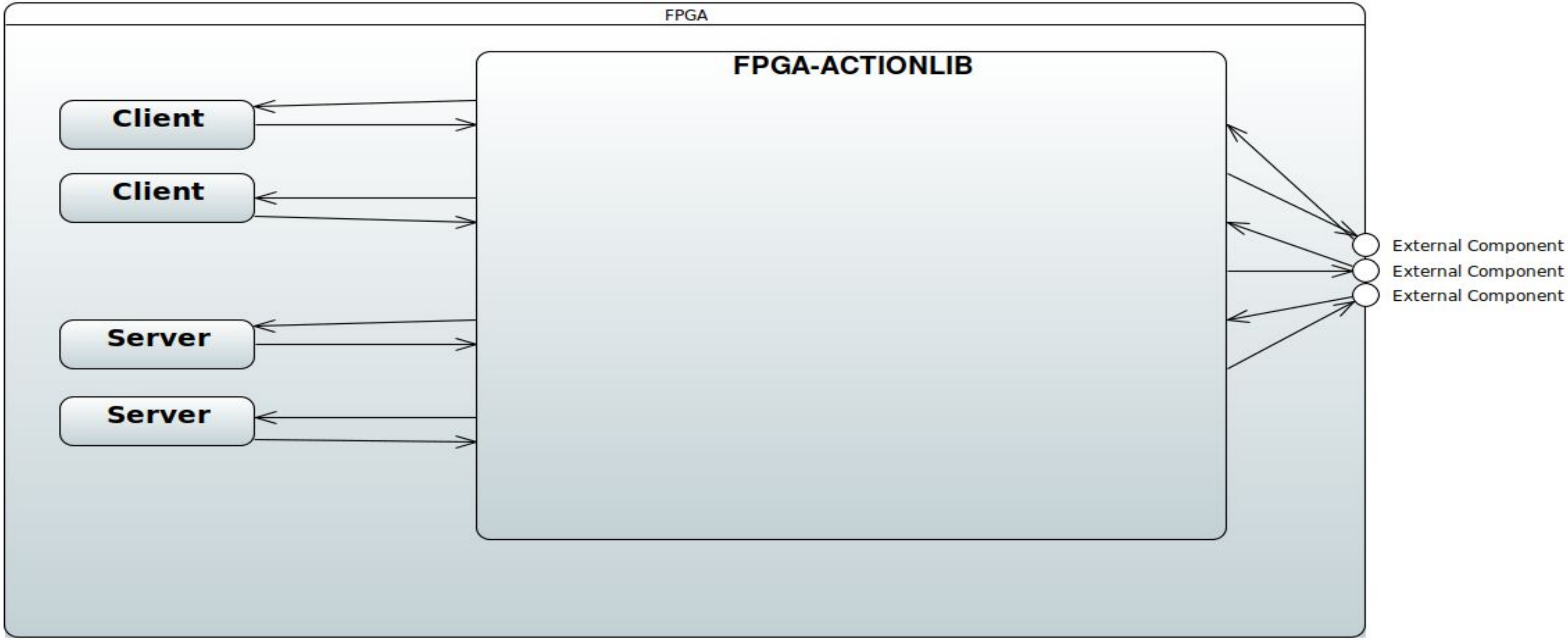
Actionlib Software Structure

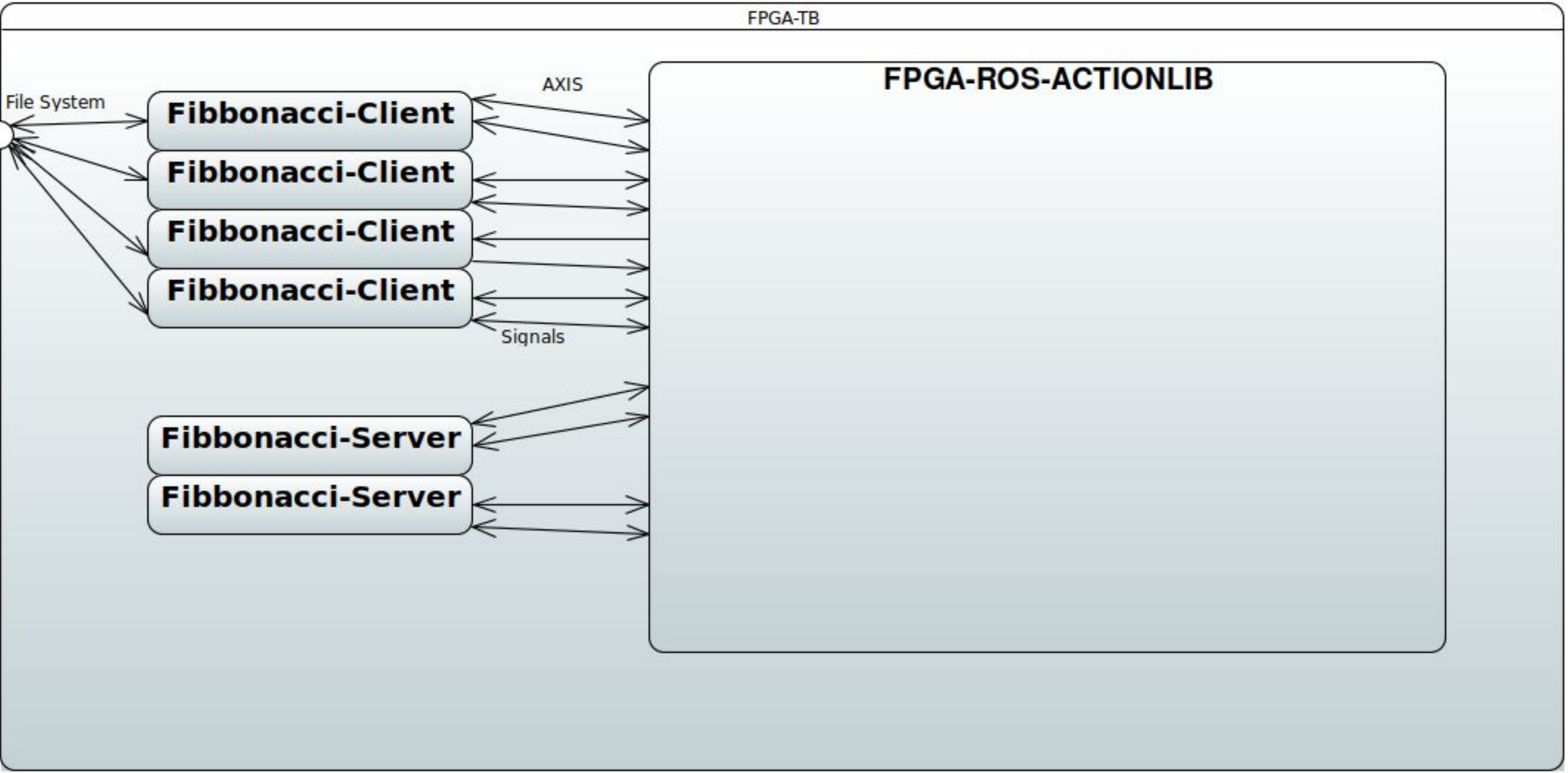
- User write code in the callbacks provided by Actionlib framework.

- Scheduler and control are designed by actionlib user.



Hardware Architecture

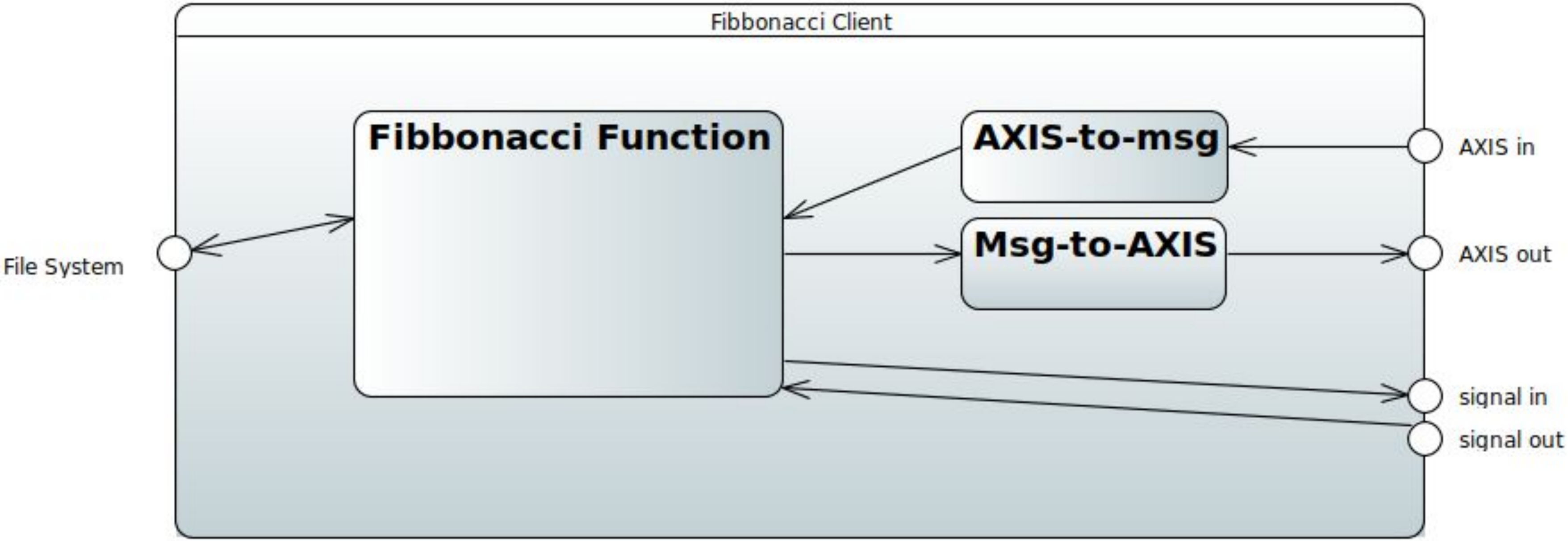



FPGA-TB
FPGA-ROS-ACTIONLIB
File System
AXIS
Fibbonacci-Client
Fibbonacci-Client
Fibbonacci-Client
Fibbonacci-Client
Signals
Fibbonacci-Server
Fibbonacci-Server

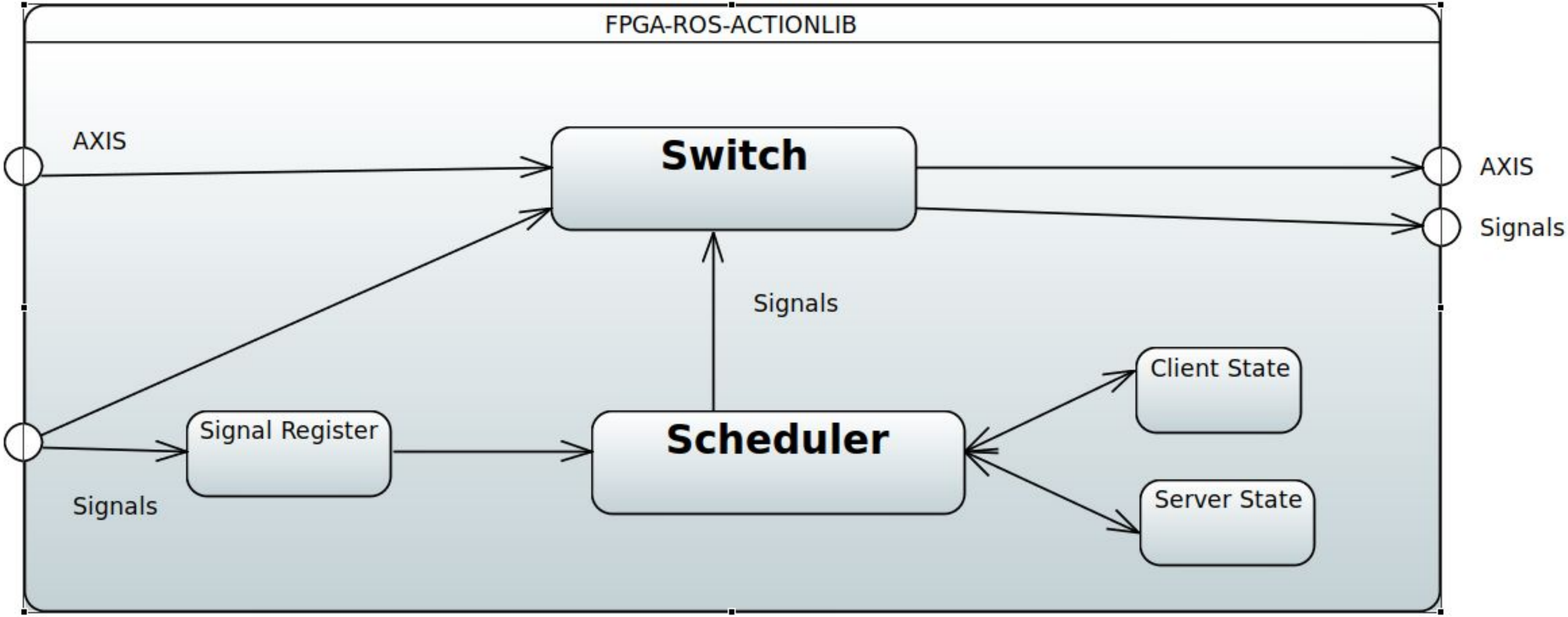
Components

- Test clients, Test servers
  - AXIS to message
- FPGA-ACTIONLIB
  - Client State Machine Vector
  - Server State Machine Vector
  - Signal Register
  - Switcher
  - Scheduler

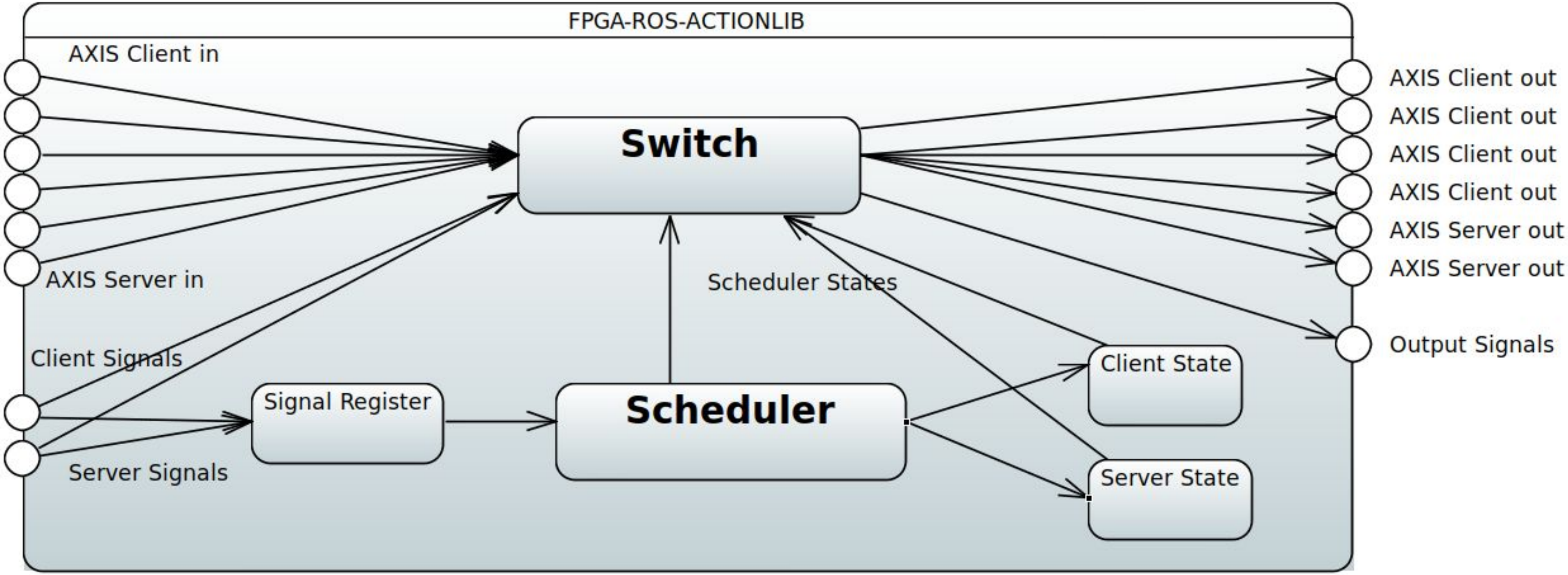
Hardware Architecture



Hardware Architecture



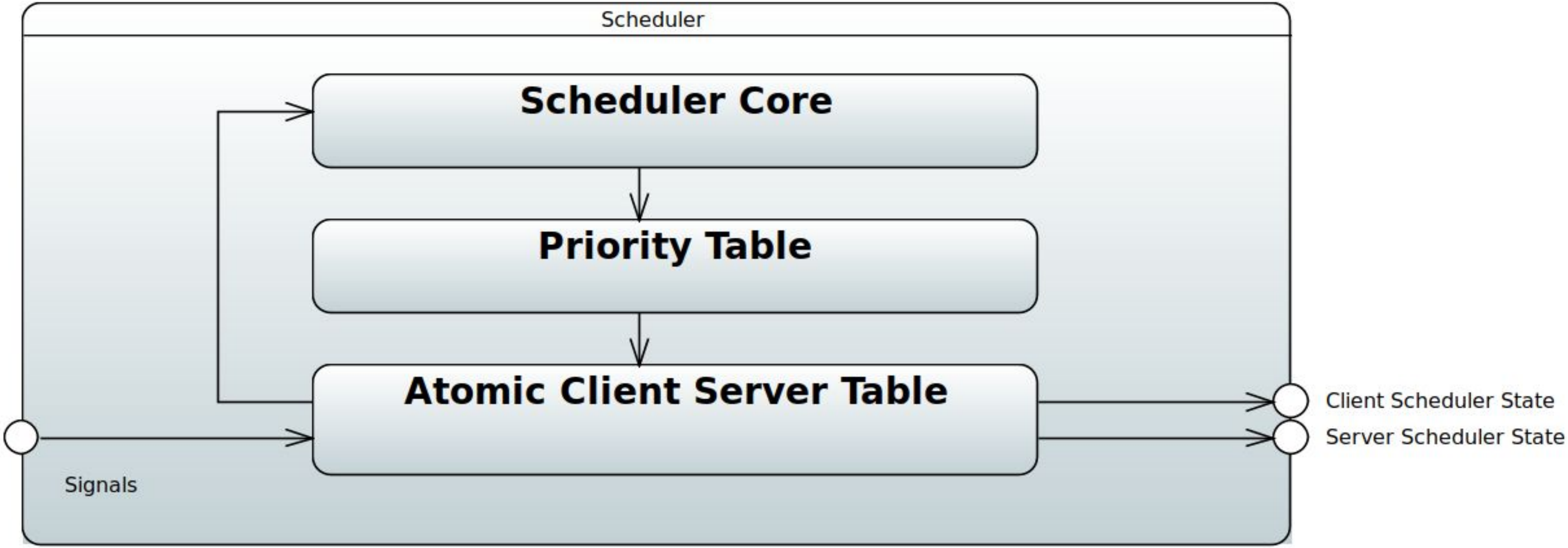
Hardware Architecture



Components

- Scheduler
    - Atomic Dual Register
    - Priority Table
    - Scheduler Core
        - First Come First Served
        - Earliest Deadline First
        - Simple Client Simple Server Assumption

Hardware Architecture



Atomic Dual Register

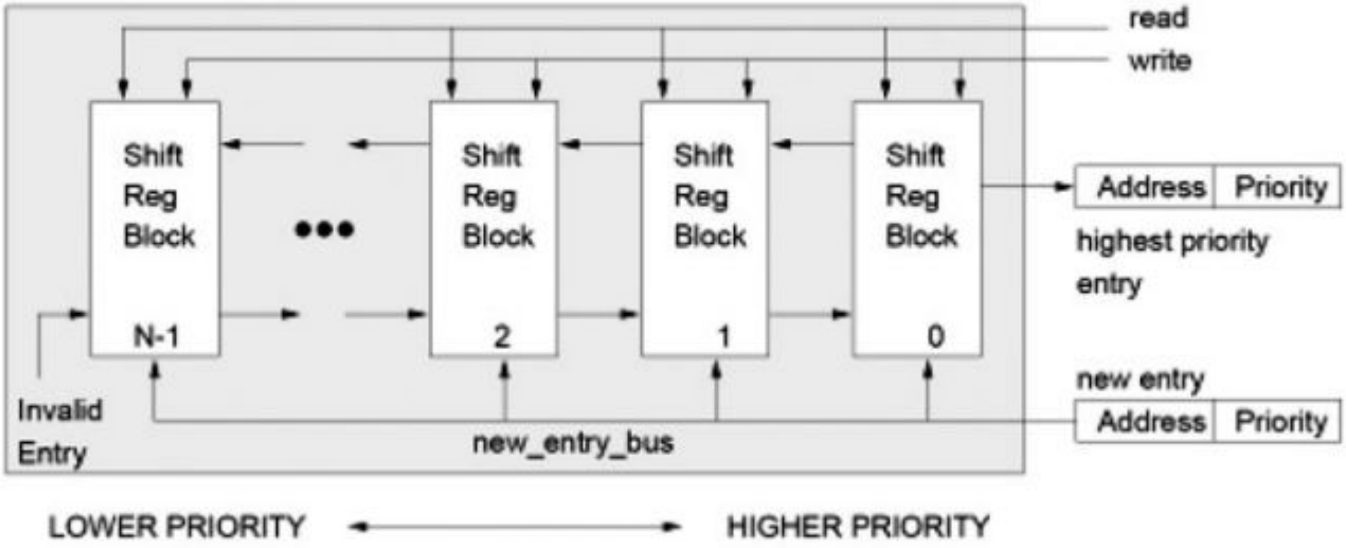
Input: Client, Server, WriteSignal, DeleteSignal.

Output: Client Active, Server, Server Active, Serving Client.

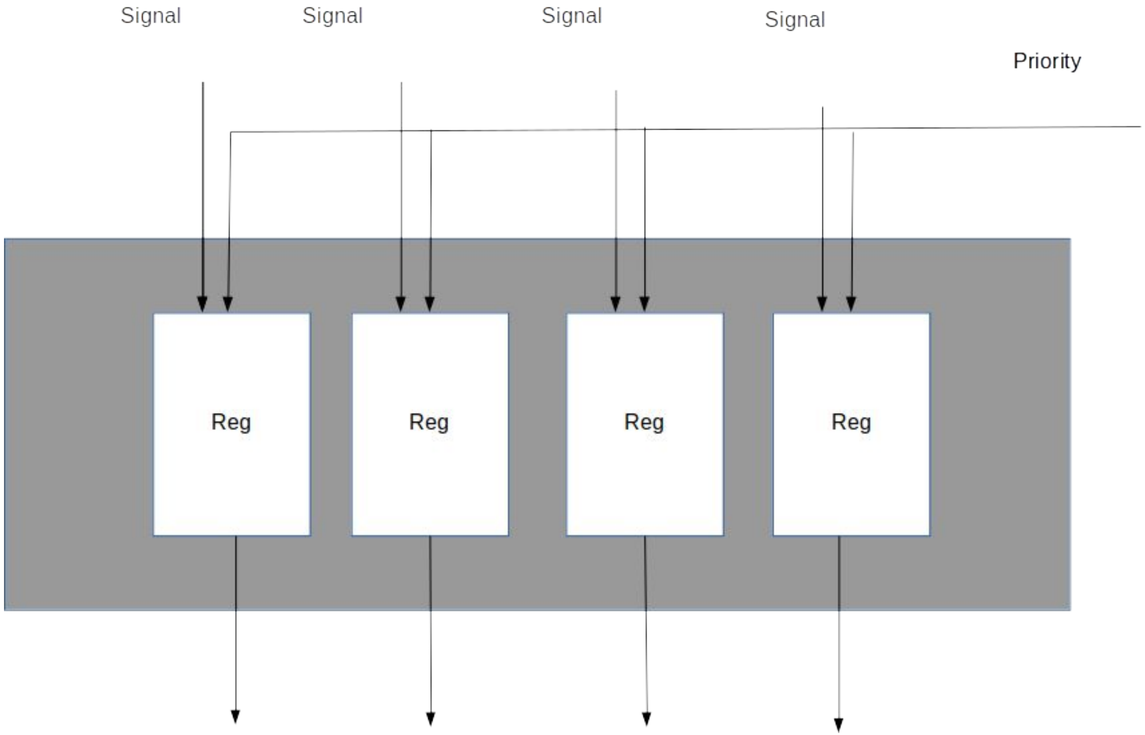
Example for 4 Clients and 2 Servers

Client Active	Server	Server Active	Serving Client
0	U	0	U
0	U	1	3
0	U		
1	1		

Priority Table: Shift Queue



Priority Table: Client list

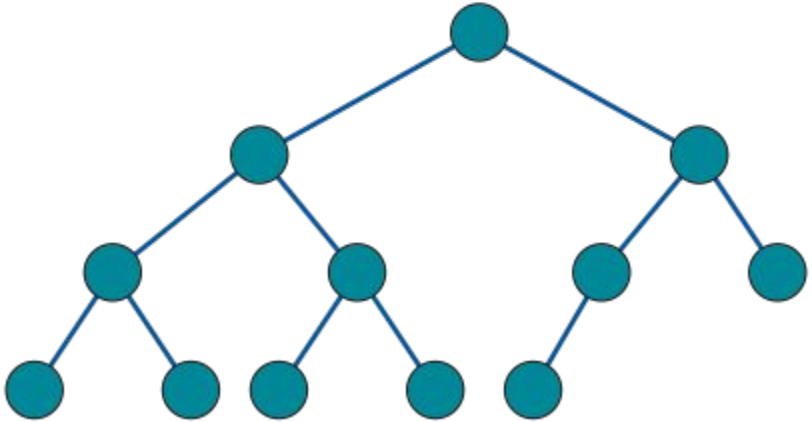


Assistant Modules

- Max
- Min
- Multiplexer
- One-hot to unsigned

Max/Min

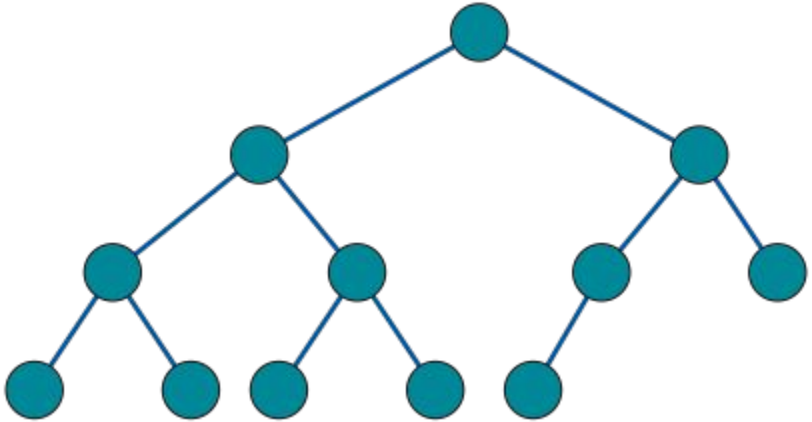
- Input: list of unsigned, Output; unsigned and one-hot list
- Example
- Input [1,3,5,2,6,4,7,0]
- Output 7, [0,0,0,0,0,0,1,0]



https://www.wikiwand.com/en/Binary_tree

One-hot to unsigned

- Input: one-hot list, Output; unsigned
- Example
- Input [0,0,0,0,0,0,1,0]
- Output 6



https://www.wikiwand.com/en/Binary_tree

One-shot schedulers

- Least Recently Used (LRU) / Most recently used (MRU)
- Least-frequently used (LFU)
- First Come First Served

Execution time

- Shortest Job(Processing Time) First (SJF)

Deadline

- Earliest Deadline First
- Least Slack Time

What next: Hardware Experiments and Metrics

- **Makespan**: the time required to execute all tasks.
- **Computation cost**: execution time of an algorithm.
- **Due date based**: Lateness should be minimum.
- **Deadline:** Task executed befor

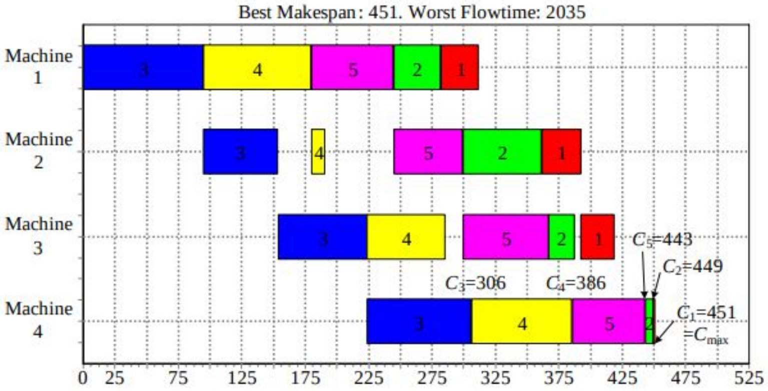


Image: Handbook of Heuristics (pp.1-24) Authors: Rubén Ruiz Universitat Politècnica de València

What next: Data Dependency (presented by DAG)

- To provide an easy way to add “Callbacks”.

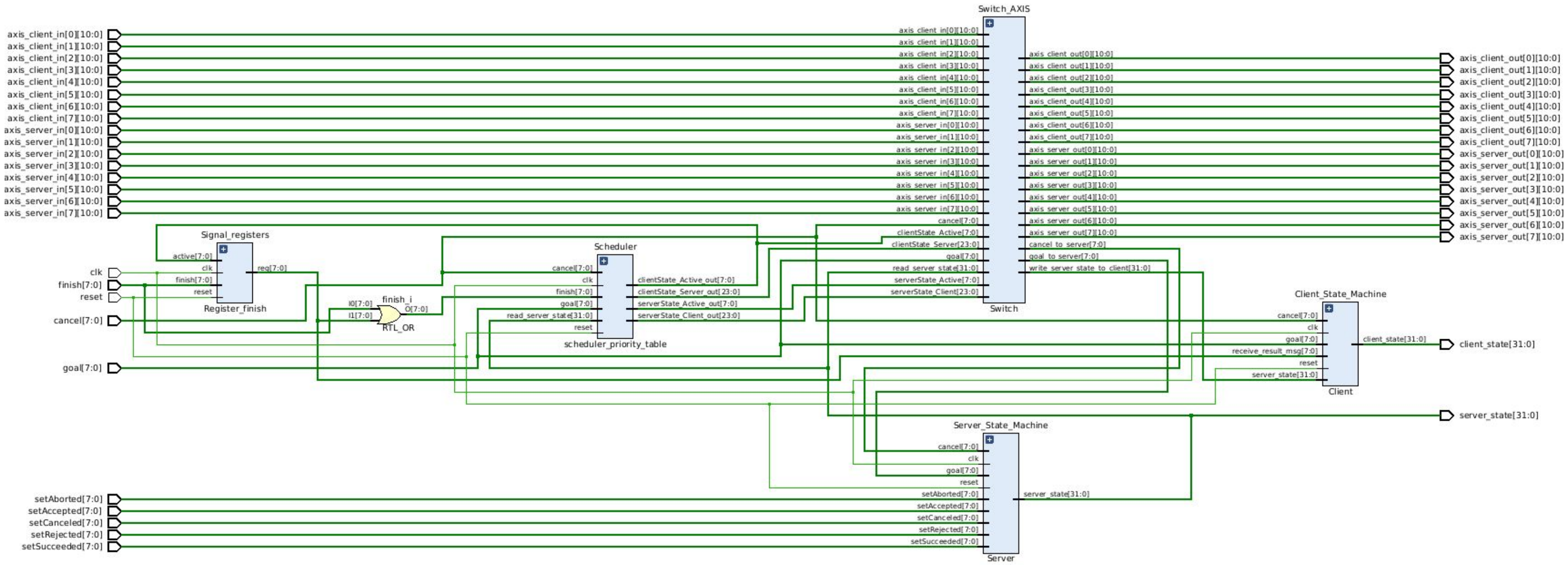
What next: Complex One-shot heuristic schedulers

Data Dependency

- Min-min[1]
- Chaining[2]
- HLFET[3]

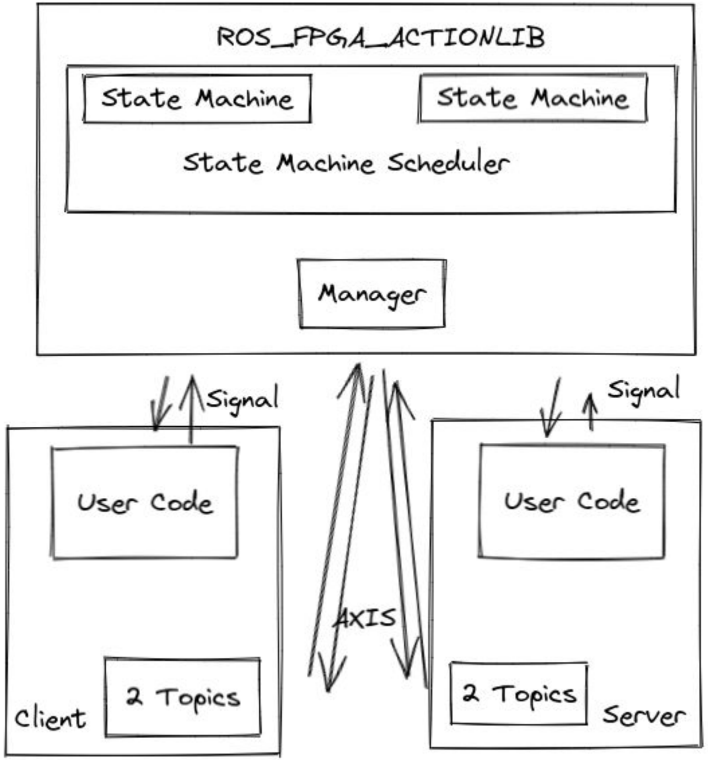
[1] Ibarra O, Kim C (1977) Heuristic algorithms for scheduling independent tasks on non-identical processors. J Assoc Comput Mach 24(2):280–289
[2] Djordjevic G, Tosic M (1996) A heuristic for scheduling task graphs with communication delays onto multiprocessors. Parallel Comput 22(9):1197–1214
[3] Adam T, Chandy K, Dickson J (1974) A comparison of list schedules for parallel processing systems. ACM Commun 17:685–690

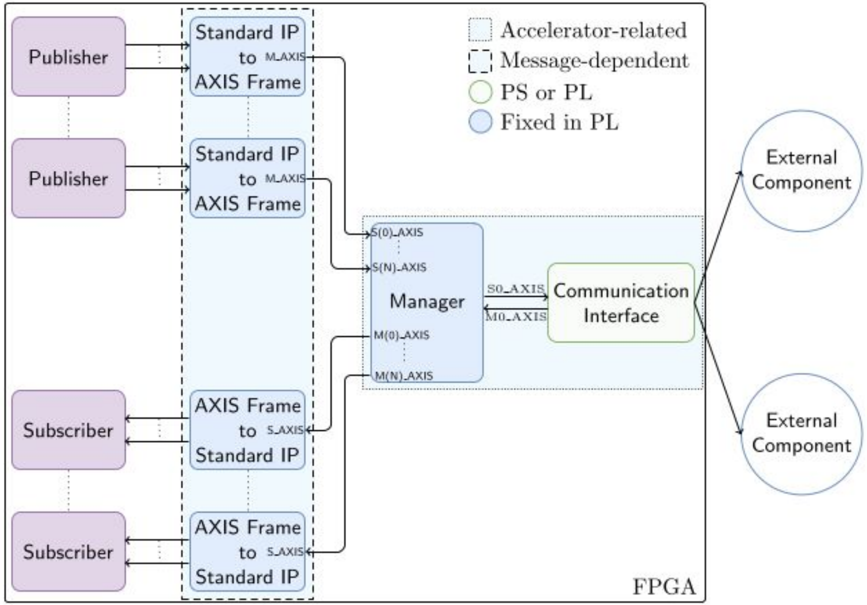
Schematic



axis_client_in[0][10:0]
axis_client_in[1][10:0]
axis_client_in[2][10:0]
axis_client_in[3][10:0]
axis_client_in[4][10:0]
axis_client_in[5][10:0]
axis_client_in[6][10:0]
axis_client_in[7][10:0]
axis_server_in[0][10:0]
axis_server_in[1][10:0]
axis_server_in[2][10:0]
axis_server_in[3][10:0]
axis_server_in[4][10:0]
axis_server_in[5][10:0]
axis_server_in[6][10:0]
axis_server_in[7][10:0]
clk
finish[7:0]
reset
cancel[7:0]
goal[7:0]
setAborted[7:0]
setAccepted[7:0]
setCanceled[7:0]
setRejected[7:0]
setSucceeded[7:0]
Signal_registers
active[7:0]
clk
finish[7:0]
reset
req[7:0]
Register_finish
finish_i
I0[7:0]
I1[7:0]
O[7:0]
RTL_OR
Scheduler
cancel[7:0]
clk
finish[7:0]
goal[7:0]
read_server_state[31:0]
reset
clientState_Active_out[7:0]
clientState_Server_out[23:0]
serverState_Active_out[7:0]
serverState_Client_out[23:0]
scheduler_priority_table
Switch_AXIS
axis_client_in[0][10:0]
axis_client_in[1][10:0]
axis_client_in[2][10:0]
axis_client_in[3][10:0]
axis_client_in[4][10:0]
axis_client_in[5][10:0]
axis_client_in[6][10:0]
axis_client_in[7][10:0]
axis_server_in[0][10:0]
axis_server_in[1][10:0]
axis_server_in[2][10:0]
axis_server_in[3][10:0]
axis_server_in[4][10:0]
axis_server_in[5][10:0]
axis_server_in[6][10:0]
axis_server_in[7][10:0]
cancel[7:0]
clientState_Active[7:0]
clientState_Server[23:0]
goal[7:0]
read_server_state[31:0]
serverState_Active[7:0]
serverState_Client[23:0]
axis_client_out[0][10:0]
axis_client_out[1][10:0]
axis_client_out[2][10:0]
axis_client_out[3][10:0]
axis_client_out[4][10:0]
axis_client_out[5][10:0]
axis_client_out[6][10:0]
axis_client_out[7][10:0]
axis_server_out[0][10:0]
axis_server_out[1][10:0]
axis_server_out[2][10:0]
axis_server_out[3][10:0]
axis_server_out[4][10:0]
axis_server_out[5][10:0]
axis_server_out[6][10:0]
axis_server_out[7][10:0]
cancel_to_server[7:0]
goal_to_server[7:0]
write_server_state_to_client[31:0]
Switch
Client_State_Machine
cancel[7:0]
clk
goal[7:0]
receive_result_msg[7:0]
reset
server_state[31:0]
client_state[31:0]
Client
Server_State_Machine
cancel[7:0]
clk
goal[7:0]
reset
setAborted[7:0]
setAccepted[7:0]
setCanceled[7:0]
setRejected[7:0]
setSucceeded[7:0]
server_state[31:0]
Server
axis_client_out[0][10:0]
axis_client_out[1][10:0]
axis_client_out[2][10:0]
axis_client_out[3][10:0]
axis_client_out[4][10:0]
axis_client_out[5][10:0]
axis_client_out[6][10:0]
axis_client_out[7][10:0]
axis_server_out[0][10:0]
axis_server_out[1][10:0]
axis_server_out[2][10:0]
axis_server_out[3][10:0]
axis_server_out[4][10:0]
axis_server_out[5][10:0]
axis_server_out[6][10:0]
axis_server_out[7][10:0]
client_state[31:0]
server_state[31:0]

A task scheduler for ROS Structure



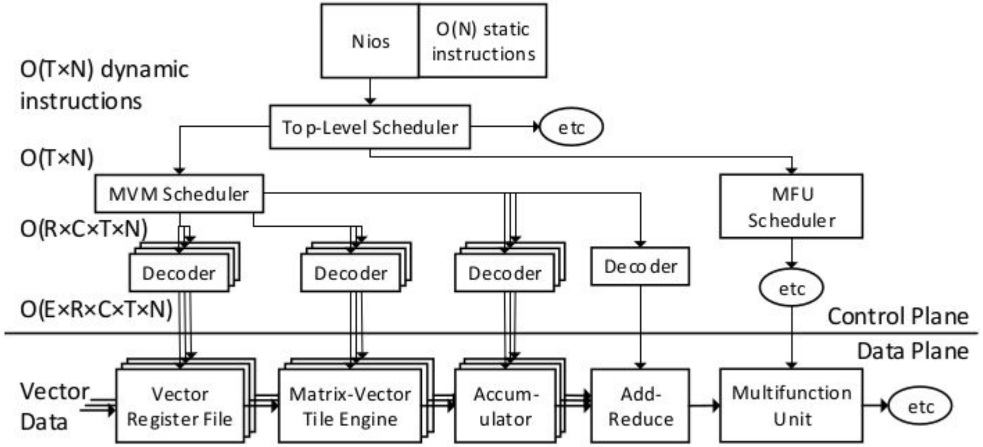


[1] A. Podlubne and D. Göhringer, “FPGA-ROS: Methodology to augment the robot operating system with FPGA designs,” in Proc. Int. Conf. ReConFigurable Comput. FPGAs (ReConFig), Dec. 2019, pp. 1–5.

Thank You

Brainwave - A Configurable Cloud-Scale DNN Processor for Real-Time AI

- Microsoft Project, ISCA18
- Intel Stratix 10 280 FPGA
- Neural Processing Unit (NPU) architectures is taking place (instead of GPGPU)
- BW NPU is able to extract sufficient SIMD and pipeline parallelism to provide high utilization from individual requests.



Callbacks and DAG

- Immediate mode: one node DAG.
- If you treat the c/s as monolithic, then actionlib states are not interested for scheduler design.
- Actionlib states is interested for client / server designer.
- Actionlib SW is a framework of c/s designs (scheduler is left to programmer, or just use its subclass "simpleActionServer").
-
- Callbacks are designed for c/s designers to put the code to react signals. we can treat them as independent servers with data dependencies.
-
-
-

Iterative Search Schedulers

Most scheduling problems belong to the NP-Hard class of computational problems (Brucker, 2007)

- genetic algorithms[1]
- simulated annealing[2]
- tabu search[3]
- A∗[4]

[1]Russell S, Norvig P (2003) Artificial intelligence, a modern approach. Pearson Education, Ch 5, pp 139–172
[2]Hou E, Ansari N, Ren H (1994) A genetic algorithm for multiprocessor scheduling. IEEE Trans Parallel Distrib Syst 5(2):113–120
[3]Chamberlain R, Edelman M, Franklin M, Witte E (1988) Simulated annealing on a multiprocessor. In: Proceedings of the 1988 IEEE international conferences on computer design: VLSI in computers and processors, pp 540–544
[4]Tian Y, Sannomiya N, Xu Y (2000) A tabu search with a new neighborhood search technique applied to flow shop scheduling problems. In: Proceedings of the 39th IEEE conference on decision and control, vol 5, pp 4606–4611