Adaptive Fault Tolerance on ROS: A Component-Based Approach

Jean-Charles Fabre, Michael Lauer, Matthieu Amy

LAAS-CNRS, Ave du Colonel Roche, F-31400 Toulouse, France

Definitions

Dependability: Ability to provide **services** that can defensibly be **trusted** within a time-period.

1. <u>Prevention</u>: means to prevent the occurrence or introduction of faults

PREV

2. <u>Removal</u>: means to reduce the number and severity of faults

REM

3. <u>Forecasting</u>: means to estimate the present number, the future incidence, and the likely consequences of faults.

FOR

4. <u>Tolerance</u>: means to avoid service failures in the presence of faults.

TOL

Definitions

Dependability: Ability to provide **services** that can defensibly be **trusted** within a time-period.

Fault Tolerance (FT) : Design and implementation of **mechanisms** to control **errors** (residual, random, systematic…) by detecting them and ensuring transitions to a **safe state**

Resiliency: The persistence of **dependability** when facing **changes**

Adaptive Fault Tolerance (AFT): Design and implementation of **Fault Tolerant Mechanisms (FTM)** to ensure the **dependability** of the system at **runtime** when facing changes

Problem statement and key concepts

Once the system is deployed, it faces changes.

System designers cannot predict everything.

Persistence of dependability requires the adaptation of safety mechanism

Problem statement and key concepts

Once the system is deployed, it faces changes.

System designers cannot predict everything.

Persistence of dependability requires the adaptation of safety mechanism

Key concepts for Adaptive Fault Tolerance

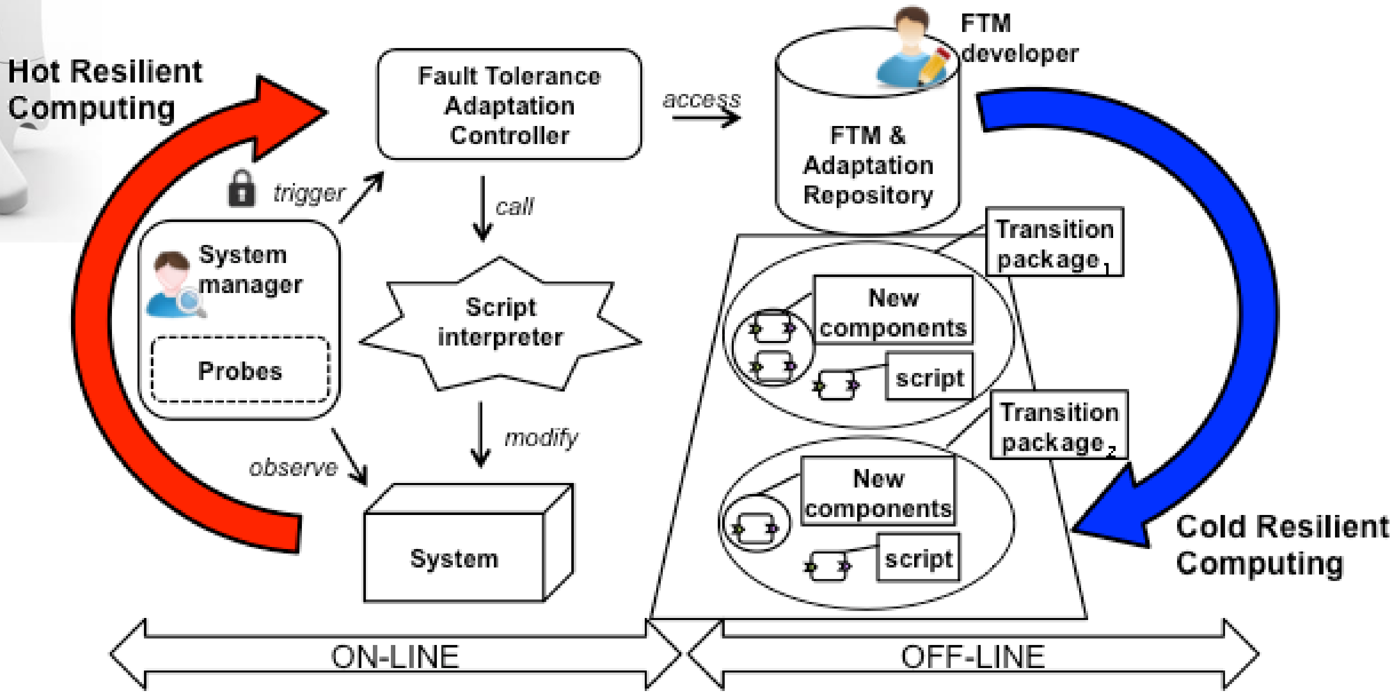
- Separation of concerns
- Design for adaptation
- Remote fine-grained updates

Overall process

FTM as a Lego system

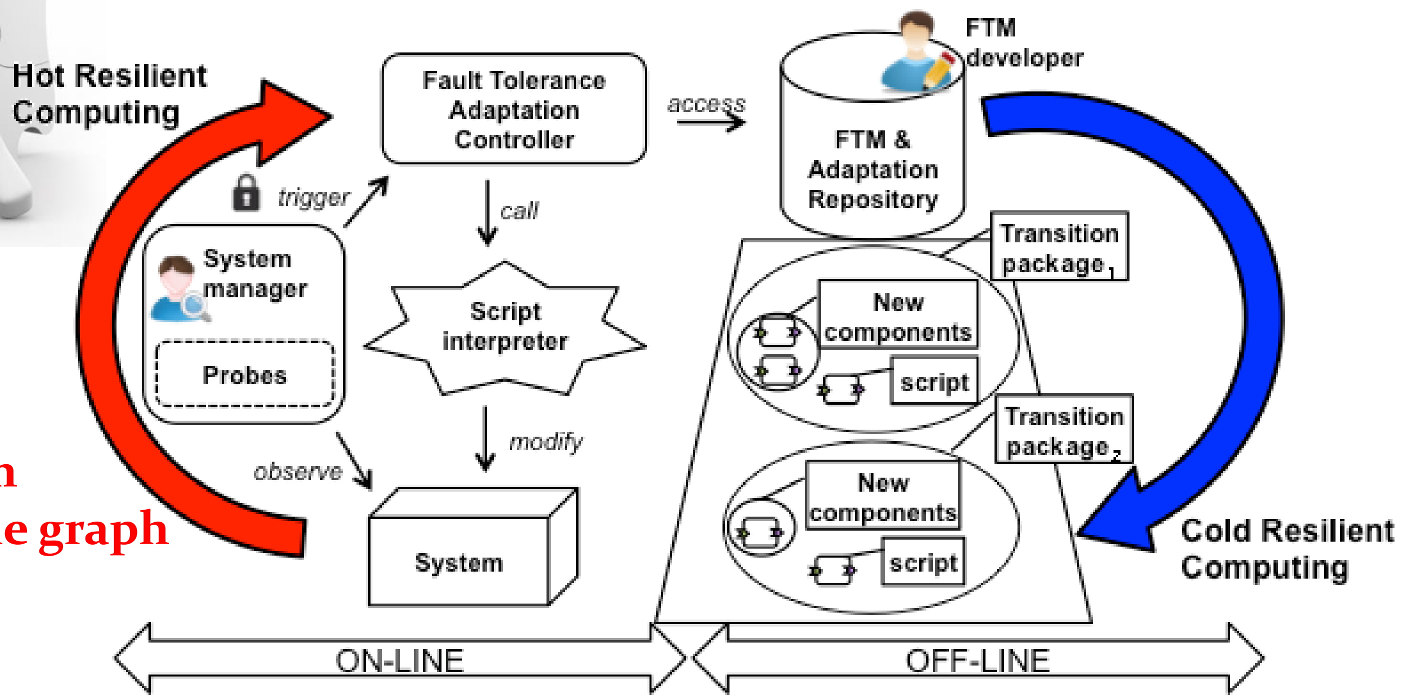
Overall process

FTM as a Lego system



Overall process

FTM as a Lego system



Remote update

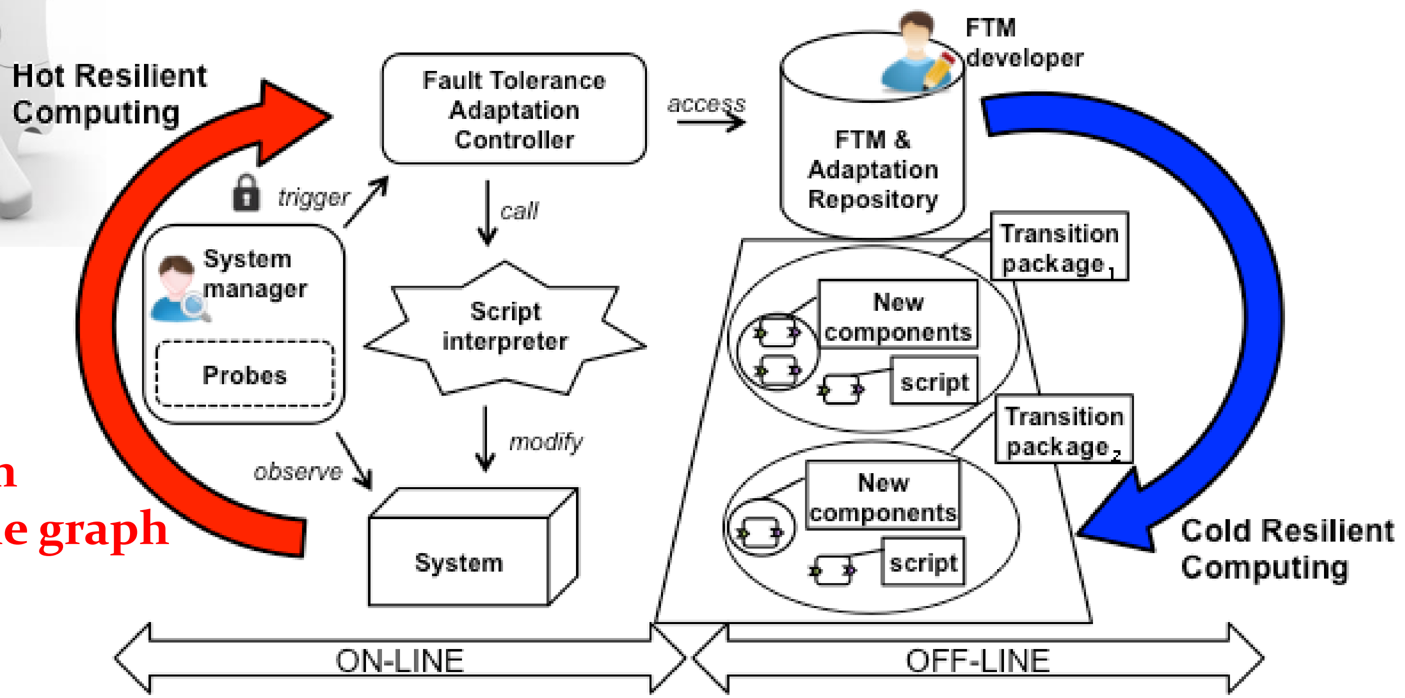
- **Component graph**
- **Suspend execution**
- **Modification of the graph**
- **Re-activate**

Overall process

FTM as a Lego system

Change

- ➔ Safety analysis / FMECA
- ➔ Impact on safety mechanism
- ➔ Agile update of FTM
- ➔ Remote update



Remote update
- Component graph
- Suspend execution
- Modification of the graph
- Re-activate

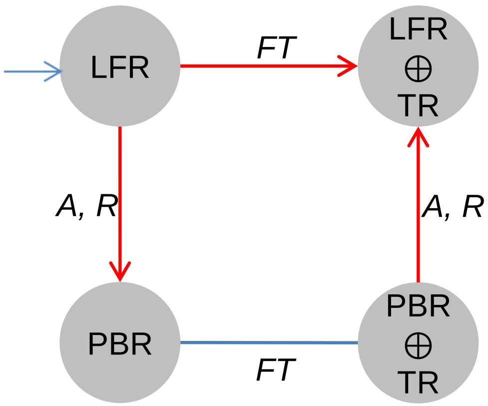
Assumptions and FTM Characteristics

Assumptions / FTM		PBR	LFR	TR
Fault Model (FT)	Crash	✓	✓	
	Transient			✓
Application behaviour (A)	Deterministic		✓	✓
	State access	✓		
Resources (R)	Bandwidth	high	low	nil
	# CPU	2	2	1

PBR=Primary-Backup Replication
LFR=Leader-Follower Replication
TR=Time Redundancy

Assumptions and FTM Characteristics

Assumptions / FTM		PBR	LFR	TR
Fault Model (FT)	Crash	✓	✓	
	Transient			✓
Application behaviour (A)	Deterministic		✓	✓
	State access	✓		
Resources (R)	Bandwidth	high	low	nil
	# CPU	2	2	1

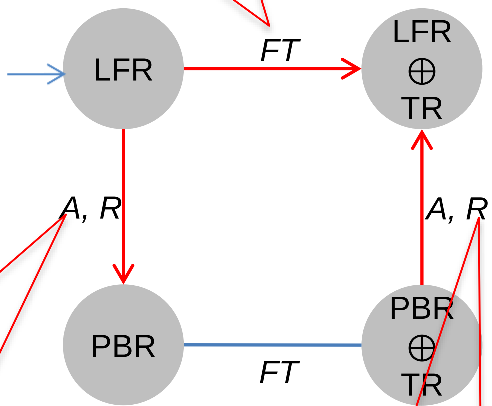


PBR=Primary-Backup Replication
LFR=Leader-Follower Replication
TR=Time Redundancy

Assumptions and FTM Characteristics

TRANSITIONS

Assumptions / FTM		PBR	LFR	TR
Fault Model (FT)	Crash	✓	✓	
	Transient			✓
Application behaviour (A)	Deterministic		✓	✓
	State access	✓		
Resources (R)	Bandwidth	high	low	nil
	# CPU	2	2	1



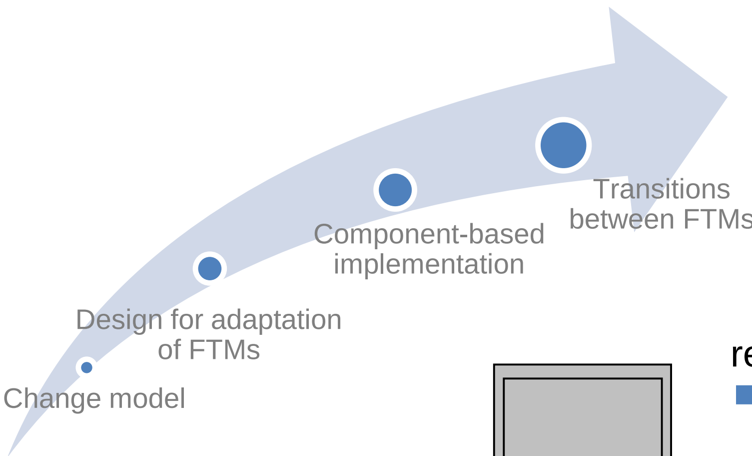
Trigger: high rate of HW transient faults observed

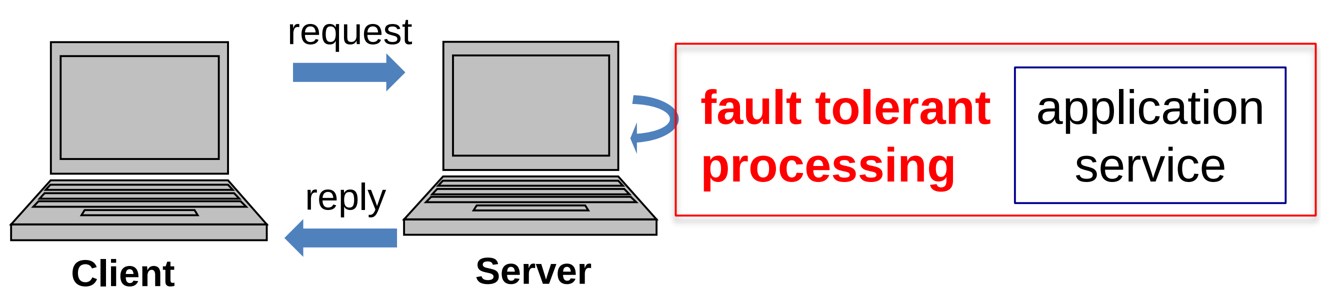
Trigger: Non deterministic SW application version

Trigger: bandwidth drop below a given threshold

PBR=Primary-Backup Replication
LFR=Leader-Follower Replication
TR=Time Redundancy

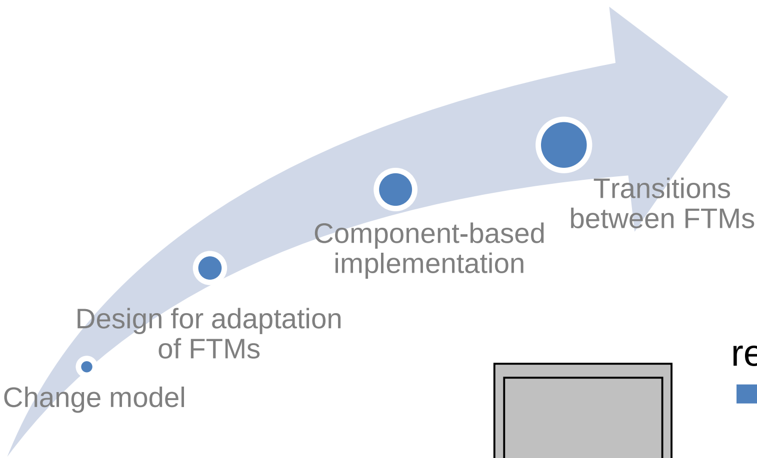
Componentization of FTM

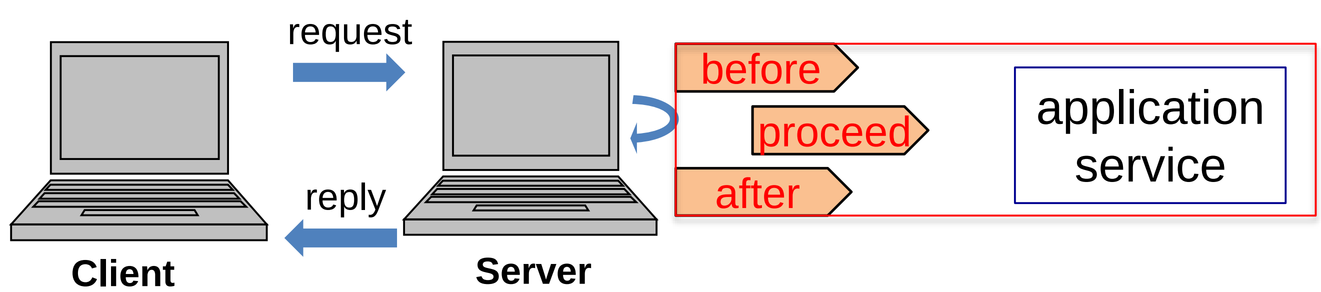




FTM	Before	Proceed	After
PBR (primary)		Compute	Checkpointing
PBR(backup)			State update
LFR (leader)	Forward request	Compute	Notify
LFR (follower)	Handle request	Compute	Handle notification
TR	Save/restore state	Compute	Compare

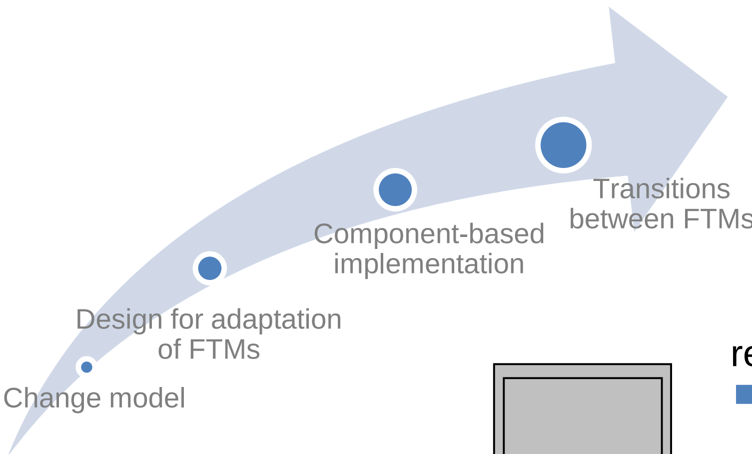
Componentization of FTM

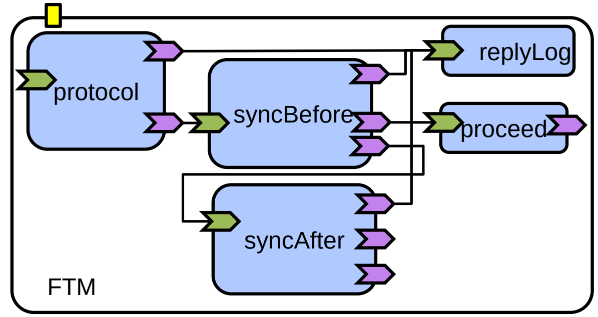


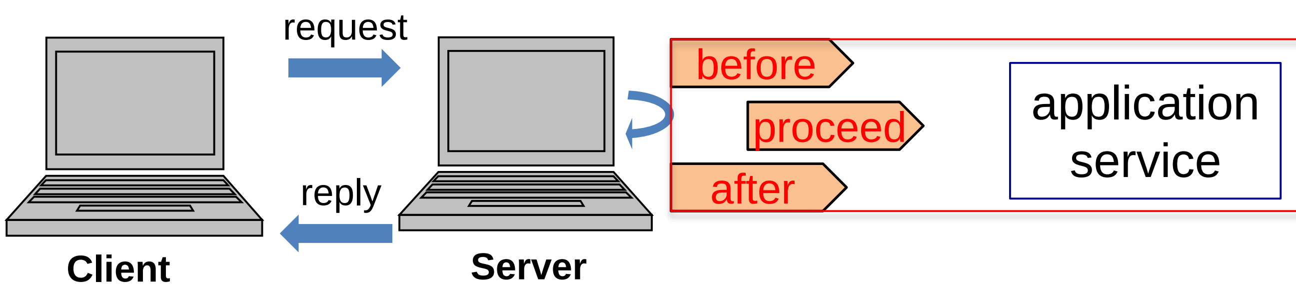


FTM	Before	Proceed	After
PBR (primary)		Compute	Checkpointing
PBR(backup)			State update
LFR (leader)	Forward request	Compute	Notify
LFR (follower)	Handle request	Compute	Handle notification
TR	Save/restore state	Compute	Compare

Componentization of FTM



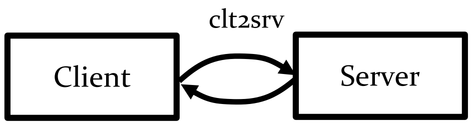




FTM	Before	Proceed	After
PBR (primary)		Compute	Checkpointing
PBR(backup)			State update
LFR (leader)	Forward request	Compute	Notify
LFR (follower)	Handle request	Compute	Handle notification
TR	Save/restore state	Compute	Compare

Design for FTM adaptation on ROS

Generic computation graph for FTM
(Boxes represent nodes)

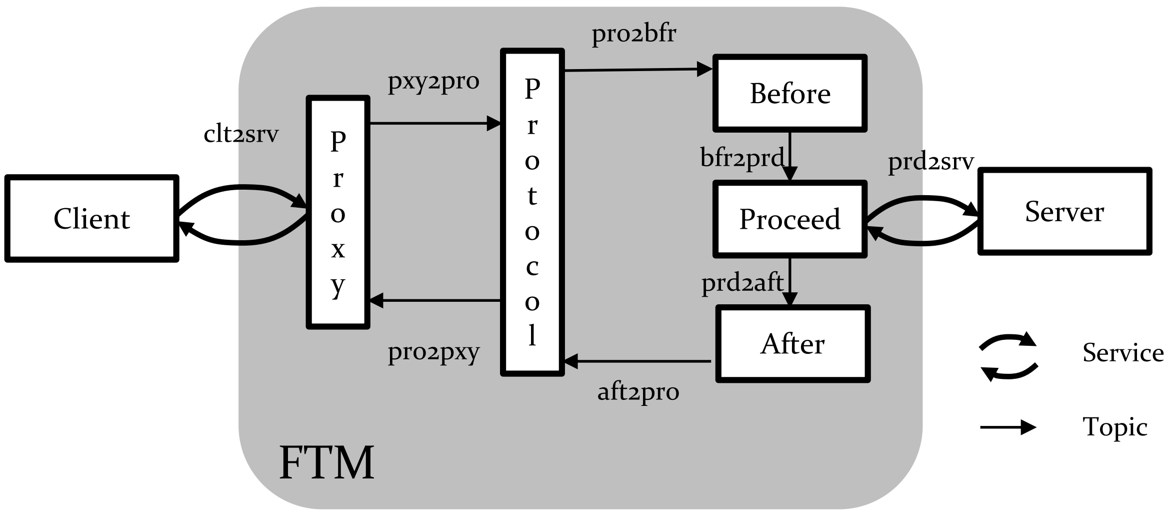


- **Topics(0)**
- **Nodes(2)**
  - Client
  - Server

Services: clt2srv (client to server)

Design for FTM adaptation on ROS

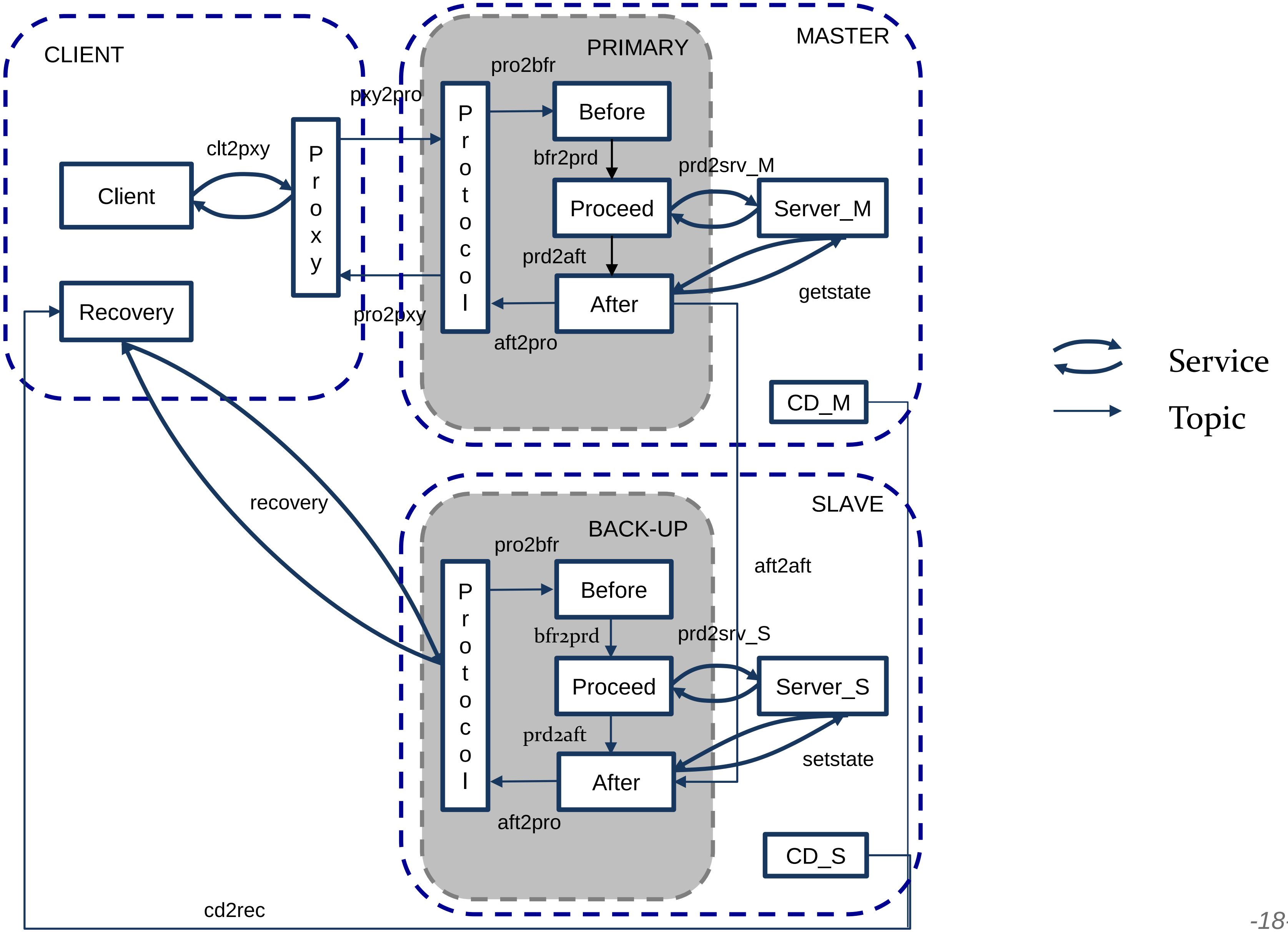
Generic computation graph for FTM
(Boxes represent nodes)



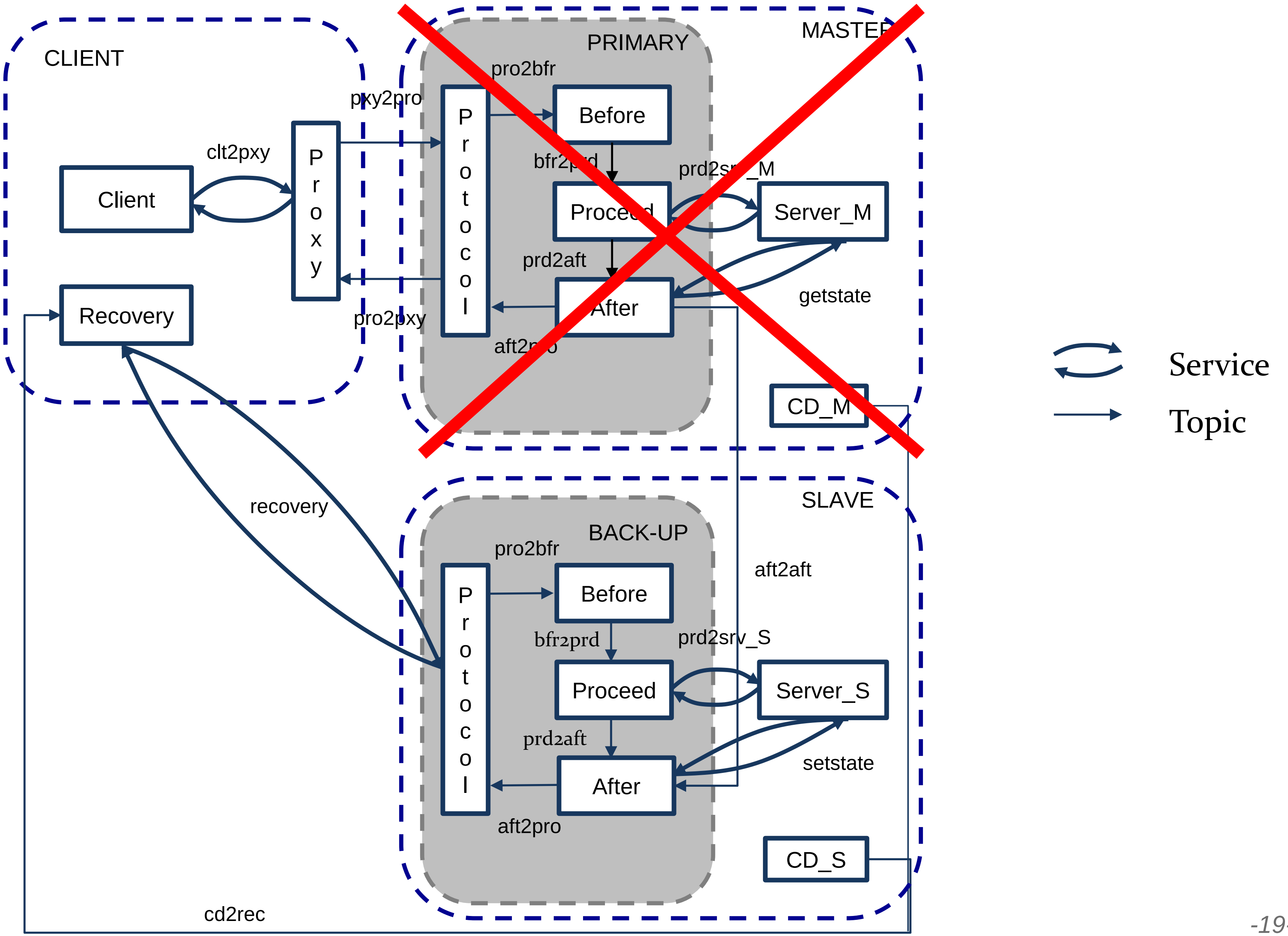
- **Topics(6)**
  - pxy2pro
  - pxy2bfr, bfr2prd,prd2aft
  - aft2pro
  - pro2pxy
- **Nodes(5+2)**
  - Client
  - Server
  - Proxy
  - Protocol
  - Before, Proceed, After

Services: clt2pxy (client to proxy) and prd2srv (proceed to server)

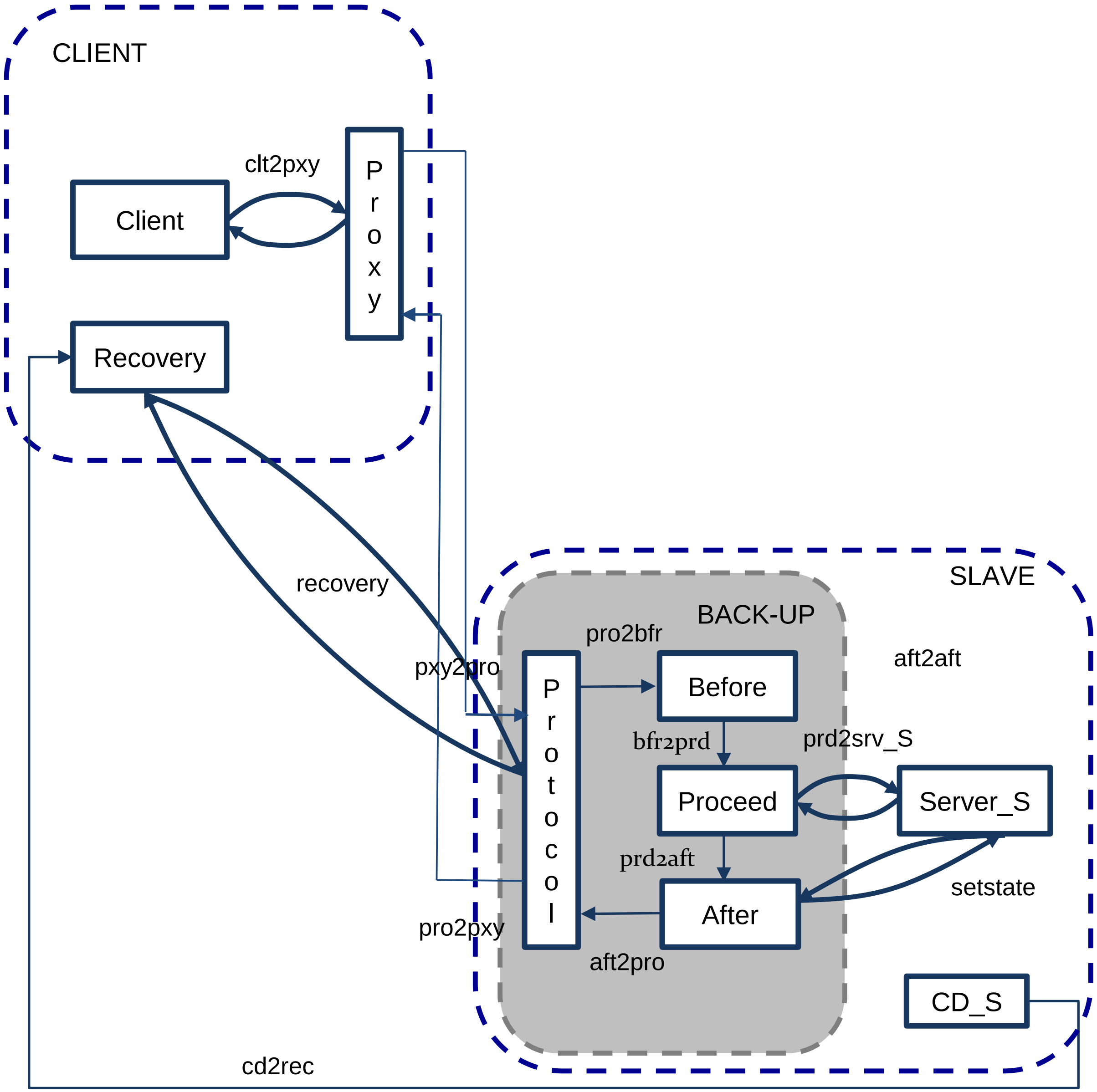
Implementing PBR on ROS



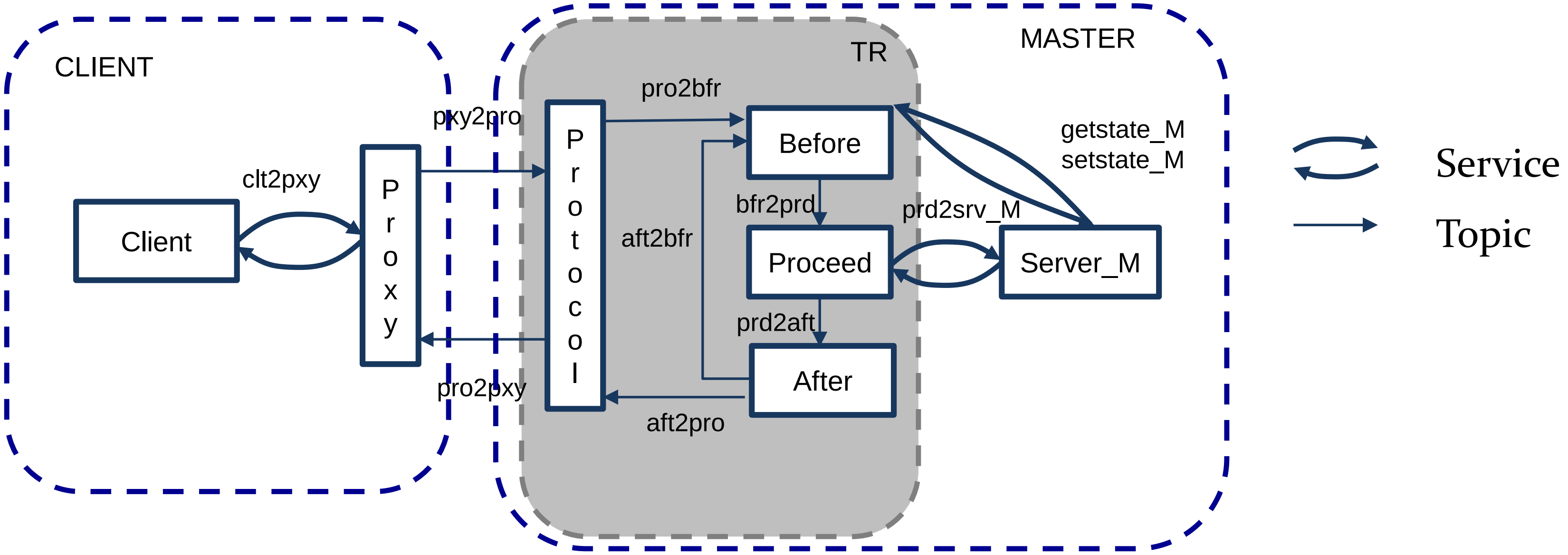
Implementing PBR on ROS



Implementing PBR on ROS

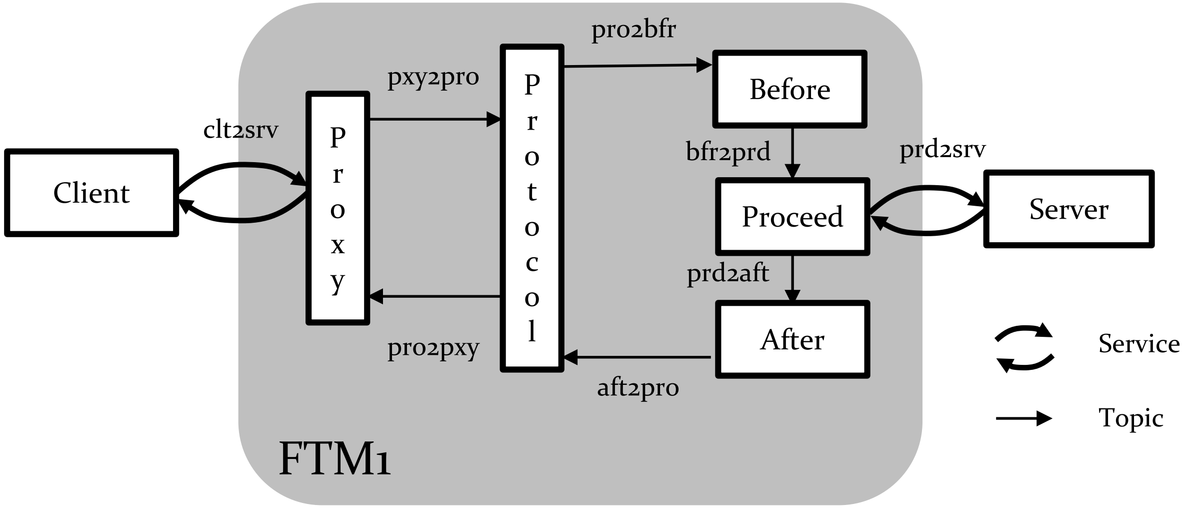


Implementing TR on ROS



Combining FTM on ROS

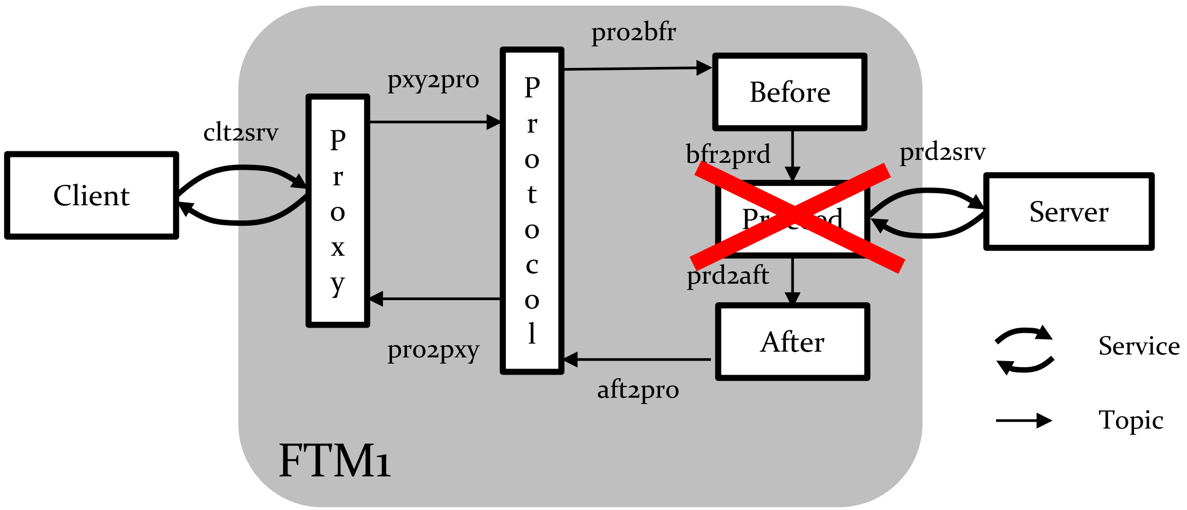
Generic composition graph for FTM



- **Protocol node is a software rack of nodes**
  - Before
  - Proceed ➔ activation of services or protocols
  - After
- **Protocol node can substitute for proceed node**
  - It can be view as a frontend of the server…

Combining FTM on ROS

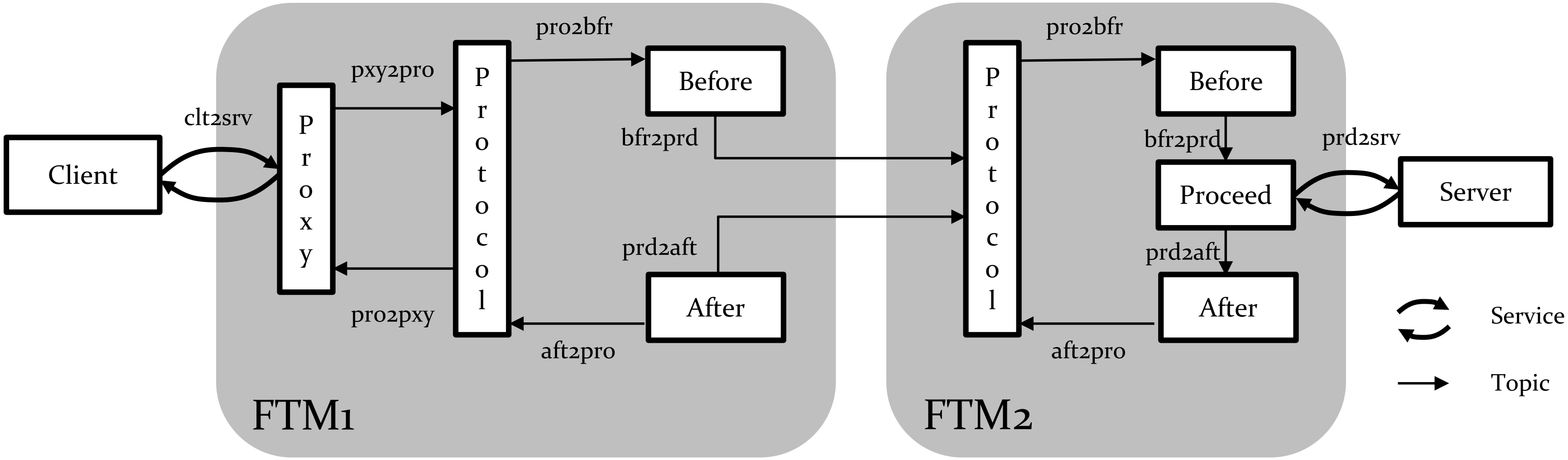
Generic composition graph for FTM



- **Protocol node is a software rack of nodes**
  - Before
  - Proceed ➔ activation of services or protocols
  - After
- **Protocol node can substitute for proceed node**
  - It can be view as a frontend of the server…

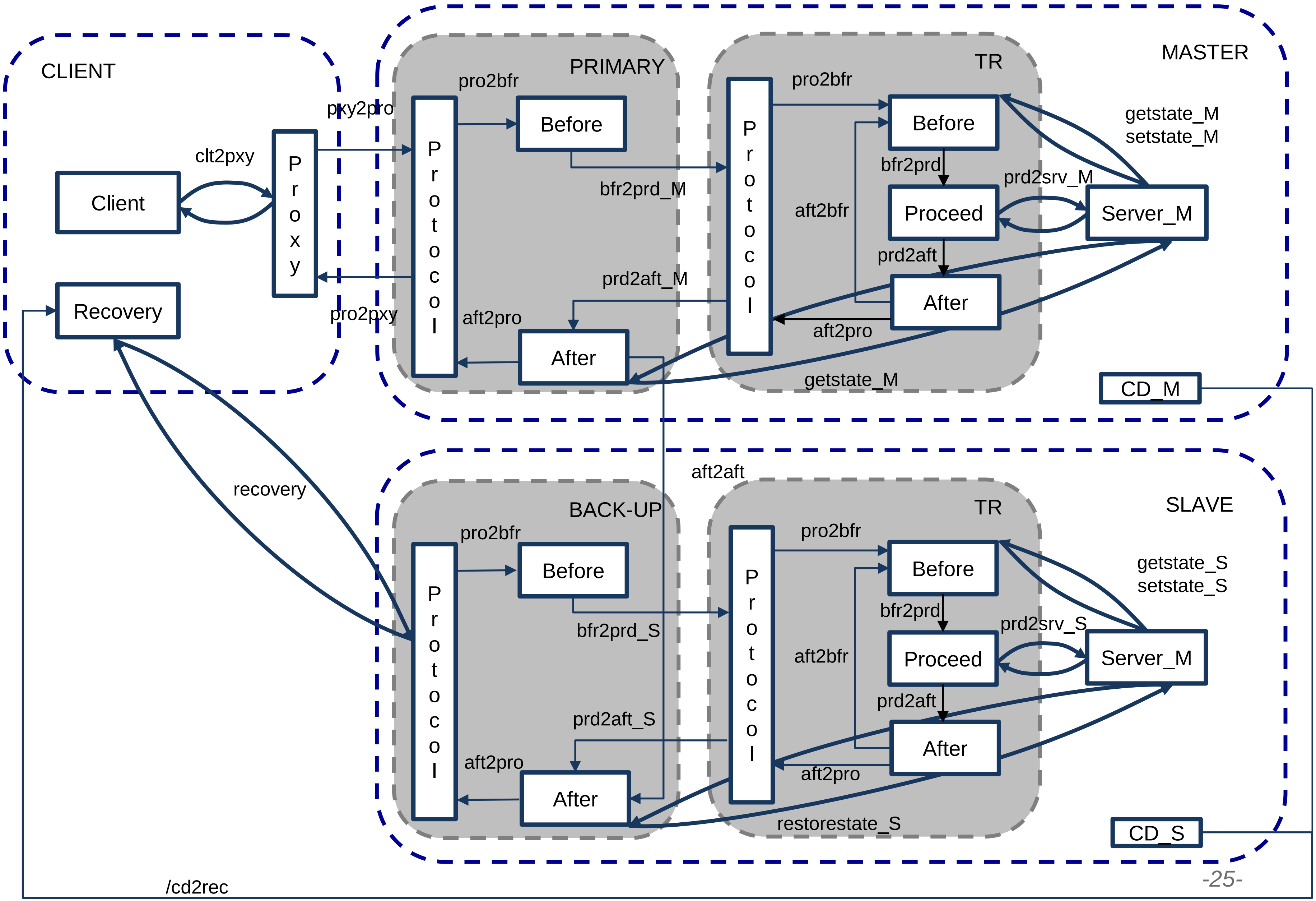
Combining FTM on ROS

Generic composition graph for FTM



- **Protocol node is a software rack of nodes**
  - Before
  - Proceed ➔ activation of services or protocols
  - After
- **Protocol node can substitute for proceed node**
  - It can be view as a frontend of the server…

Combining PBR+TR on ROS



Case Study

- **Initialization**
  - Initialisation time around **0,5s**
  - Time due to the **initialization of communications** by the ROS Master
- **Execution**
  - Around **5ms** for the PBR and **2ms** for the TR
  - Requests every 7cm for a car driving at 50km.h^{-1}
- **Recovery**
  - Recovery → Reactivation of **2 Topics**
  - Recovery time around **1ms**
- **Adaptation & Composition**
  - Adaptation → Initialization of new nodes
  - Same order as Initialization time (**≈ 0,3s**)

Ubuntu Trusty 14.04
I5 Dual Core 2,5GHz
8Go DDR3 RAM

ROS Master : A single point of failure

- **The ROS Master is requisite for:**
  - The control over the system
  - The control over communication
  - The control over the graph
  - The control over the Nodes

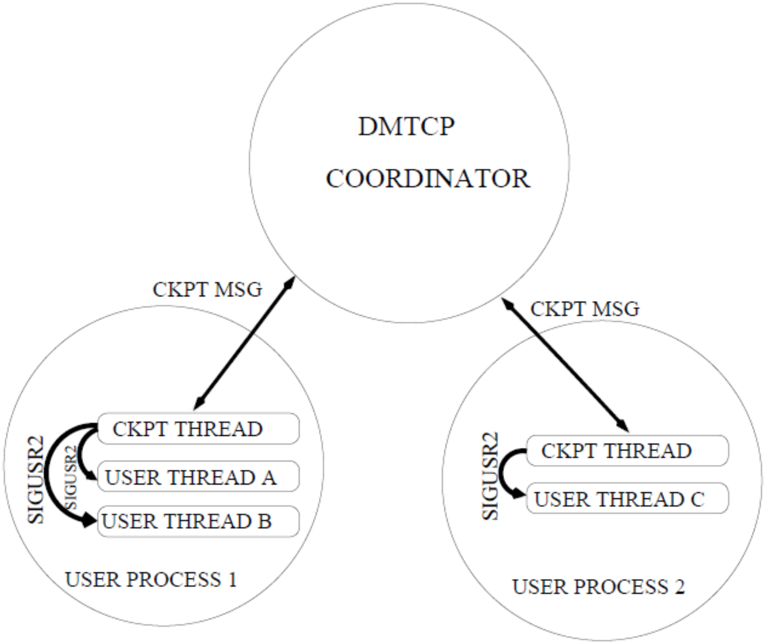
- **If the ROS Master crash:**
  - Loss of the software architecture
  - ***Nodes*** have to be reloaded
  - The state of the system is reinitialized
  - Critical loss in case of embedded systems

Solutions to assure the reliability of the ROS Master:

- **Launching it on a distinct and reliable machine**
- **Check-pointing its state and restoring it**

DMTCP: Check Pointing the ROS Master

- **DMTCP, how does it work:**
  - Works with Linux kernel 2.6.9 and later
  - Transparent (no recompilation…)
  - Virtualization of Process ID



- **Check pointing with DMTCP:**
  - Process is launch along the coordinator
  - A checkpoint image is created for each process
  - A restart script is created by coordinator

→ DMTCP should be able to checkpoint the ROS Master

→ The lost of the ROS Master should no longer be a problem

Lessons learnt

- **Adaptive fault tolerance**
  - Separation of Concern
  - Design for Adaptation

  SoC+D4A ➔ FTM isolation and componentization

- **Installation or adaptation of an FTM online**
  - Node can be started and stopped
  - Mapping at initialization
- **Node Management**
  - APIs are not provided by ROS for Node Management
  - User signals and System calls fulfill the missing requirements
- **Implementing dynamic binding**
  - Natural dynamic binding is also not provided by ROS
  - Topics and Services are remapped at the initialization

Summary of dynamic adaptation

SoC	ROS nodes, component mapping to nodes
D4A	Componentized FT design patterns Protocol-Before-Proceed-After
Nodes Mngmnt	Unix system calls and ROS commands
Dynamic Binding	ROS services, ports, topics Additional logic to create ports and topics
Master CKPT	Check point of the ROS Master ROS Master is no longer a Single Point of Failure

Conclusion

- **Now…**
  - Adaptive Fault Tolerance for Resilient Computing is possible on ROS
  - Design and validation of FTMs is always carried out offline
  - If application can be terminated and re-launched : adaptation OK
  - Dynamic adaptation :
    - Extended API for dynamic binding
    - Consistency of reconfiguration?

- **Proceeding...**
  - Experiments on ADAS with ***Renault SAS***
  - Evolution of ***AUTOSAR*** into ***Adaptive AUTOSAR***
  - Experimentation on ***ROS Master*** with ***DMTCP***