SeMaFoR Project
& Concerto-D for decentralized reconfiguration of Fog systems

Jolan PHILIPPE

PostDoc - SeMaFoR project

Thomas LEDOUX
(STACK)

Hélène COULLON
(STACK)

Charles PRUD'HOMME
(TASC)

Hugo BRUNELIERE
(Naomod)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

STACK

SeMaFoR
Self Management of Fog Resources

Naomod PotW
July 3rd, 2023

Inria

LS2N

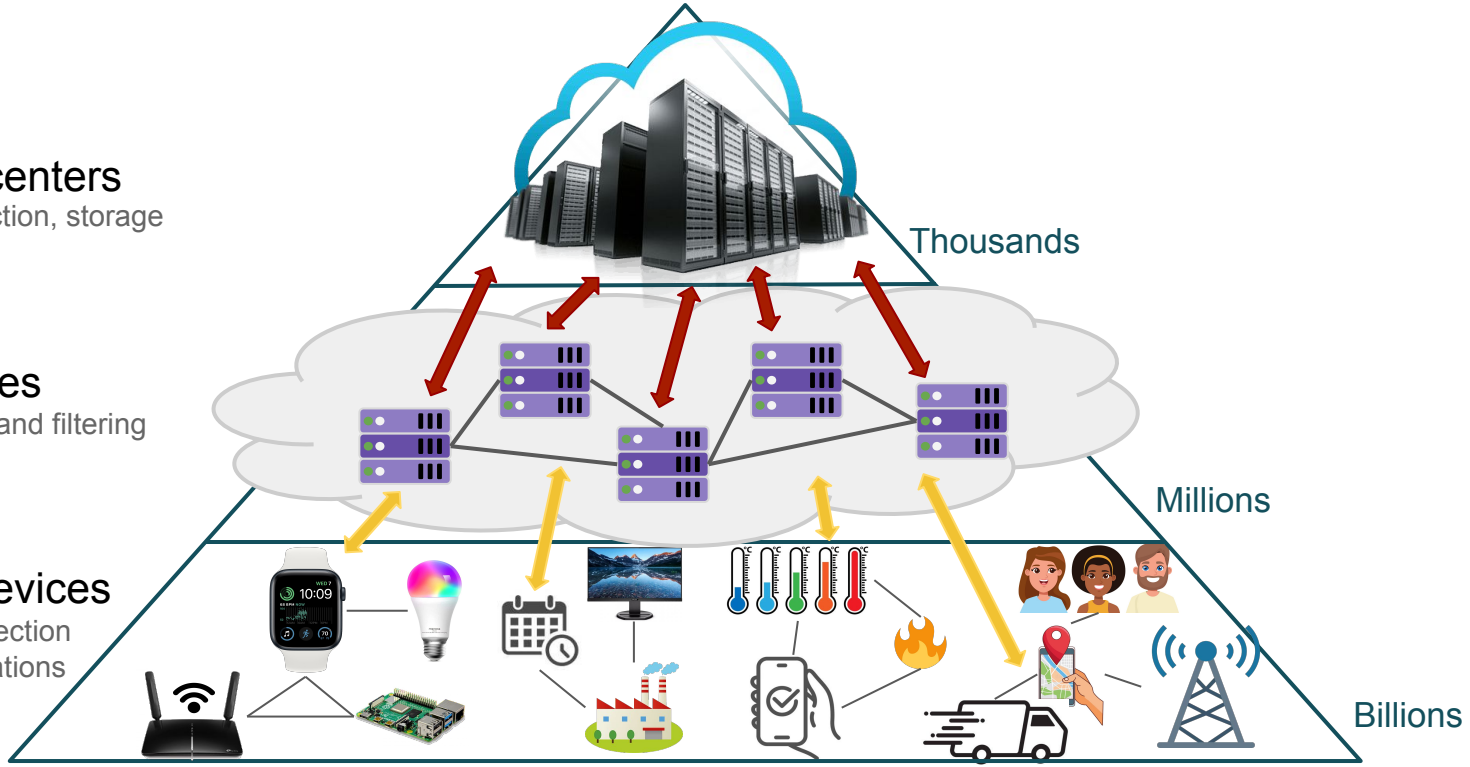
PROJET FINANCÉ PAR L'ANR - PROJECT FUNDED BY THE ANR

Context: Fog Architectures

Cloud data centers
Non-real-time data action, storage

Fog nodes
Local data analysis and filtering

Edge / IoT devices
Local data collection
+ small calculations



"The Fog extends the Cloud to be closer to the thing that produce and act on IoT data" [Cisco, mar. 2015]

SeMaFoR Project

Problem

- How to administrate a Fog infrastructure? (size, reliability, dynamic, heterogeneous,...)

Objectives [SeMaFoR, 2023]

- Designing and developing a decentralized, generic solution for self-administration of resources.
- Coordinate a fleet of autonomous controllers in a distributed manner, with each controller having a local view of its resources.

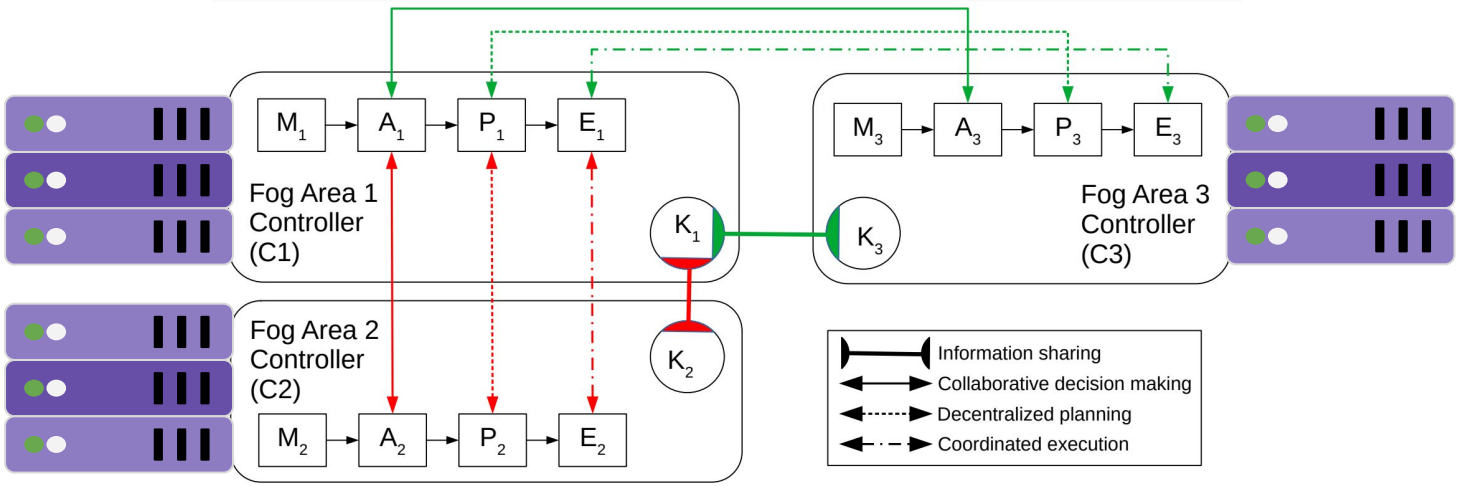
STACK TASC naomod alter way SMILE

SORBONNE UNIVERSITÉ CRÉATEURS DE FUTURS DEPUIS 1257 LIP6

SeMaFoR proposal for controller coordination

MAPE-K [IBM, 2006]**:** ***Coordinated Control Pattern*** model

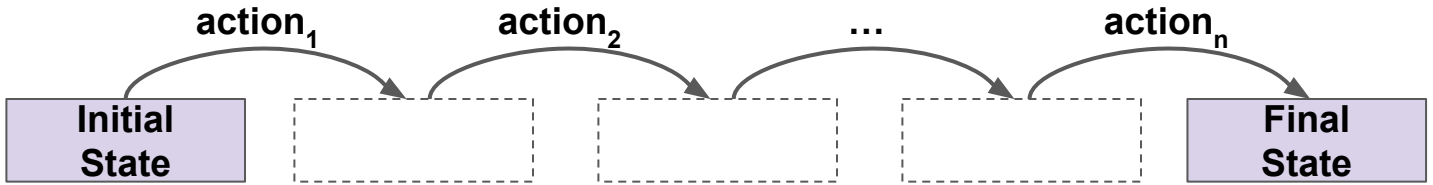
- **M**onitor its state and the state of the environment
- **A**nalyze to decide which state to reach
- **P**lan the reconfiguration
- **E**xecute the reconfiguration to reach the new state
  - **K**nowledge that is common, to take a decision



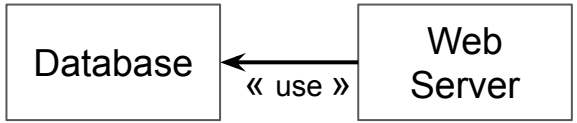
Reconfiguration plan 📋

A reconfiguration ≔ a set of actions, answering

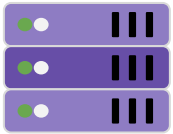
- **WHERE**
- **WHAT**
- **HOW**
- **WHEN**



Deployment example



Machine 1:



Database (DB)

1. Install
2. Configure
3. Start the service
4. Prepare the service

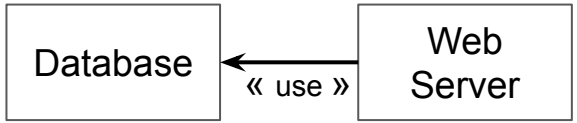
Machine 2:

Web Server (WS)

1. Install
2. Configure firewall
3. Download
4. Configure parameters
5. Start the service

Component granularity: DB ≪ WS
Lifecycle granularity: DB(3) ≪ WS(4), DB(4) ≪ WS(5)

Deployment example



Machine 1: **WHERE**

Database (DB) **WHAT**

HOW

1. Install
2. Configure
3. Start the service
4. Prepare the service

Machine 2: **WHERE**

Web Server (WS) **WHAT**

HOW

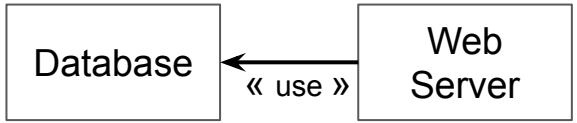
1. Install
2. Configure firewall
3. Download
4. Configure parameters
5. Start the service

WHEN

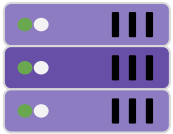
Component granularity: DB ≪ WS

Lifecycle granularity: DB(3) ≪ WS(4), DB(4) ≪ WS(5)

Reconfiguration example: update database



Machine 1:



Database (DB)

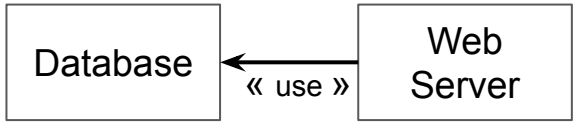
1. Backup data
2. Stop the service
3. Download update
4. Configure parameters
5. Start the service
6. Restore data

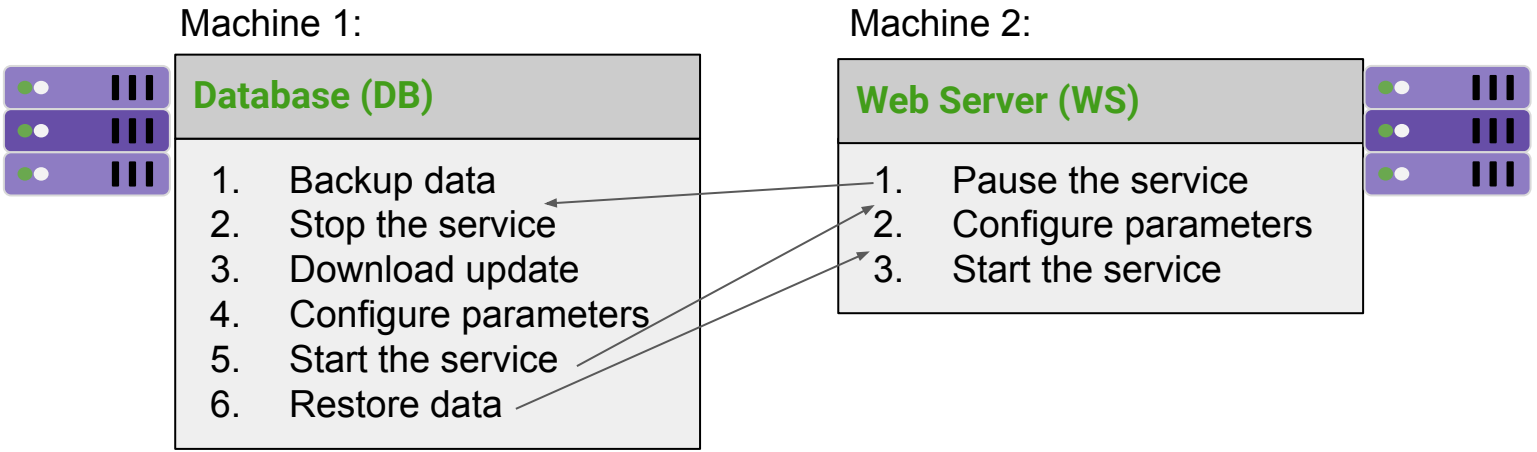
Machine 2:

Web Server (WS)

1. Pause the service
2. Configure parameters
3. Start the service

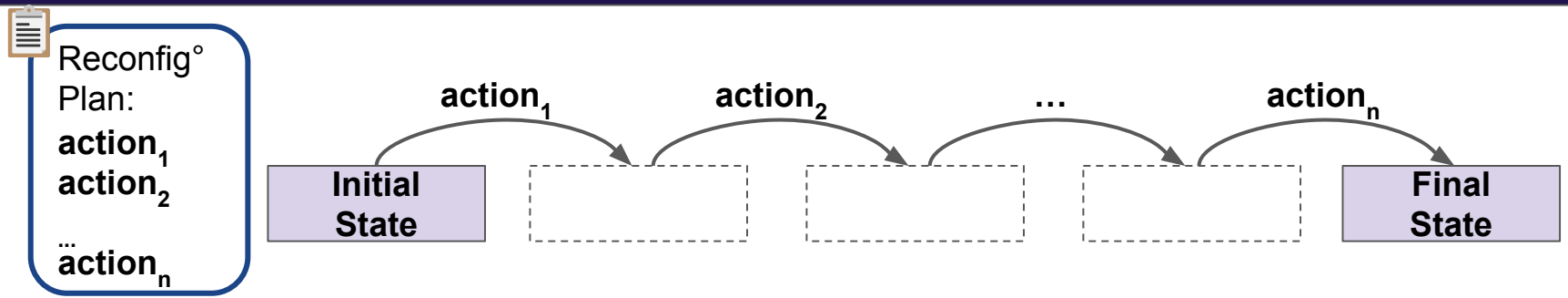
Reconfiguration example: update database





$WS(1) \ll DB(2), \quad DB(5) \ll WS(2), \quad DB(6) \ll WS(3)$

Reconfiguration plan of Fog resources



Postdoc objectives:

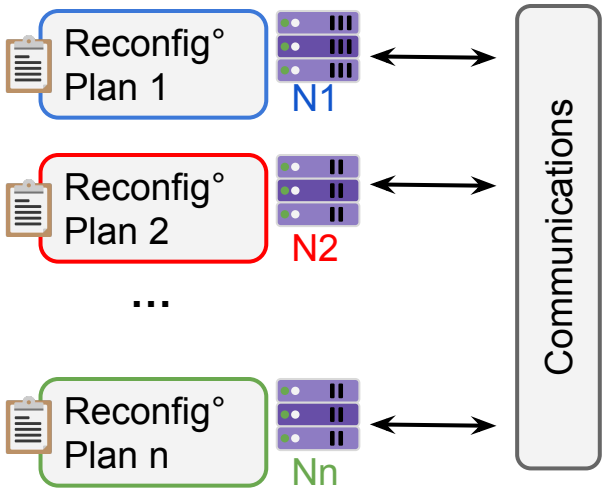
- ➢ Infer reconfiguration actions
- ➢ Optimal overall reconfiguration

Challenges:

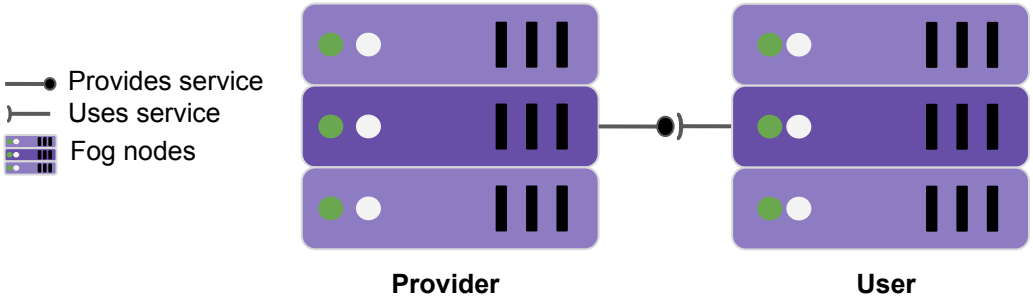
- ■ Locally: partial view of the system
- ■ Collaboration with other nodes

Inspiration:

- ■ SMT-based [Robillard, apr. 2022]



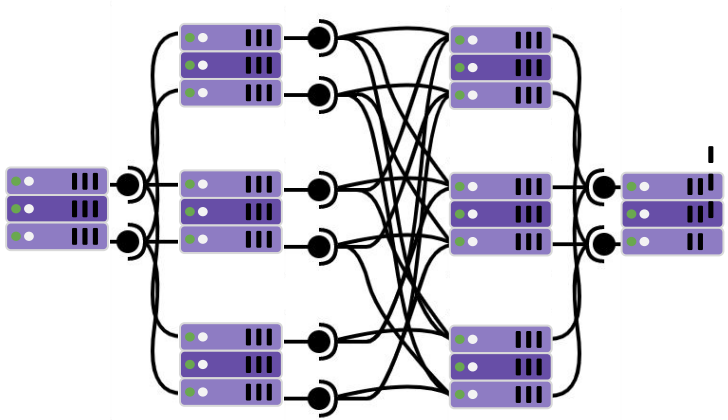
Constraint with providers and users



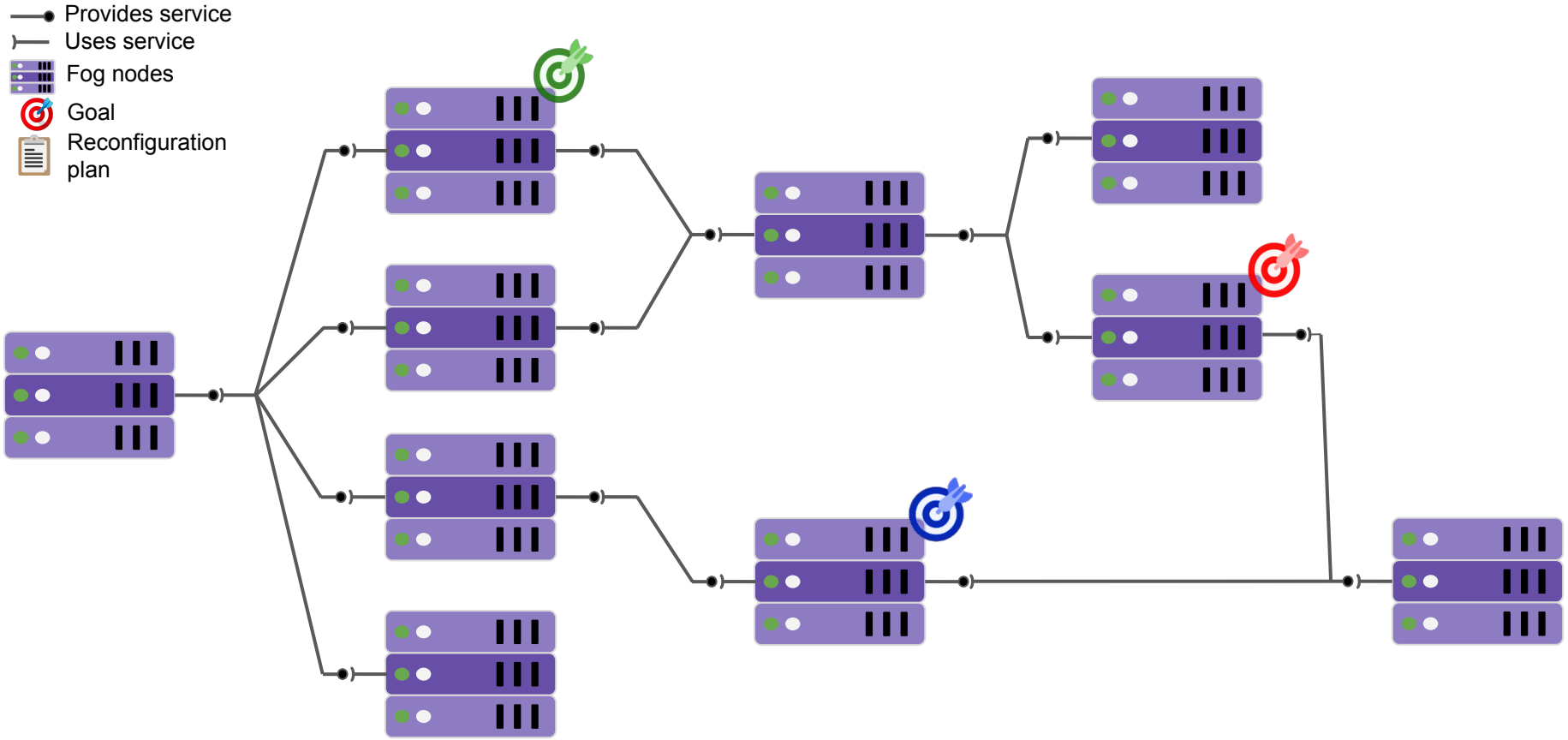
Nodes are connected using interfaces to:

- **Provide** services
- **Use** external services

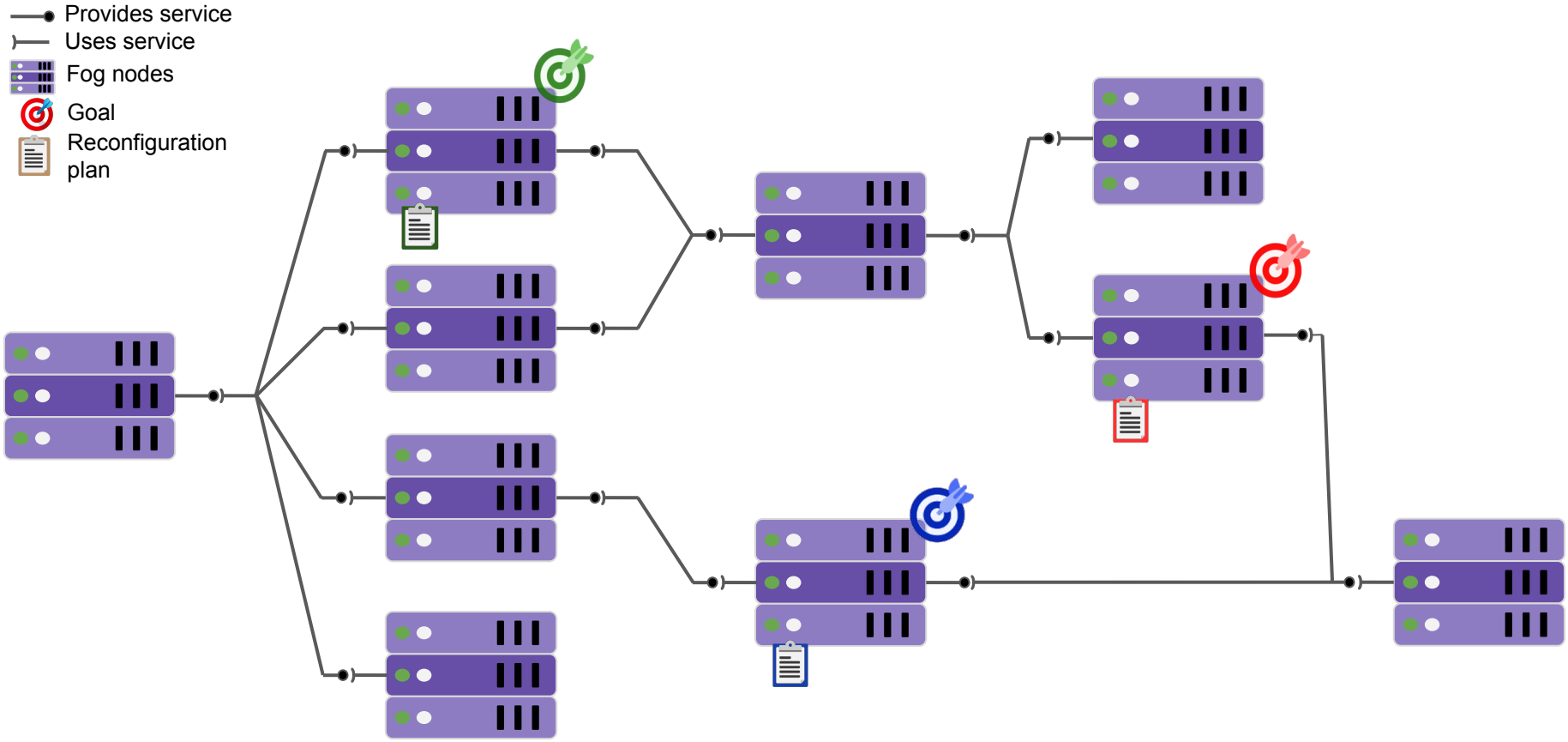
creating coordination constraints (behavioral and sync.)



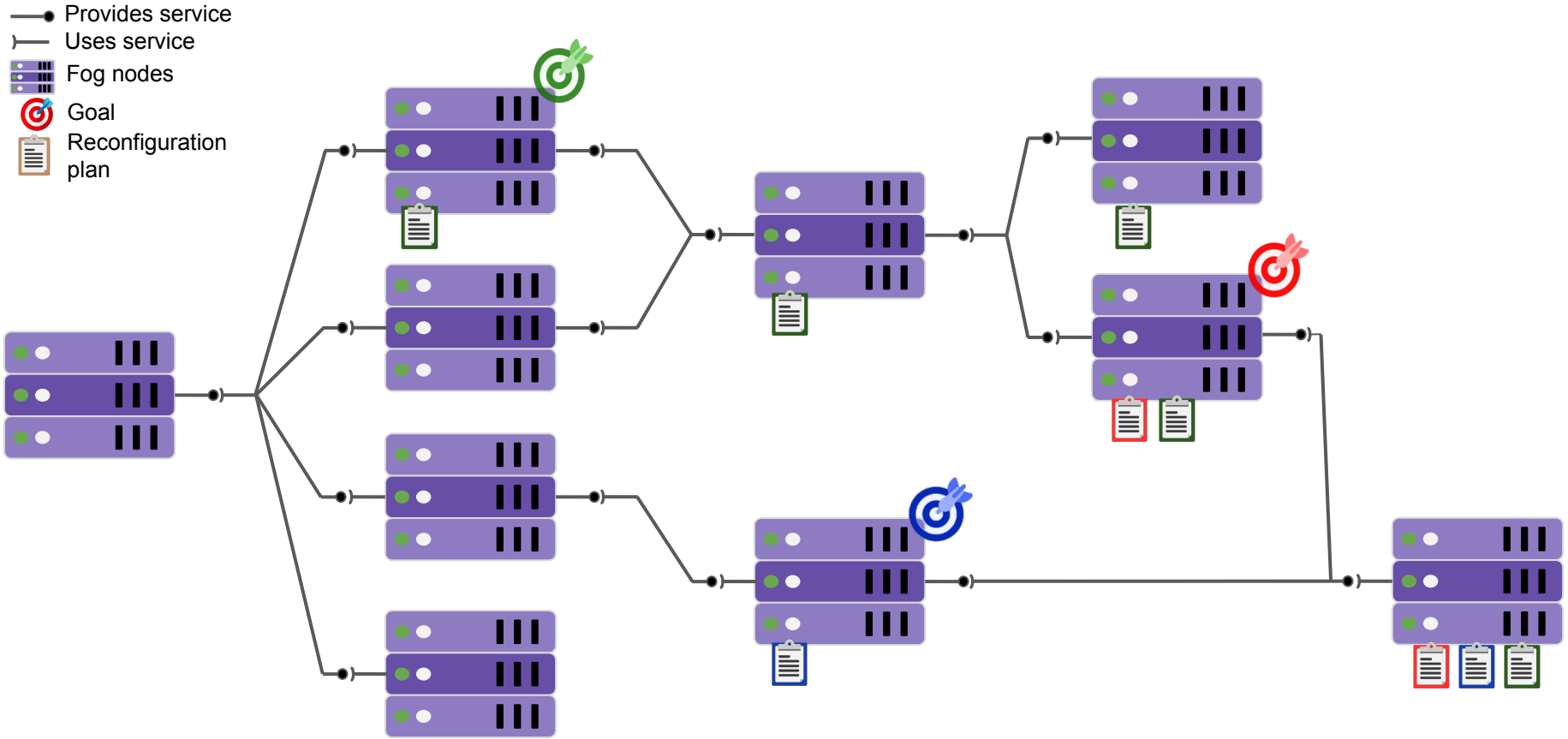
Reconfiguration of Fog resources: Local goal



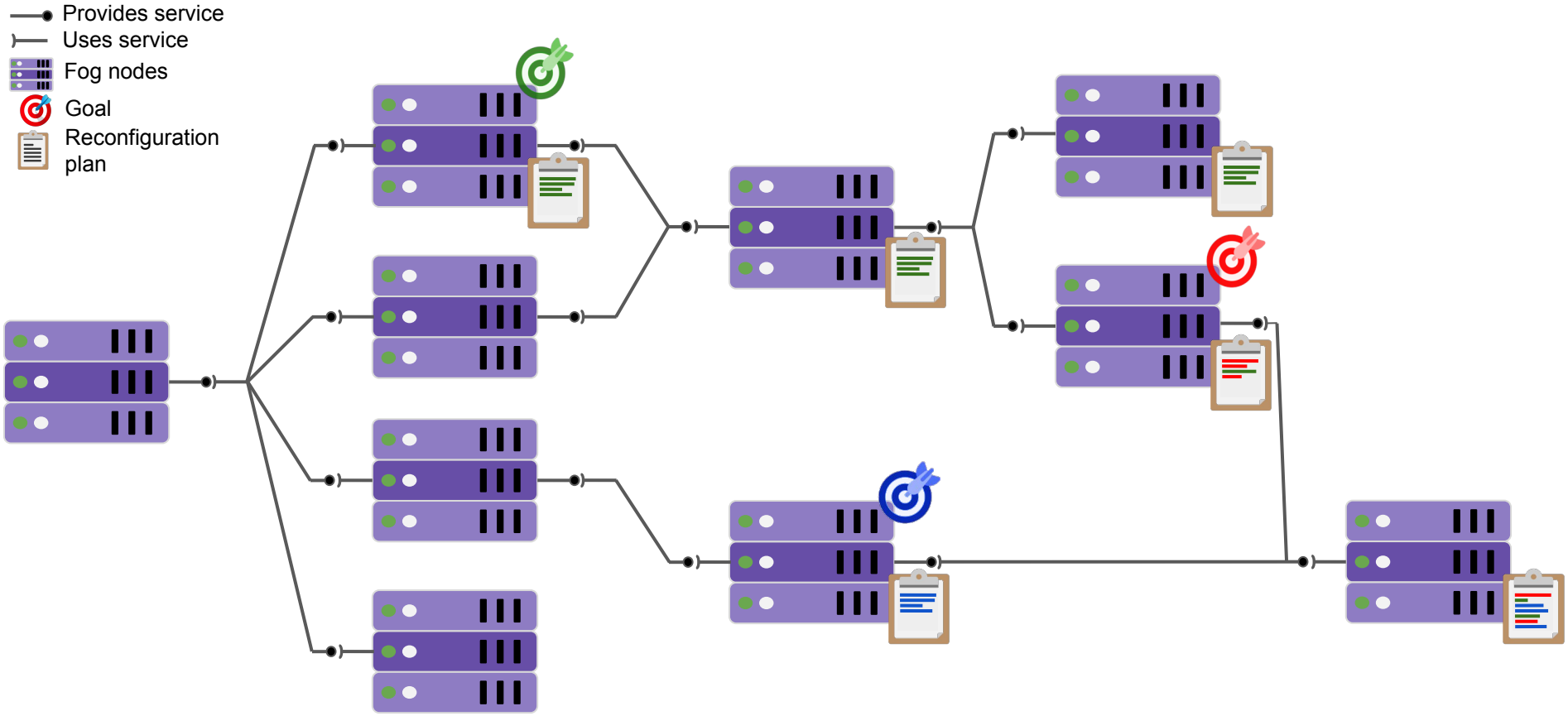
Reconfiguration of Fog resources: Local decision



Reconfiguration of Fog resources: Local decision propagation

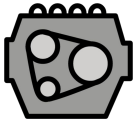


Reconfiguration of Fog resources: Local plan (Sync + Optimization)



Approach

- **Sharing protocol** with message passing (rumor-spreading)
  - **<u>Local inference of behaviors</u>** with Constraint Programing (CP)
    - Modelisation as automata
    - **Goal:** Find a sequence matching the automata
      - Goal constraints 🎯
      - Coordination constraints 📋📋📋
  - **<u>Local planning</u>** with CP
    - Overload the automata from local decision
      - Add synchronization constraints
    - **Goal:** Find a sequence matching the automata
      - Goal constraints
      - Coordination constraints

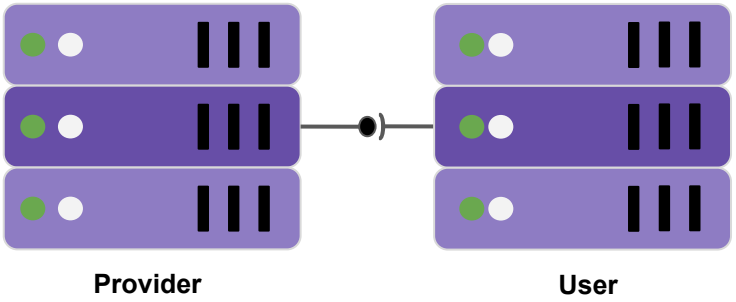


- Produced plan for the **Concerto-D** language

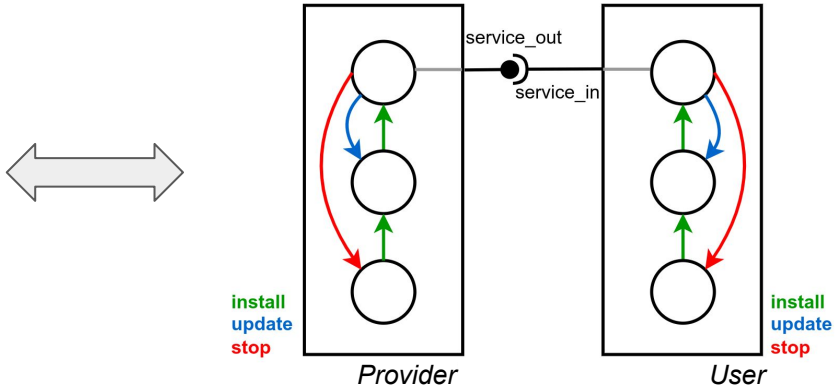
Modeling Fog component reconfiguration with Concerto-D

Concerto-D: A reconfiguration language for decentralized components

- Involved components
- Interactions / connections between components
- Changes in the component

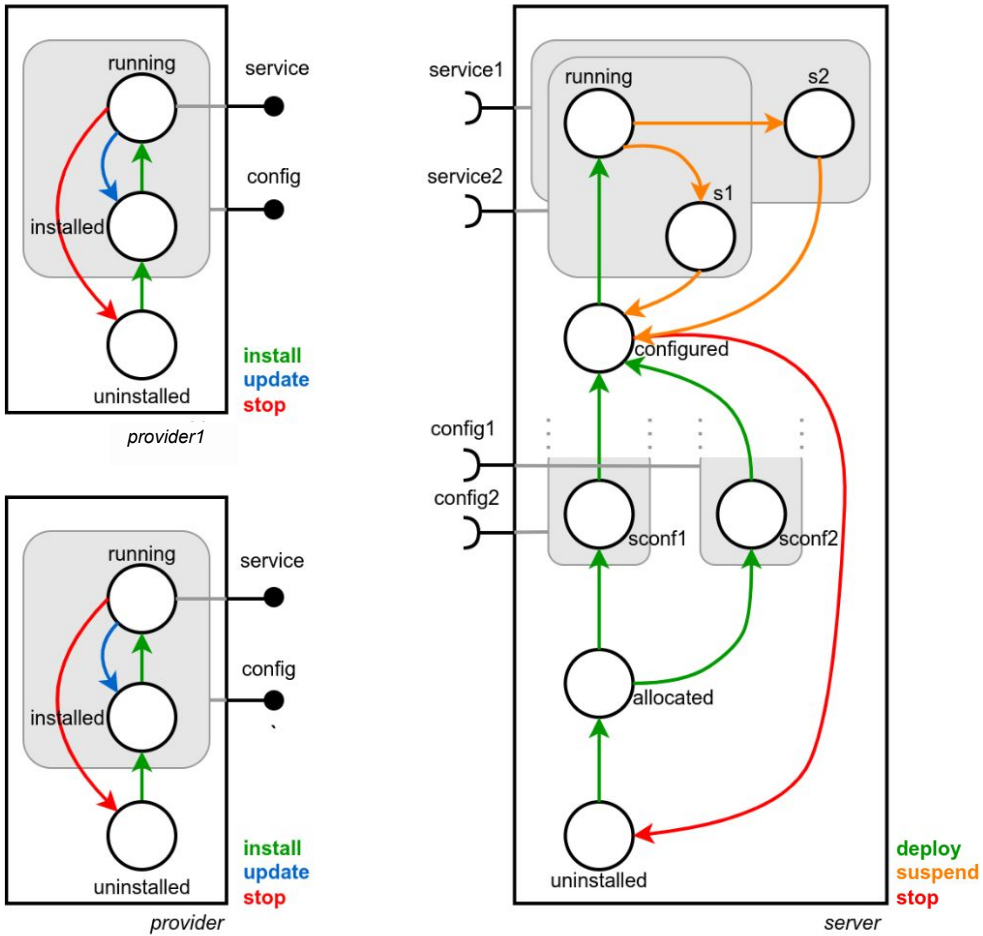


Fog view



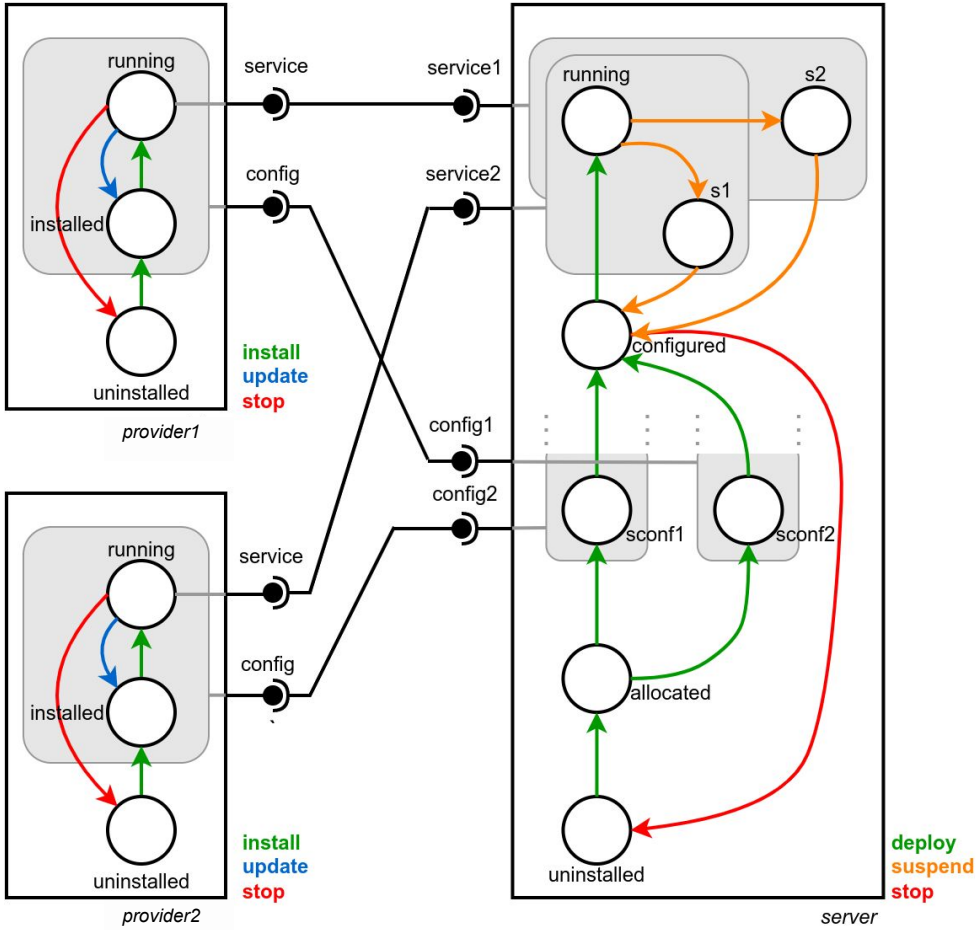
Concerto-D model

Concerto-D: Involved components



add("*provider1*", Provider)
add("*provider2*", Provider)
add("*server*", Server)

Concerto-D: Connections between components

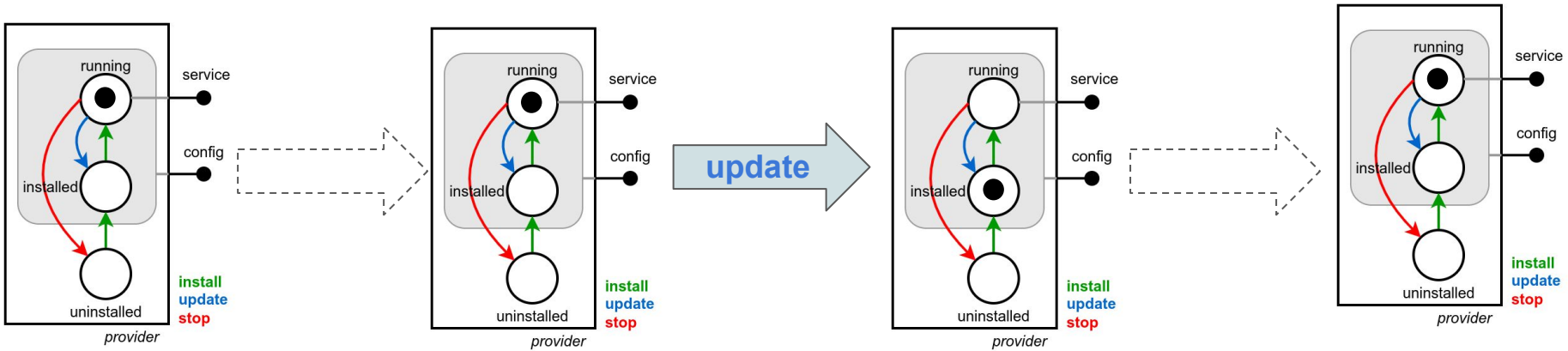


add("*provider1*", Provider)
add("*provider2*", Provider)
add("*server*", Server)
connect("provider1", "service",
"server", "service1")
connect("provider1", "config",
"server", "config1")
connect("provider2", "service",
"server", "service2")
connect("provider2", "config",
"server", "config2")

Concerto-D: State and changes in components

Example of objective:

- **Update** a running *provider*
- End the reconfiguration with a running *provider*

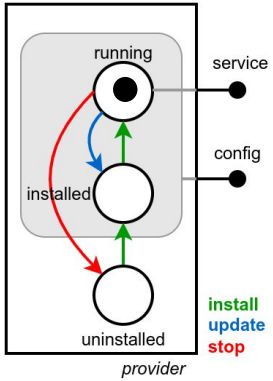


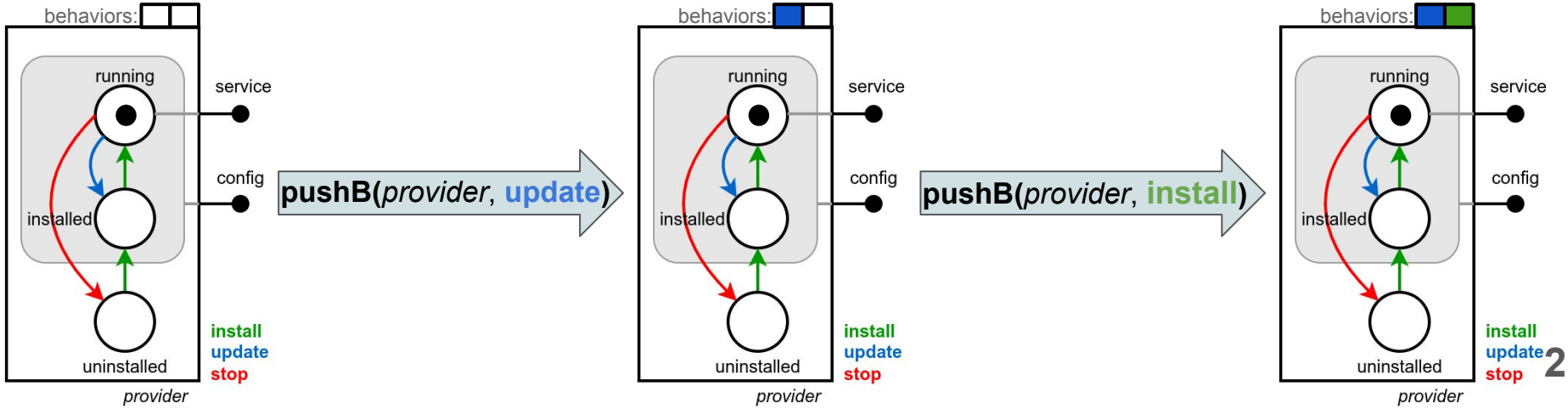
Inferred actions:

- **update** *provider*
- **install** *provider*

Concerto-D: Connections between components

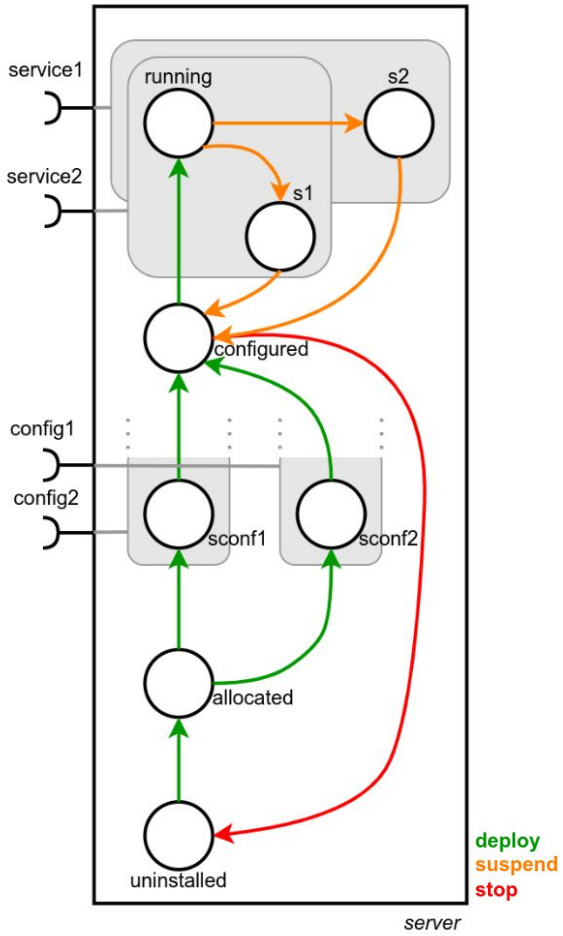
non-blocking **pushB(*provider*, update)**
non-blocking **pushB(*provider*, install)**
blocking (syncro) **wait(*provider*, install)**

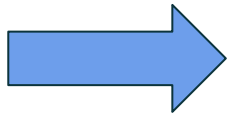


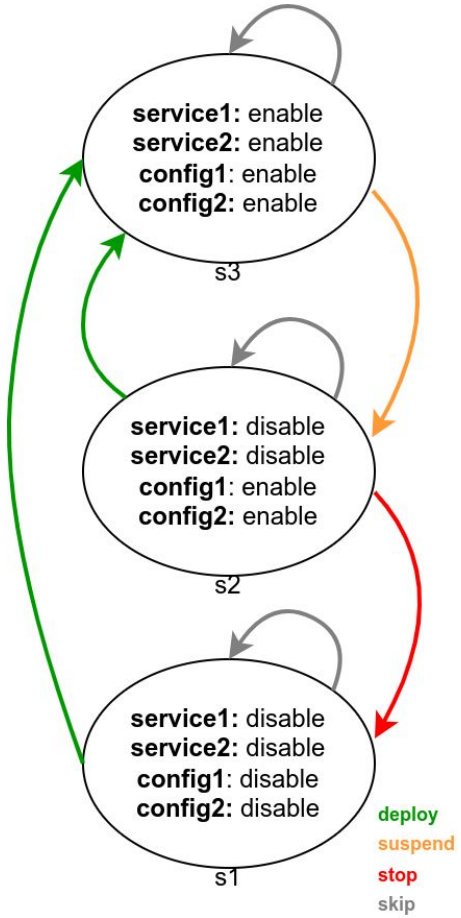


(

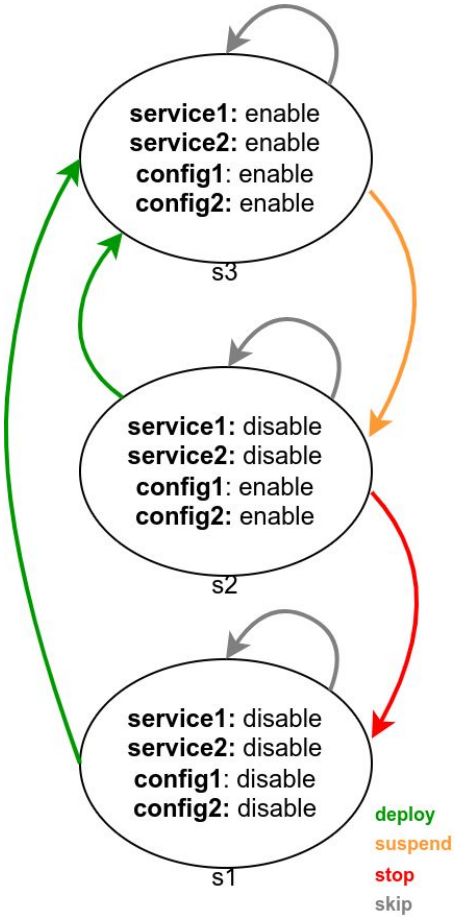
Constraint resolution: Concerto-D to a labeled automata







Constraint resolution: MiniZinc model



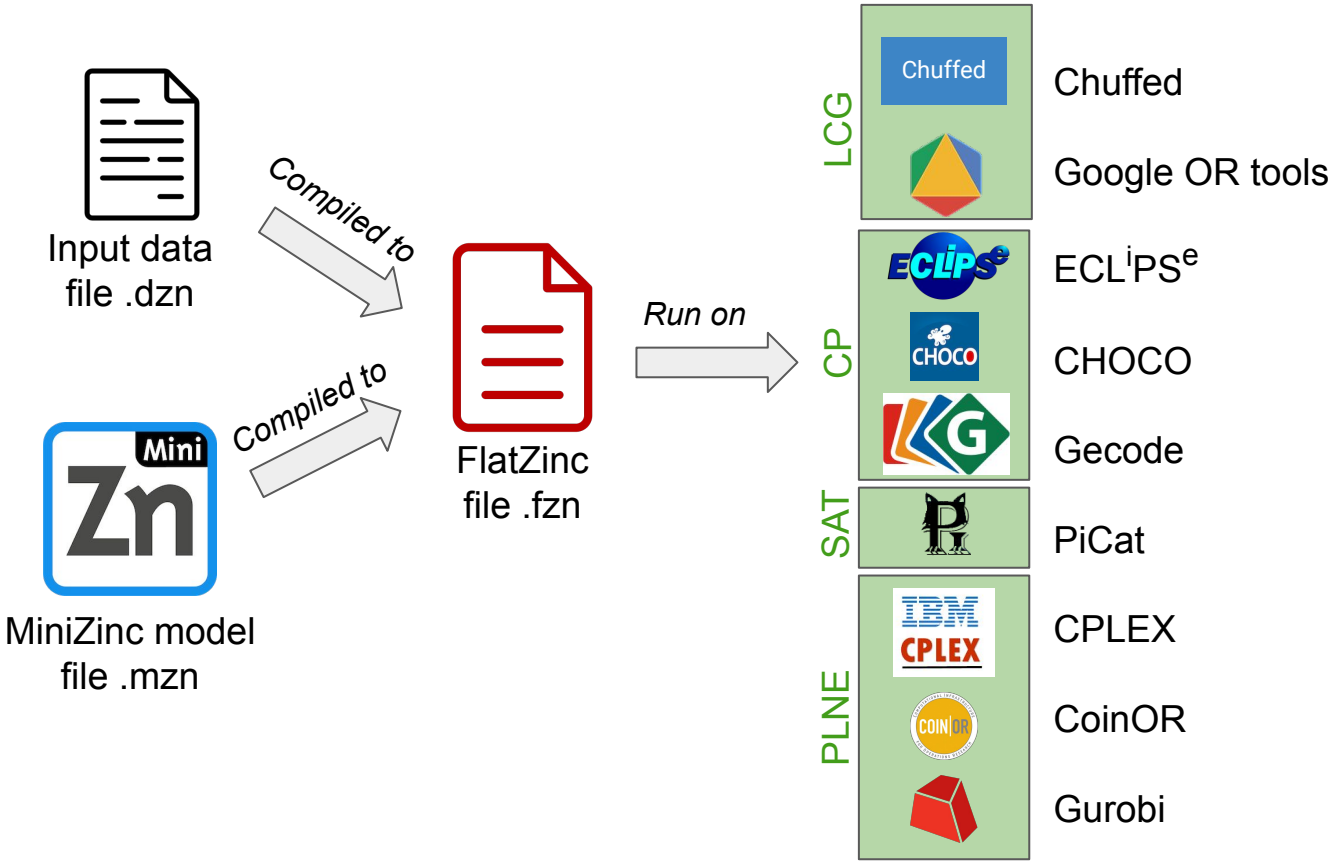

```
BEHAVIOR := {deploy, suspend, stop, skip}
STATE := {s1, s2, s3}
STATUS := {enabled, disabled}
transitions: Array[STATE][BEHAVIOR] of STATE =...

sequence: Array[1..n] of BEHAVIOR
state: Array[1..n+1] of STATE

service1: Array[1..n+1] of STATUS
…

constraint regular(sequence, transitions)
constraint sequence[i] = skip ⇒ sequence[i+1] = skip

constraint ∀i ∈ 1..n, state[i+1] = transition[state[i]][sequence[i]]
constraint ∀i ∈ 1..n+1, config1[i] = enabled ⇔ state[i] ∈ {s1, s2}
…

solve maximize count(skip, sequence)
```

Output with *init=s3; final=s3; goal=stop; n=10*

sequence = [suspend, stop, deploy, skip, skip, skip, skip, skip, skip, skip]

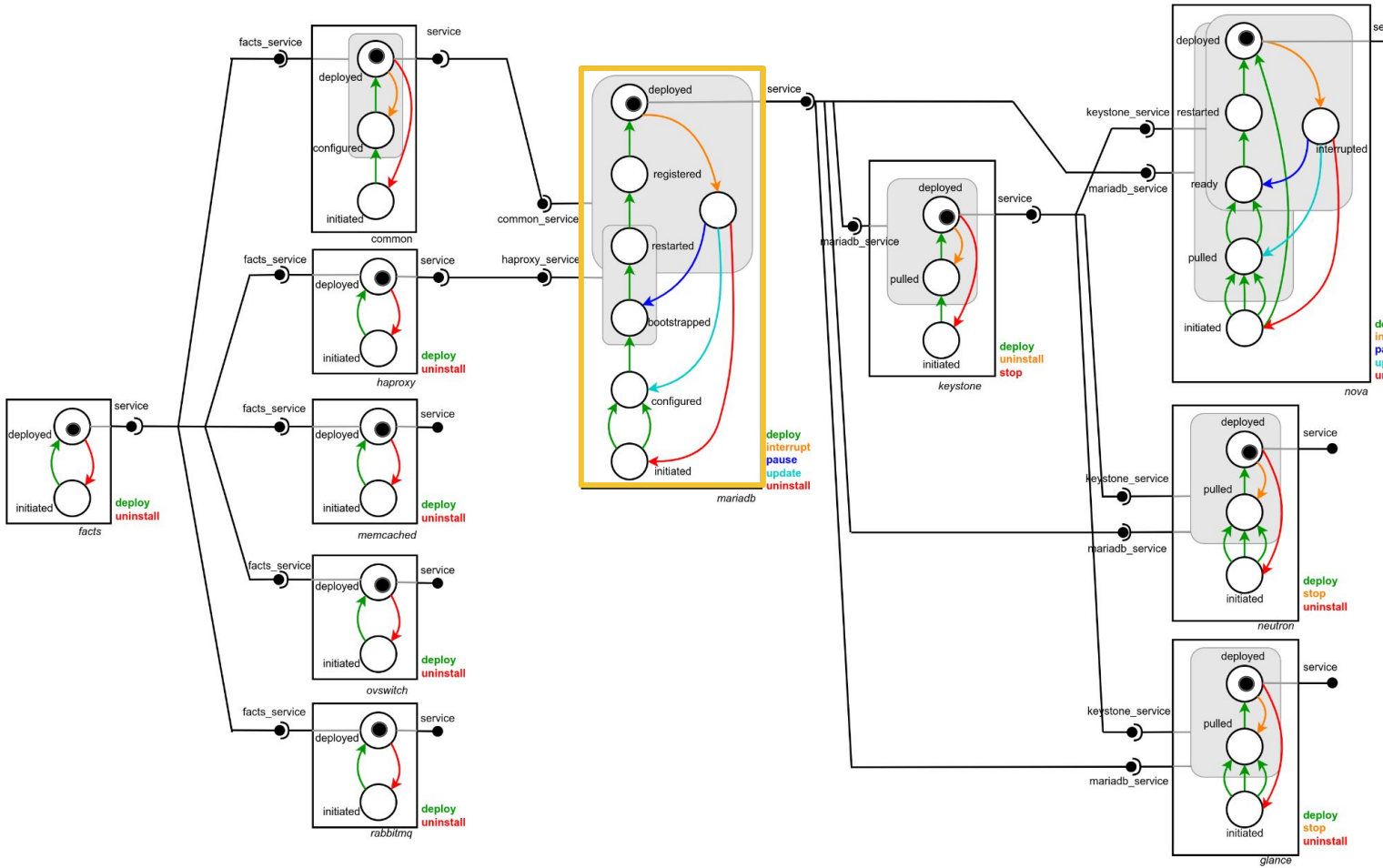
state = [s3 , s2 , s1 , s3 , s3 , s3 , s3 , s3 , s3 , s3 , s3]

Constraint resolution: MiniZinc into FlatZinc into Solvers



)

Example of Concerto-D model and reconfiguration goal: OpenStack



11 components, all **deployed:**

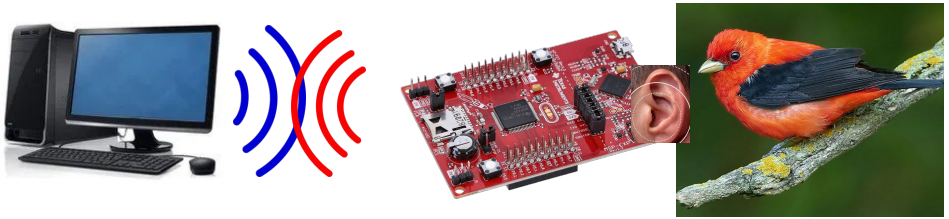
- *facts*
- *common*
- *haproxy*
- *memcached*
- *ovswitch*
- *rabbitmq*
- *mariadb*
- *keystone*
- *nova*
- *neutron*
- *glance*

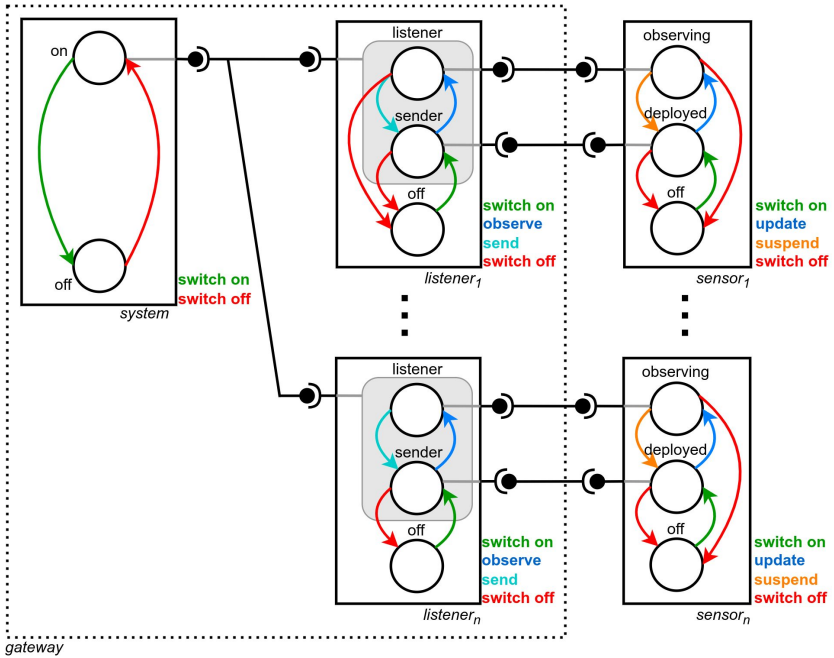
Goal:

update *mariadb*

goal(*mariadb*, update)
goal(*, **deploy**, final=true)

Example of Concerto-D model and reconfiguration goal: CPS





Collaboration with STR team from Centrale Nantes (w. Antoine BERNABEU)**:**

- CPS
  - Listen animals sounds
  - Communicate with a gateway
  - Need to be reconfigured
    - Update of system
    - Change freq. of observation

1 + 2n components:

- *system*
- *n listeners*
- *n sensors*

Goal: Reconfigure a sensor

State of the art of modeling languages for Fog

Concerto-D is not a language for modeling Fog systems

Table 2
A comparison of existing Fog modeling languages (✓ = feature supported, ~= feature partially supported).

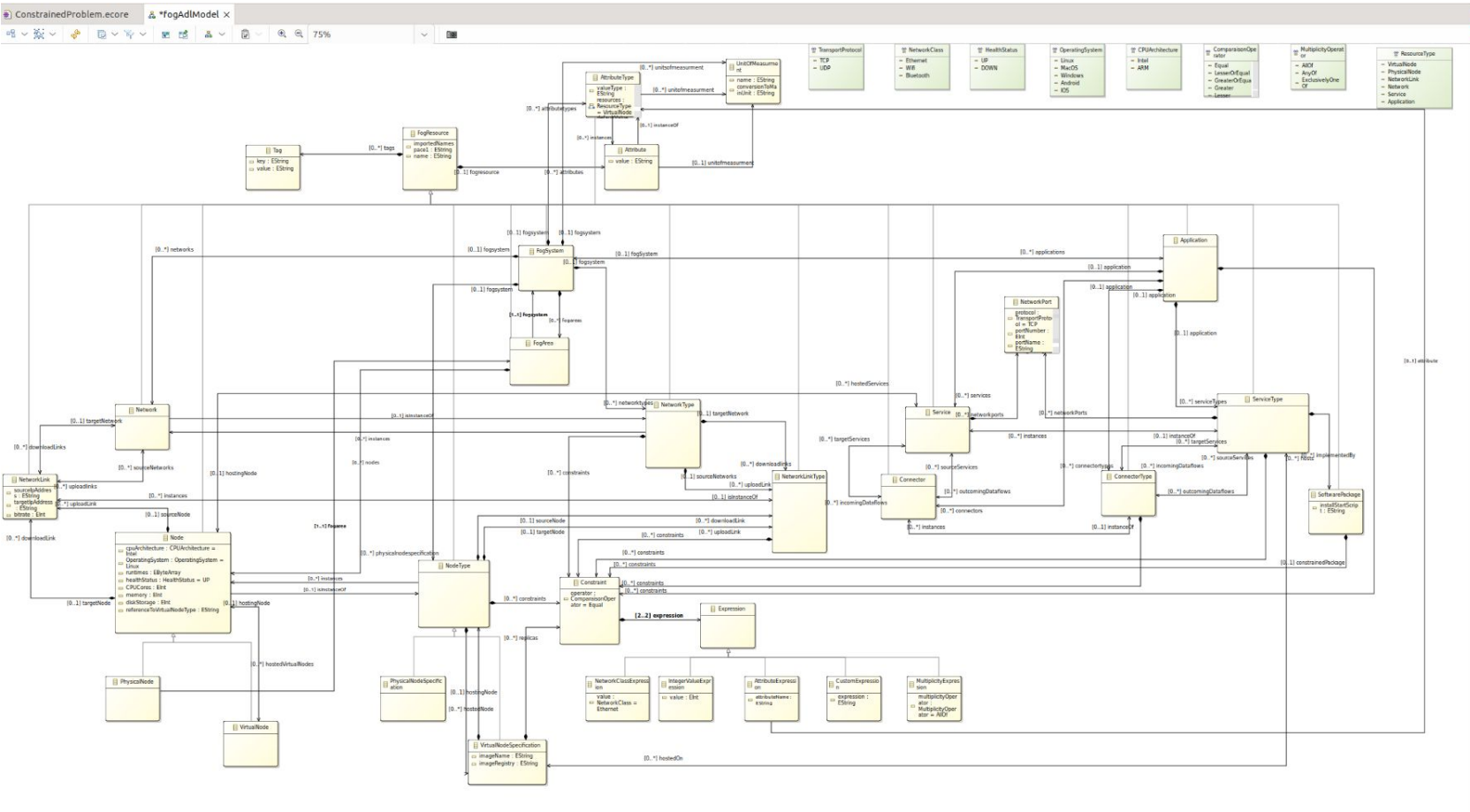
			Smada-Fog	Khebbeb et al.	Sahli et al.	FogDirSim	AcOP	DITAS	Engelsberger et al.	MobileFog	iFogSim	FogNetSim++	COMPSs	Extended TOSCA	CloudPath	Distributed Node-RED	YAFS	Fogify
Language																		
Scope	Dimension	Structure	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		Behavior	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓		
	Layer	IoT	✓		✓		✓	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓
		Fog	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
		Cloud	✓	✓	✓			✓	✓	✓	✓	~		✓	✓	✓	✓	✓	✓
	Architecture type	Areas			✓		✓			✓			~						
		Layers	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		Views							~										
	Control type	Centralized	✓	✓	✓	✓		✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
		Decentralized					~						~						
	Resources	Physical Res.	✓	✓	✓	✓	✓	~	✓	✓	✓	✓				✓	✓	✓	✓
		Software Res.	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	~	✓	
		Workflow Res.	✓	✓	✓	✓	✓	✓		✓	✓	✓					✓		
	Properties	Privacy/Security	~					~								~			
		Health			✓	✓								✓			✓	✓	
		Performance	✓	✓	✓	✓	✓		✓	✓	✓	~	✓	✓	~	✓	✓	✓	
		Energy	✓		✓	✓					✓	✓							
		Other properties						✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
	Genericity	Domain-specific				✓	✓		✓				✓				✓		
		Domain-independent	✓	✓	✓			✓		✓	✓	✓		✓	✓	✓			✓
Definition																			
	Abstract syntax	Standard-based						✓						✓		✓			
		Custom	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓		✓		✓	✓	
	Concrete syntax	Graphical	✓		✓			✓	✓		✓	✓		✓		✓			
		Textual		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓	
	Semantics	Formal		✓	✓	✓			✓										
		Semi-formal	✓				✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Extension mechanism						✓	✓	✓	✓	~			✓		✓	✓		
Support																			
Implementation	Open source		✓			✓	✓				✓	✓	✓	✓		✓	✓	✓	
	Proprietary																		
Capabilities	Editing		✓	✓							✓	✓	✓	✓		✓			
	Model transformation		✓											~					
	Code generation		✓																
	V&V			✓	✓														
	Simulation		✓	✓		✓					✓	✓					✓	✓	
	Execution					✓	✓			✓			✓		✓	✓			
	Adaptation	Scheduling	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓		
		Offloading		✓			✓					✓							
		Reconfiguration	✓		✓										✓			✓	
		Scaling	✓	✓	✓	✓				✓				✓	✓				
		Other adaptation		✓	✓			✓	✓		✓							✓	
Interoperability	Standard languages		✓			✓					✓			✓	✓		✓	✓	
	Existing tools		✓			✓								✓		✓	✓	✓	
Exploitation	Design time		✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	
	Runtime		✓	✓	✓		✓	✓	✓	✓			~	✓	✓				
Validation	Experiments		✓			✓	✓	✓		✓	✓	✓	✓		✓		✓	✓	
	Use cases		✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Documentation	Repository		~			✓					✓	✓	✓	✓		✓	✓	✓	
	Community										✓		✓						
	Commercial support																		

A **survey** of languages for modeling Fog:

Abdelghani Alidra, Hugo Bruneliere, Thomas Ledoux.
A feature-based survey of Fog modeling languages.
Future Generation Computer Systems, 2023, 138, pp.104-119.
⟨10.1016/j.future.2022.08.010⟩.

- Lack of homogenization
- No separation of concerns
- Need for multiple representations and abstractions
- Lack of extensibility and refinement capabilities
- Security and privacy not represented
- …

SeMaFoR's FML and VeriFog

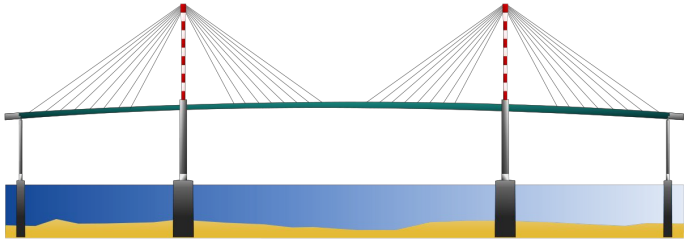


Future work: FML model to Concerto-D

Model verification

VeriFog

EMF Model of FML



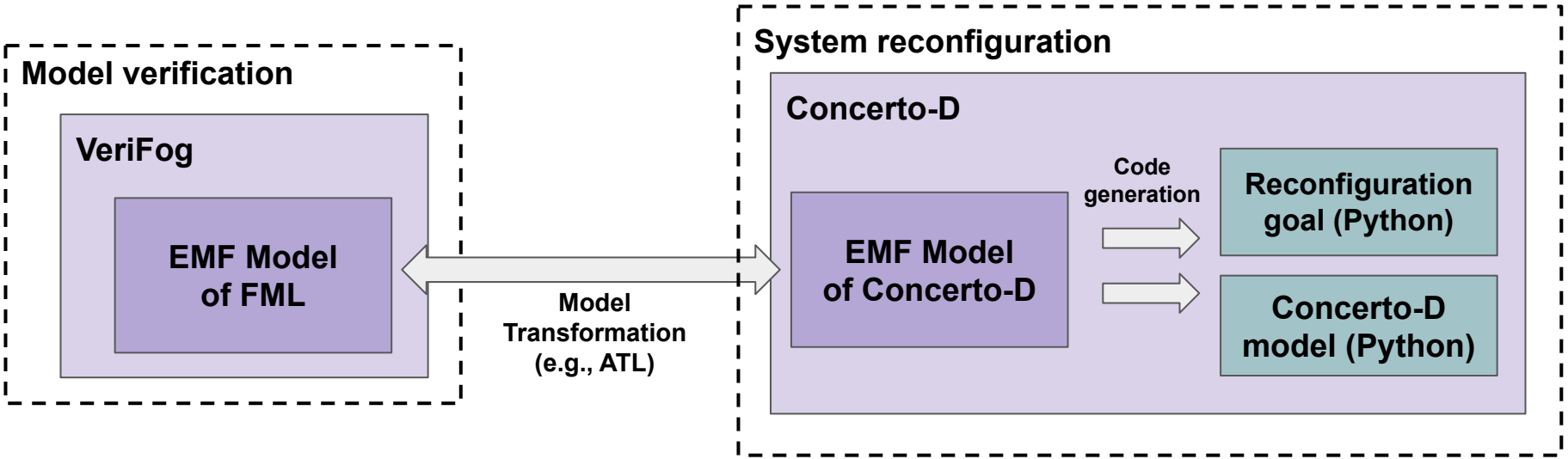
System reconfiguration

Concerto-D

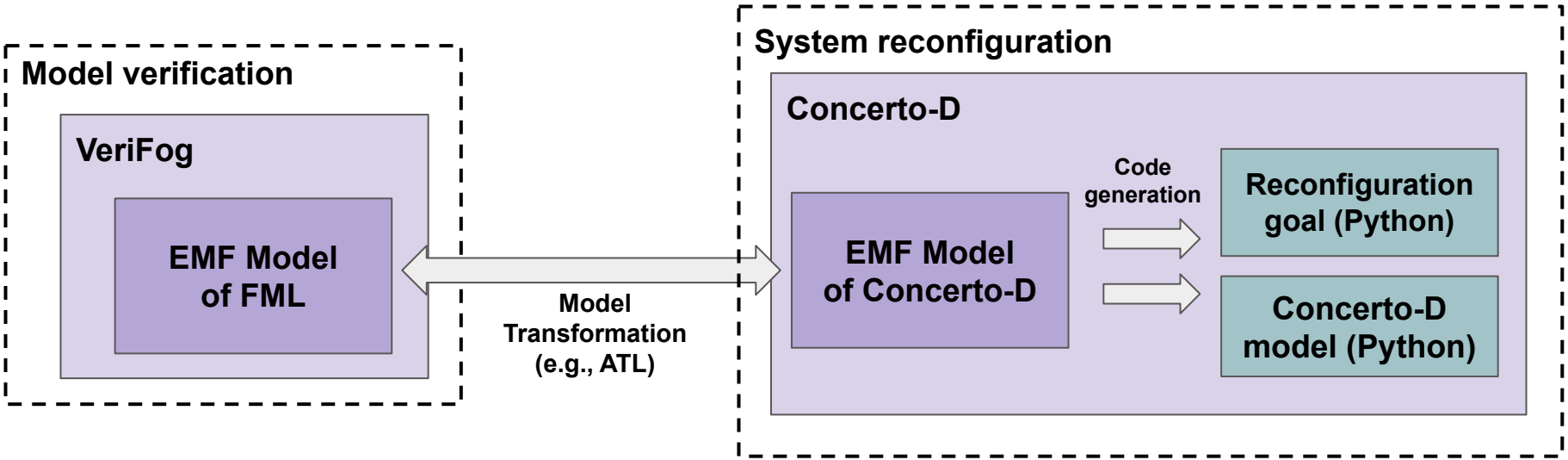
Reconfiguration goal (Python)

Concerto-D model (Python)

MDE approach



Student project? Internship?



- Task 1: Model Concerto-D using EMF
- Task 2: Allow Python code generation from EMF model of Concerto-D
- Task 3: Study FML and (probably) extend it
- Task 4: Write FML2ConcertoD transformation
- Task 5: Write a workshop article

Concluding remarks

Postdoc contributions

- Concerto-D and SeMaFoR project
- Infer reconfiguration actions (CP-based approach)
- Communication protocol

Target applications:

- (SeMaFoR) Smart cities, smart buildings, smart factories, etc.
- CPS nodes

Perspectives:

- Benchmarking (solvers, comm. protocols, dist. architectures)
- Optimization of plan (energetic cost, time, financial cost)
- MDE approach for bridging Concerto-D to Fog models

Questions ?

References:

[Cisco, mar. 2015] Maher Abdelshkour. From Cloud to Fog Computing. Cisco, 2015

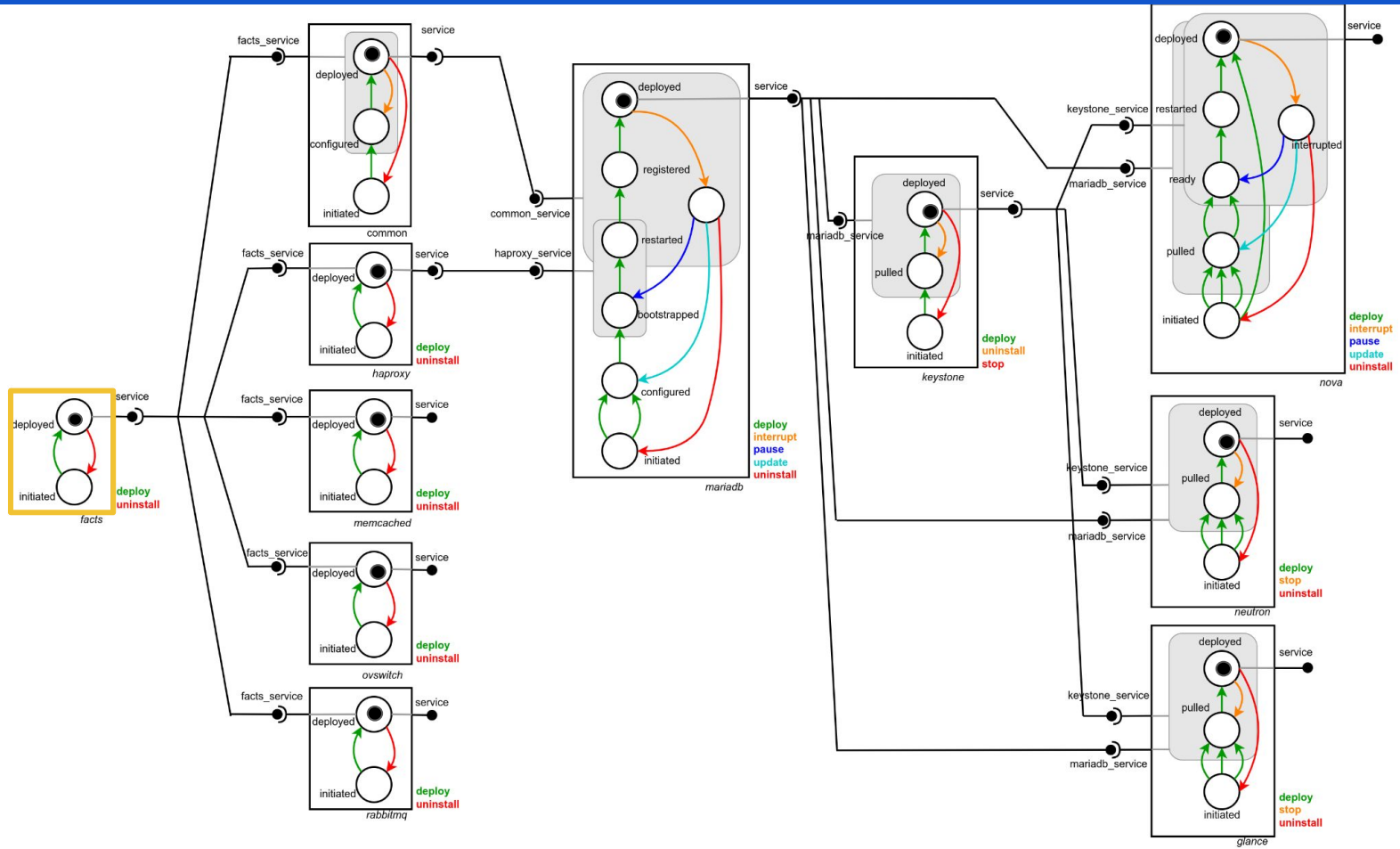
[IBM, 2006] A. Computing et al. An architectural blueprint for autonomic computing. IBM White Paper, 2006.

[SeMaFoR, 2023] SeMaFoR - Self-Management of Fog Resources with Collaborative Decentralized Controllers

[Robillard, apr. 2022] Simon Robillard et al. SMT-Based Planning Synthesis for Distributed System Reconfigurations. FASE 2022

Backup

Example of stratified assembly and reconfiguration



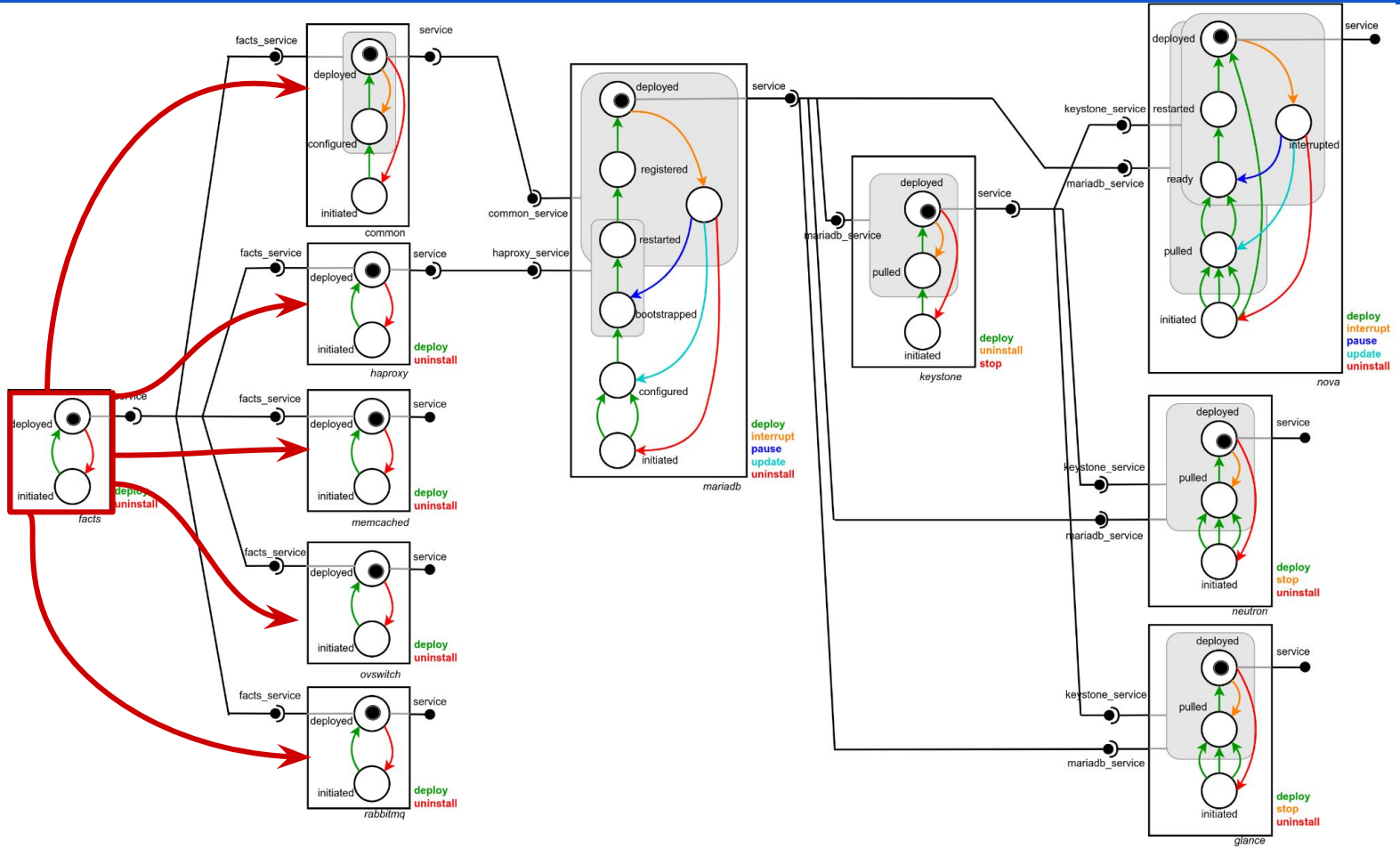
11 components, all **deployed:**

- *facts*
- *common*
- *haproxy*
- *memcached*
- *ovswitch*
- *rabbitmq*
- *mariadb*
- *keystone*
- *nova*
- *neutron*
- *glance*

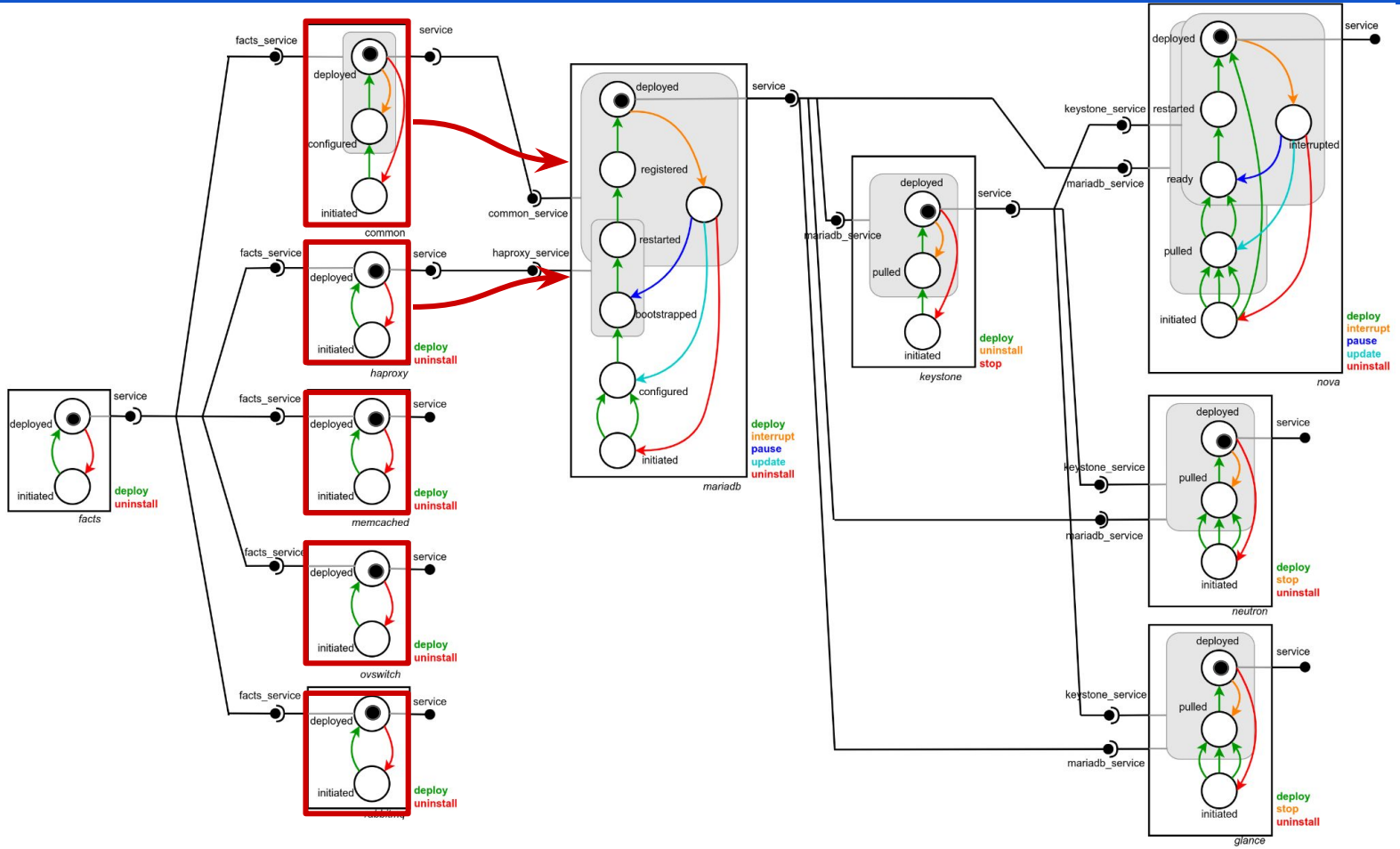
Goal: reboot *facts*

facts
pushB(facts, uninstall) pushB(facts, deploy)

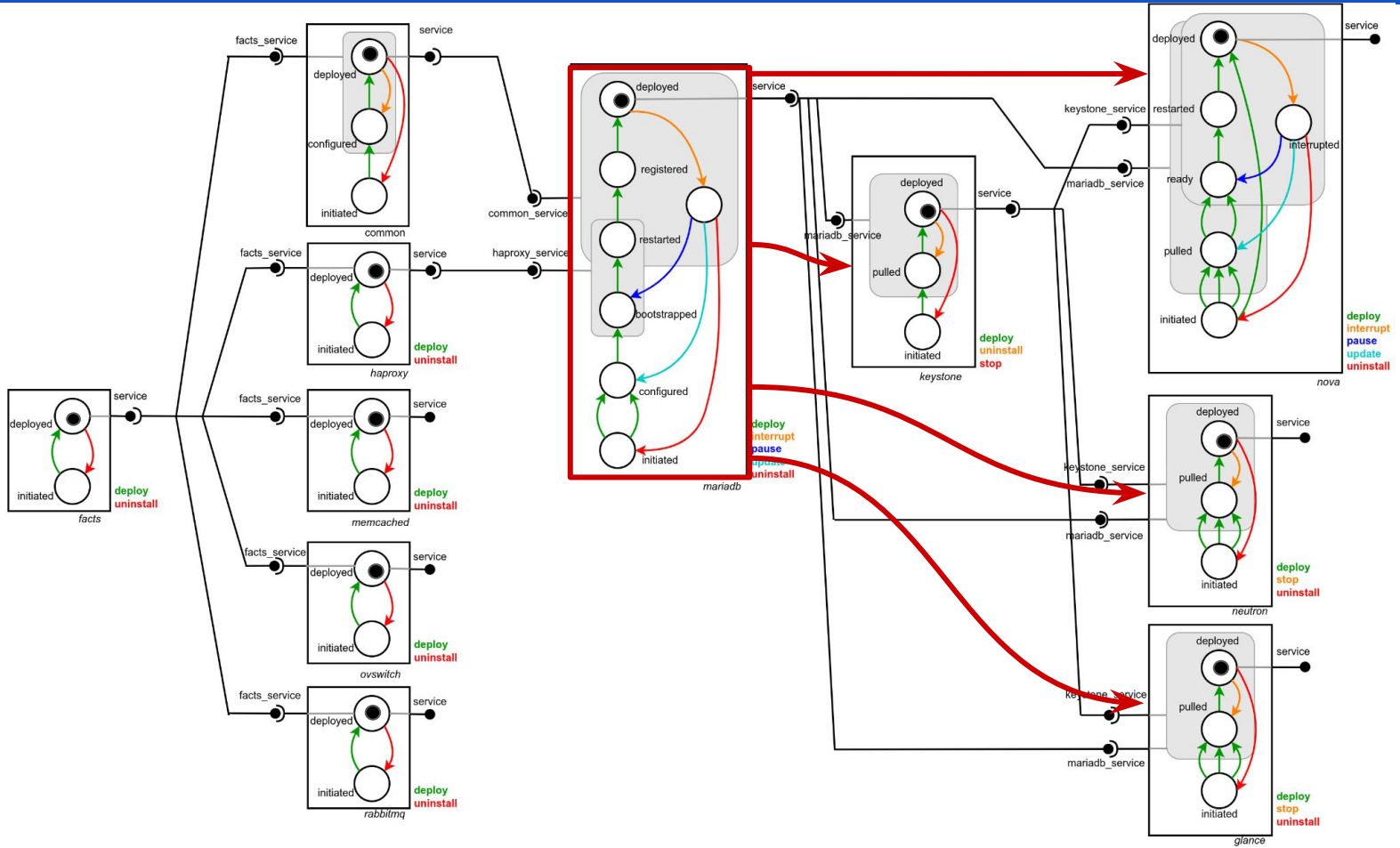
Information sharing protocol - Step I: Propose



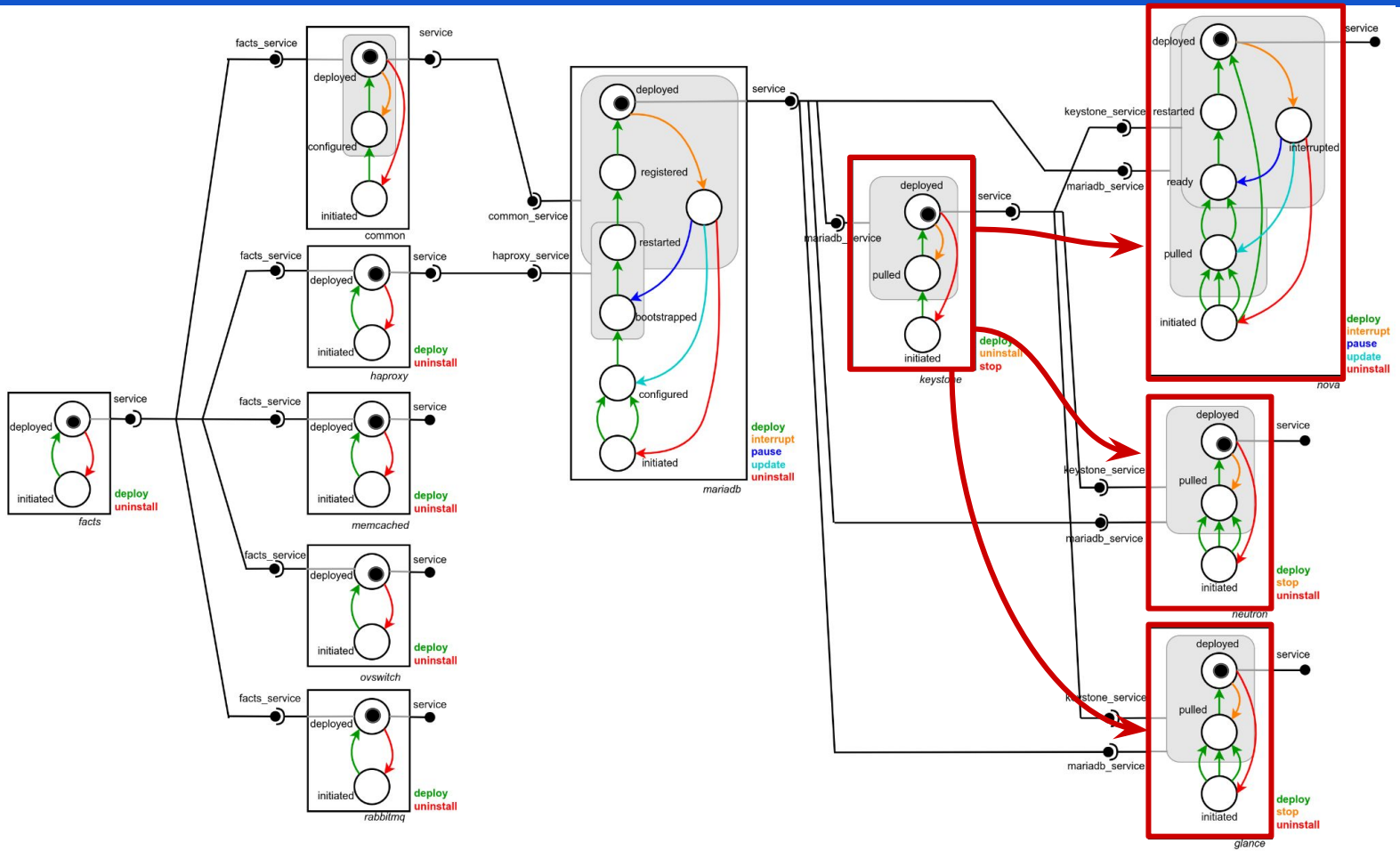
Information sharing protocol - Step I: Propose



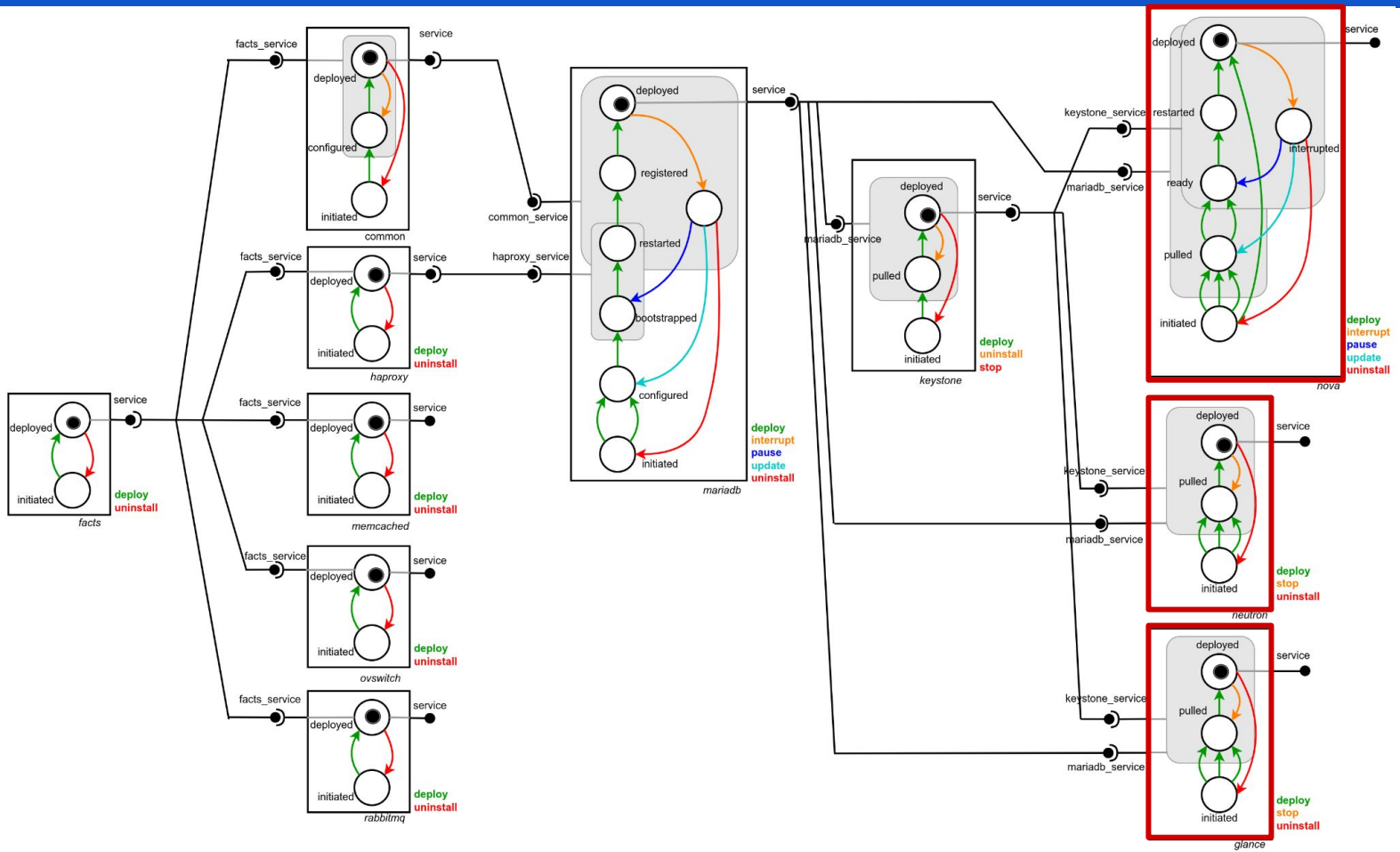
Information sharing protocol - Step I: Propose



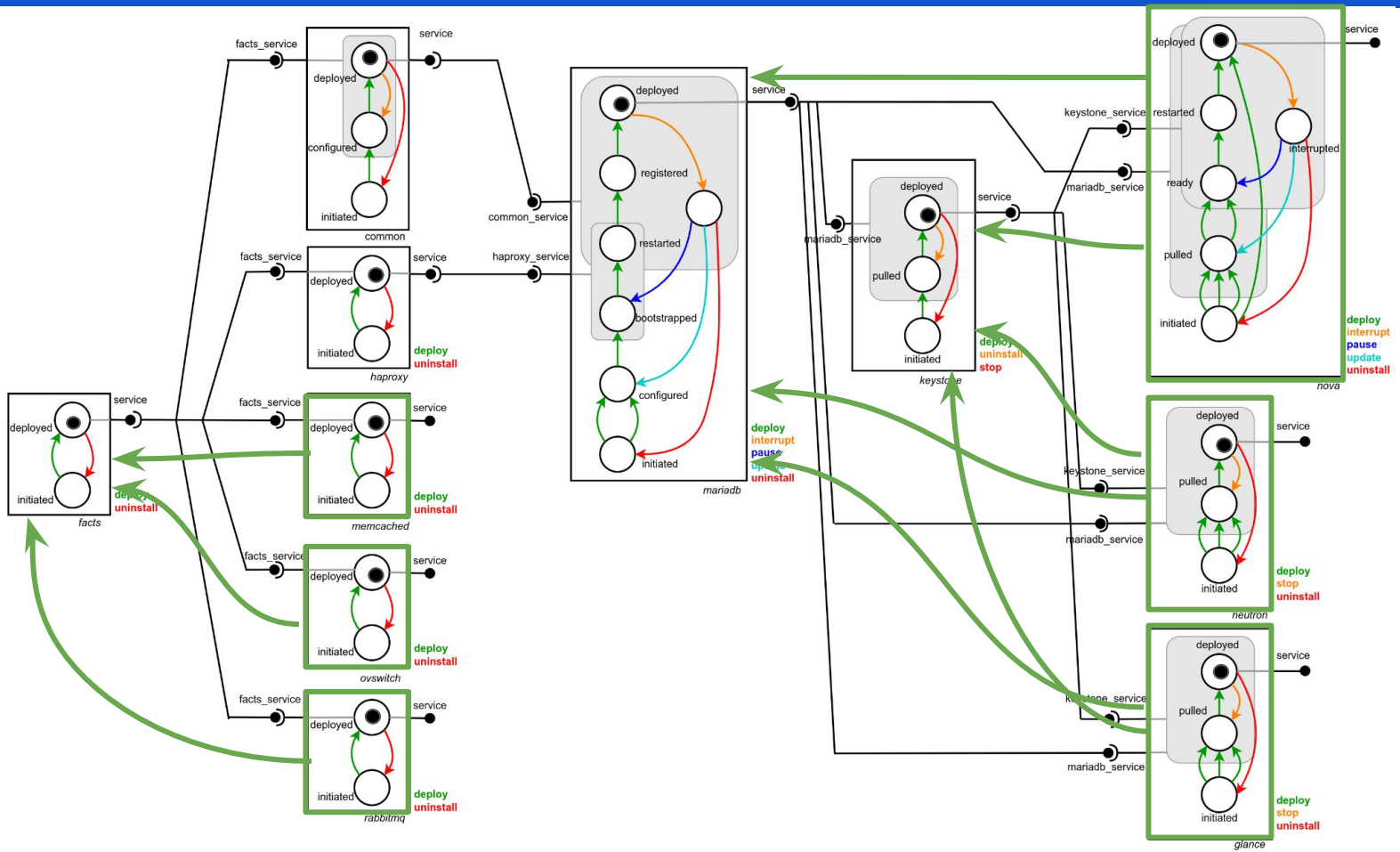
Information sharing protocol - Step I: Propose



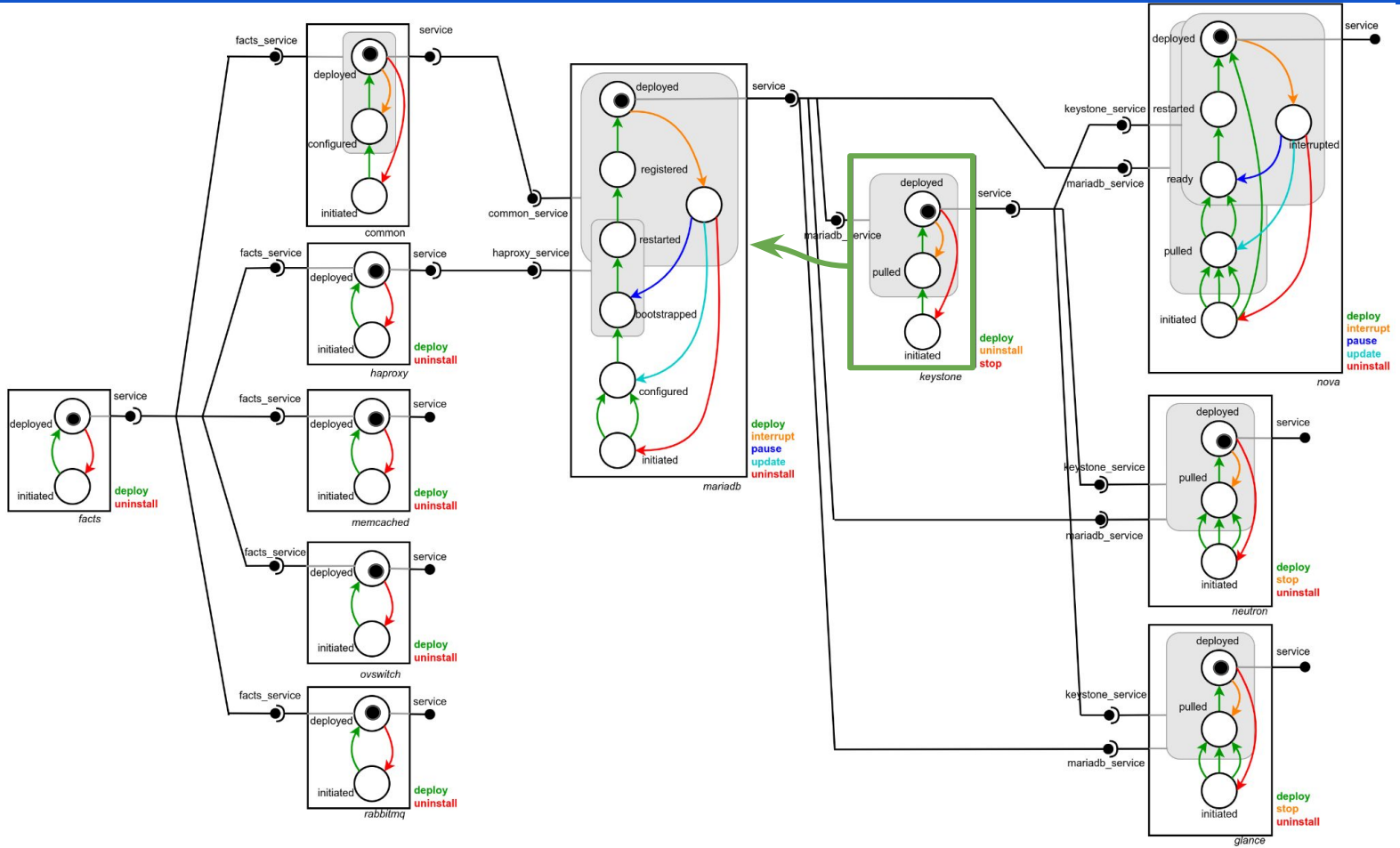
Information sharing protocol - Step I: Propose



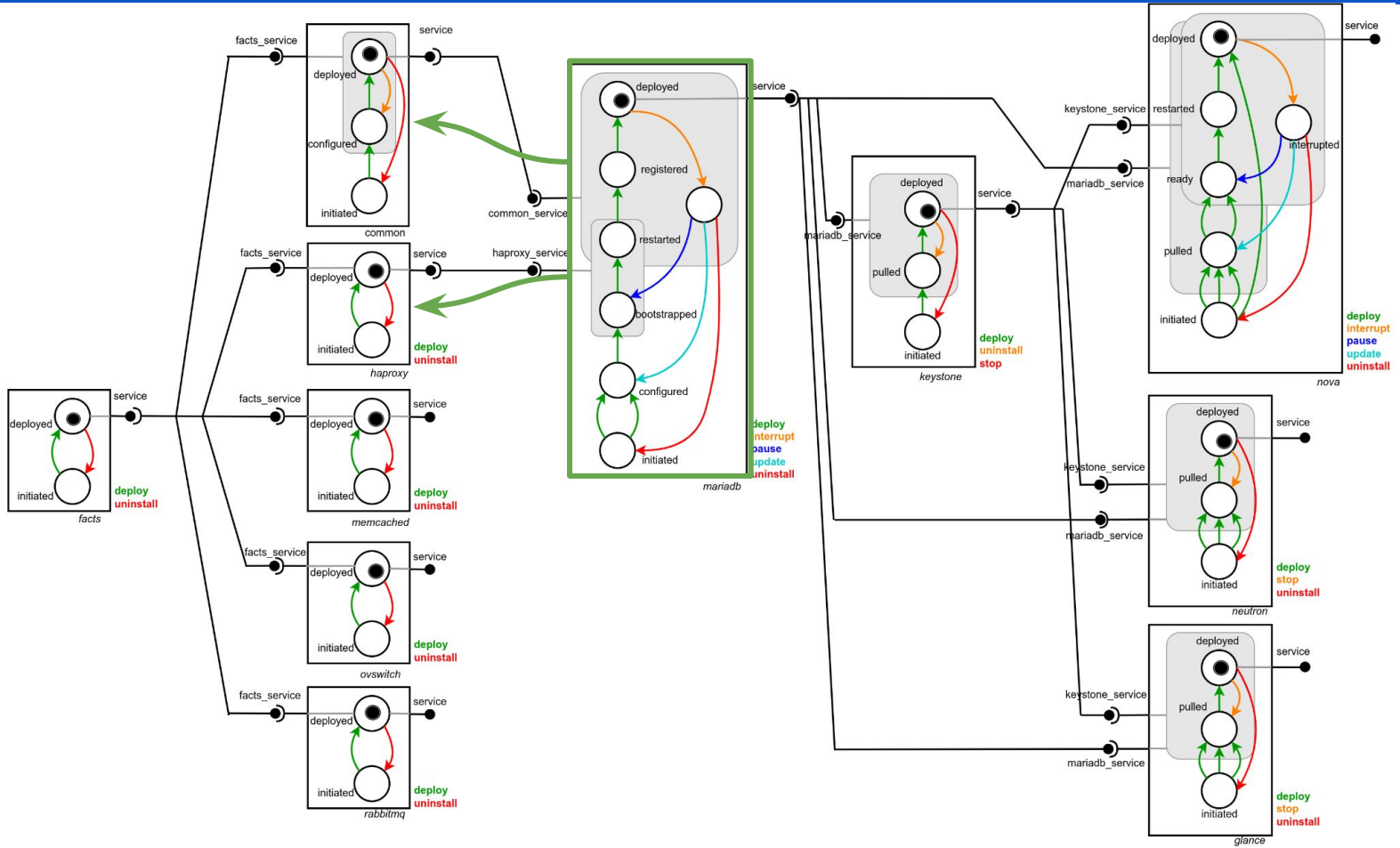
Information sharing protocol - Step II: Send ack



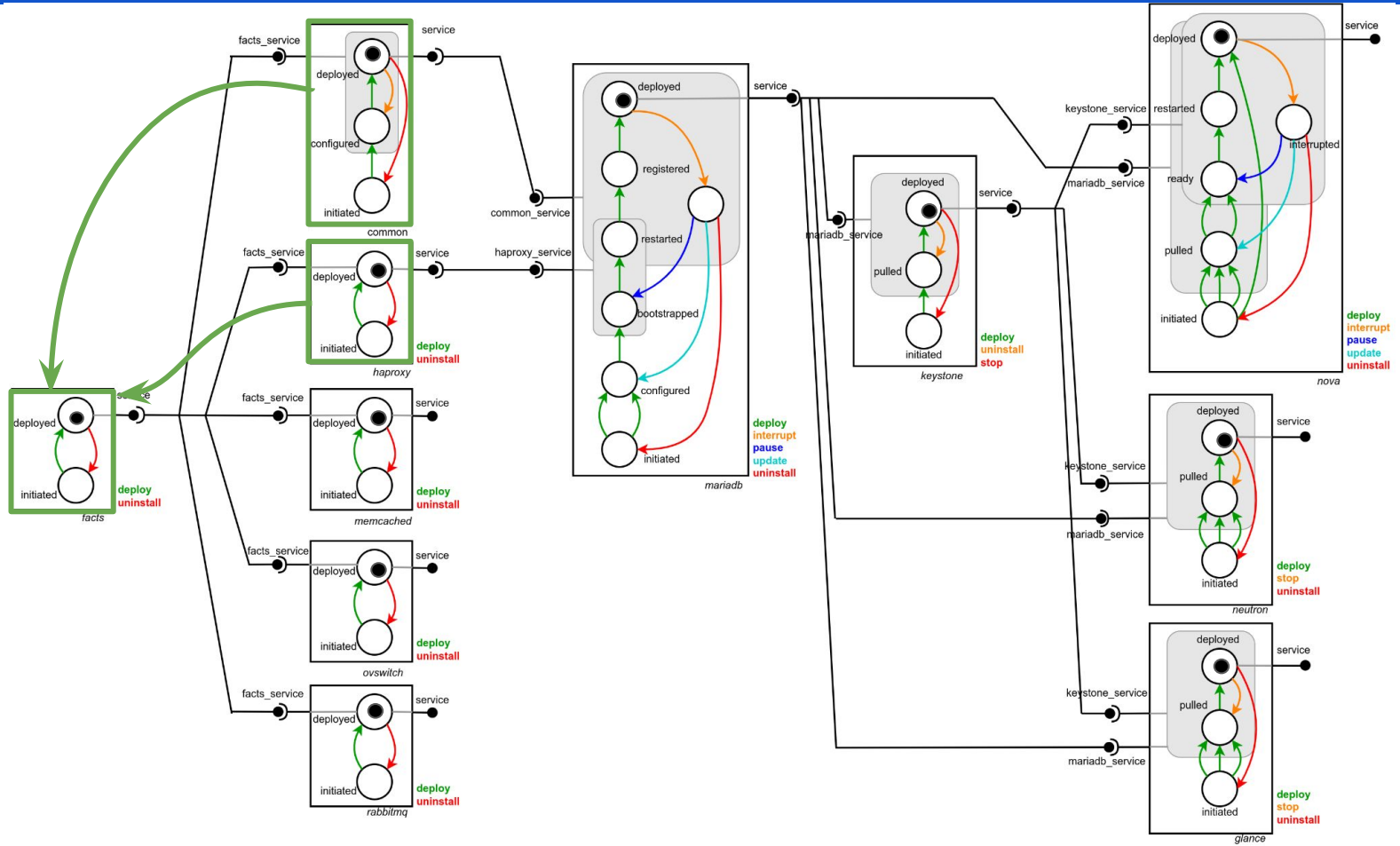
Information sharing protocol - Step II: Send ack



Information sharing protocol - Step II: Send ack



Information sharing protocol - Step II: Send ack



Information sharing protocol - Step III: Global ack from root