SeMaFoR Project
& Decentralized reconfiguration plan synthesis

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

Seminaire DAPI
6th June 2023

Inria

LS2N

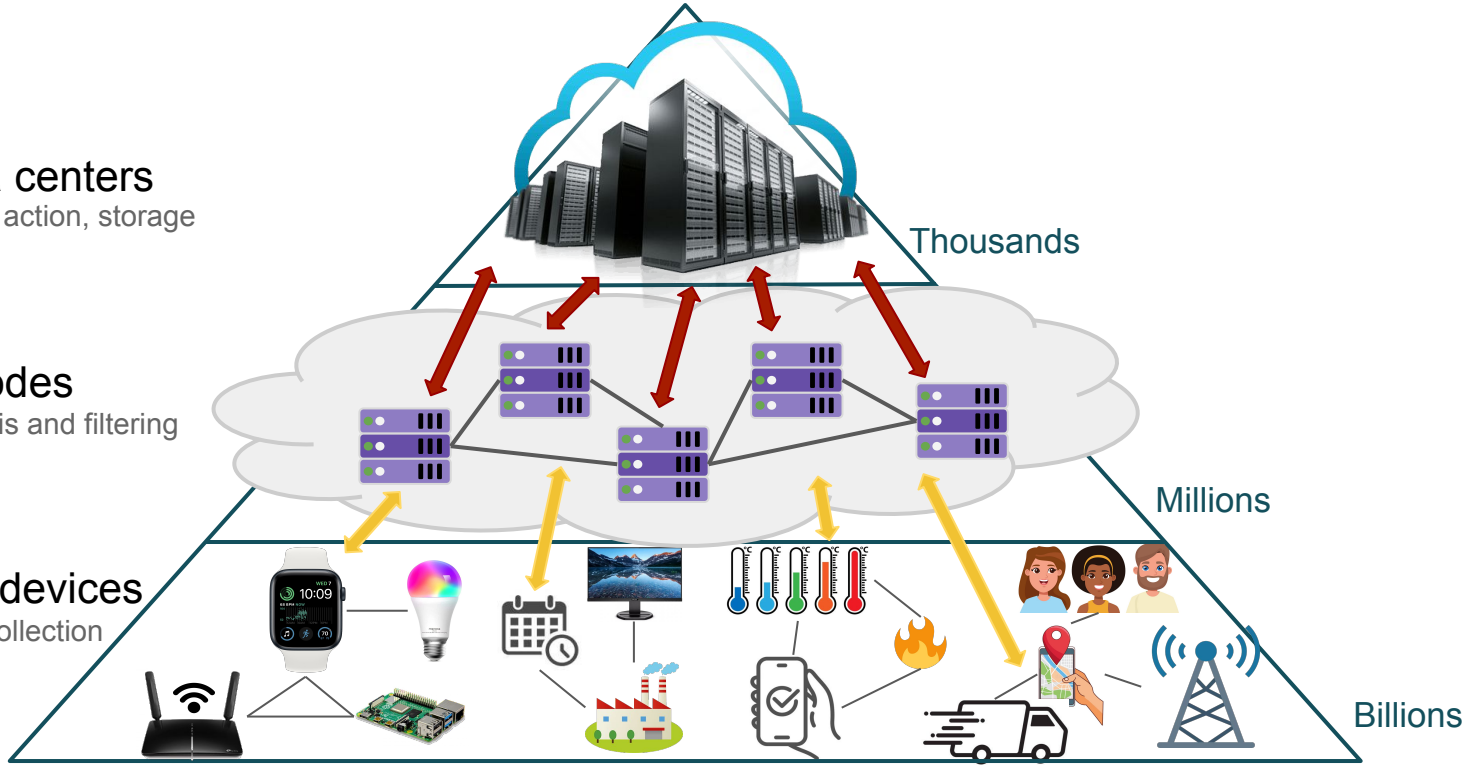
PROJET FINANCÉ PAR L'ANR - PROJECT FUNDED BY THE ANR
ANR

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



"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.

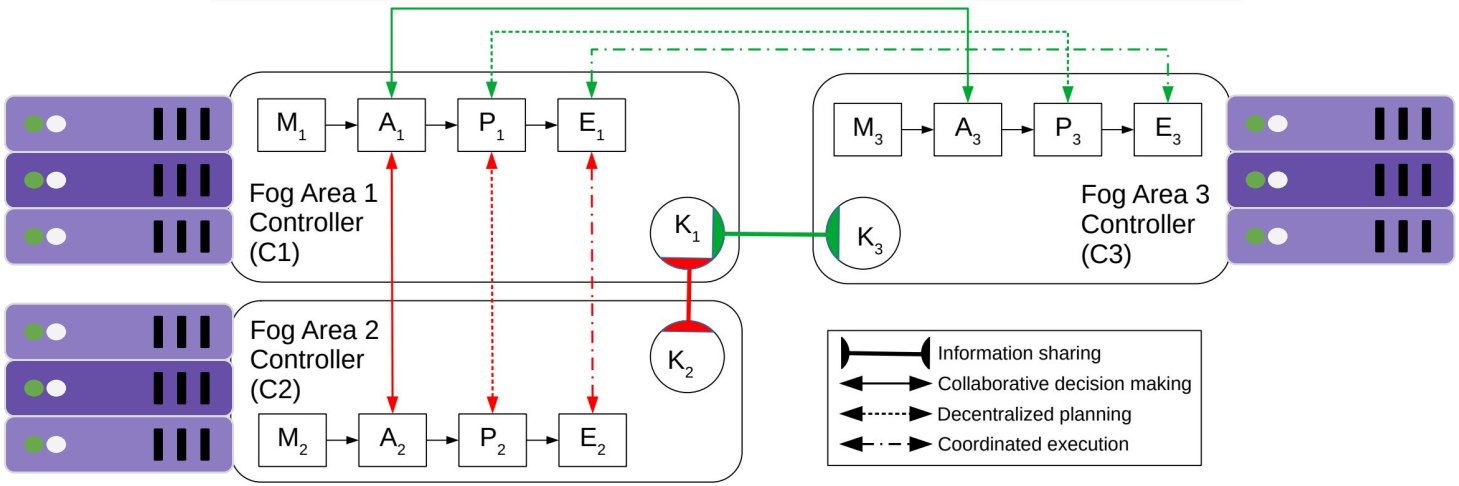
STACK TASC naomod alter way SMILE

SORBONNE UNIVERSITÉ CRÉATEURS DE FUTURS DEPUIS 1257 LIP6

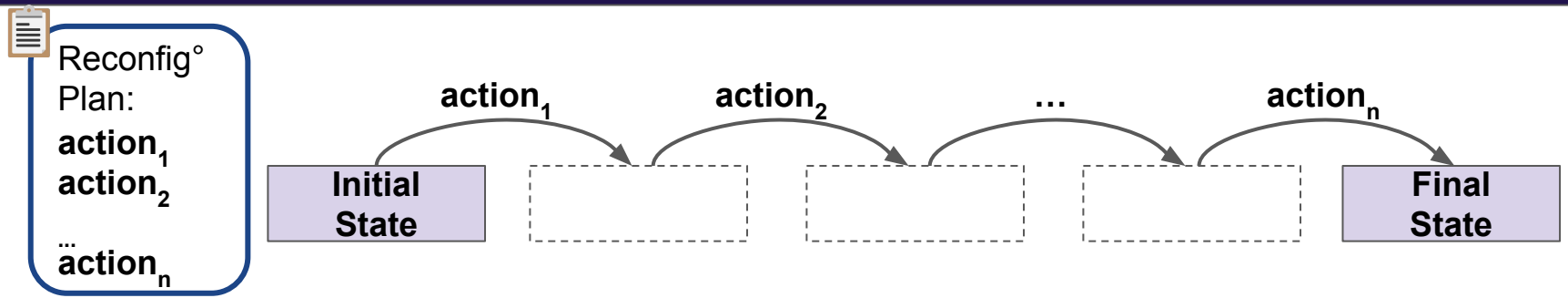
SeMaFoR proposal for controller coordination

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 of Fog resources



Postdoc objectives:

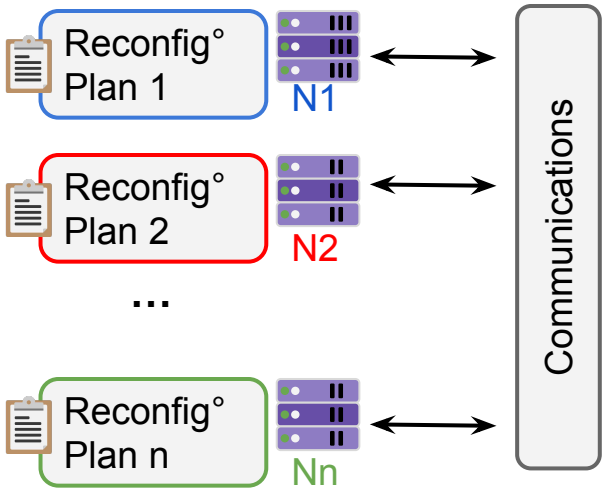
- ➢ Infer reconfiguration actions
- ➢ Optimal overall reconfiguration

Challenges:

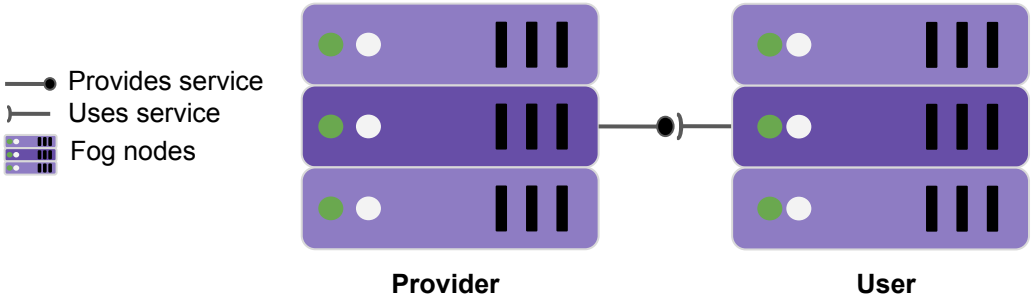
- Locally: partial view of the system
- Collaboration with other nodes

Inspiration:

- SMT-based [Robillard, apr. 2022]



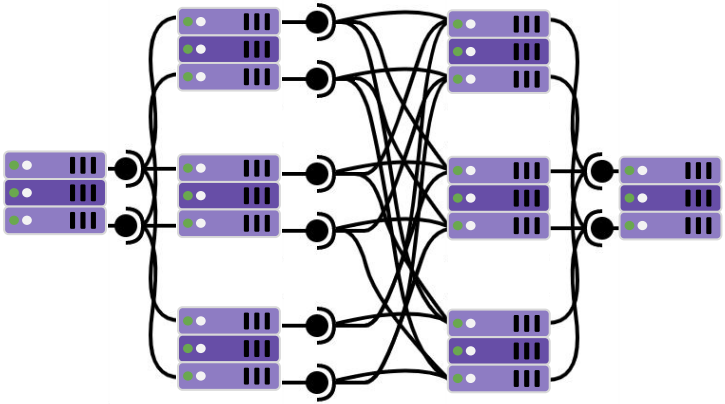
Constraint with providers and users



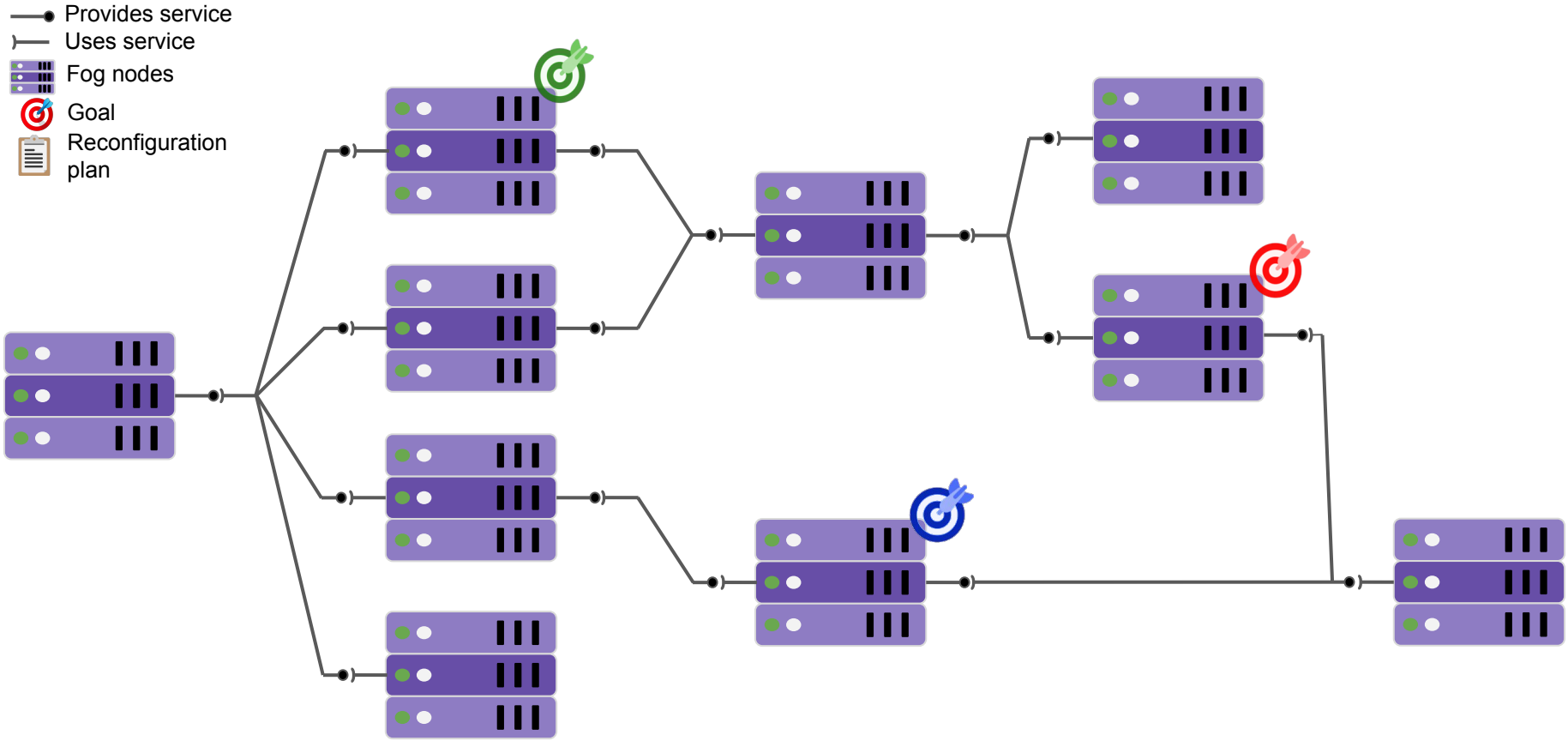
Nodes are connected using interfaces to:

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

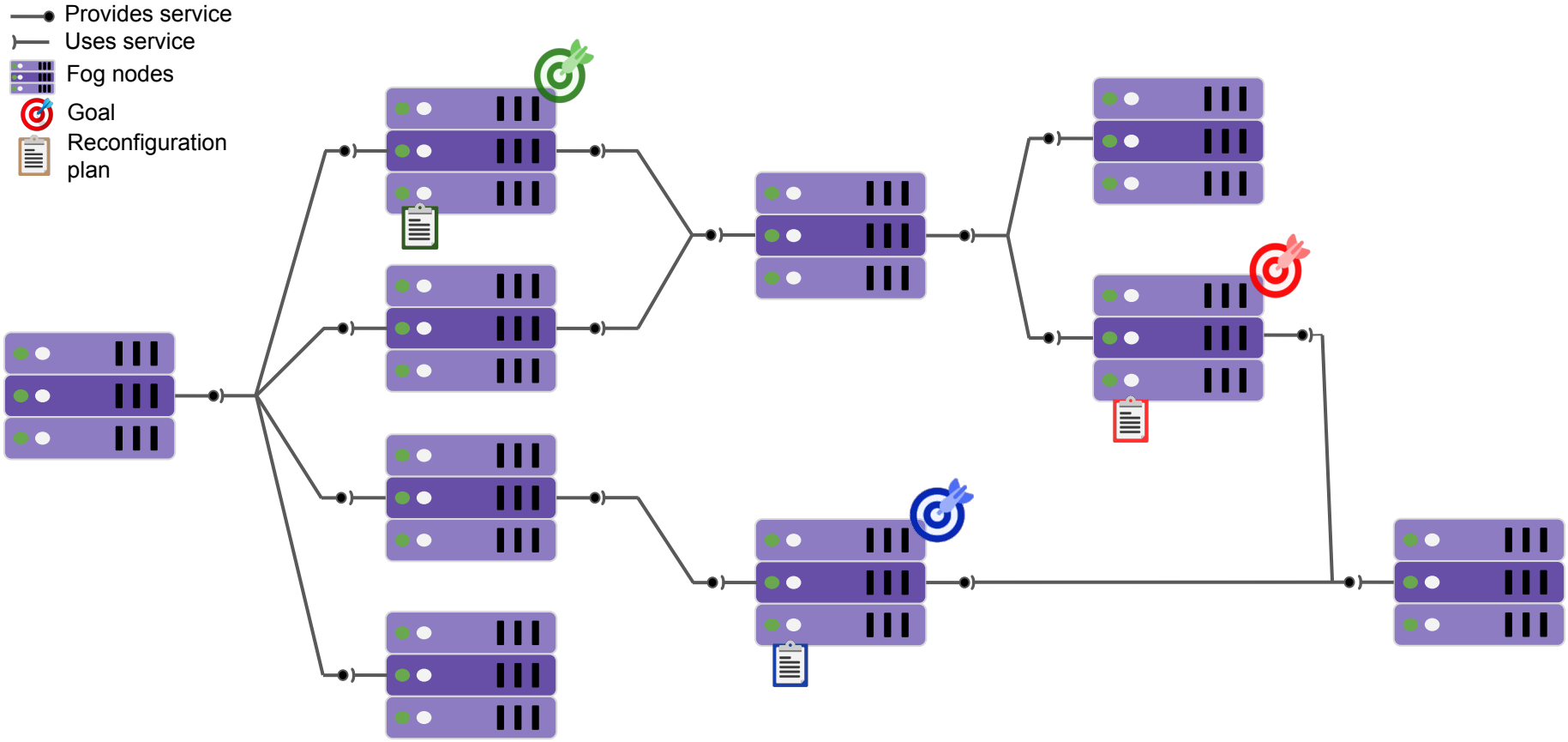
creating coordination constraints (behavioral and sync.)



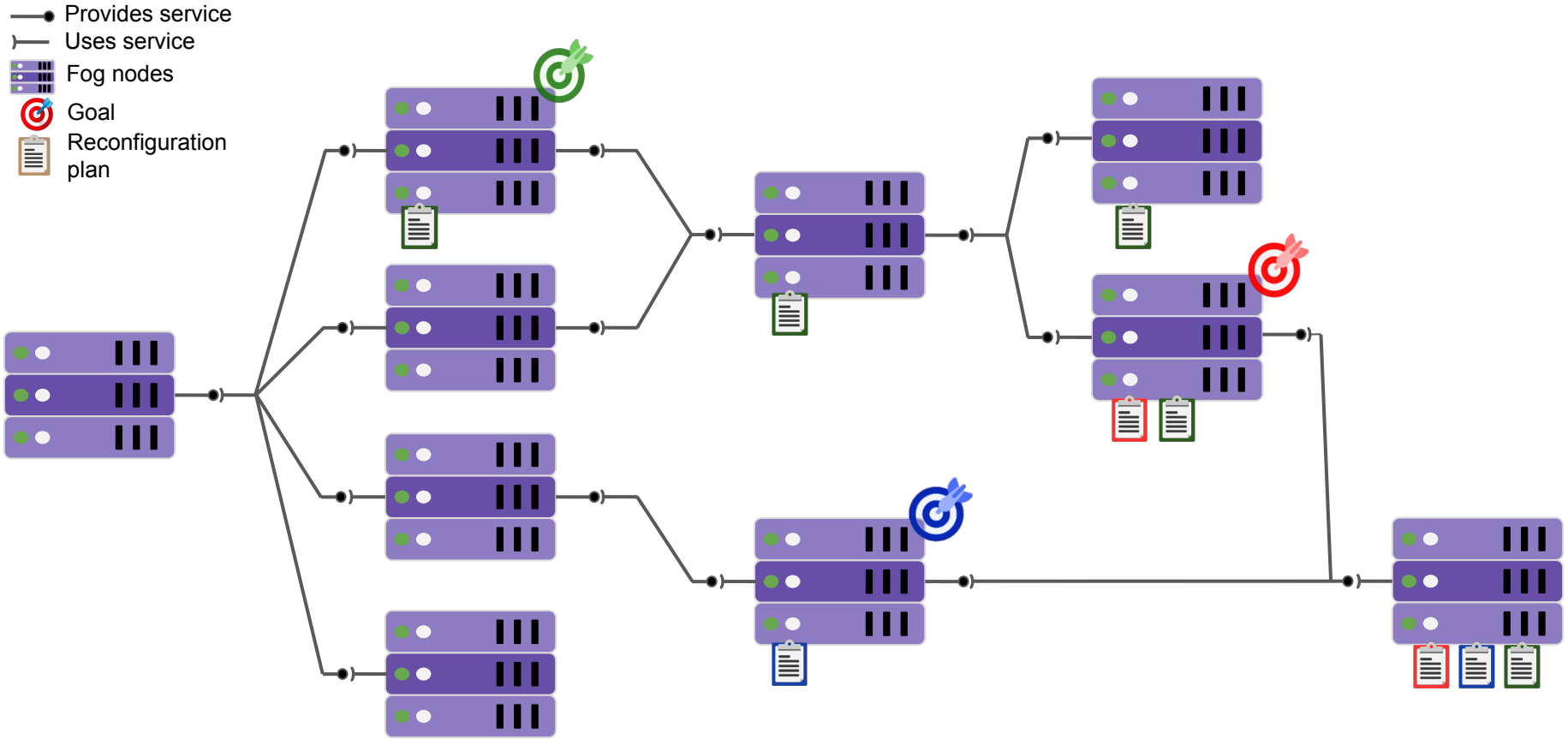
Reconfiguration of Fog resources: Local goal



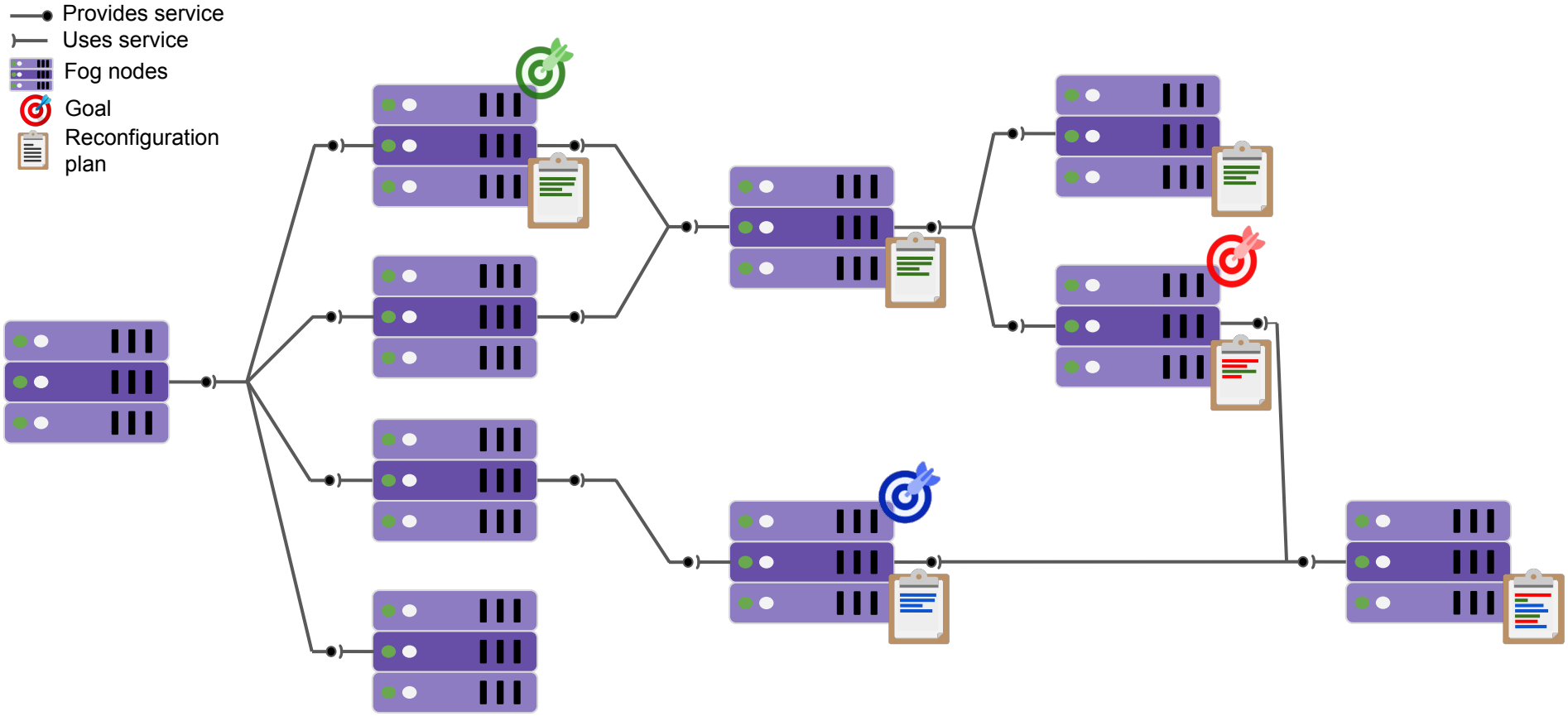
Reconfiguration of Fog resources: Local decision



Reconfiguration of Fog resources: Local decision propagation

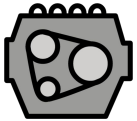


Reconfiguration of Fog resources: Local plan (Sync + Optimization)



Approach

- **Sharing protocol** with message passing (rumor-spreading)
  - **<u>Local decision</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

Concluding remarks

Contributions

- Infer reconfiguration actions (CP-based approach)
- Communication protocol

Target applications for SeMaFoR projects:

- Smart cities, smart buildings, smart factories, etc.
- CPS nodes with Distributed Arctic Observatory

Perspectives:

- Explore communication protocols
- Benchmarking (≠ solvers, dist. architectures)
- Optimization of plan (energetic cost, time, financial cost)

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

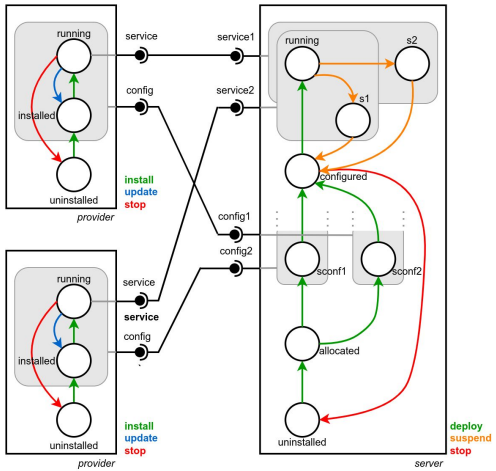
[Robillard, apr. 2022] Simon Robillard et al. SMT-Based Planning Synthesis for Distributed System Reconfigurations. FASE 2022

Backup

Fog Modeling: ADL and Concerto-D

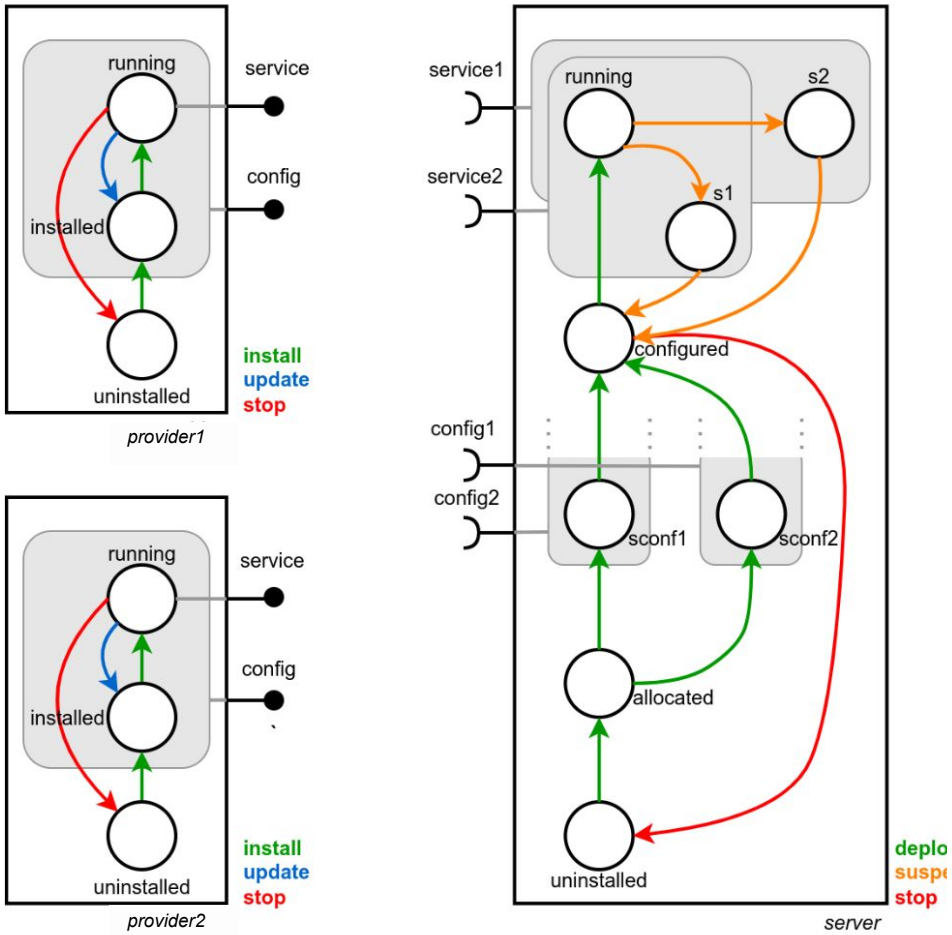
A **survey** of languages for modeling Fog:

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⟩. ⟨hal-03759010⟩



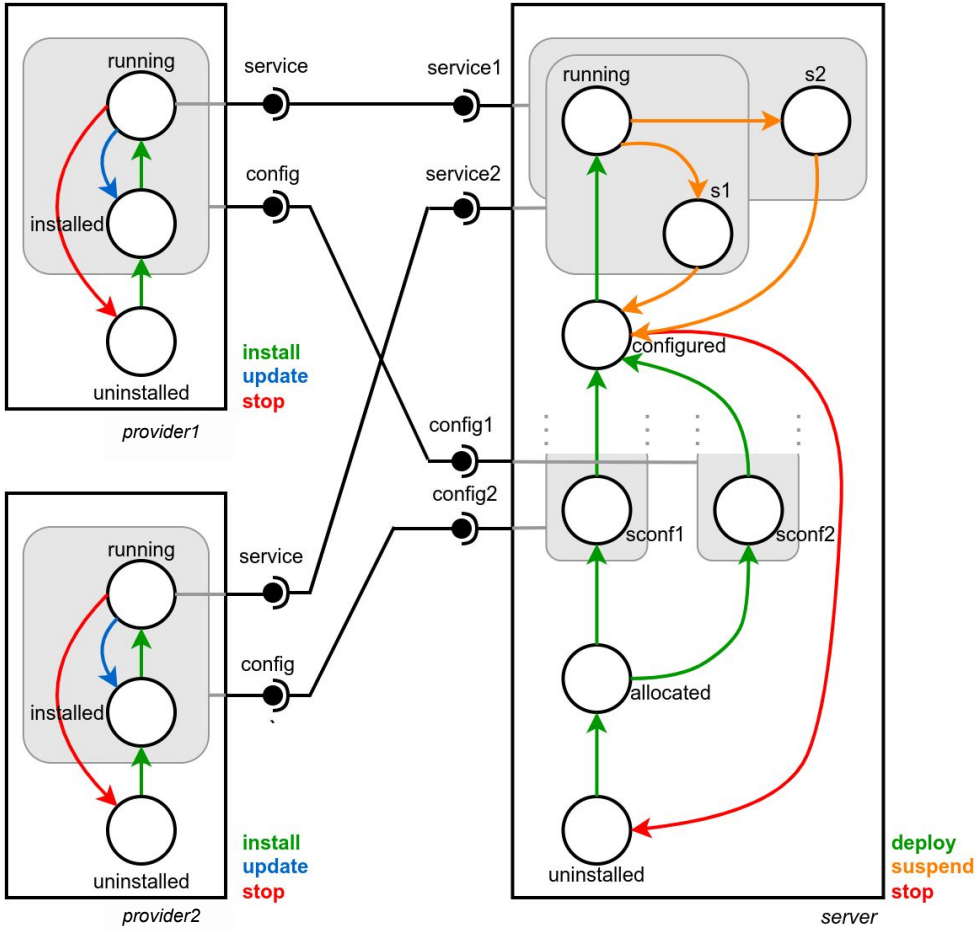
- **ADL** for modeling and verifying properties on Fog
- **Concerto-D: A reconfiguration language** for decentralized components
  - Involved components
  - Interactions / connections between components
  - Changes in the component

Fog Modeling: Concerto-D example



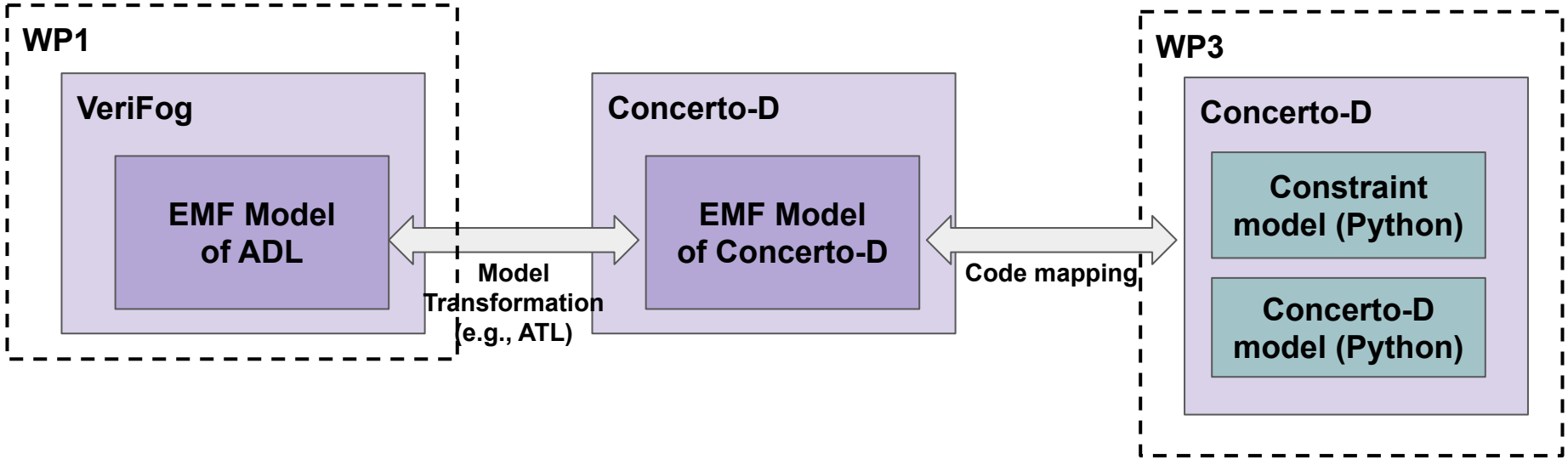
add("*provider1*", Provider)
add("*provider2*", Provider)
add("*server*", Server)

Fog Modeling: Concerto-D example



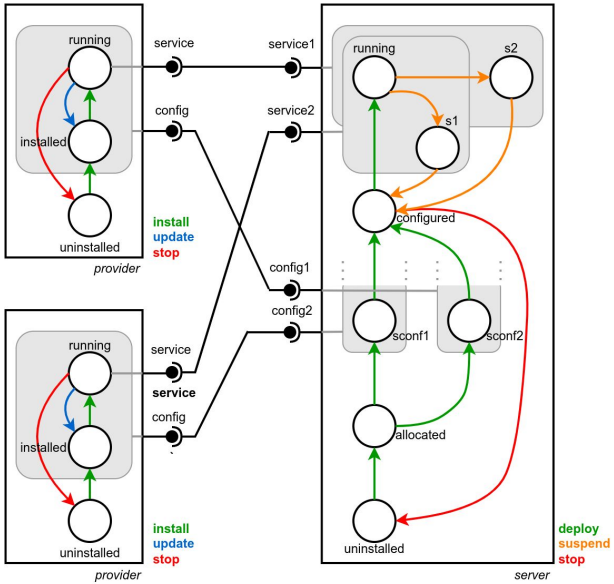

```
add("provider1", Provider)
add("provider2", Provider)
add("server", Server)
connect("provider1", "service",
        "server", "service1")
connect("provider1", "config",
        "server", "config1")
connect("provider2", "service",
        "server", "service2")
connect("provider2", "config",
     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

Model transformation (Future internship)



- Task 1: Modéliser Concerto-D en UML et l'implementer en EMF
- Task 2: Ecrire une transformation Model2Text pour générer du code Python
- Task 3: (Probablement) Etendre l'ADL, pour pouvoir se caler sur Concerto-D
- Task 4: Ecrire la transformation ADL2Concerto-D
- Task 5: Ecrire un article de Workshop

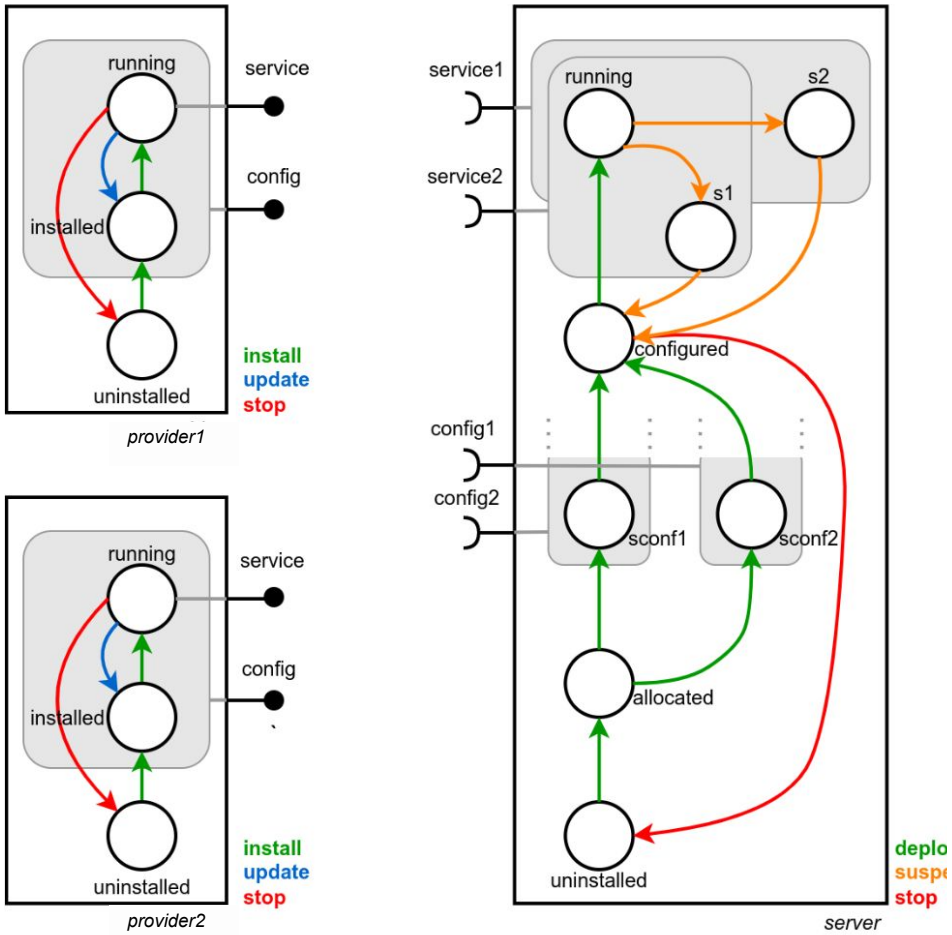
Fog Modeling: Concerto-D



Concerto-D: A reconfiguration language
for decentralized components

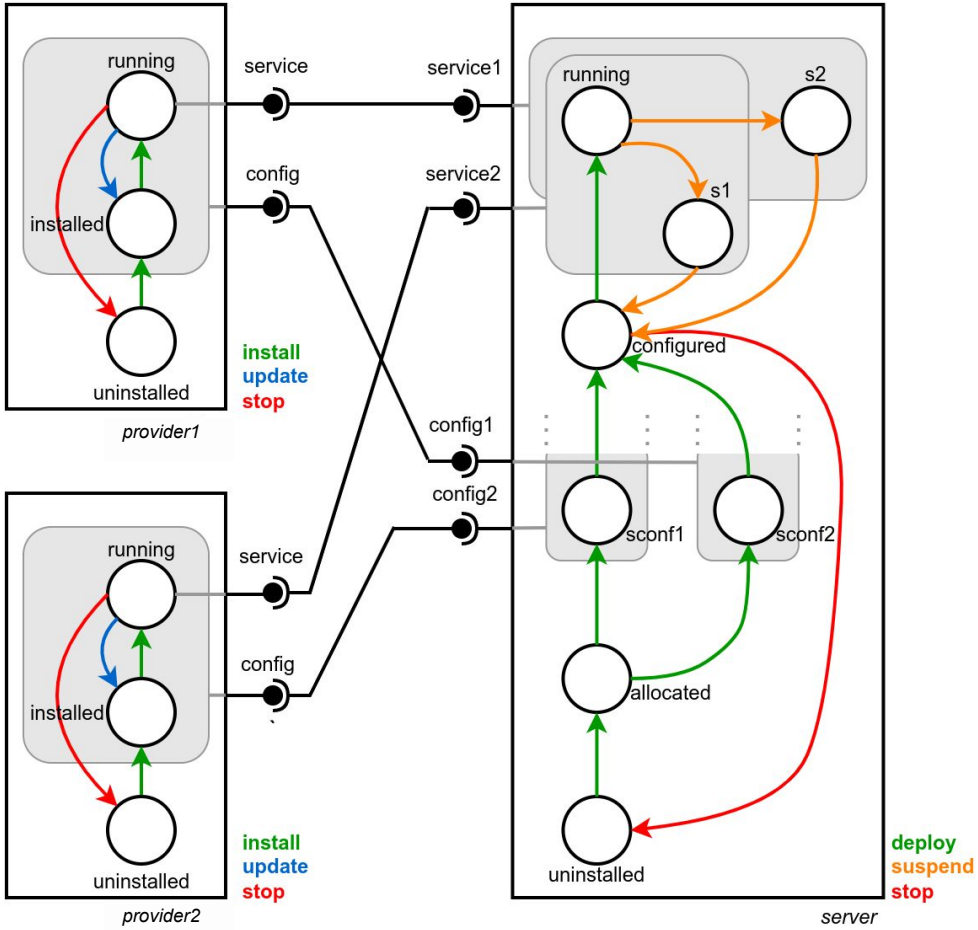
- Involved components
- Interactions / connections between components
- Changes in the component

Concerto-D: Involved components



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

Concerto-D: Connections between components

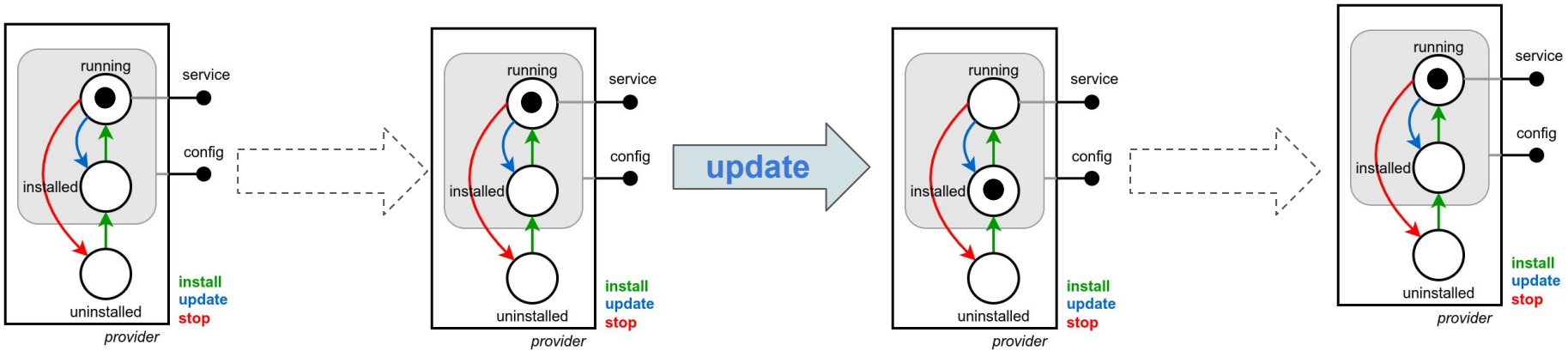


add("*provider1*", Provider)
add("*provider2*", Provider)
add("*server*", Server)
connect(“provider1”, “service”, “server”, “service1”)
connect(“provider1”, “config”, “server”, “config1”)
connect(“provider2”, “service”, “server”, “service2”)
connect(“provider2”, “config”, “server”, “config2”)

Concerto-D: State and changes in the component

Example of objective:

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

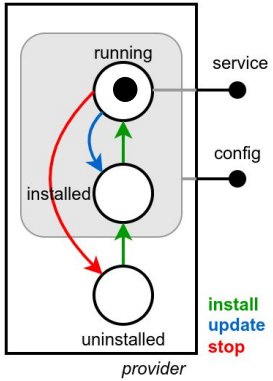


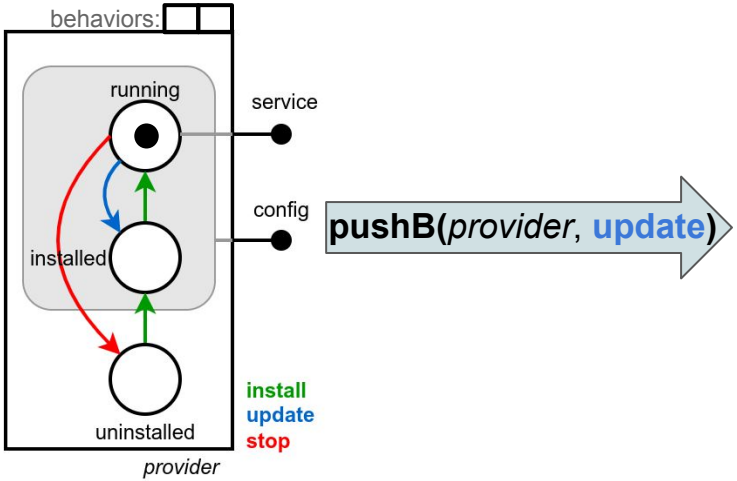
Inferred actions:

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

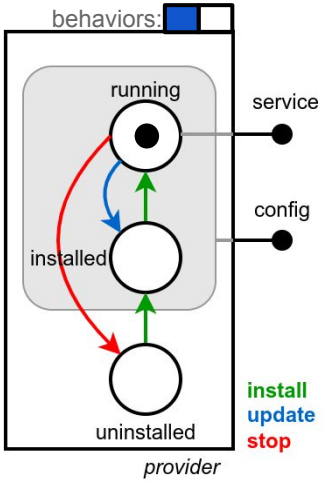
Concerto-D: State and changes in the component

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

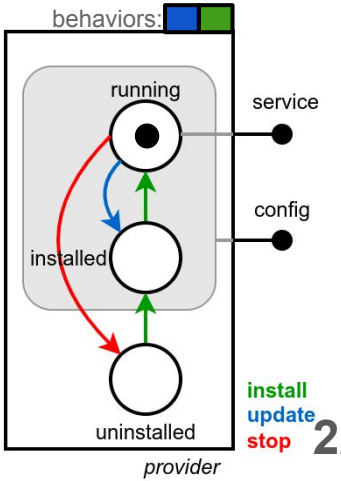




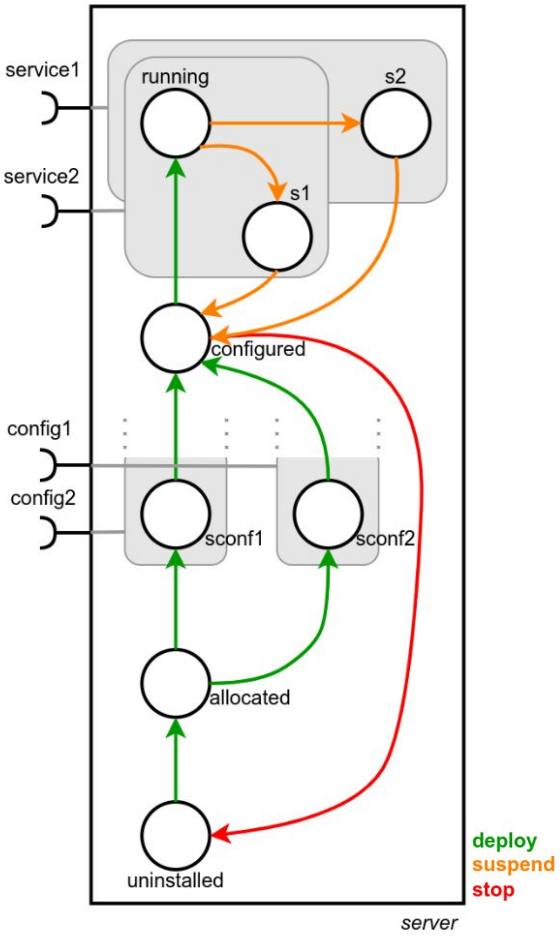
pushB(*provider*, update)

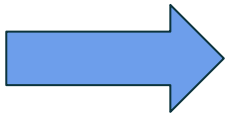


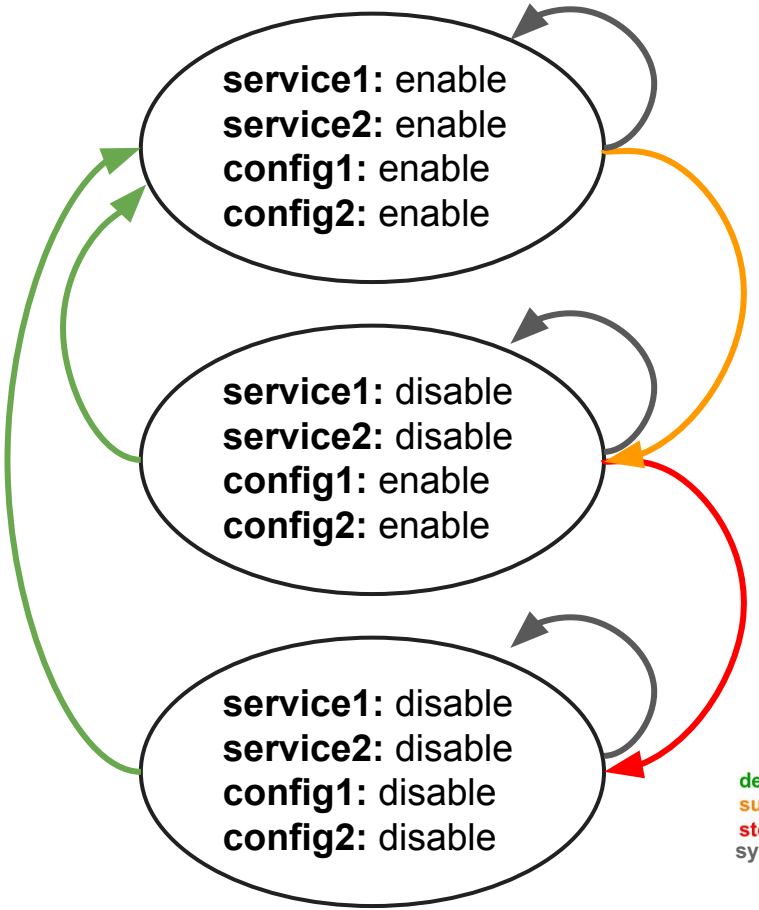
pushB(*provider*, install)



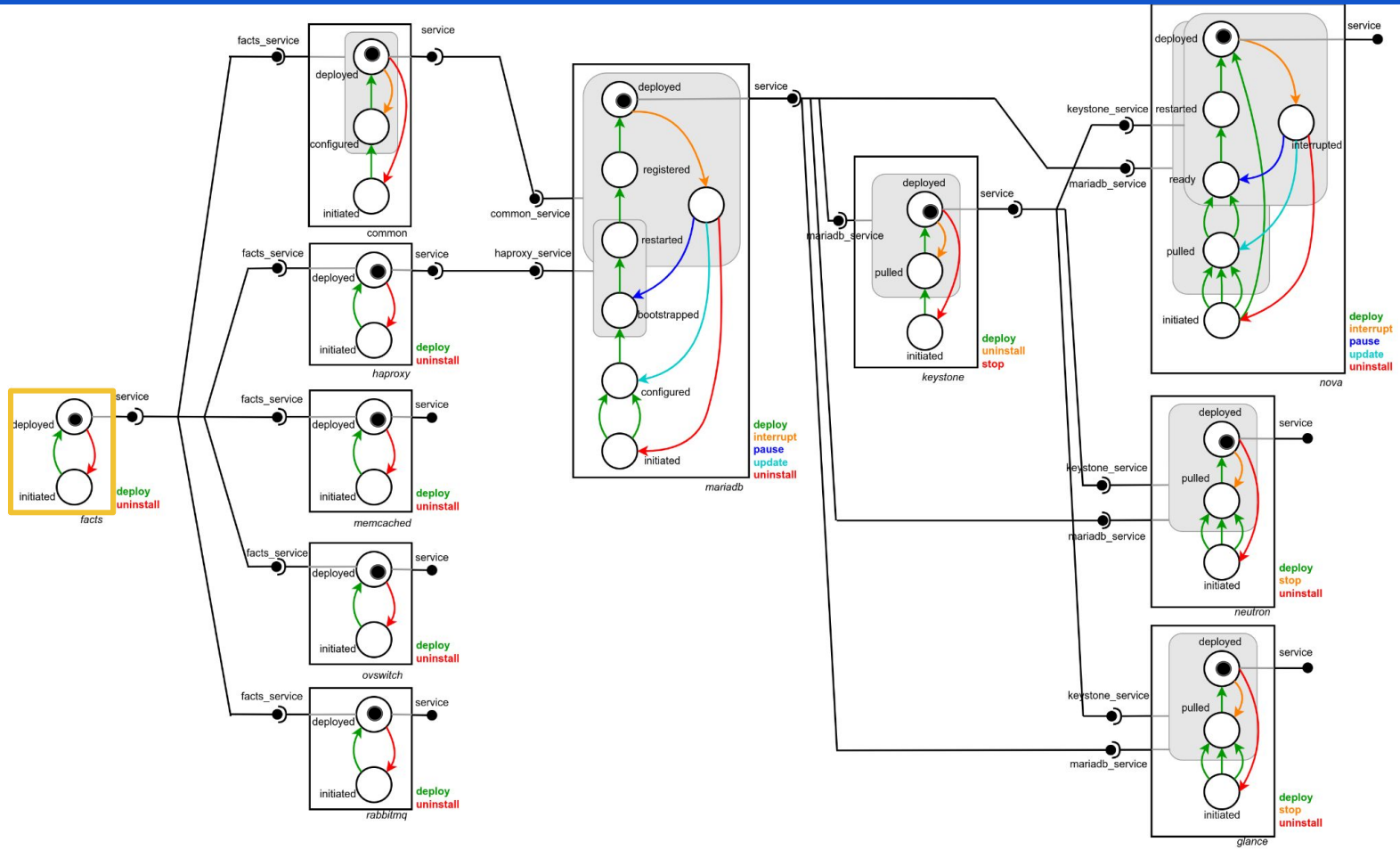
Constraint resolution: Concerto-D to a labeled automata







Example of stratified assembly and reconfiguration



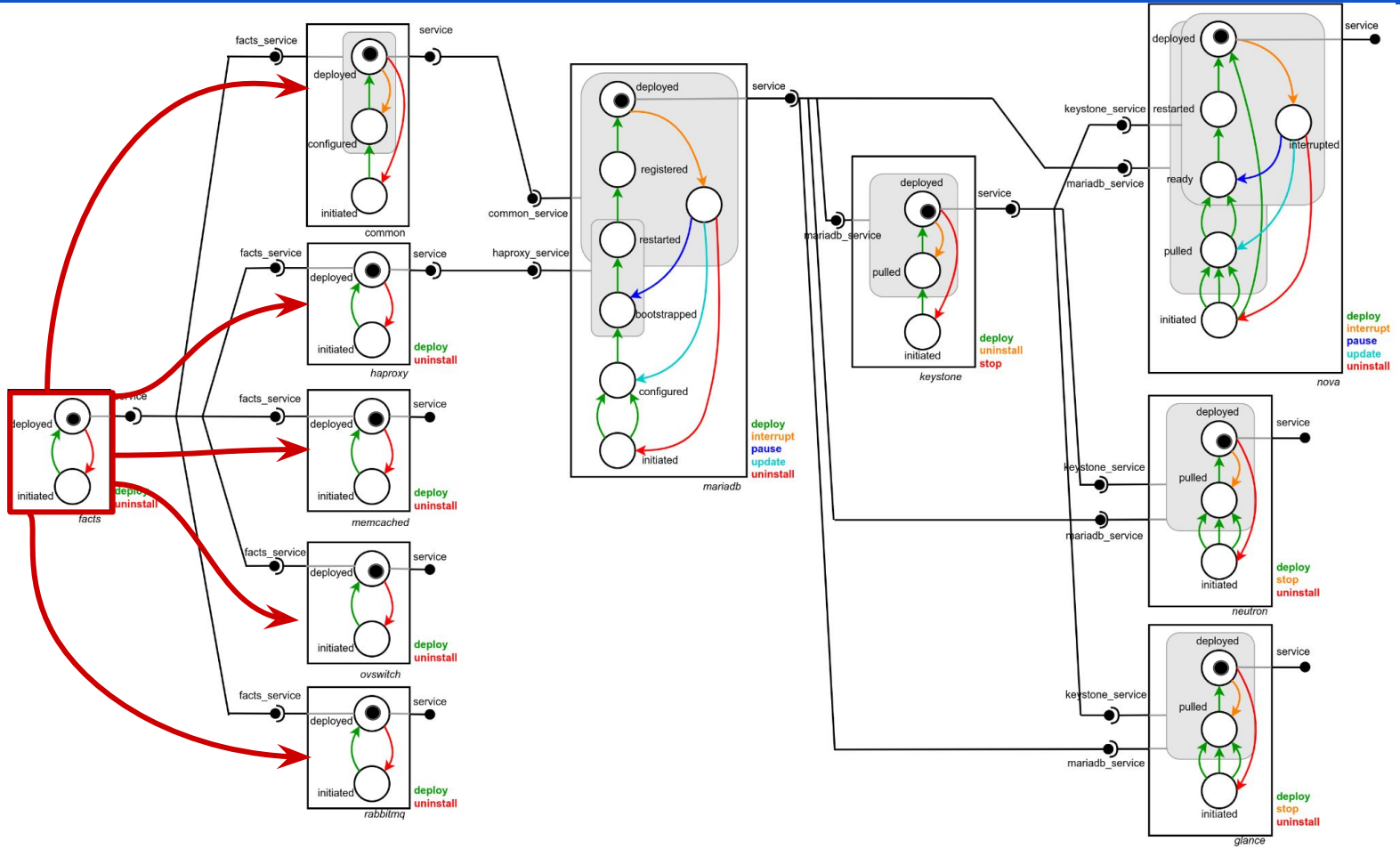
11 components, all **deployed:**

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

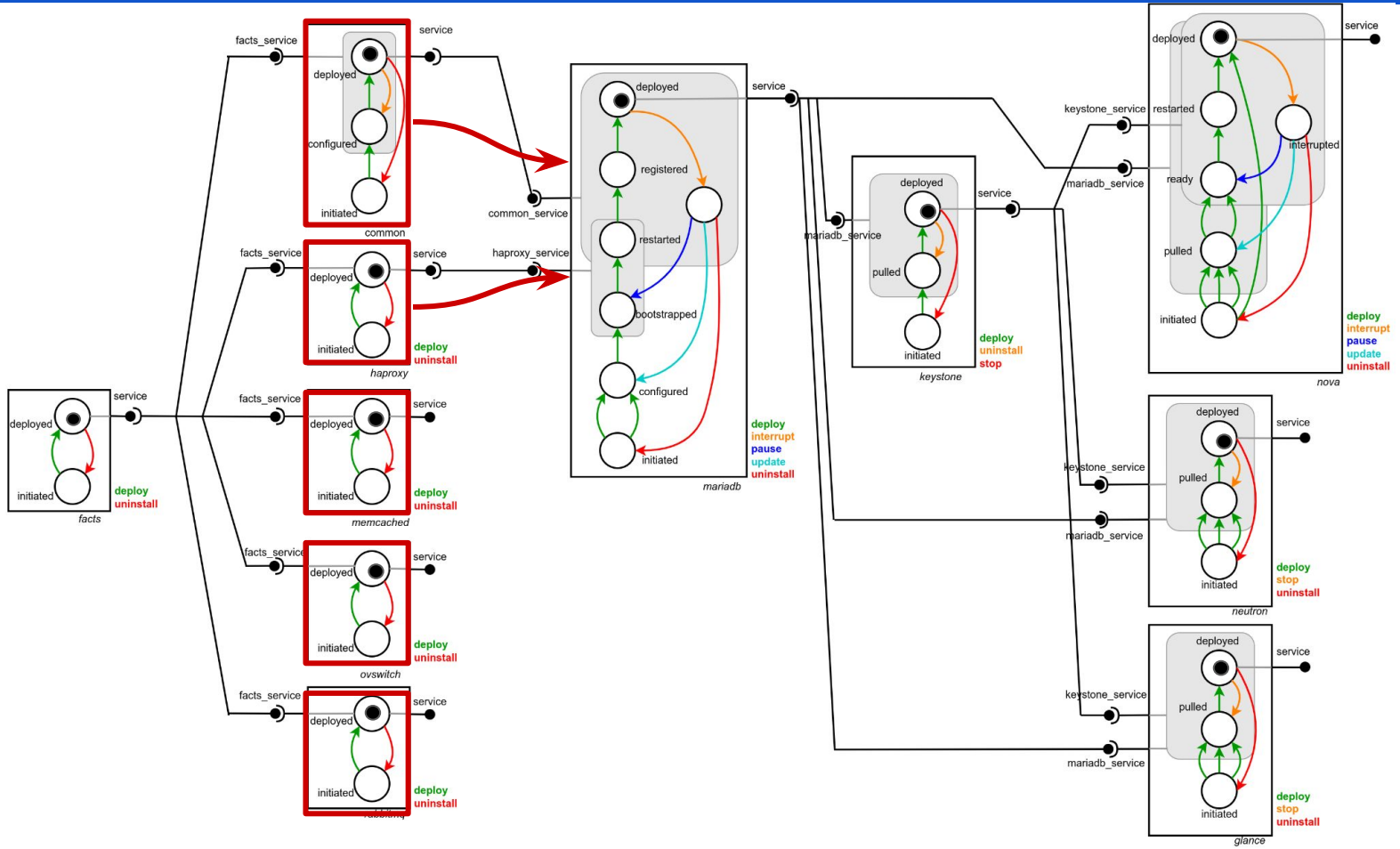
Goal: reboot *facts*

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

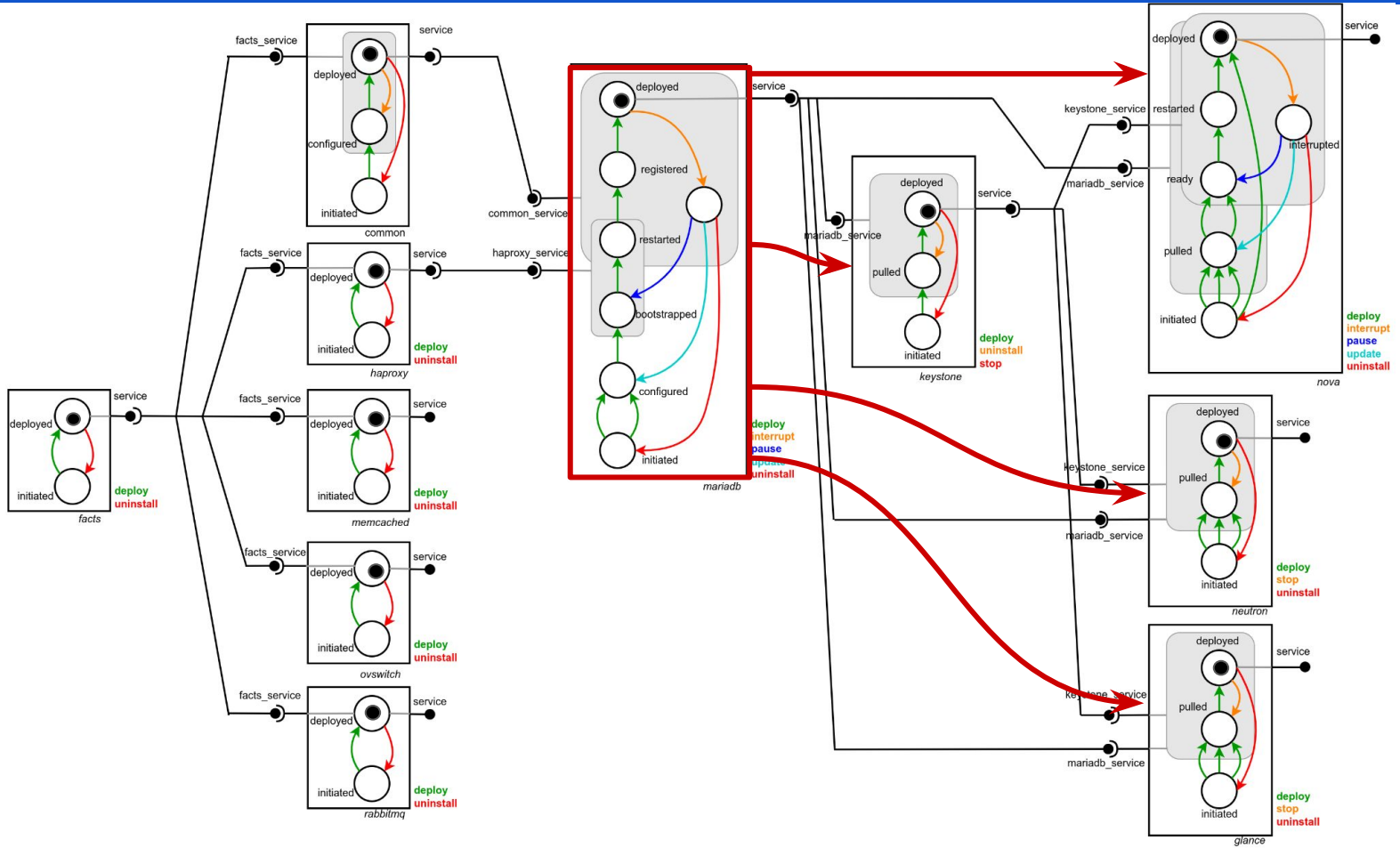
Information sharing protocol - Step I: Propose



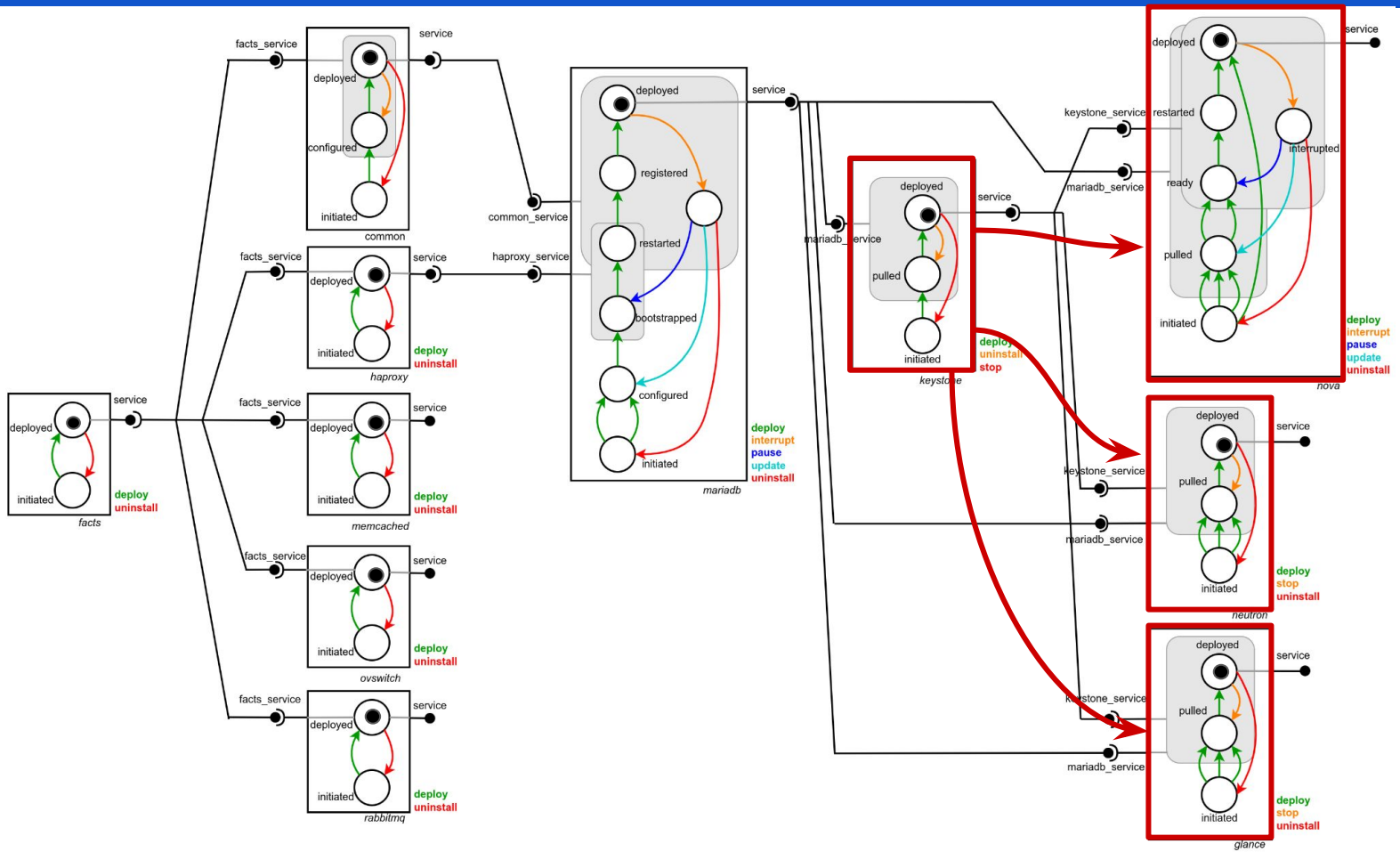
Information sharing protocol - Step I: Propose



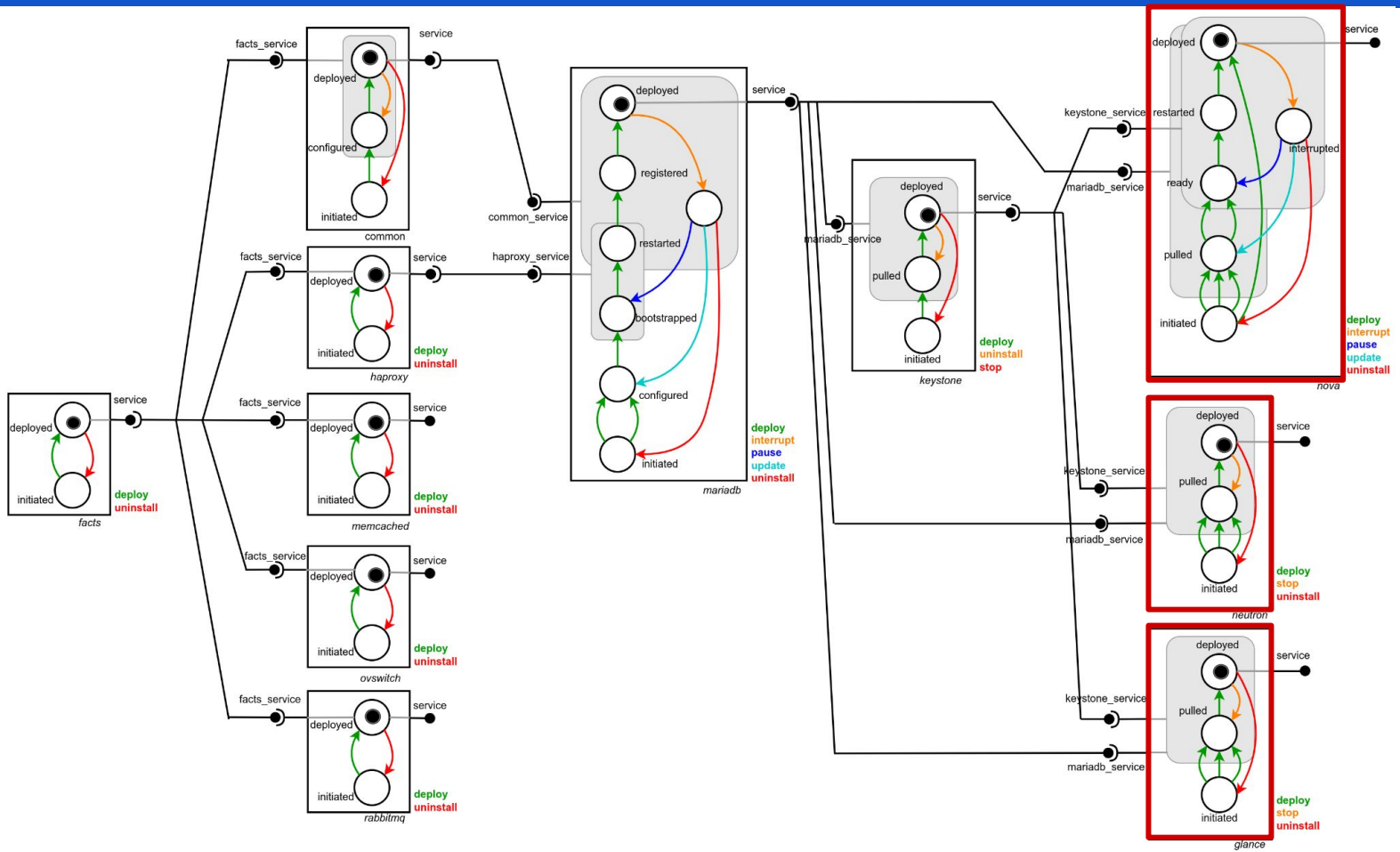
Information sharing protocol - Step I: Propose



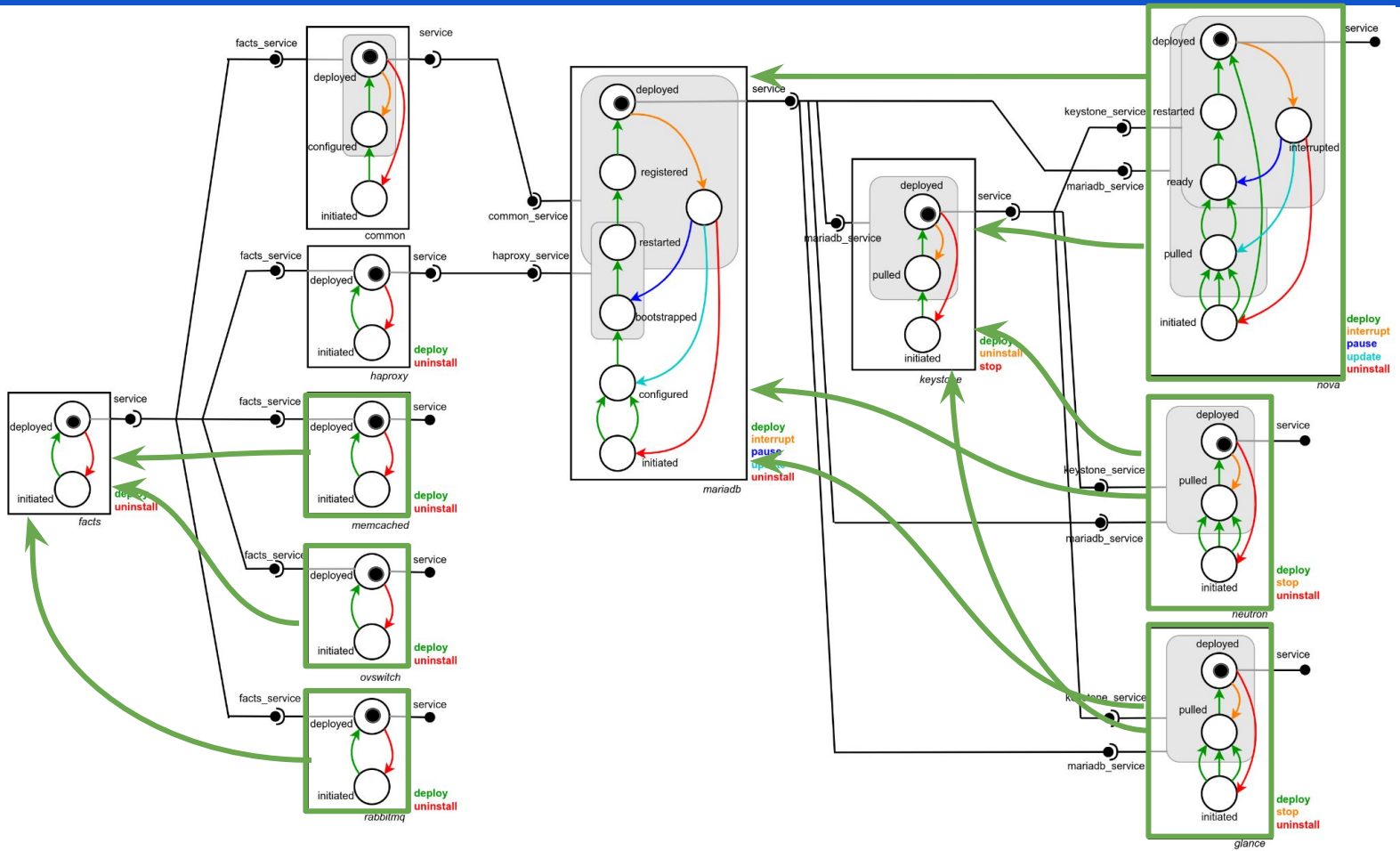
Information sharing protocol - Step I: Propose



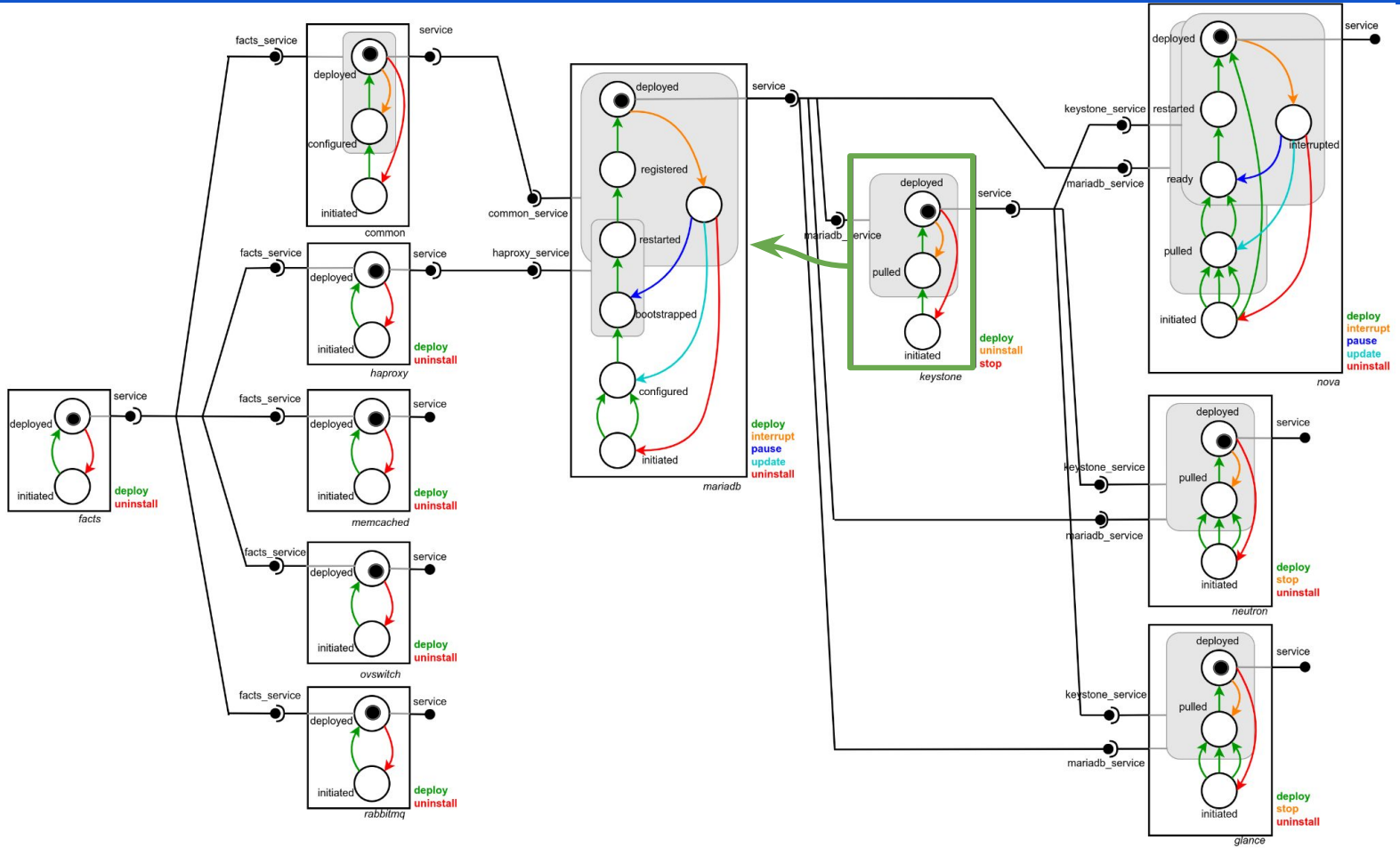
Information sharing protocol - Step I: Propose



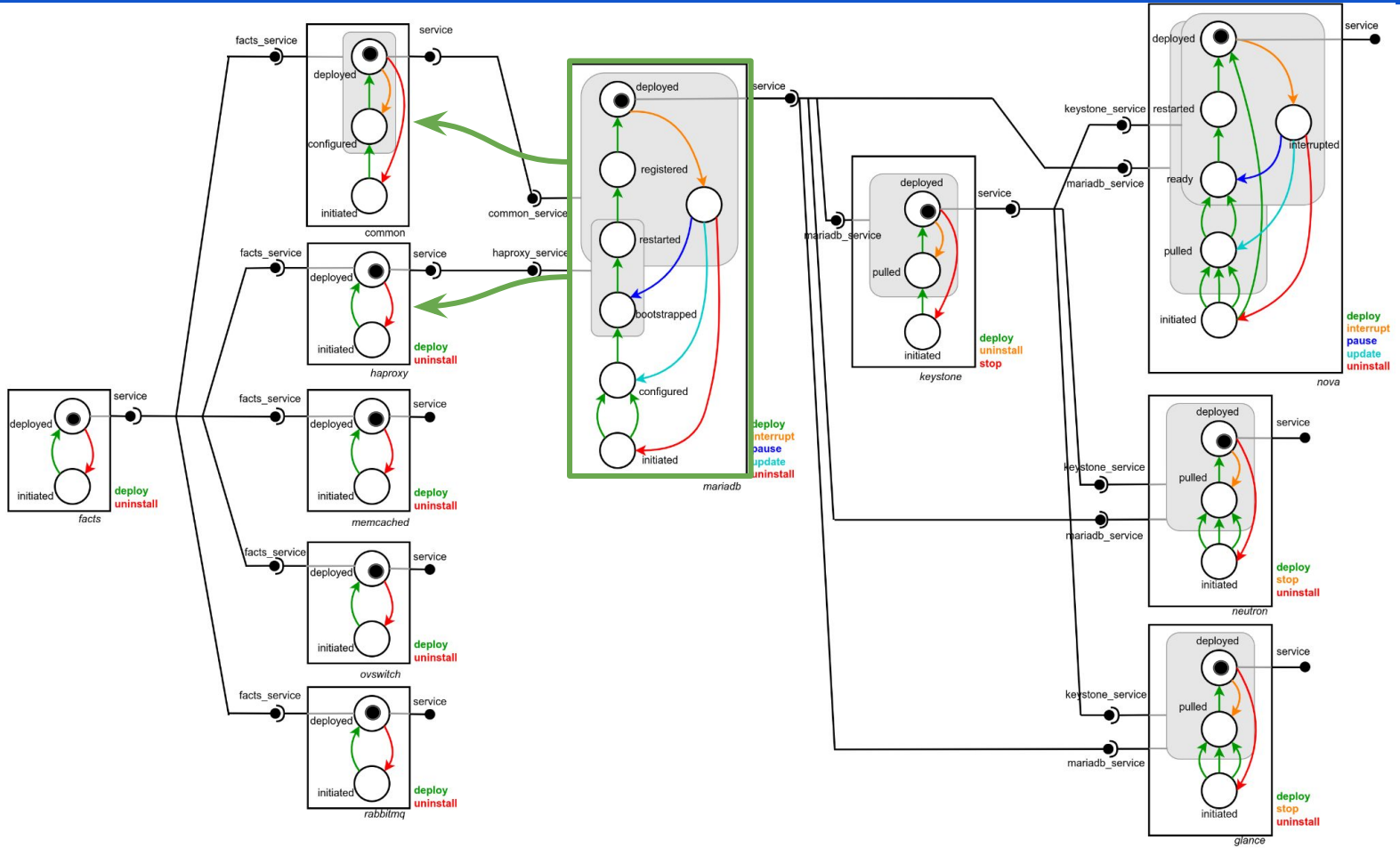
Information sharing protocol - Step II: Send ack



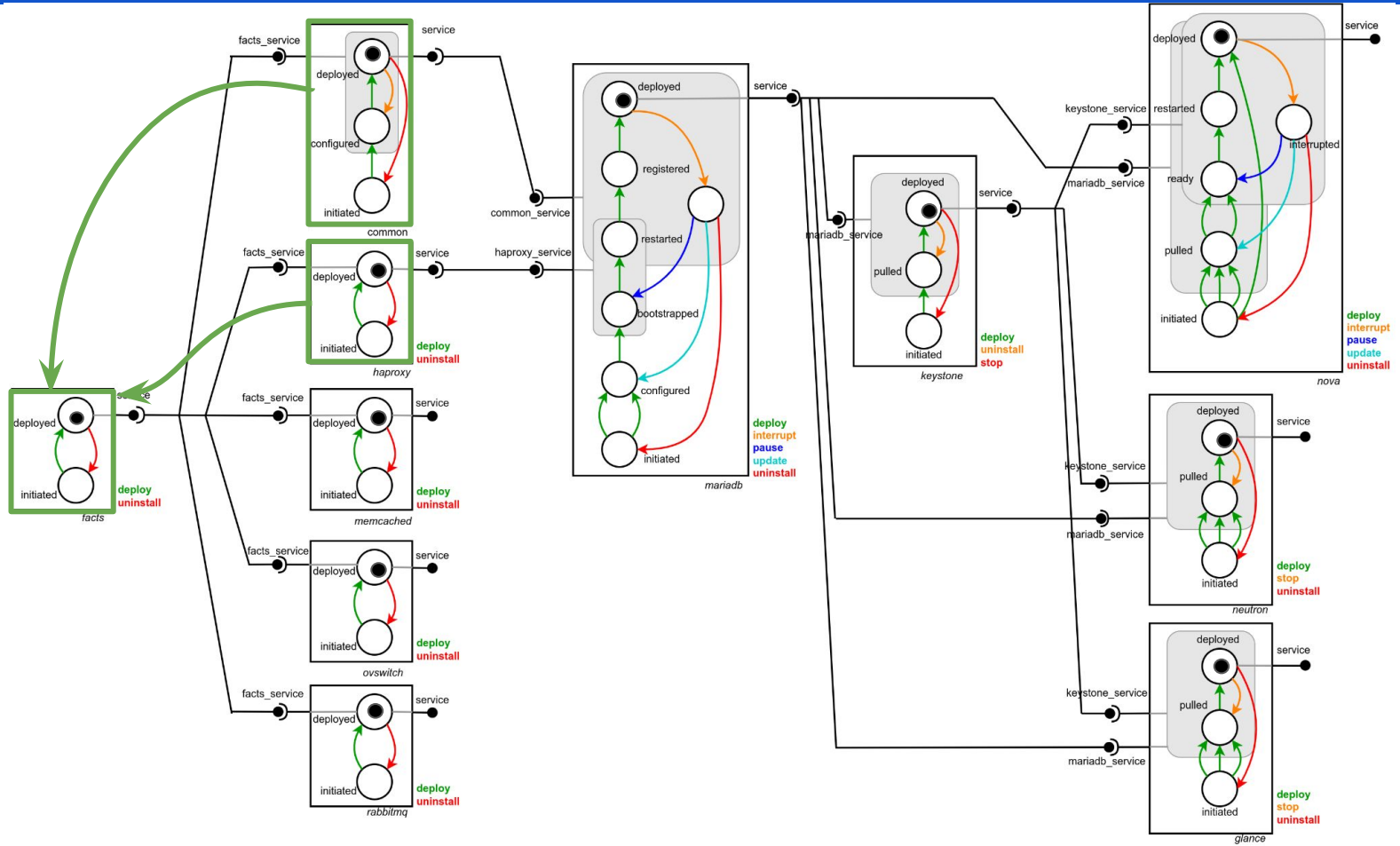
Information sharing protocol - Step II: Send ack



Information sharing protocol - Step II: Send ack



Information sharing protocol - Step II: Send ack



Information sharing protocol - Step III: Global ack from root