Secure FaaS orchestrations in the Cloud-IoT Continuum

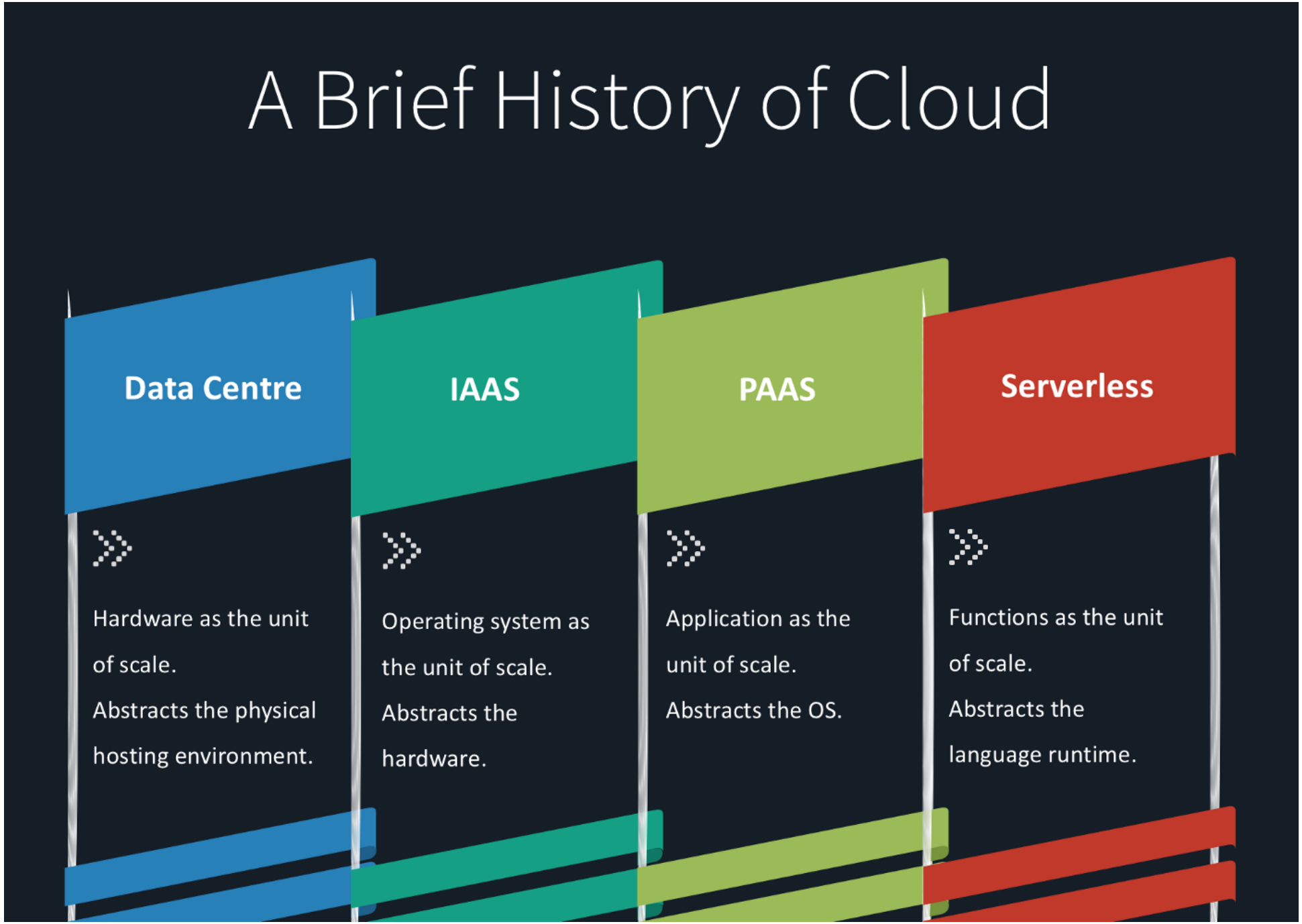
Alessandro Bocci

Advanced Software Engineering

15/12/2021

Outline

- Introduction
  - FaaS & FaaS Orchestrations
  - Securing FaaS in the Cloud-IoT Continuum
- State of the Art
  - Defining FaaS Orchestrations
  - Executing FaaS Orchestrations in the Cloud-IoT Continuum
  - Securing FaaS Orchestrations
  - Research Challenges
- Placement of FaaS Orchestrations in the Cloud-IoT Continuum
  - FaaS2Fog
  - Code & Demo
  - Ongoing work


A Brief History of Cloud
Data Centre
Hardware as the unit of scale.
Abstracts the physical hosting environment.
IAAS
Operating system as the unit of scale.
Abstracts the hardware.
PAAS
Application as the unit of scale.
Abstracts the OS.
Serverless
Functions as the unit of scale.
Abstracts the language runtime.

Function-as-a-Service

- The *function* is the core element
- Users provide the code of the function, the trigger and what to do with the results
- No concern about infrastructure
- Event-driven
- Stateless
- Low execution costs

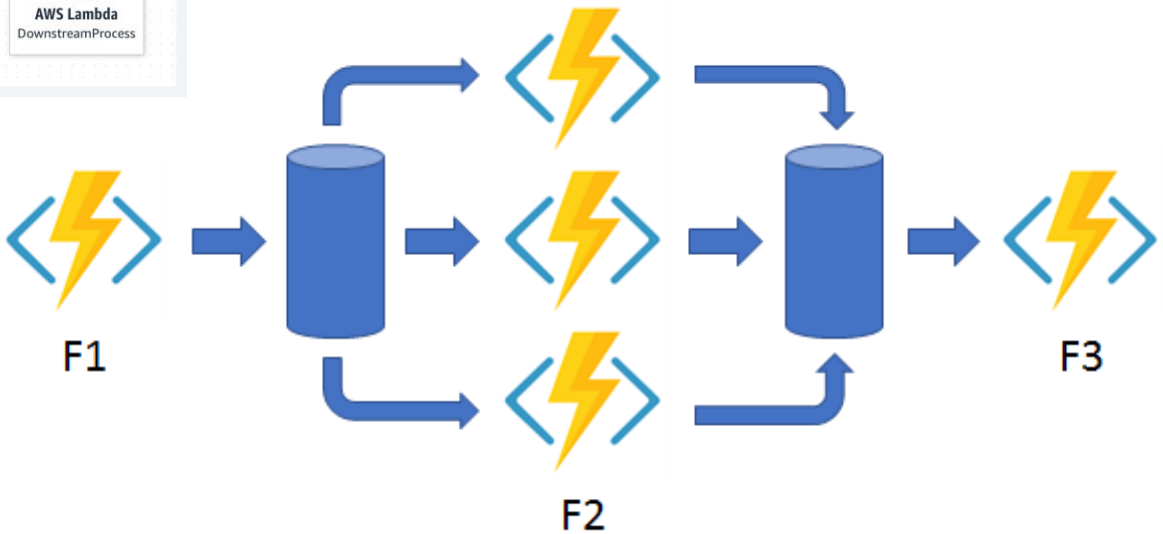
Function-as-a-Service

- Pro
  - Costs
  - Scalability
  - Productivity
  - Avg. latency
- Cons
  - Cold starts (sometimes mitigated via "stem cells")
  - No persistent state
  - Non concurrency controls

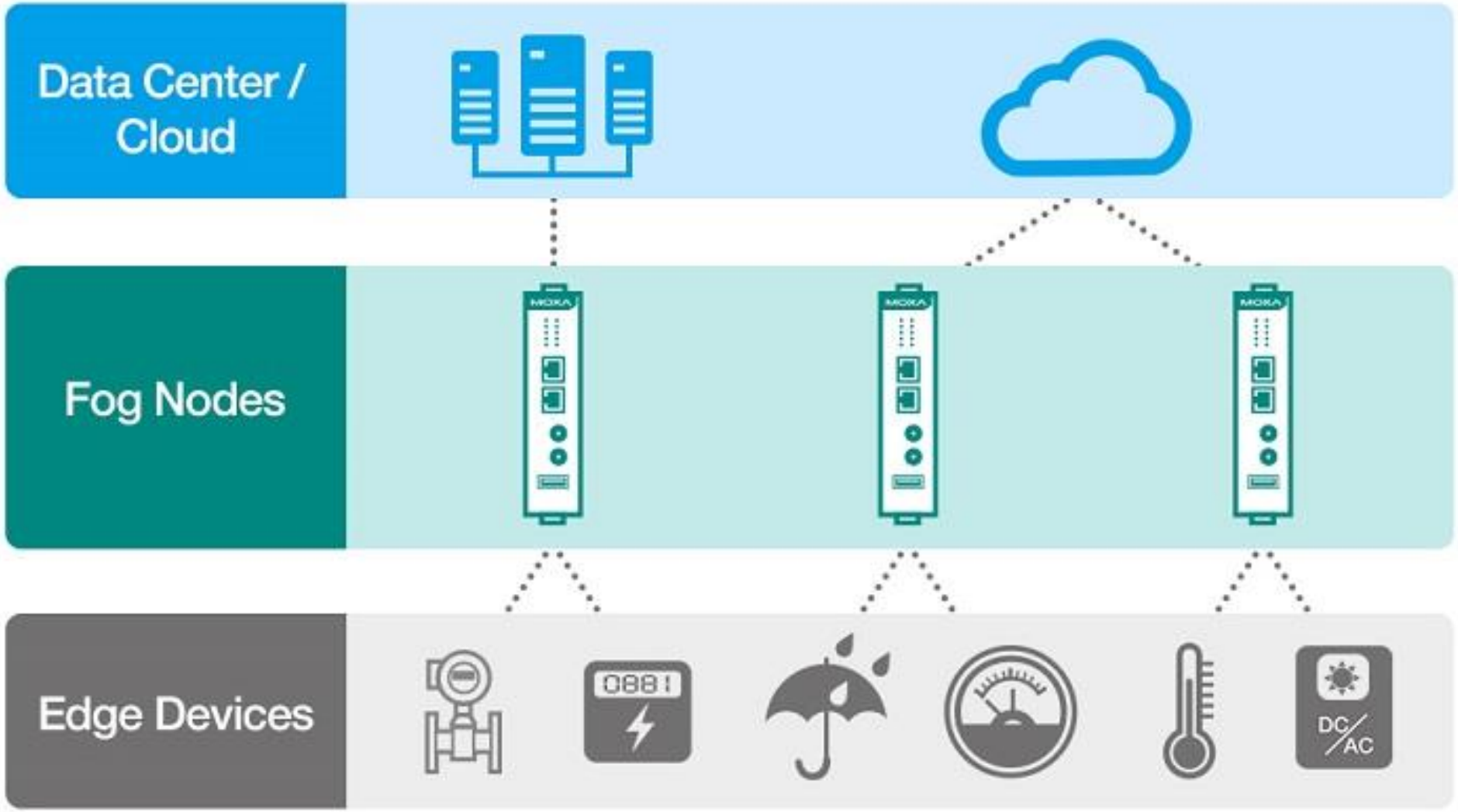
FaaS platforms

- AWS Lambda
- Azure Functions
- Google Cloud Functions
- Apache OpenWhisk
- IBM Cloud Functions
- Kubernetes-based platforms

FaaS Orchestrations



Cloud-IoT Continuum (C-I Cont.)



When FaaS meets the Cloud-IoT Continuum

Key idea: deploy functions on nodes near the edge

- Improve QoS of FaaS
- Event-driven programming
- Better resource management

But: this rises non-trivial security problems!

Security issues

- Reduced Trust Computing Base
- Devices can be easily hacked, stolen or broken
- Isolate users to calculate accounting and billing
- Privacy

…

Motivations

FaaS + C-I Cont. + Security

Applications for:

- Environmental monitoring
- Diseases tracking
- Home automation
- Vocal Assistants
- Smart Agriculture
- …

State of the art

Three Perspectives:

- P1: Defining FaaS orchestrations
- P2: Executing FaaS orchestrations in the C-I Cont.
- P3: Securing FaaS orchestrations

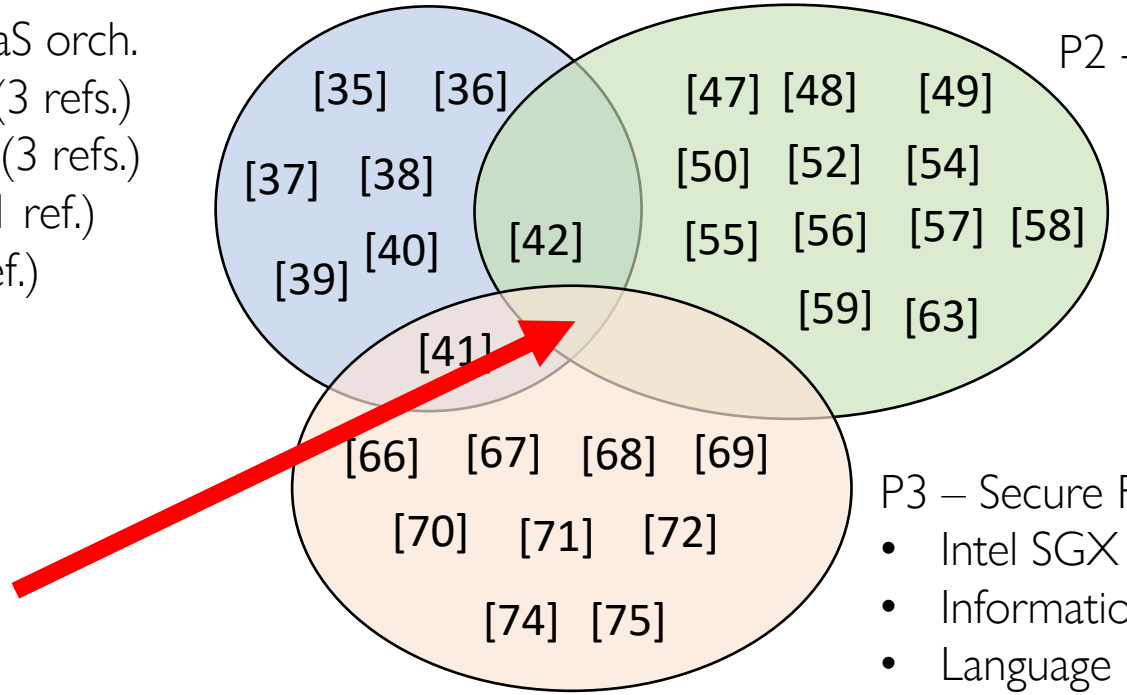
References form:

A. Bocci, S. Forti, G.-L. Ferrari, A. Brogi, *Secure FaaS orchestration in the fog: how far are we?* Computing 103.5 (2021): 1025-1056.

Literature selection

- FaaS platforms
- Initial corpus: 80+ scientific articles

P1 - Definition of FaaS orch.
- Formal methods (3 refs.)
- Workflow-based (3 refs.)
- Function typing (1 ref.)
- Actor based (1 ref.)



P2 - Execution of FaaS orch. in the C-I Cont.
- FaaS platforms (6 refs.)
- Placement optimisation (6 refs.)
- Auction-based (1 ref.)

P3 – Secure FaaS orch.
- Intel SGX based (5 refs.)
- Information flow (2 refs.)
- Language based (3 refs.)

P1: Defining FaaS orchestrations

Languages, models and methodologies to define FaaS orchestrations.

Formal methods to model FaaS platforms

- Baldini et al. [35]
- Jangda et al. [36]
- Gabbrielli et al. [37]

- Eisman et al. [35]
- Lopez et al. [36]
- Yang et al. [37]

Workflow-based

Actor Based

- Persson and Angelsmark [42]

- Gerasimov [41]

Function Typing

P1: Findings

- High support for basic programming constructs

  (sequences, conditional branches, loops…)

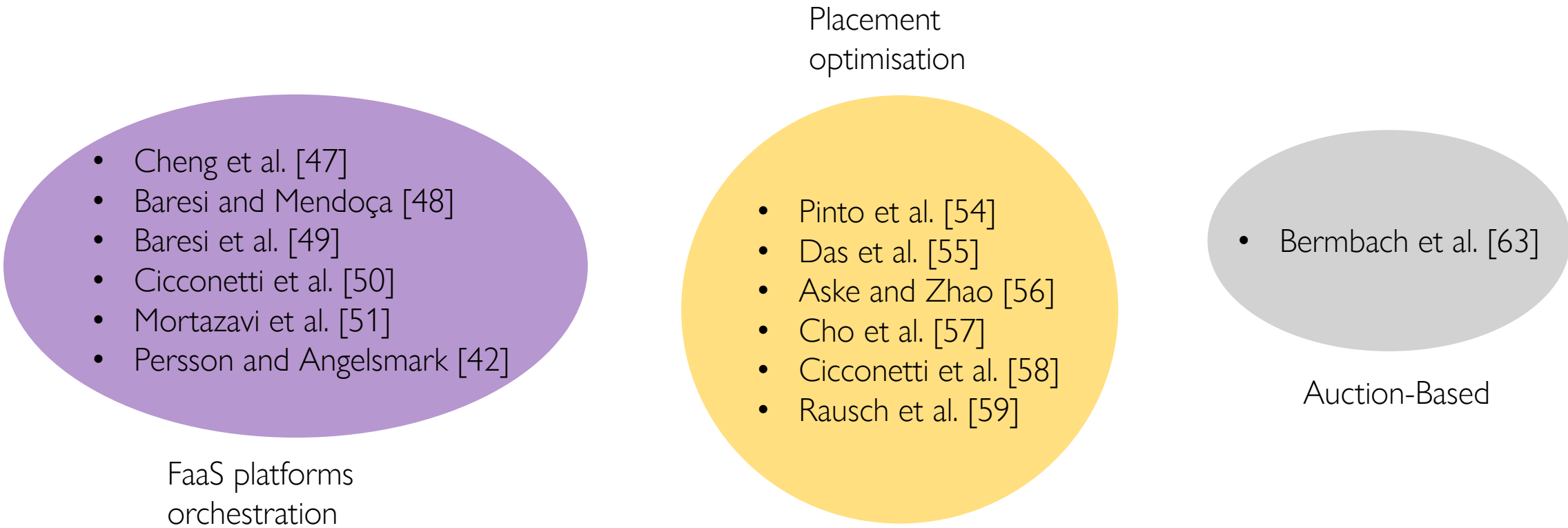
- Direct tiggers vs publish/subscribe invocation

- Recursive functions supported

- Low support of type checking of functions

P2: Executing FaaS orchestrations in the C-I Cont.

Platforms, techniques and methodologies to execute FaaS orchestrations in the C-I Cont.



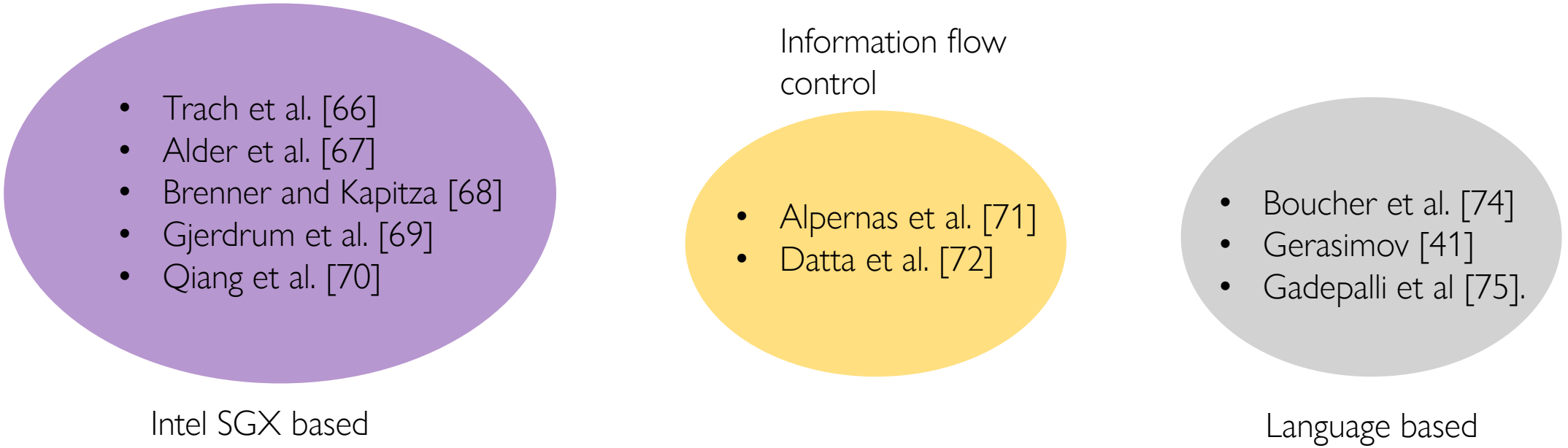
P2: Findings

- High support for latency and resource management
- Cost-awareness has good consideration
- Data- and bandwidth-awareness are considered by few works

P3: Securing FaaS orchestrations

Techniques and methodologies to secure FaaS orchestrations

both statically and at runtime.



P3: Findings

- Data confidentiality and function integrity are the main assets protected
- The main threat is given by external attacks, but also cloud providers and developer mistakes are highly considered
- The main protection techniques are hardware isolation and information flow security

Research Challenges

P1 ∩ P2 (Defining FaaS orchestrations ∩ Executing FaaS orchestrations in the C-I Cont.)

- Orchestration-aware execution:

  Exploit orchestrations to place and execute functions (instead of considering single functions)

- Definition and Execution of context- and Qos-aware orchestrations

  Add context (hardware, software, affinity) and QoS (latency, bandwidth) requirements to functions and orchestrations to support placement and execution.

Research Challenges

P1 ∩ P3 (Defining FaaS orchestrations ∩ Securing FaaS orchestrations)

- Definition of Security requirements:

  Add to function orchestrations security requirements on functions (e.g. specific nodes execution), orchestrations (e.g. policies defined on set of functions), and data (e.g. defining and exploiting security levels).

- Static analysis of FaaS orchestrations:

  Analyse statically defined FaaS orchestrations using the aforementioned requirements to support placement and execution.

Research Challenges

P2 ∩ P3 (Executing FaaS orchestrations in the C-I Cont. ∩ Securing FaaS orchestrations)

- Secure Executions in the C-I Cont.

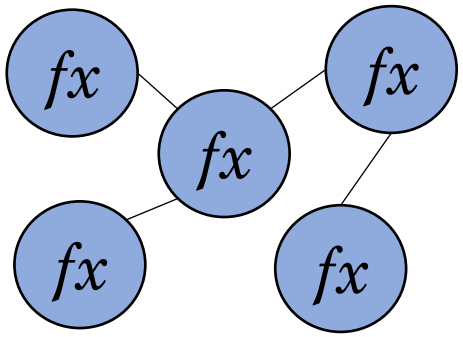
  Specific approaches to security of FaaS, e.g. information flow control security of functions and orchestrations in the C-I Cont.

Placement of FaaS Orchestration in the Cloud-IoT Continuum

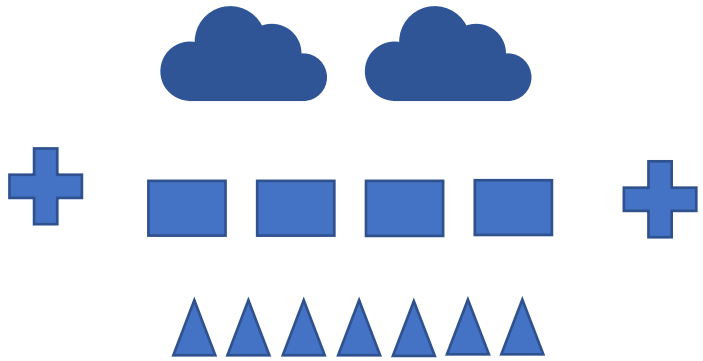
Our scenario

- We aim to protect the data confidentiality
- Opportunistic edge/fog computing
- Planning phase, before deployment and execution

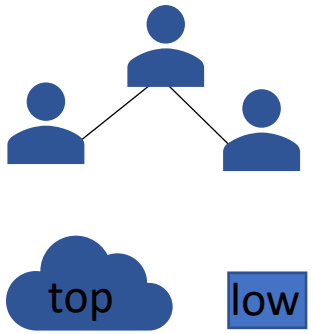
Considered Problem



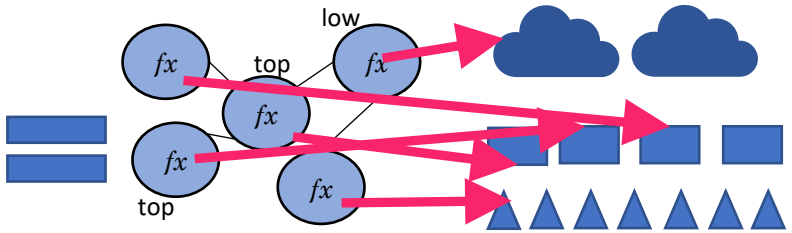
FaaS orchestration + Data security levels



C-I infrastructure + Security capabilities

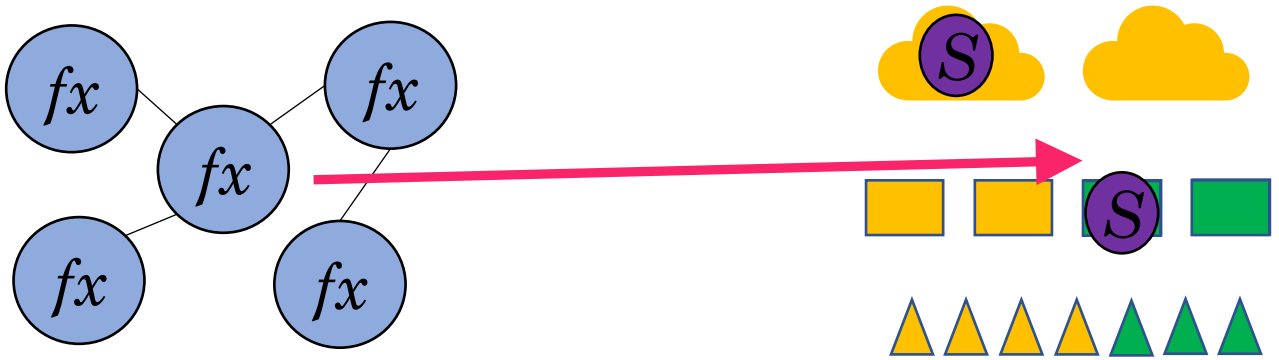


Security policies + Trust network



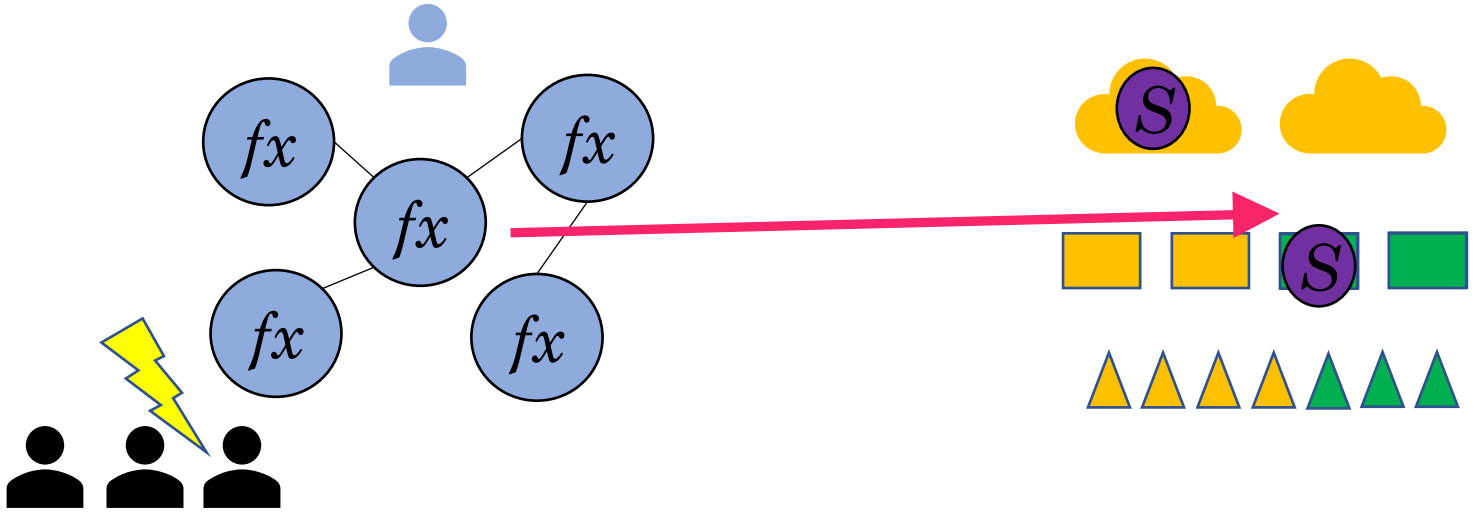
QoS-, security-, trust-aware placement of FaaS orchestrations

Stakeholders



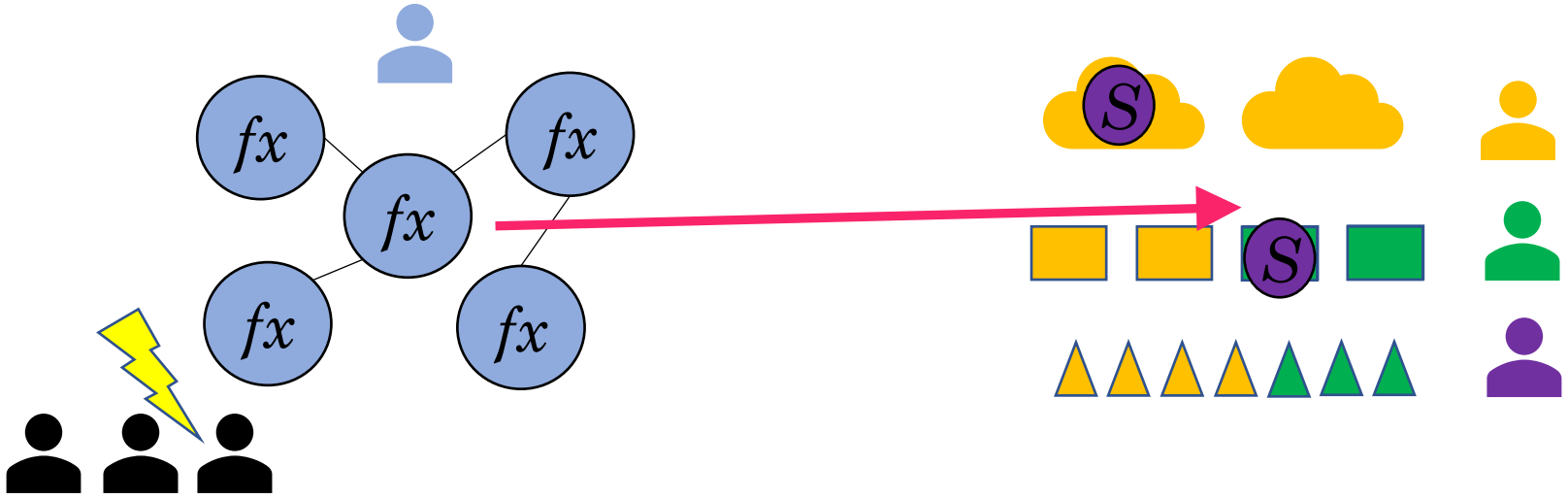
Stakeholders

- Application operators
- Clients of applications



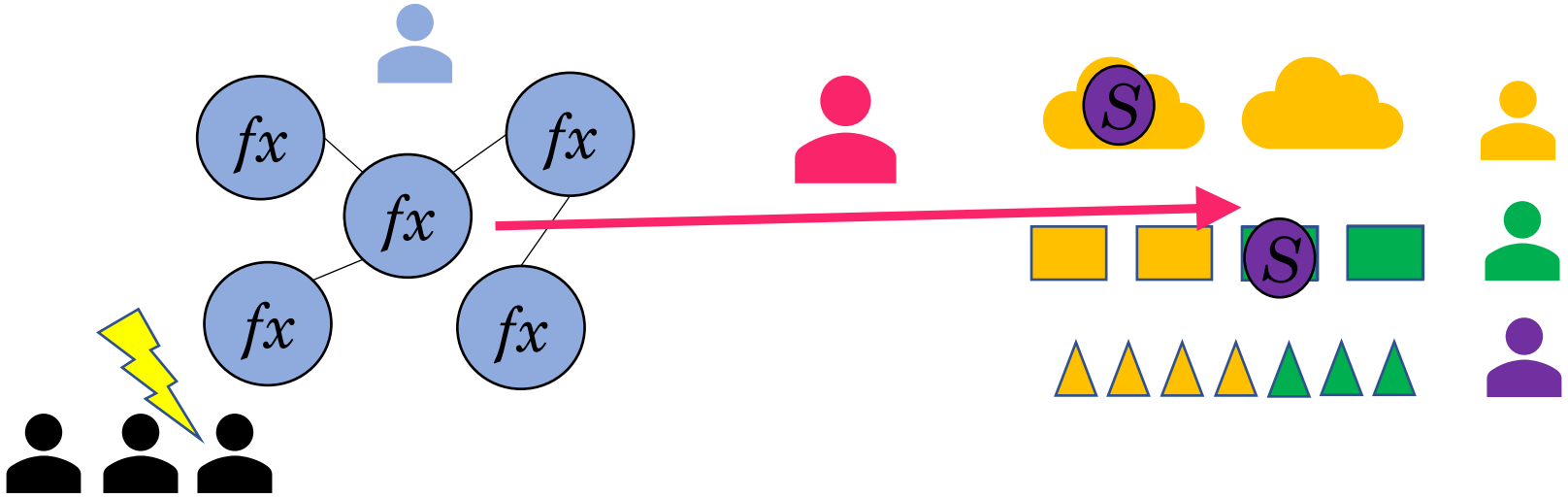
Stakeholders

- Application operators
- Clients of applications
- Node providers
- Service providers



Stakeholders

- Application operators
- Clients of applications
- Node providers
- Service providers
- Orchestration service provider



Attacker model

Try to disclose data confidentiality of application's clients.

Knowledge Base

- Public data
- Node resources
- Services placement
- Application structure
- Application external behaviour

Attacks

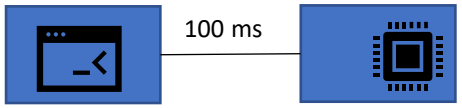
- Hack weak nodes

- Traffic monitoring

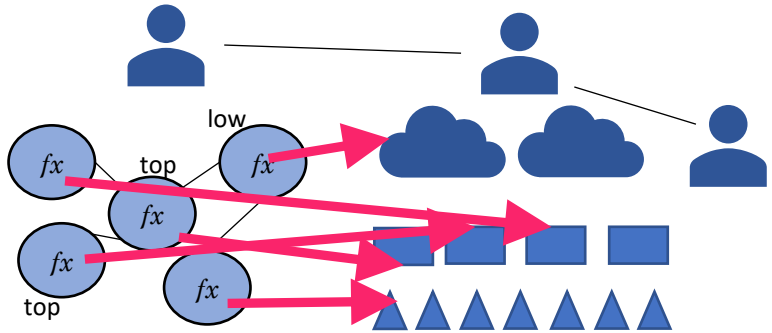
FaaS2Fog

A methodology and (alpha)Problog prototype to place orchestrated FaaS applications onto C-I Cont. infrastructures.

Declarative model



Hw, Sw, Latency constraints



Infrastructure and IF security, and trust

https://github.com/di-unipi-socc/FaaS2Fog

Based on SecFog

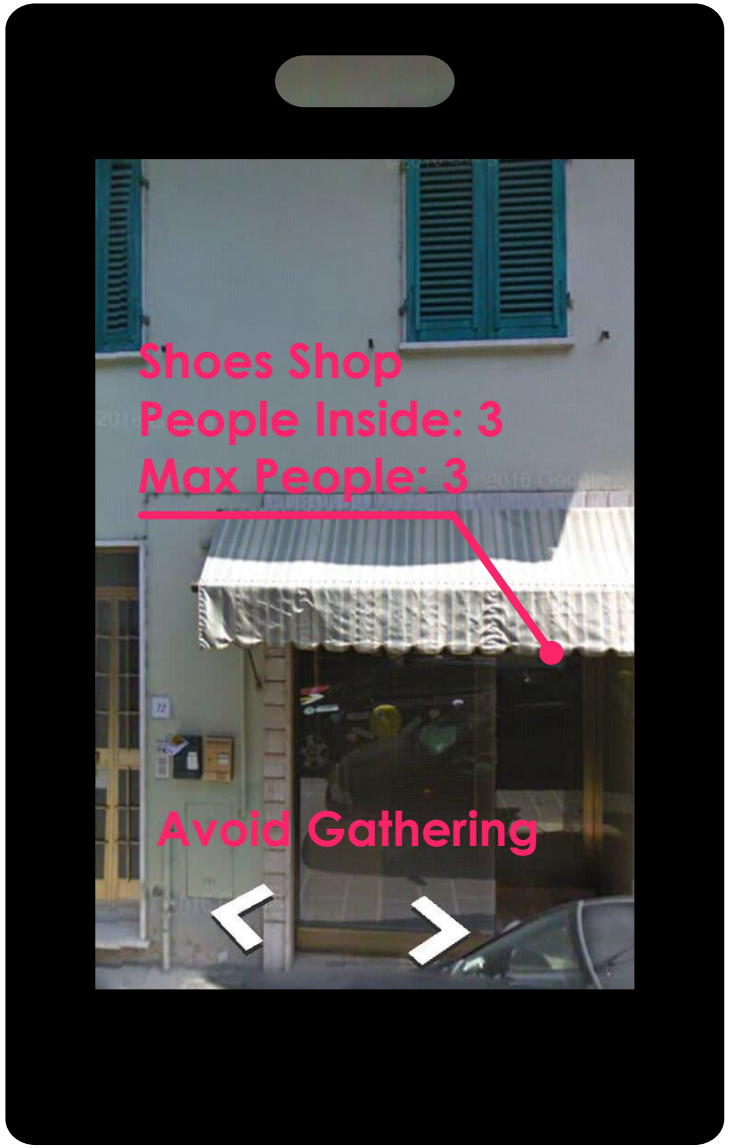
Motivating example

AR Application to prevent people gathering in commercial areas.

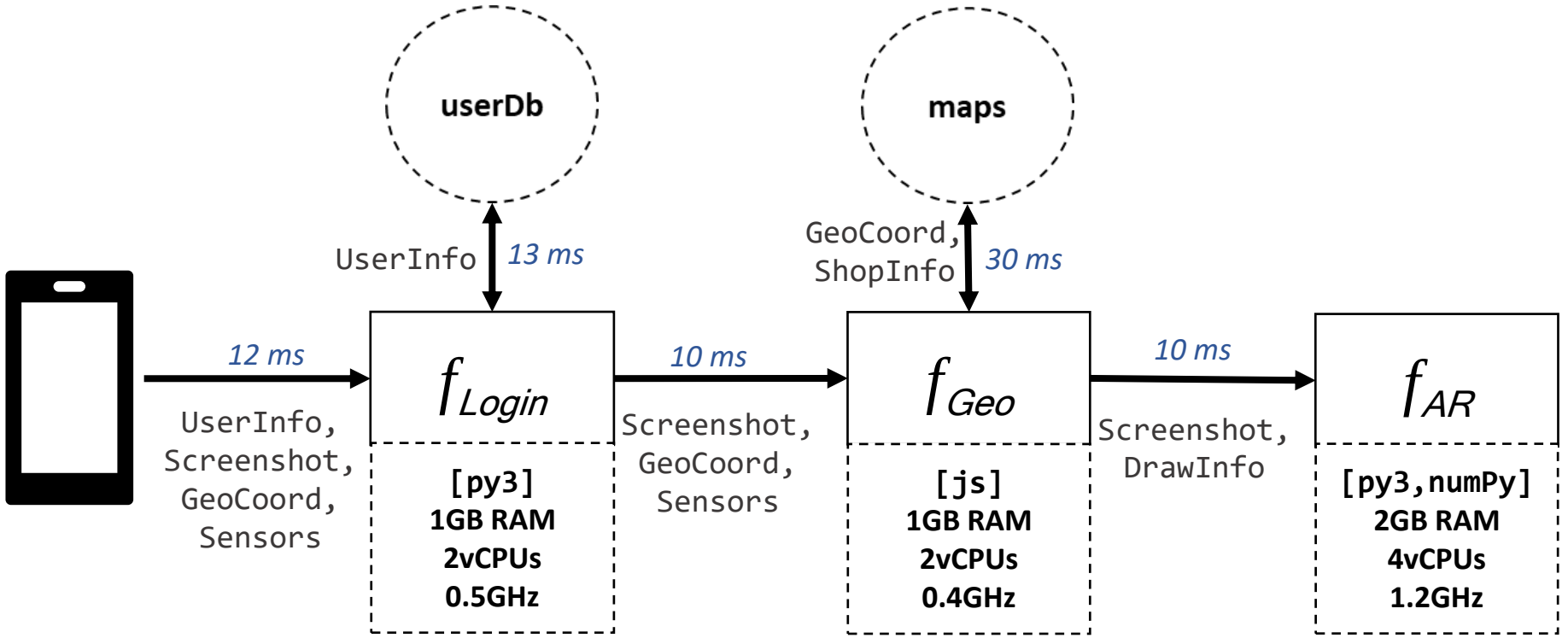
The application renders over the video

3D information about the number of people inside a shop and its overall capacity.

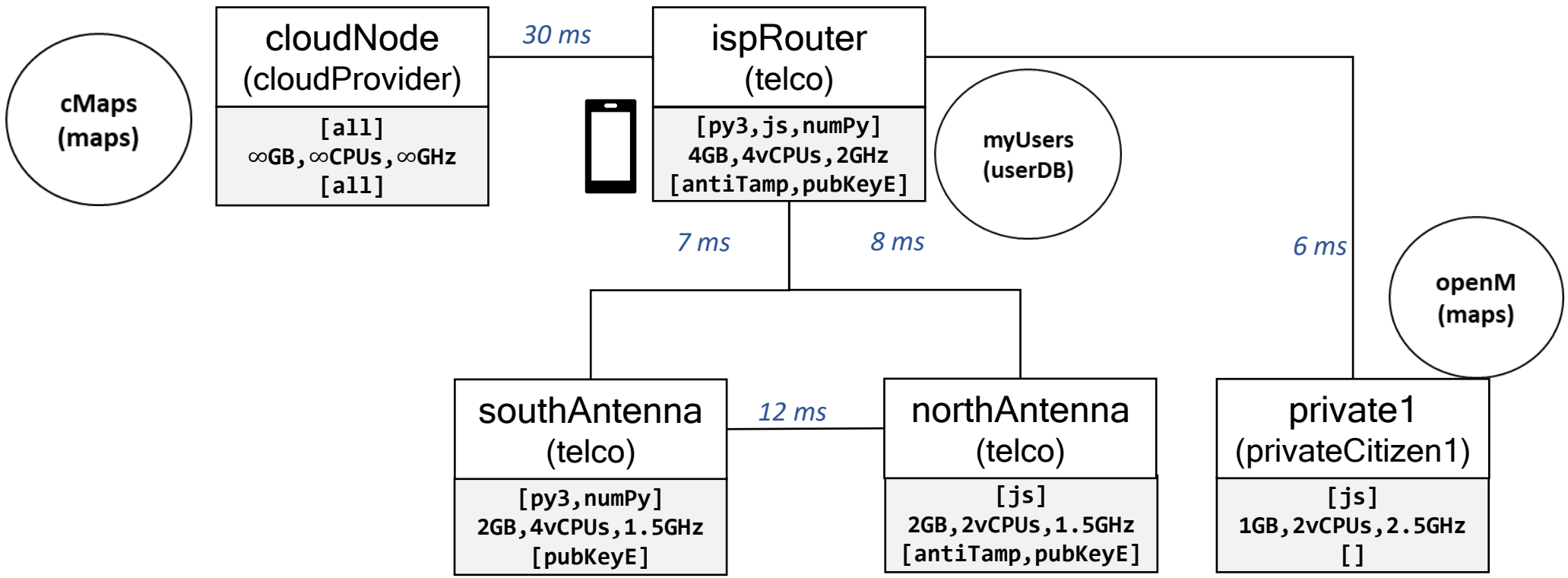
If high risk of gathering is detected, the application suggests alternative locations to move away from the gathering area.



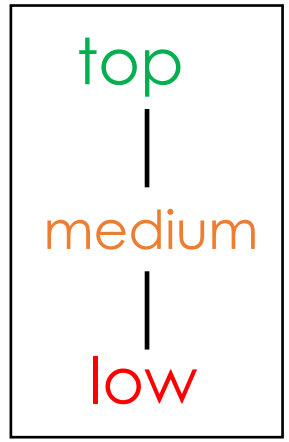
Motivating example – FaaS Chain



Motivating example – Infrastructure



Motivating example – Operator Policies



Node Labelling:

top

nodeName (provider)
[swCaps] hwCaps [secCaps]

Antitampering & Public Key Encription

medium

nodeName (provider)
[swCaps] hwCaps [secCaps]

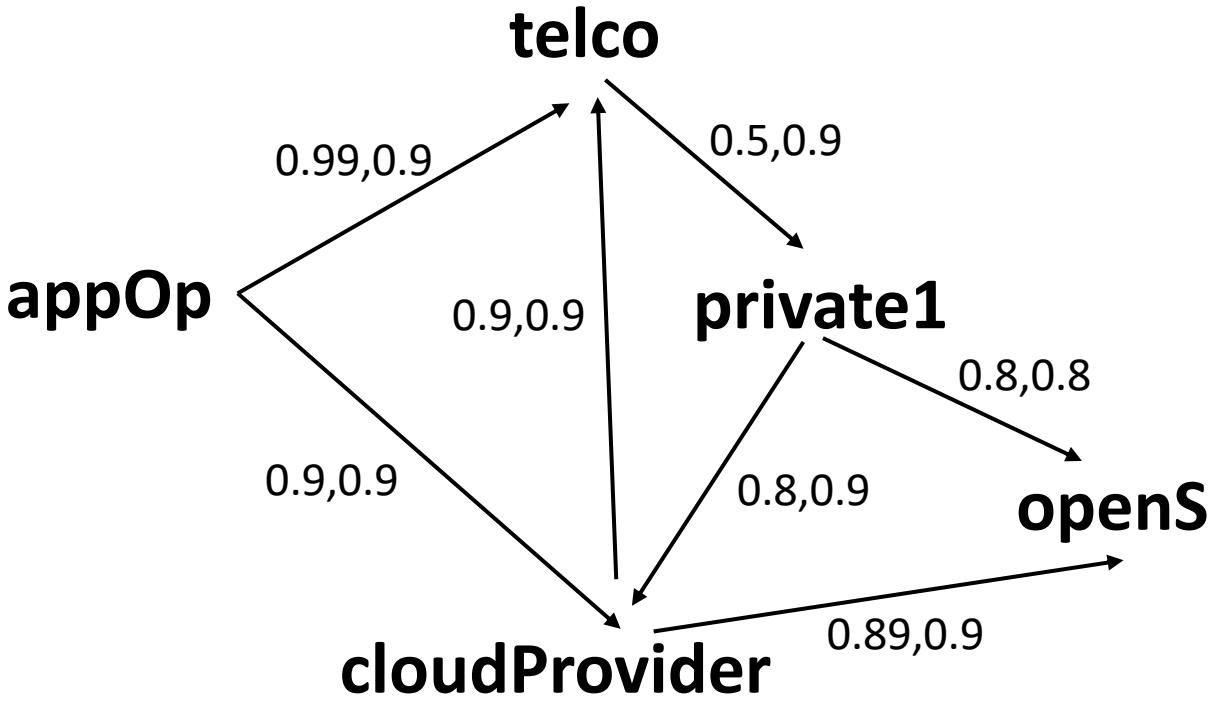
Public Key Encription

low

nodeName (provider)
[swCaps] hwCaps [secCaps]

All the others

Motivating example – Trust network

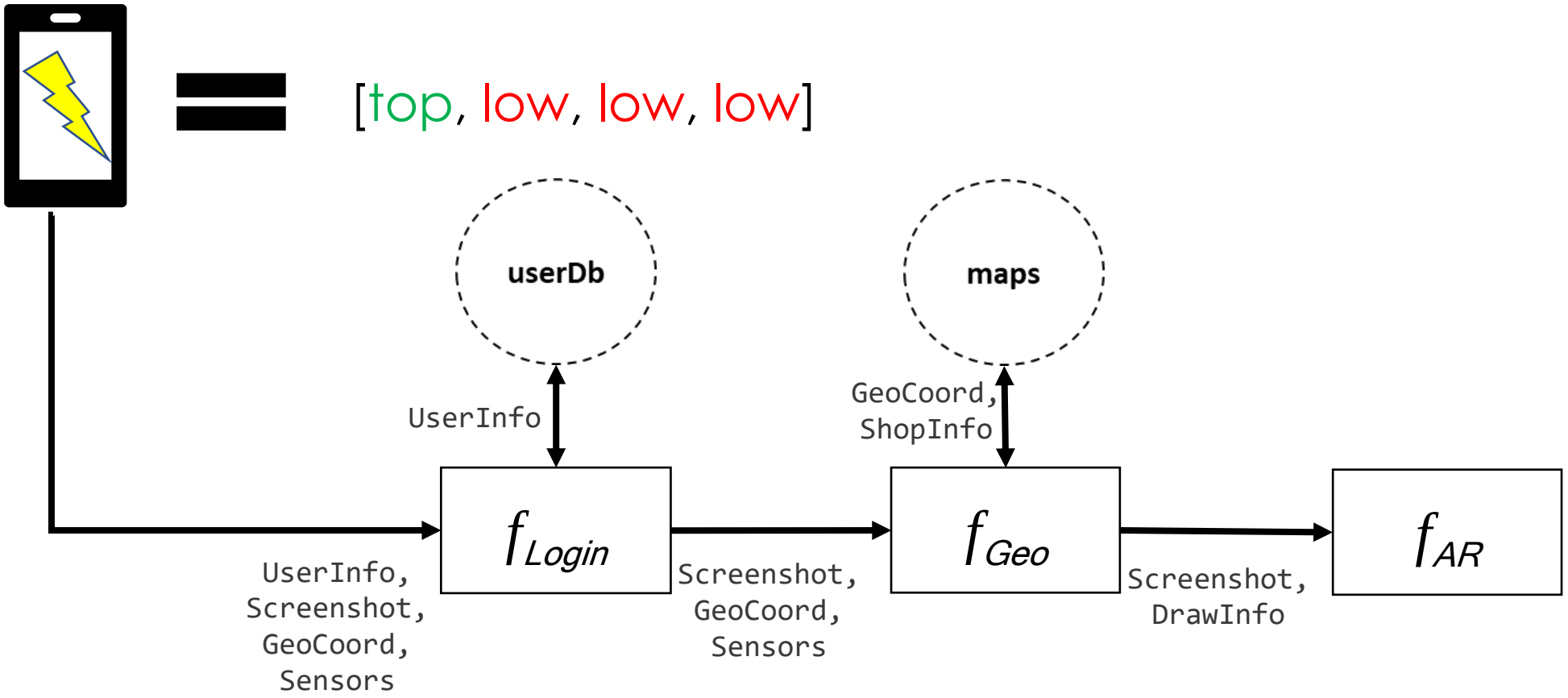


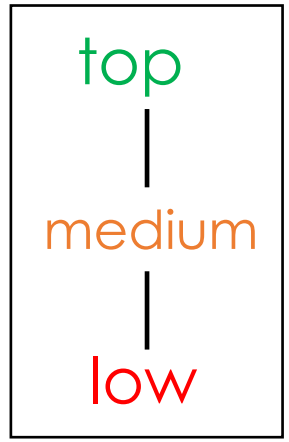
How FaaS2Fog works

Given a function chain it output a placement in two main steps:

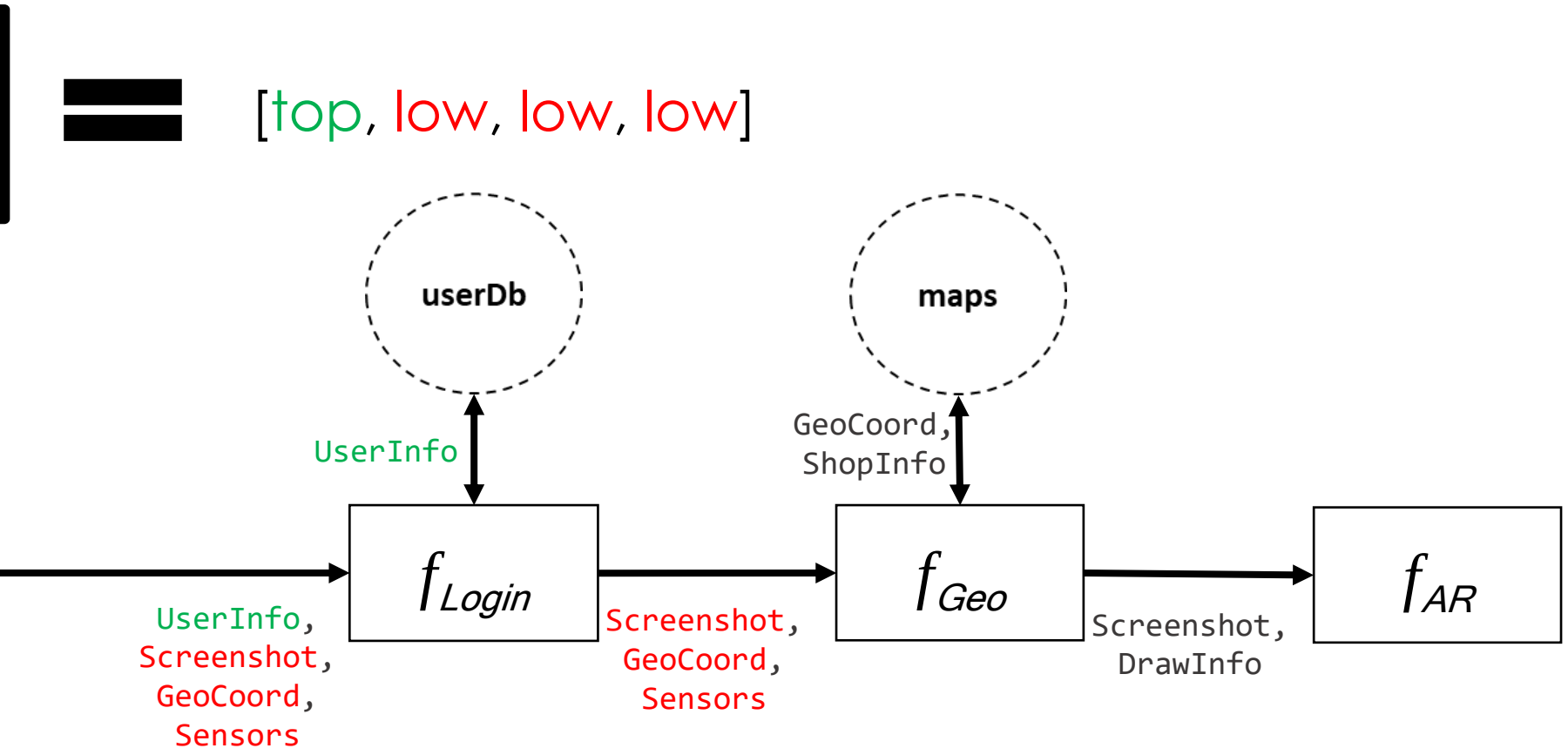
- Security type propagation trough the chain
- Mapping of typed functions onto the C-I infrastructure

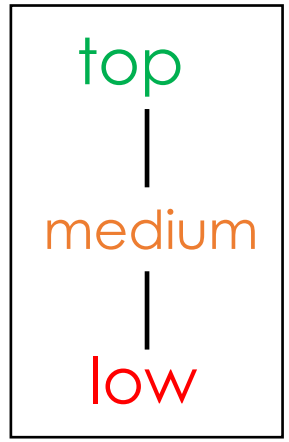
FaaS chain Label Propagation



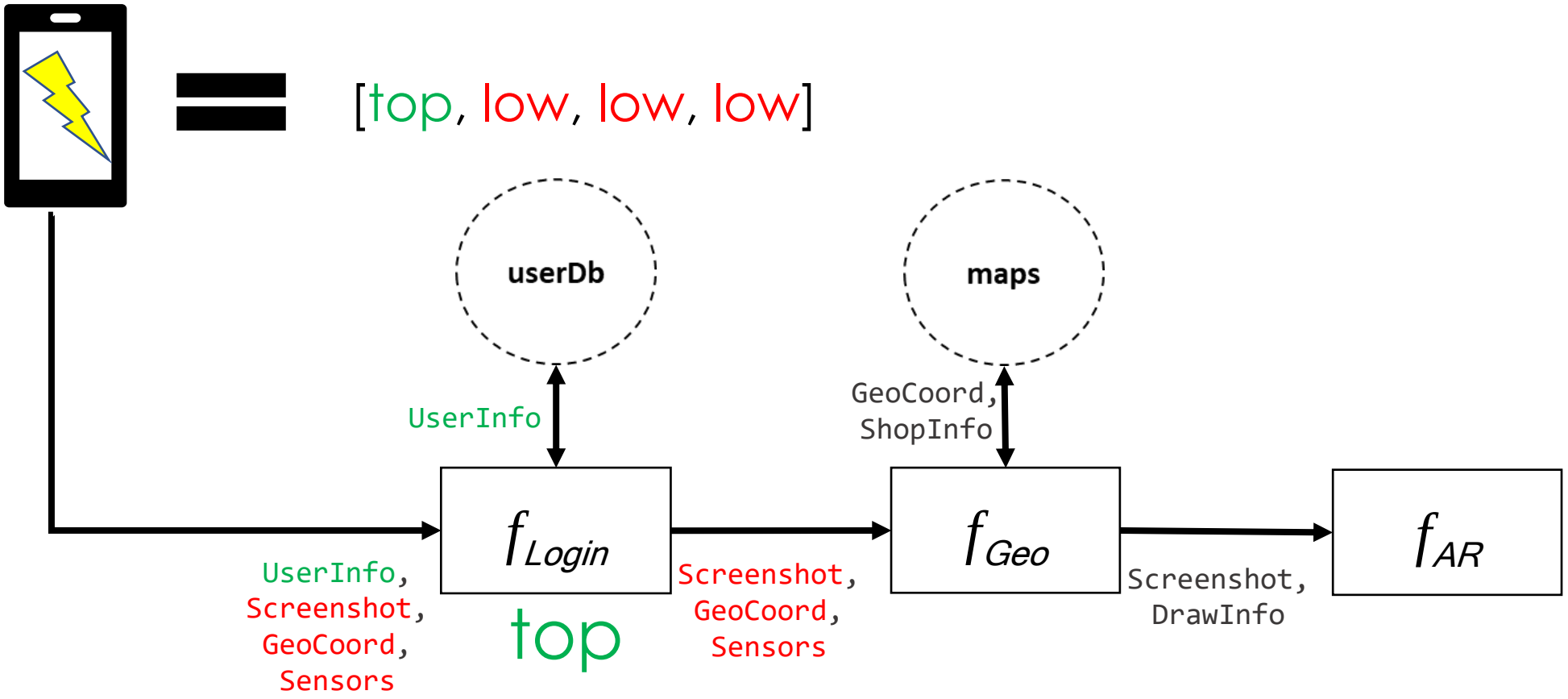


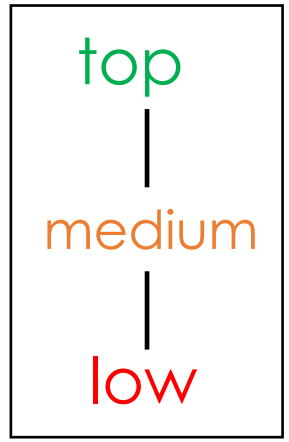
FaaS chain Label Propagation



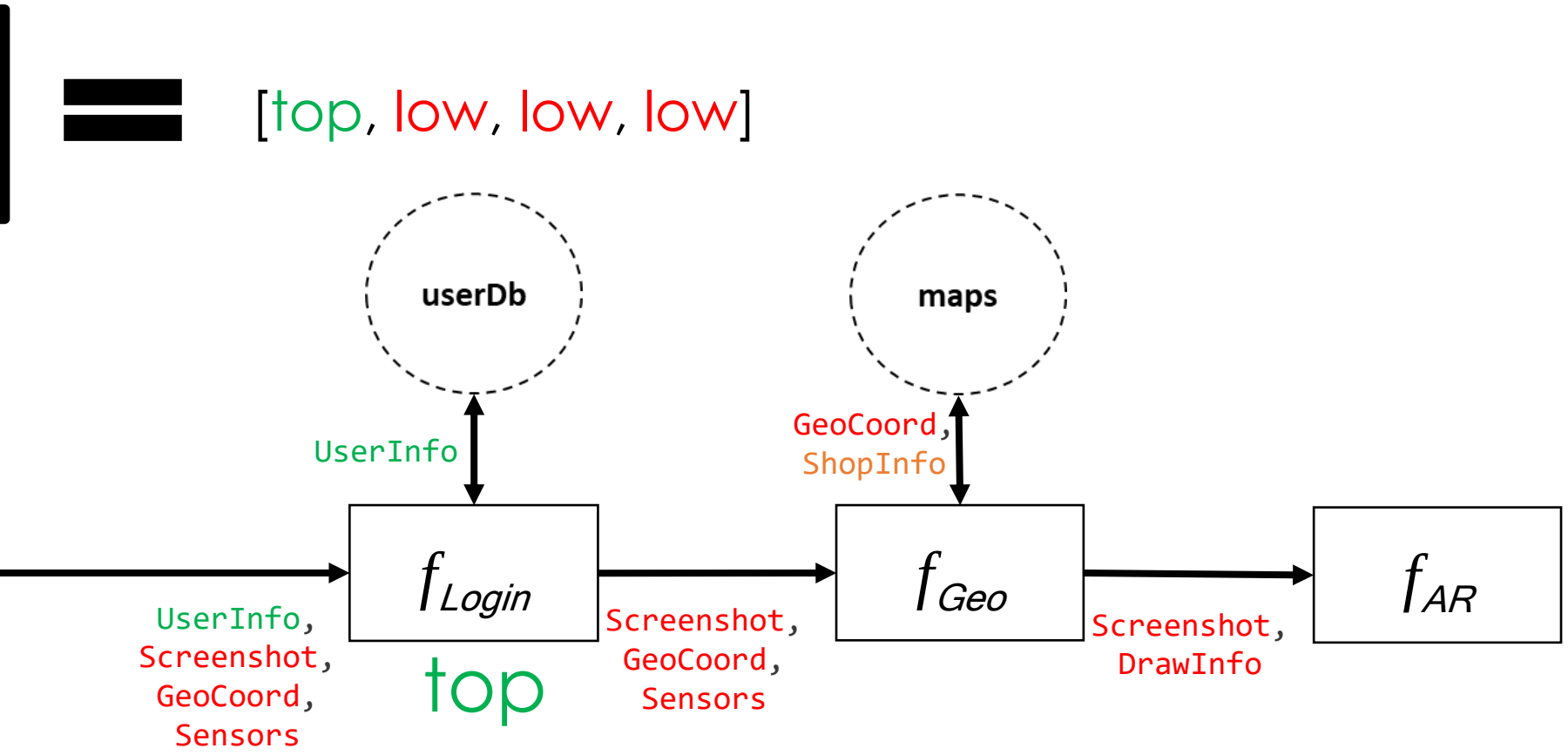


FaaS chain Label Propagation

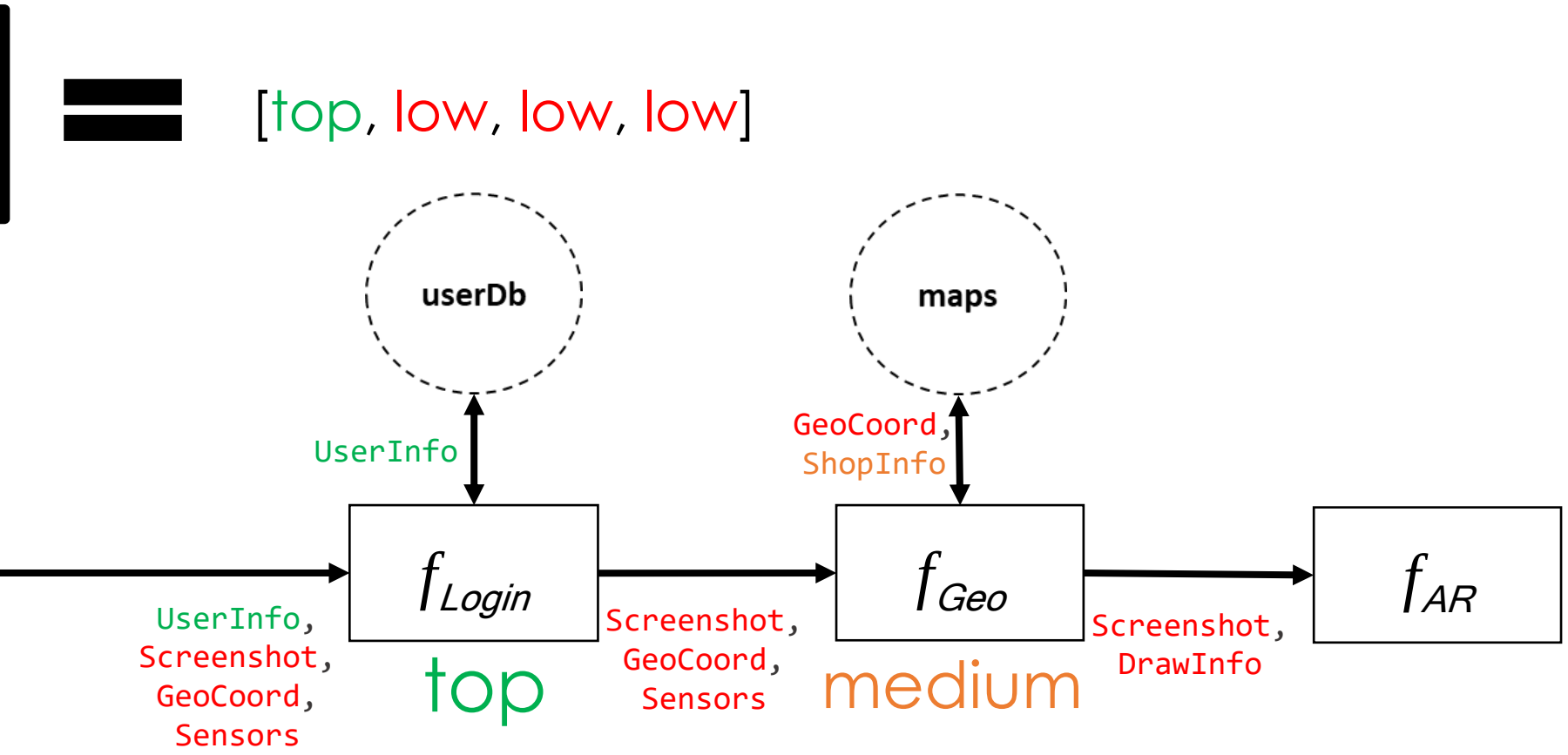




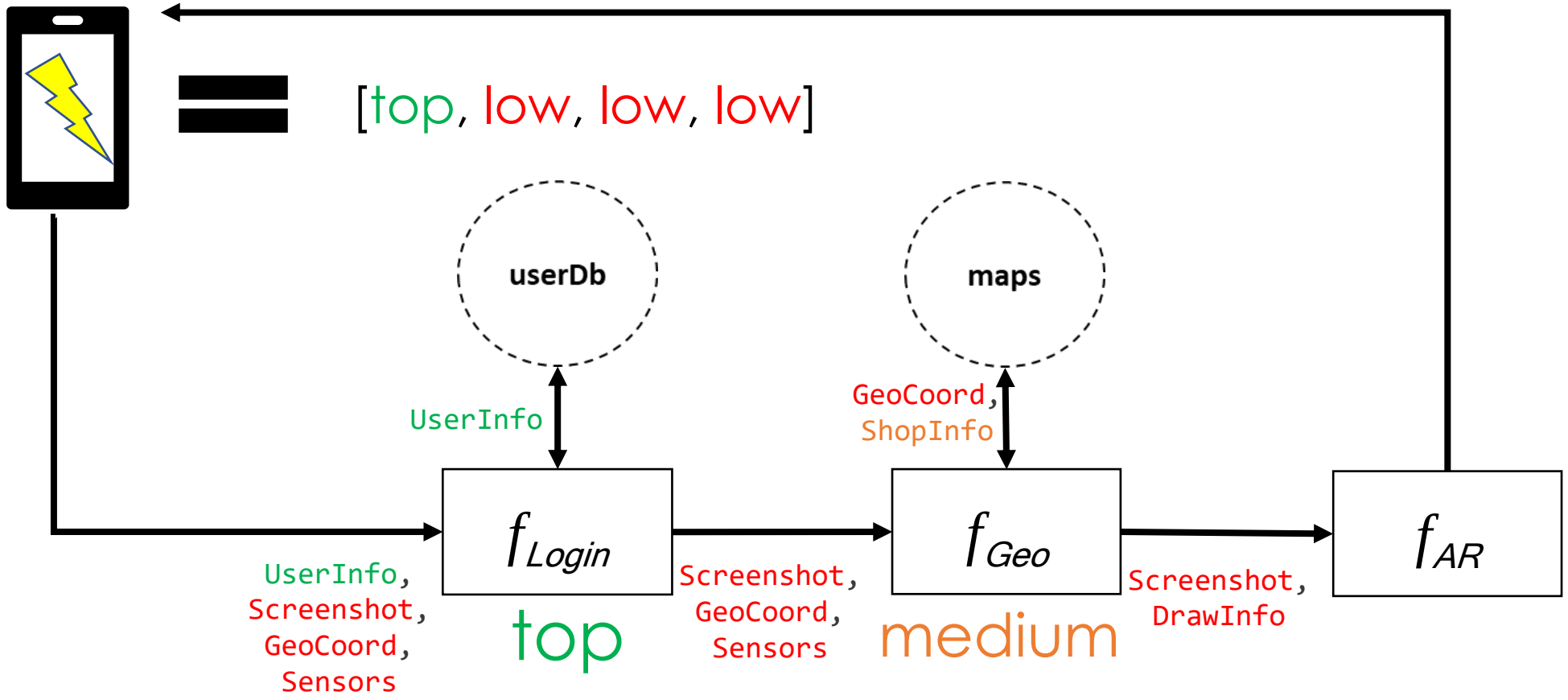
FaaS chain Label Propagation

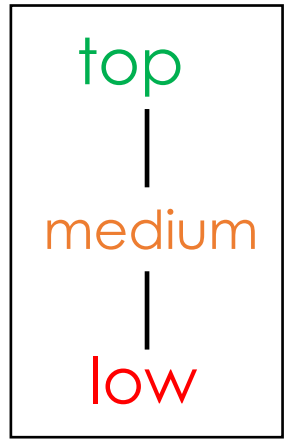


FaaS chain Label Propagation

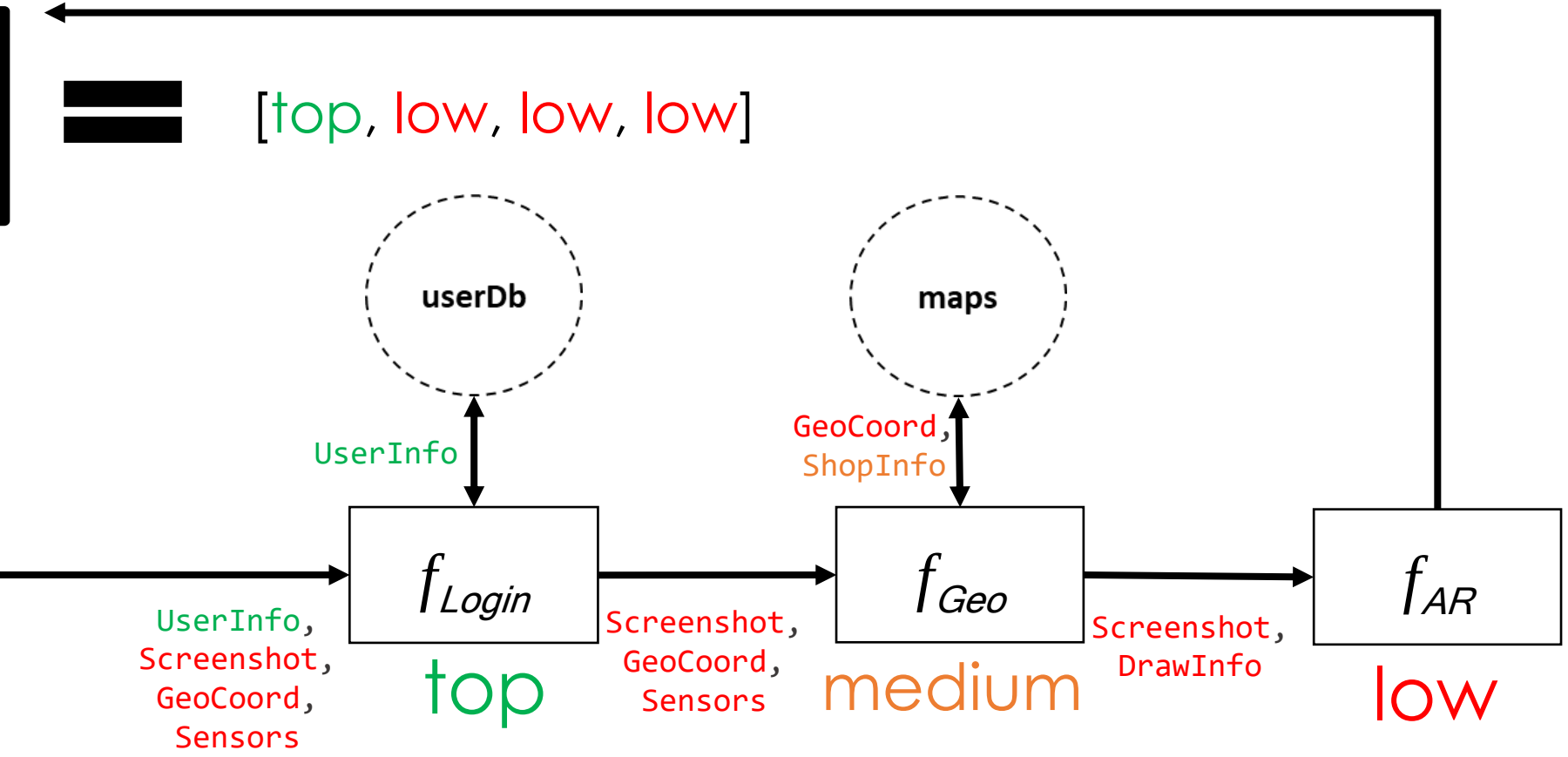


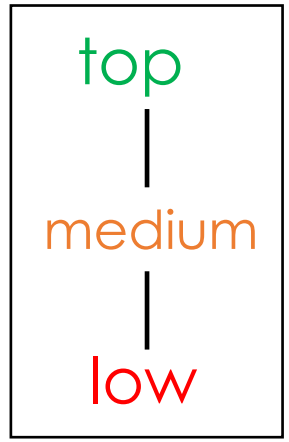
FaaS chain Label Propagation



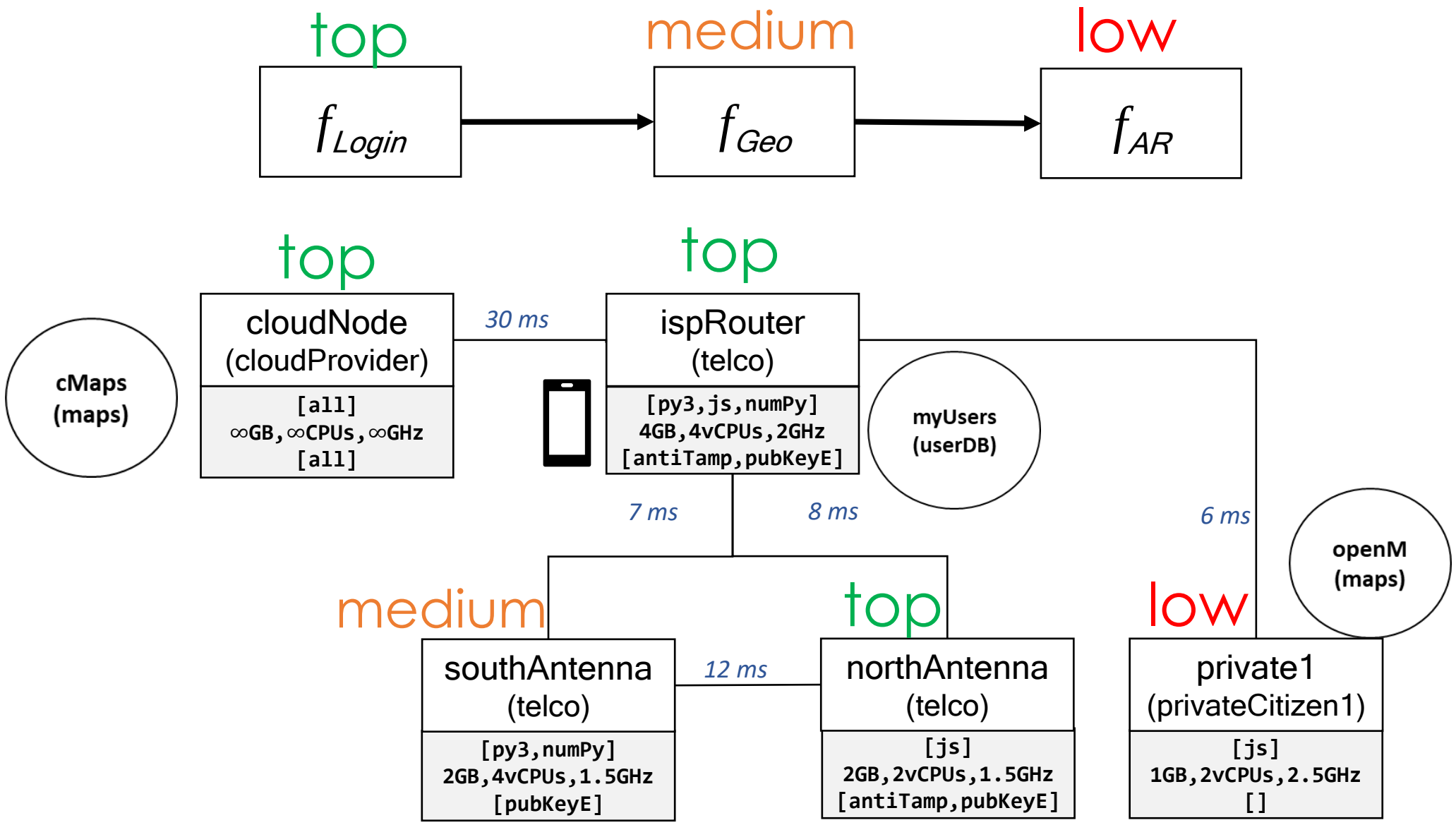


FaaS chain Label Propagation

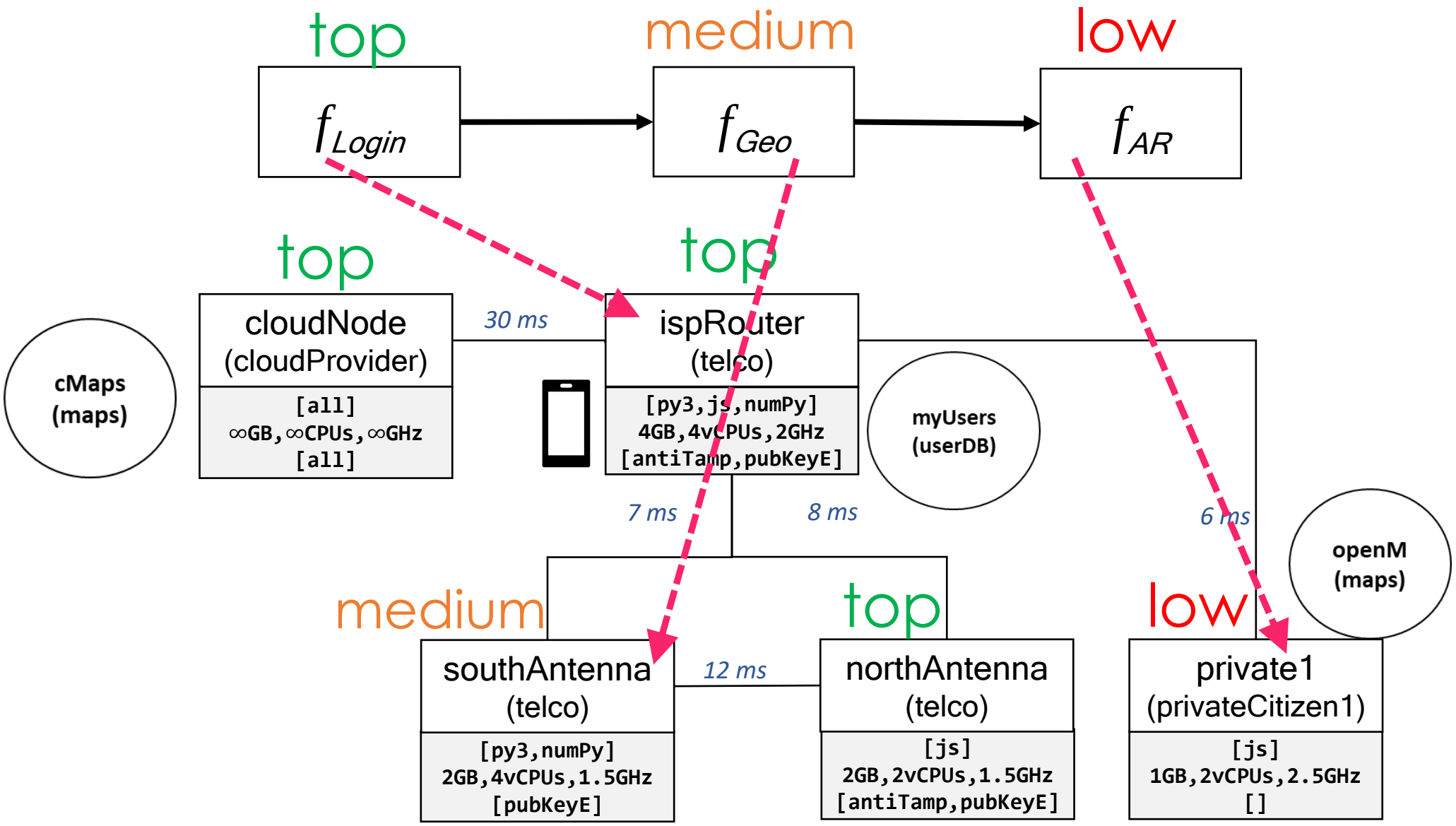




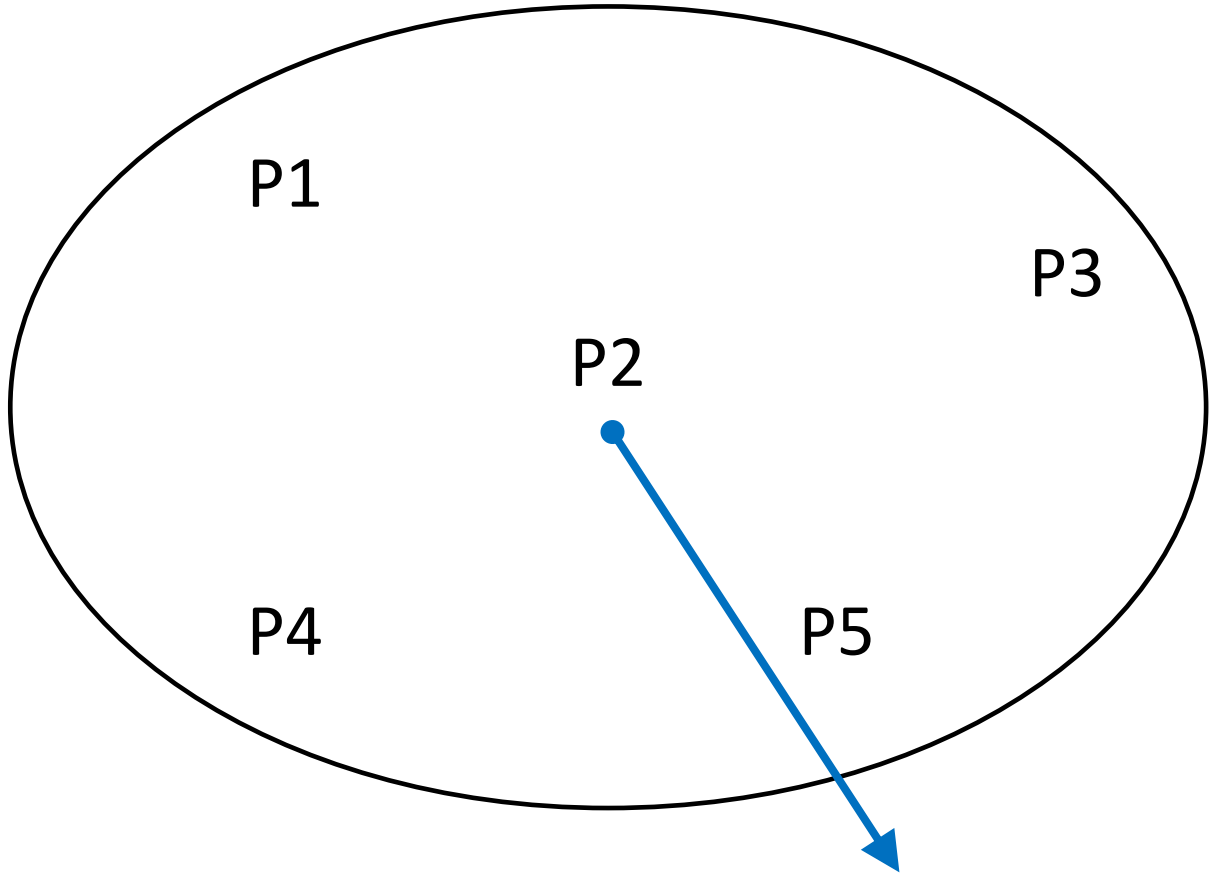
The mapping phase



The mapping phase

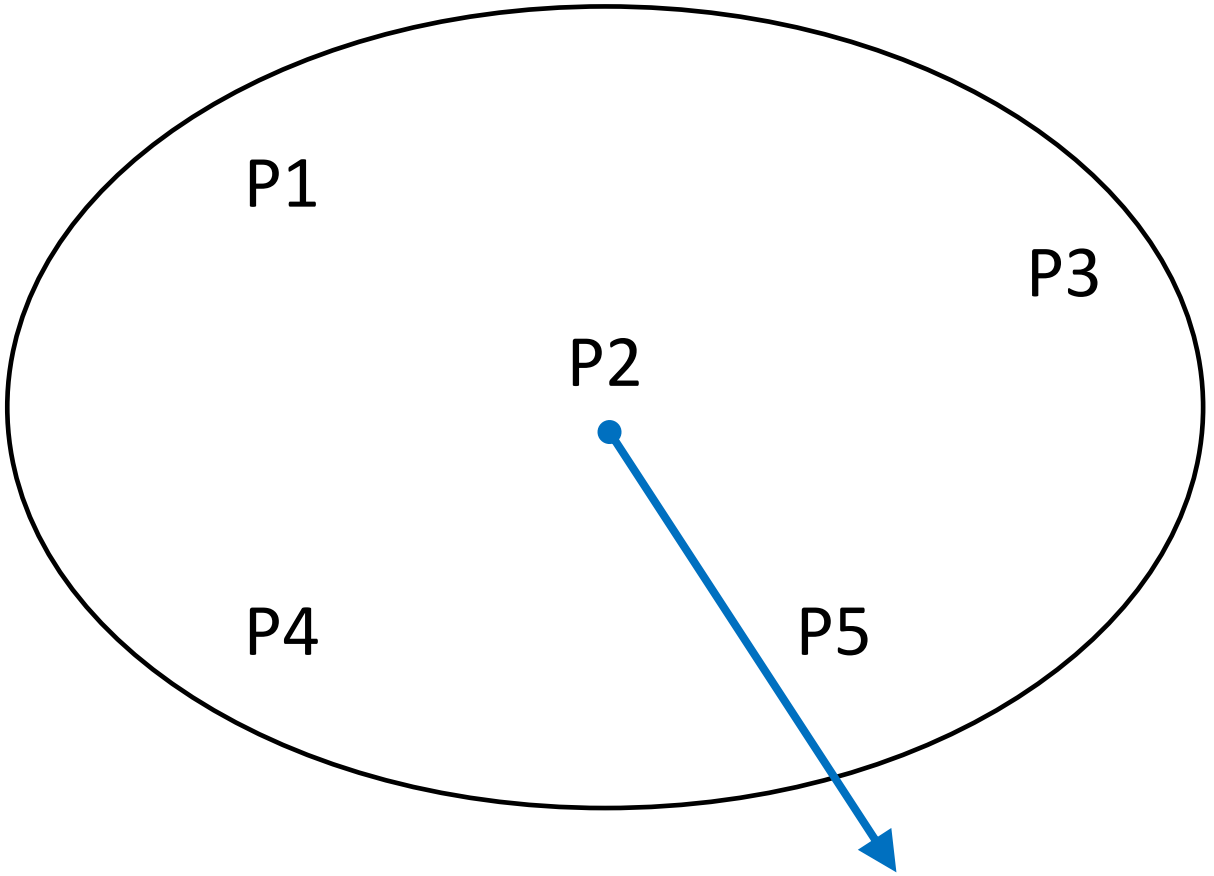


Results




```
P2 = [on(fLogin, ispRouter, [(userDB, myUsers, ispRouter)]),
on(fGeo, private1, [(maps, openM, private1)],
on(fAR, northAntenna, [])].
```

Results with trust



	Trust
P1	(0.27, 0.23)
P2	(0.61, 0.31)
P3	(0.27, 0.23)
P4	(0.77, 0.48)
P5	(0.34, 0.35)

```
P2 = [on(fLogin, ispRouter, [(userDB, myUsers, ispRouter)]),
on(fGeo, private1, [(maps, openM, private1)],
on(fAR, northAntenna, [])]: (0,61, 0,31).
```

Code & Demo

Ongoing work

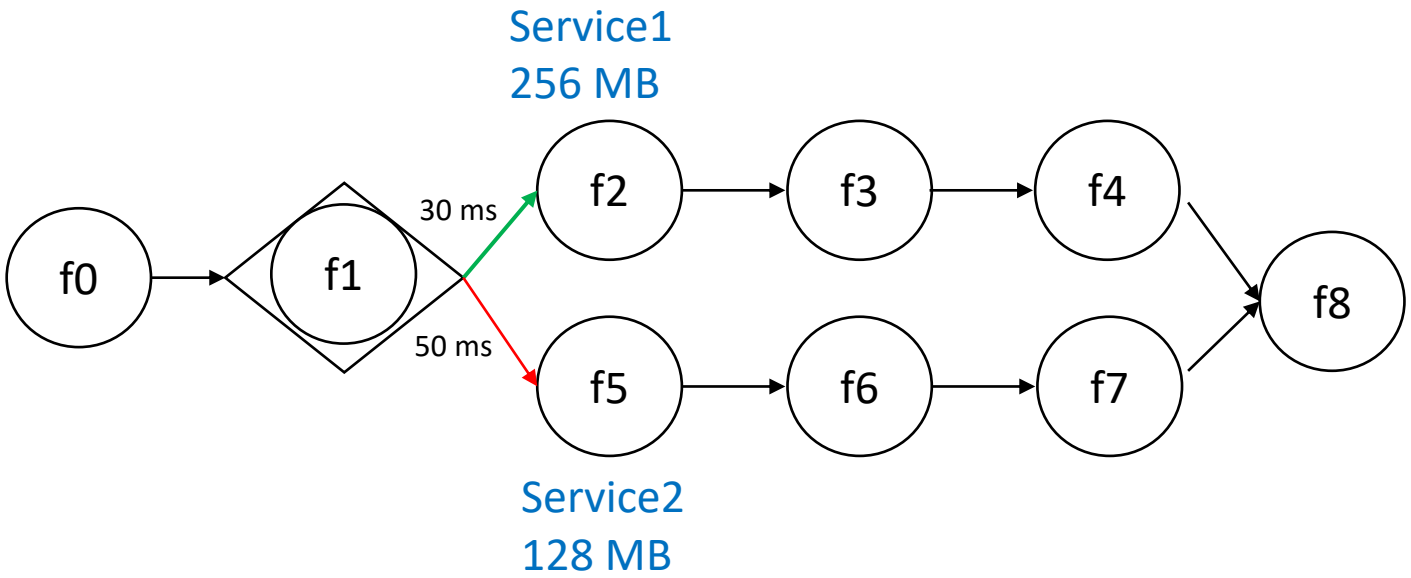
More complex function orchestrations.

Grammar accepted:

```
Exp     ::= Seq | If | F | Par
Seq     ::= seq(Exp, Exp)
If      ::= if(F, Exp, Exp)
F       ::= fun(FId, Bindings, Latency)
Par     ::= par(Explist)
Explist :: = Exp * Exp
```

Conditional Branches

Deployment could lead to leaking the if guard value



As we saw: Attacker model

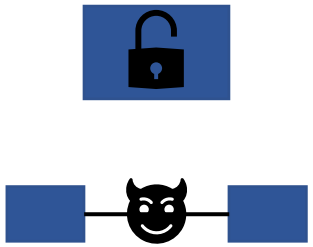
Try to disclose data confidentiality of application's clients.

Knowledge Base

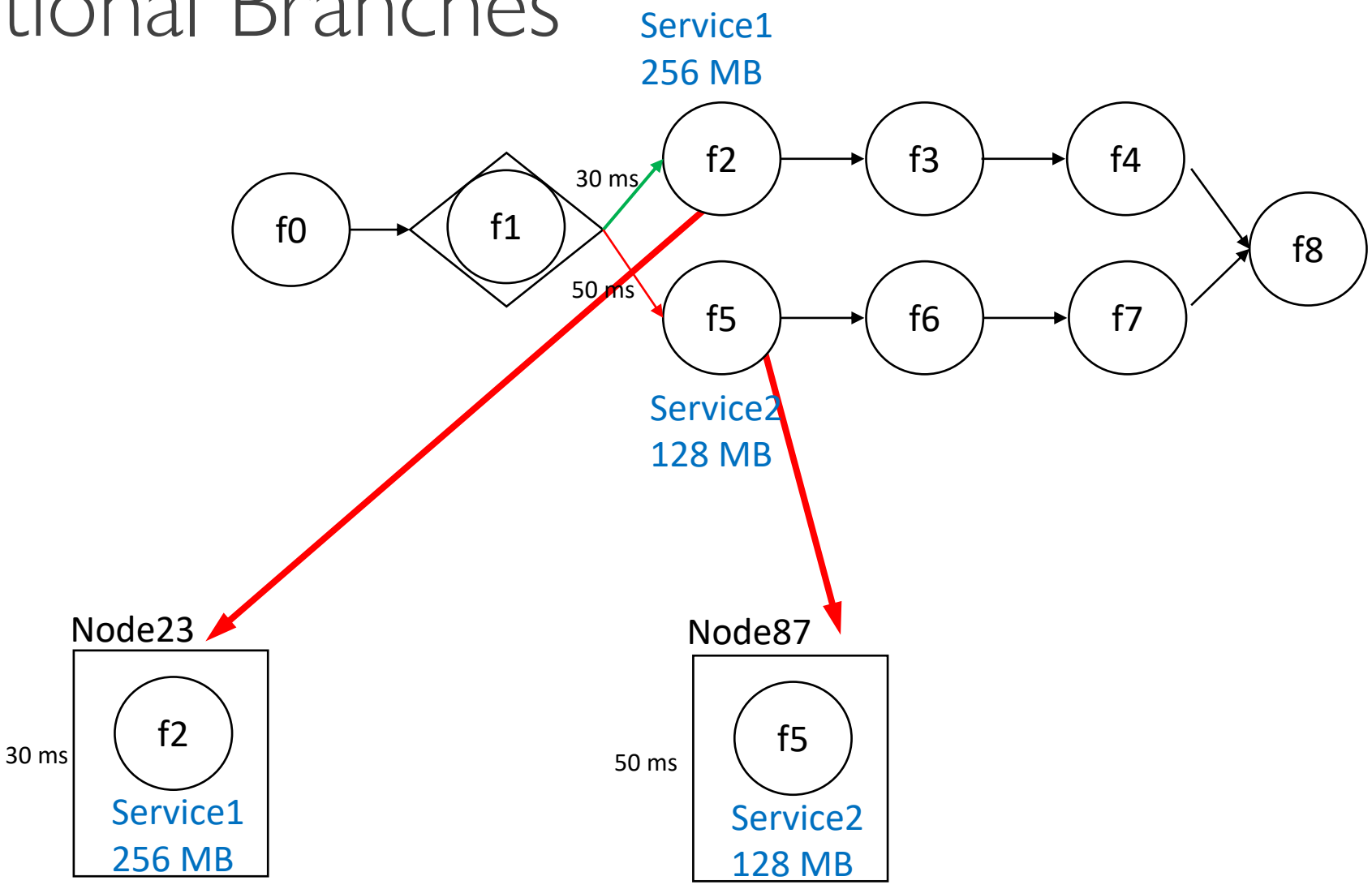
- Public data
- Node resources
- Services placement
- Application structure
- Application external behaviour

Attacks

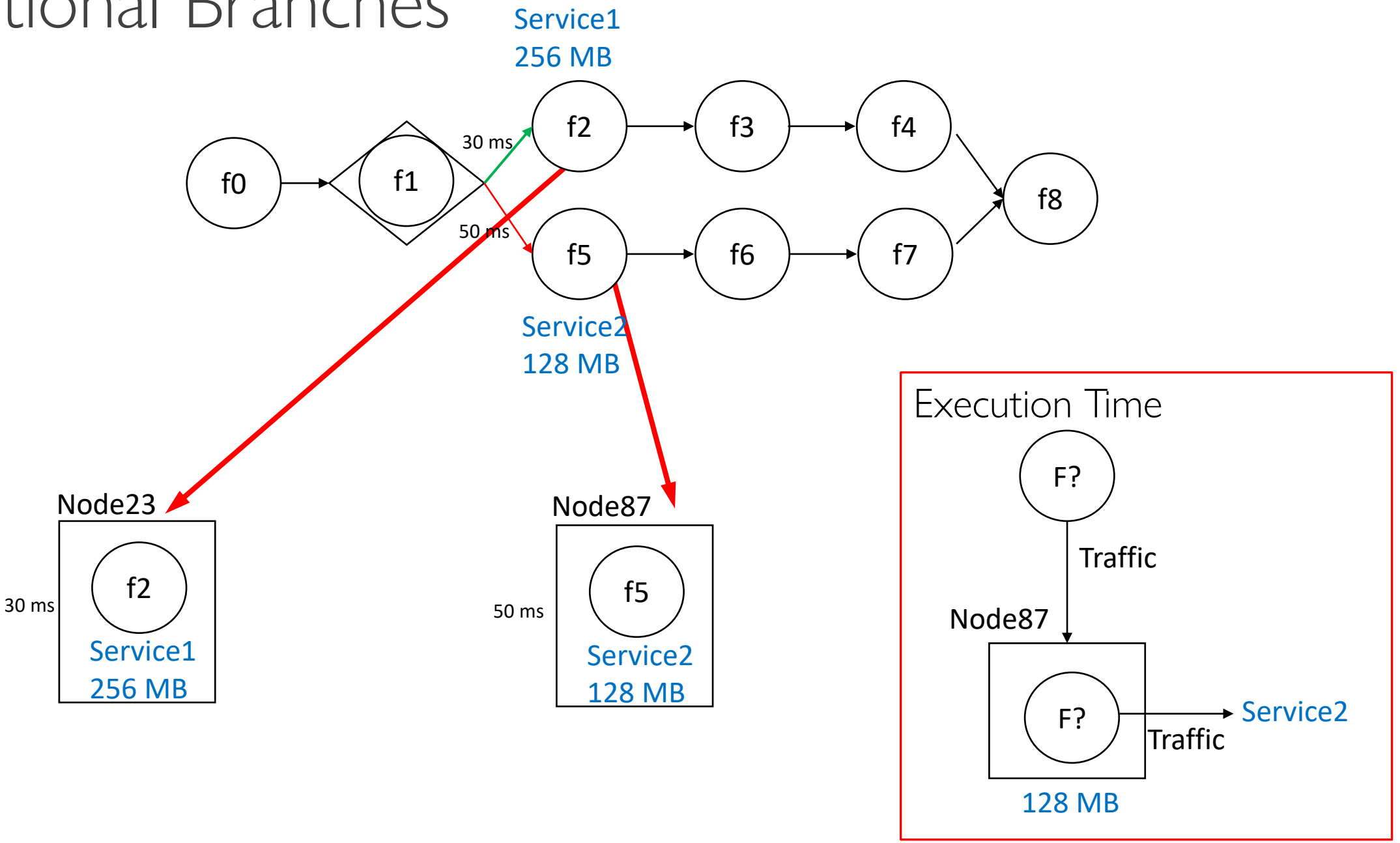
- Hack weak nodes
- Traffic monitoring



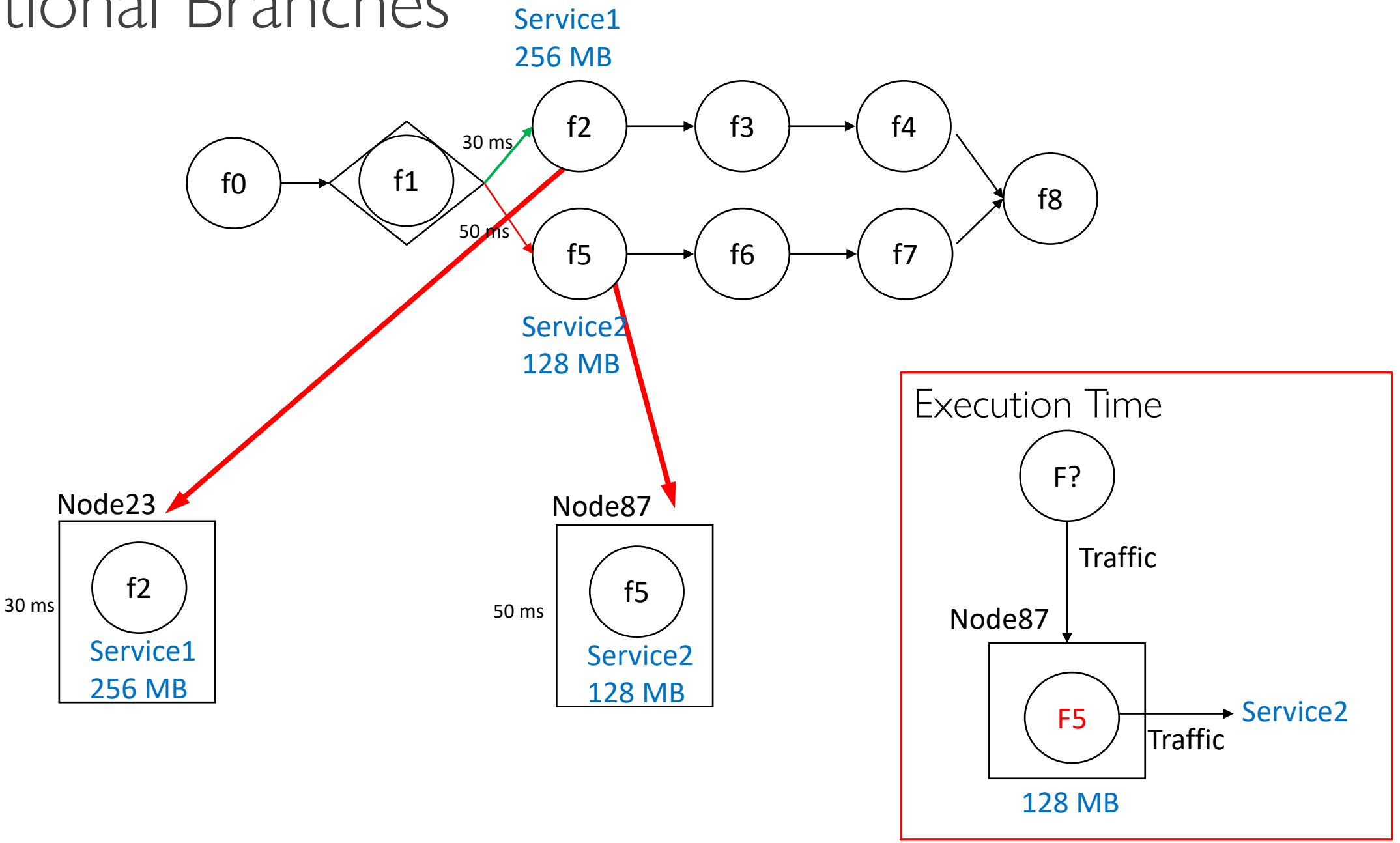
Conditional Branches



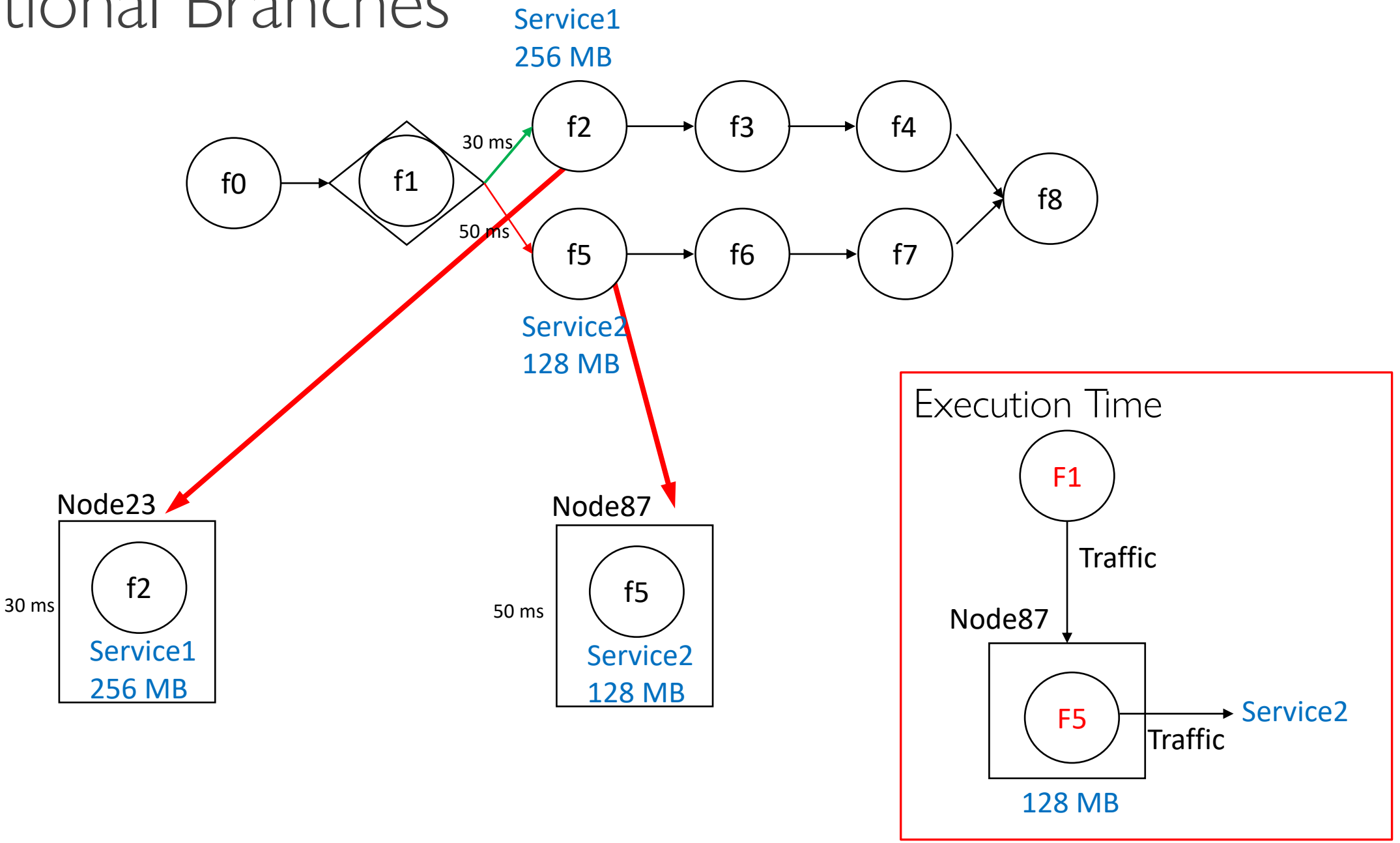
Conditional Branches



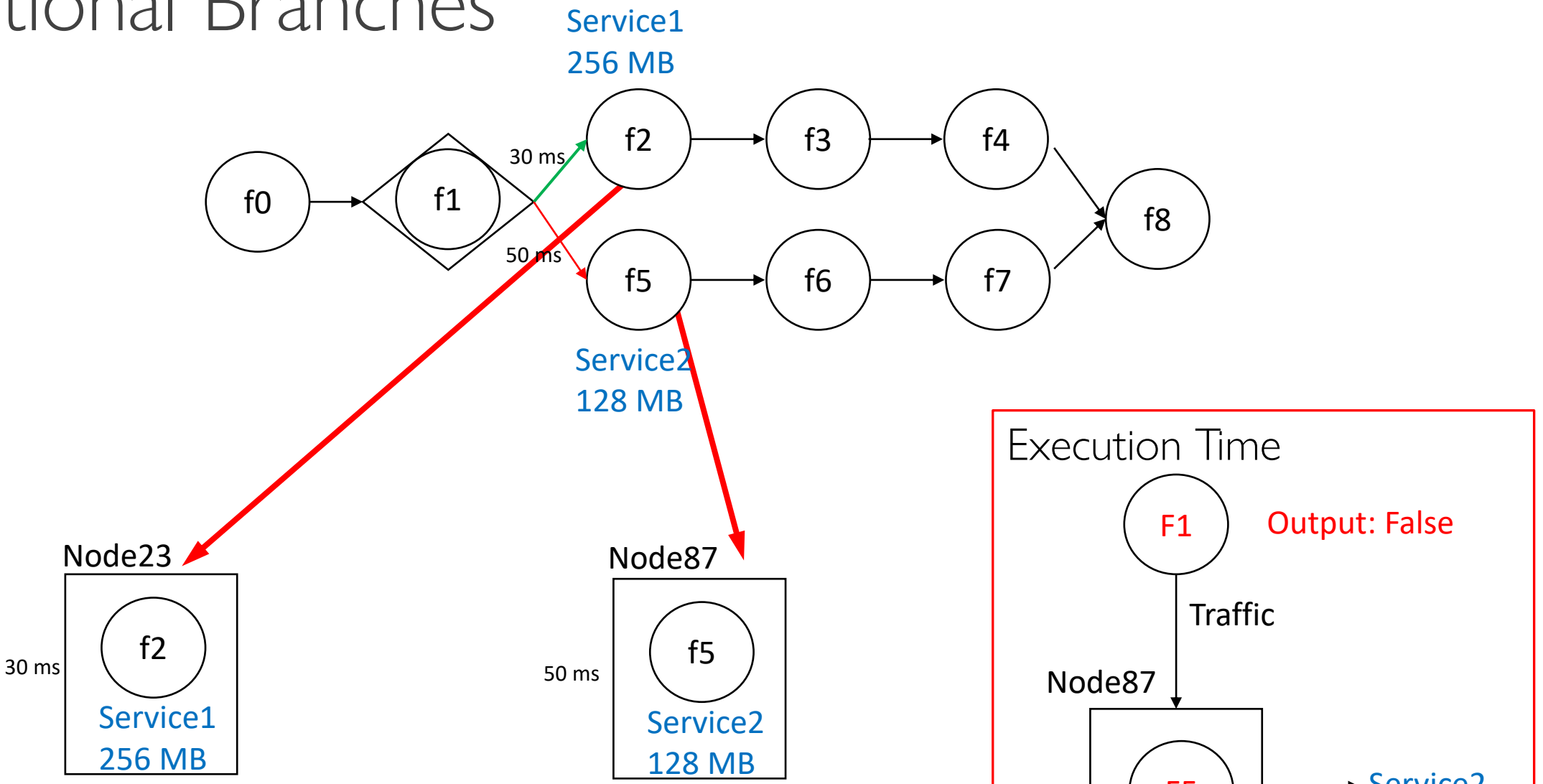
Conditional Branches



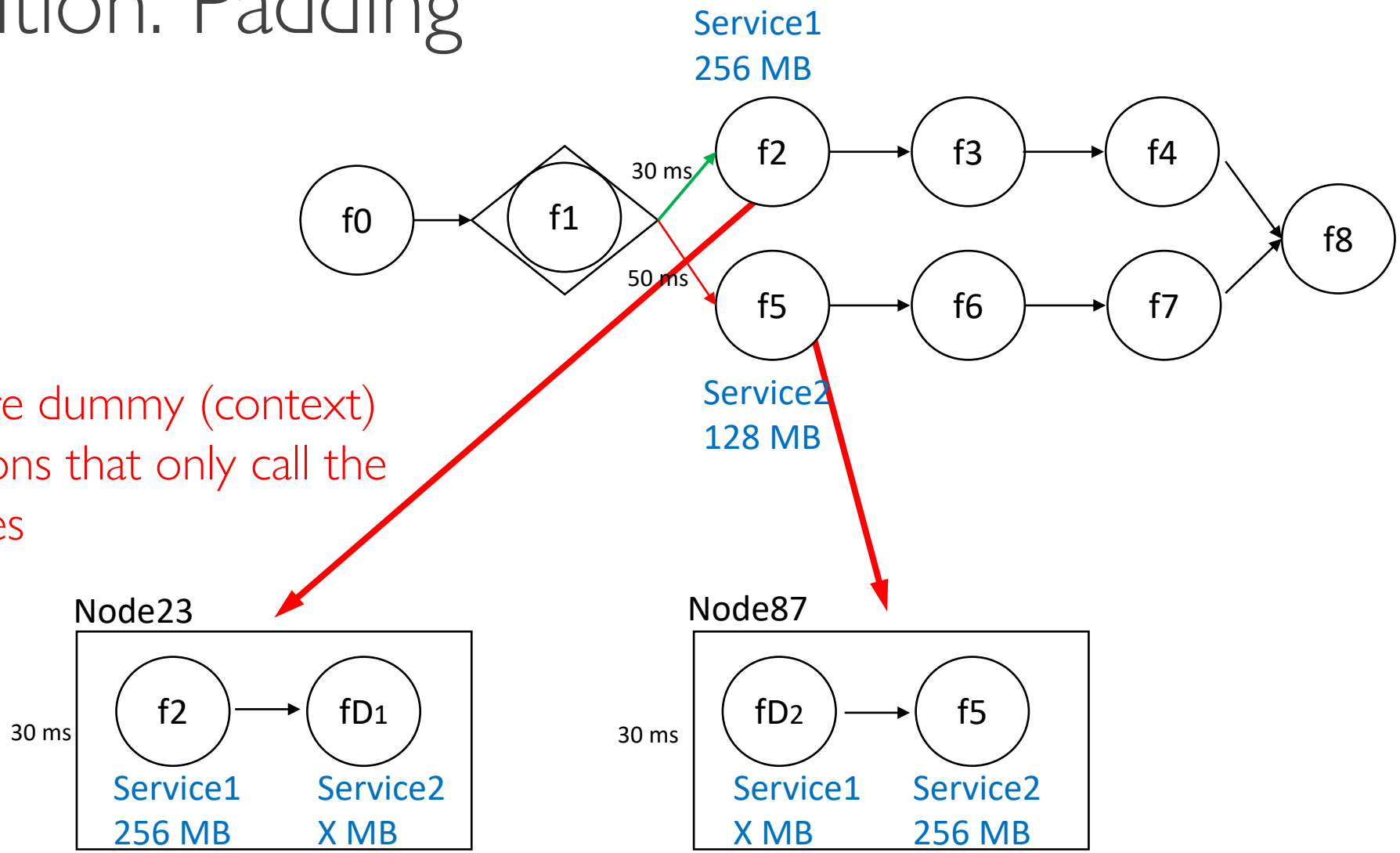
Conditional Branches



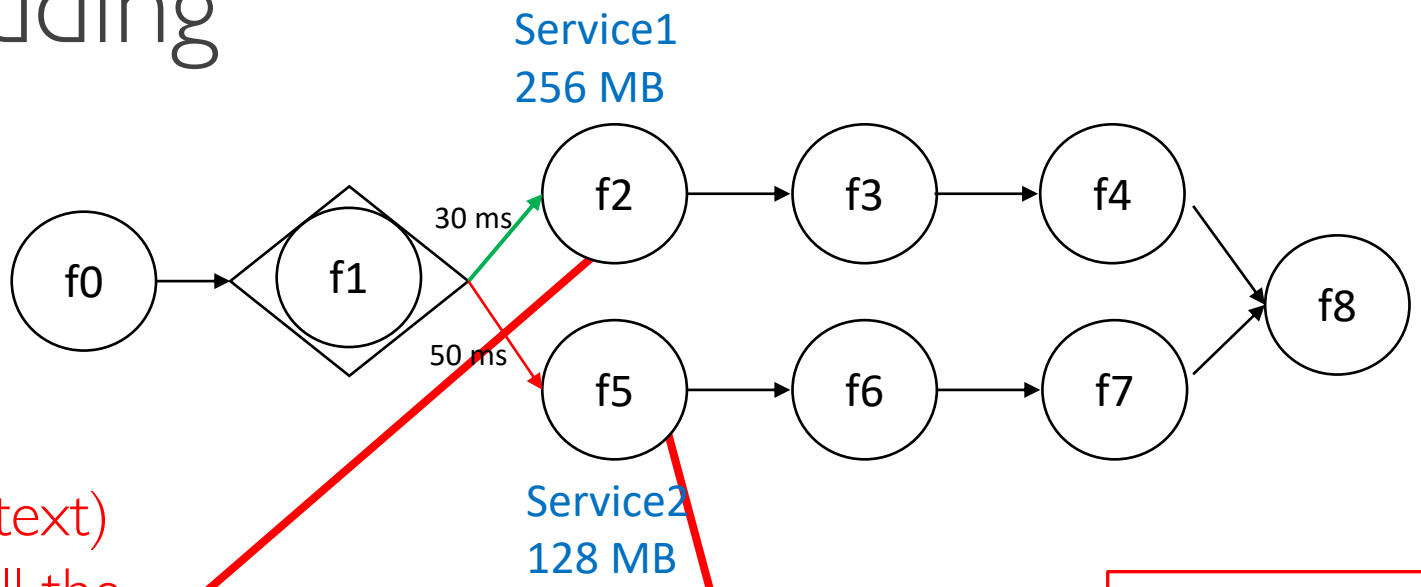
Conditional Branches



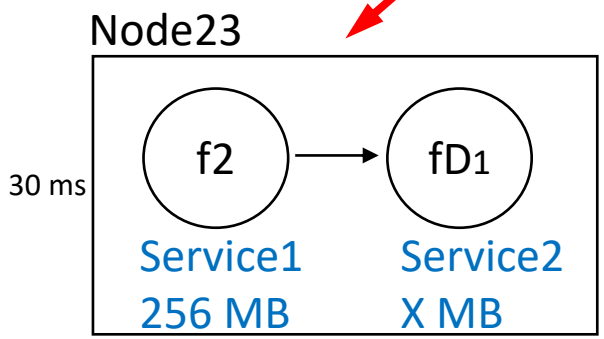
Solution: Padding

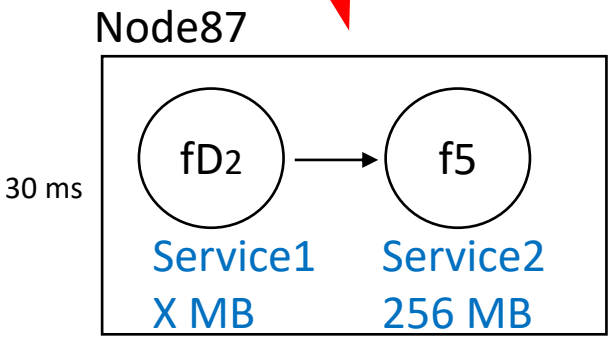


Solution: Padding

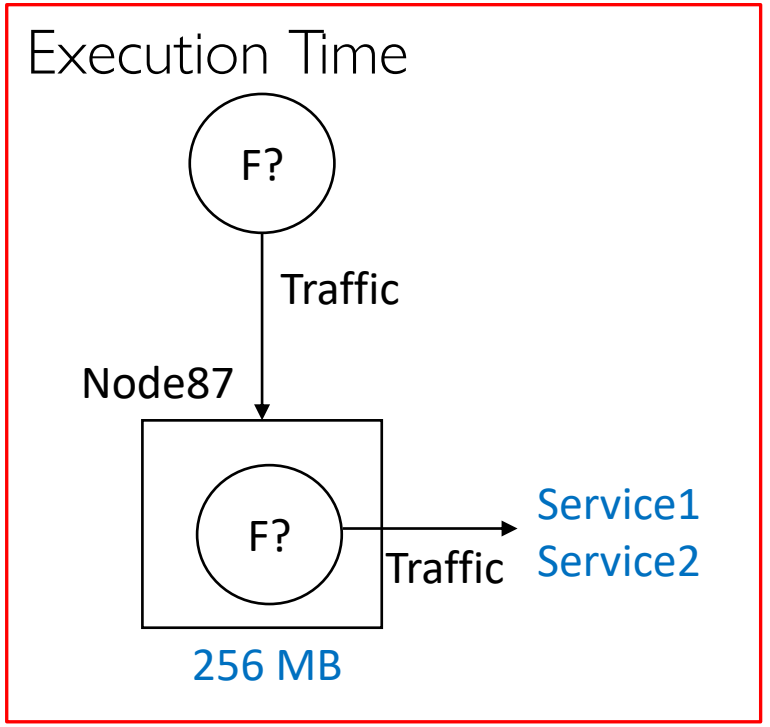


fDx are dummy (context) functions that only call the services





Seq at latency 0



Bibliography

- State of the Art:

  A. Bocci, S. Forti, G.-L. Ferrari, A. Brogi, *Secure FaaS orchestration in the fog: how far are we?* Computing 103.5 (2021): 1025-1056.

- FaaS2Fog:

  A. Bocci, S. Forti, G-L Ferrari, A. Brogi *Placing faas in the fog, securely*. ITASEC 2021. CEUR Workshop Proceedings, vol. 2940, 166-179