Directions for COIN
draft-kutscher-coinrg-dir-01

Dirk Kutscher, Jörg Ott, Teemu Kärkkäinen

22 November 2019 – IRTF COINRG

Reminder: Outline

- What does in-network really mean?
  - Exploring numerous (present and future) options

- Some thoughts on computing
  - Looking at code and its provisioning, execution, etc.

- What could/should COIN look at?

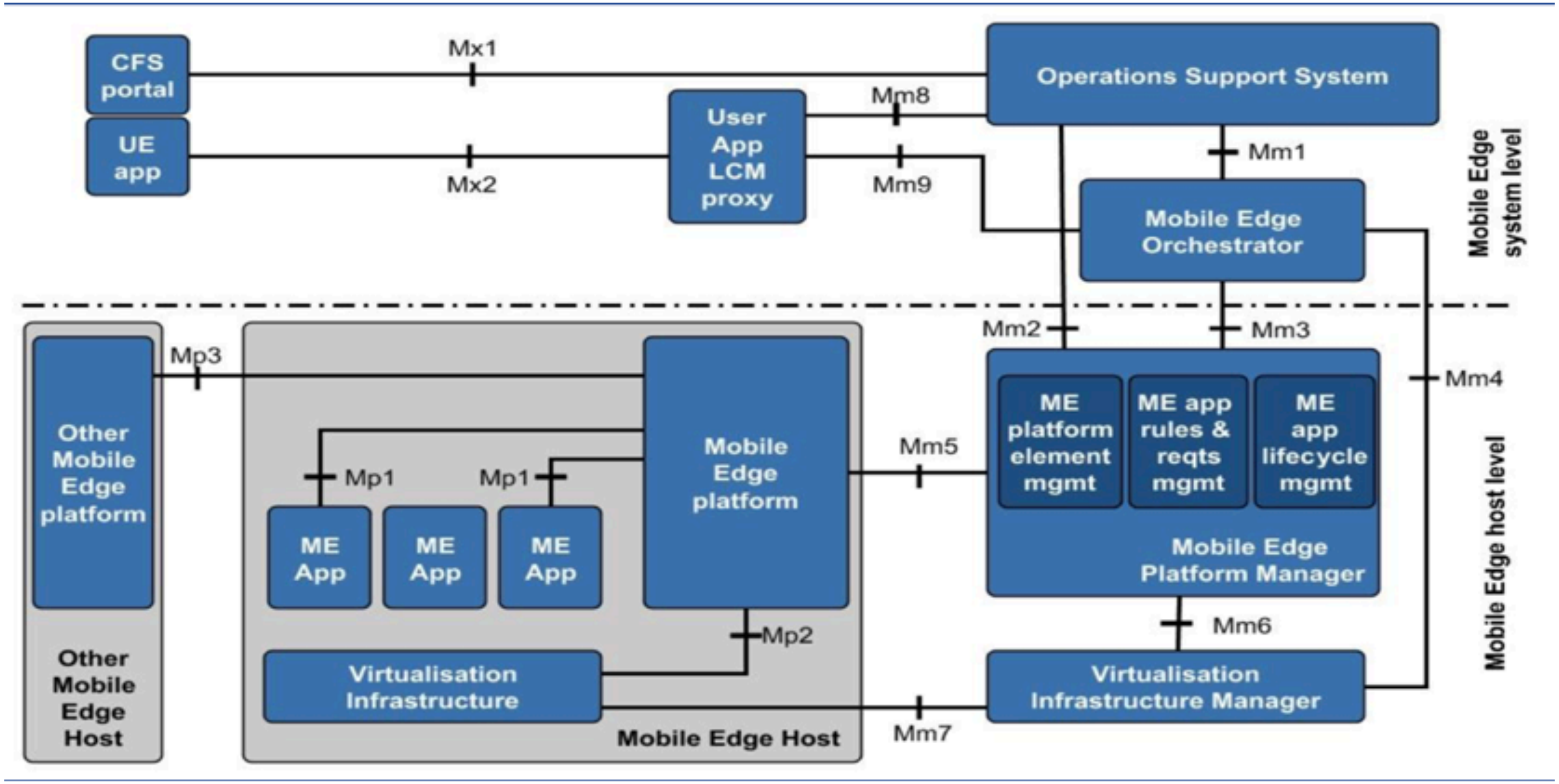
What does “in-network” really mean?

Lots of Computing “in the Network“ Today

- SmartNICs
- Web servers
- CDNs
- Cloud platforms
- Note: Some forms of „Edge Computing“ are merely about extending the cloud computing concept to specific hosts at the edge

- These approaches are applied (more or less) successfully today and do not need COIN research…
  - ...but there is lots of engineering to be done in the IETF

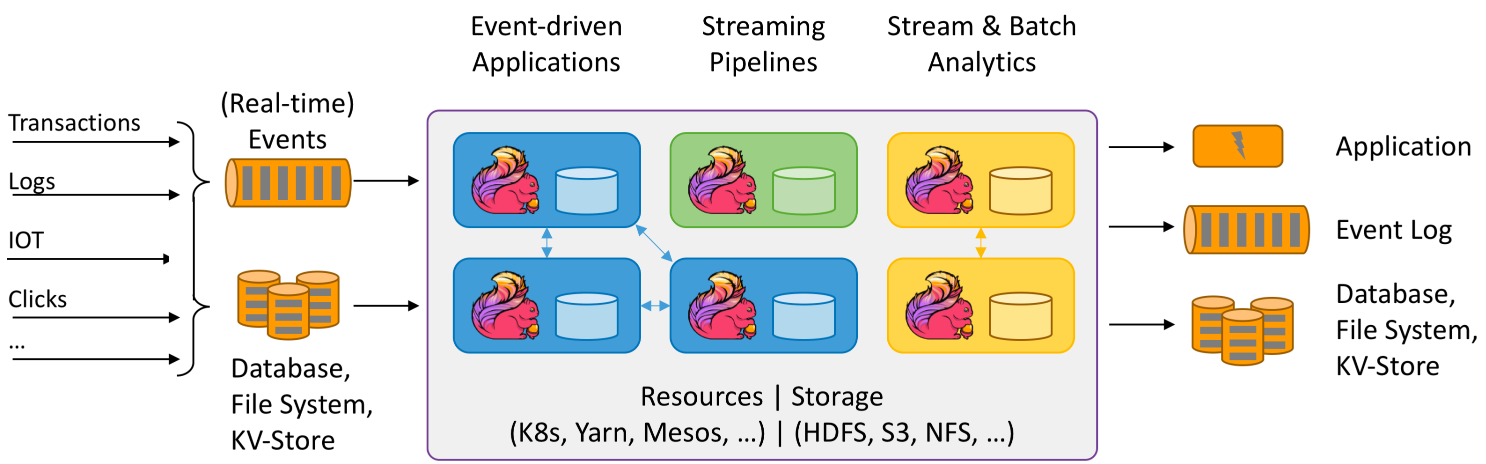
Example: Mobile Edge Computing



https://datatracker.ietf.org/meeting/98/materials/slides-98-nfvrg-sessb-12-multi-access-edge-computing-mec-applications-00

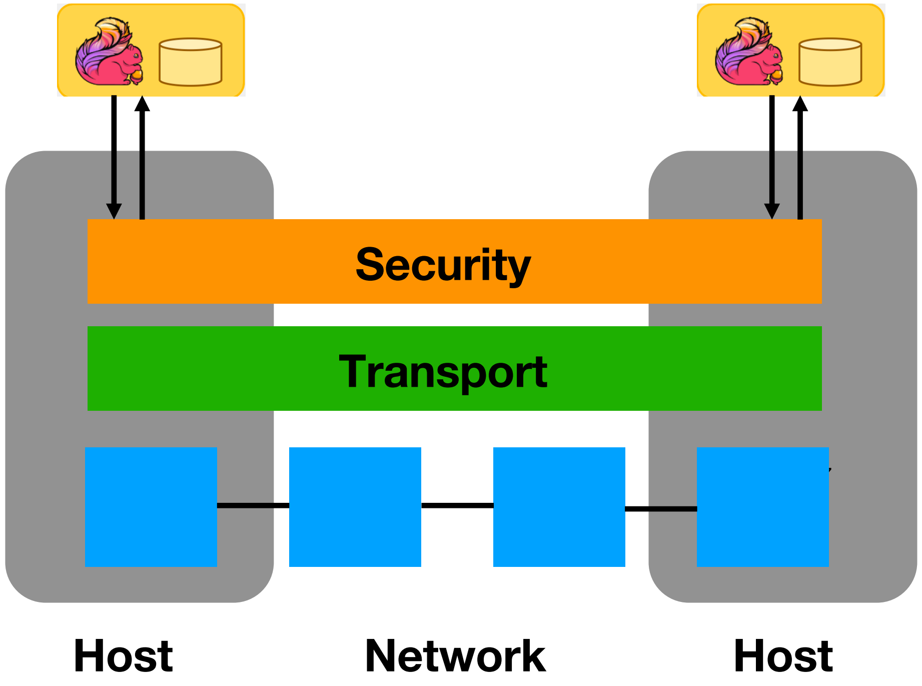
Example: Streaming Frameworks

APACHE Spark APACHE kafka A distributed streaming platform samza Flink

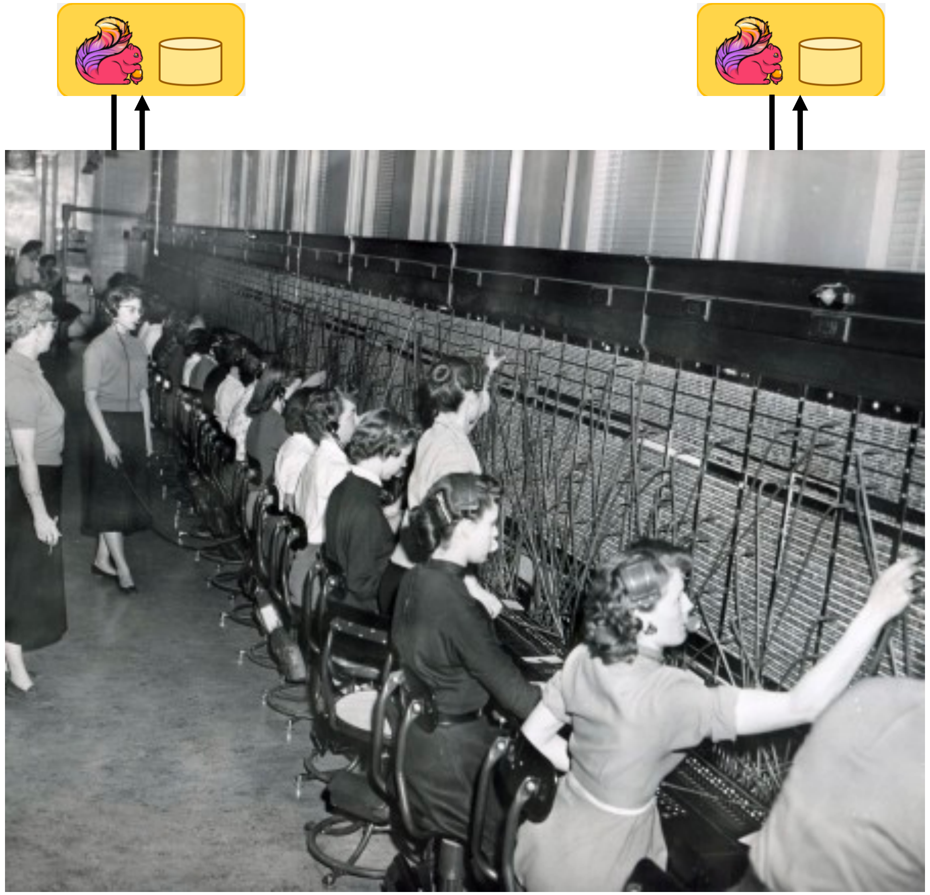

Event-driven Applications
Streaming Pipelines
Stream & Batch Analytics
Transactions
Logs
IOT
Clicks
...
(Real-time) Events
Database, File System, KV-Store
Resources | Storage
(K8s, Yarn, Mesos, …) | (HDFS, S3, NFS, …)
Application
Event Log
Database, File System, KV-Store

- Elaborate services and guarantees for different use cases
- Apache Flink: Different streaming connectors — but typically as network overlays

Decoupling Computing from the Network



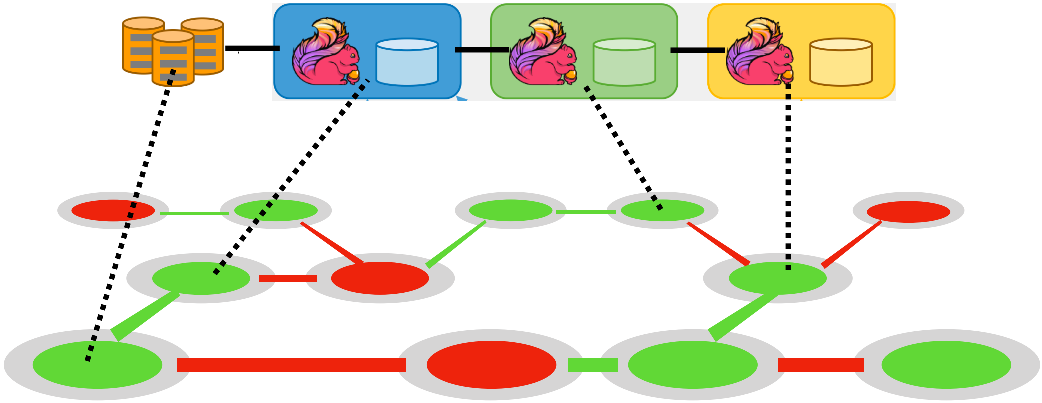
Decoupling Computing from the Network



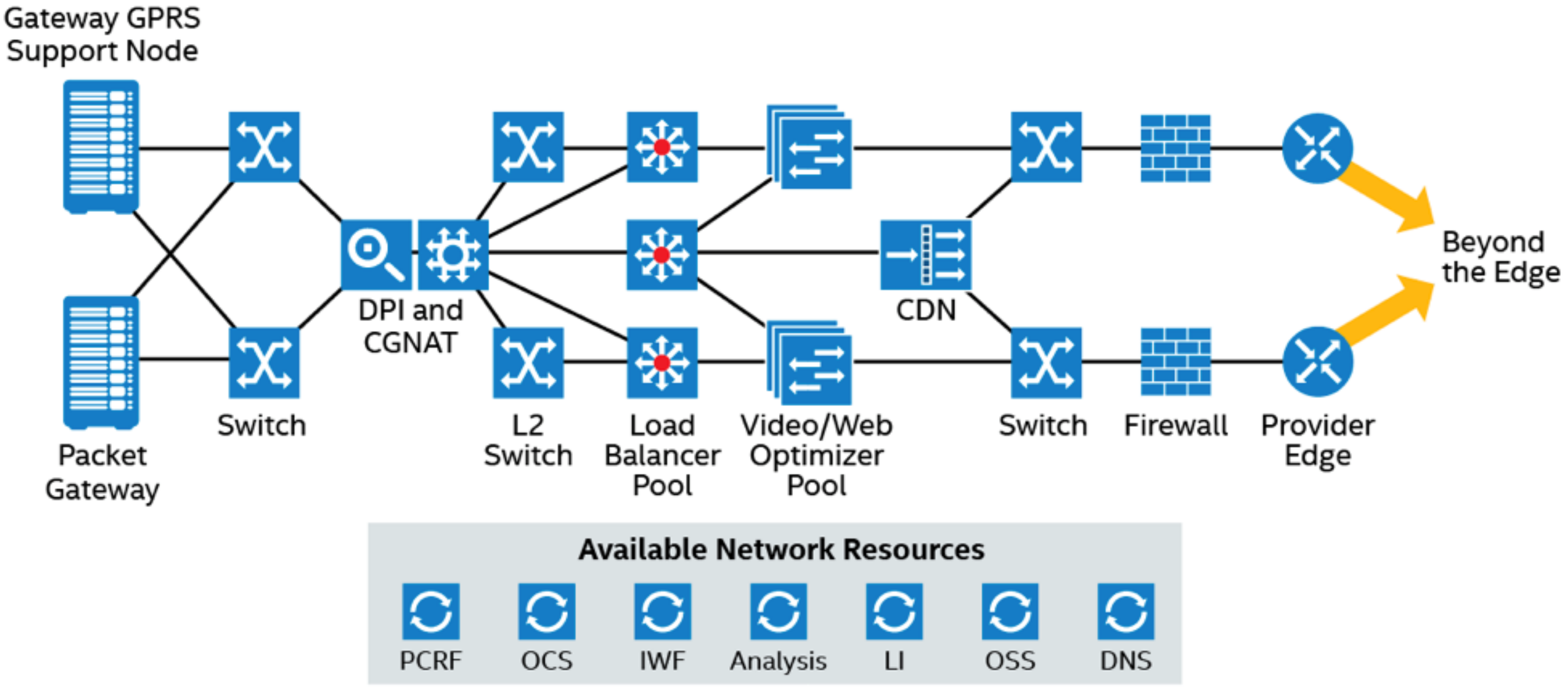
- Circuit-like connectivity
  - Limited visibility into network
- Different namespaces
  - DNS, discovery
- Trust often centralized
  - PKIs for TLS certificates etc.

Computing in the Network

- Do not require fixed locations of data and computation
- Can lay out processing graphs flexibly – meeting requirements optimally
  - Sometimes we can move functions (to be close to large data assets)
  - Sometimes we gradually move data where it is needed (e.g., where specific computations run)
- Conditions may change dynamically and constantly: network to adapt to application requirements, network conditions etc.
- **Optimization based on application requirements & view of all relevant resources**



Version 01 Updates (1/3)
Service Function Chaining



CDN – content delivery network; **CGNAT** – Steering/Carrier Grade Network Address Translation; **DPI** – deep packet inspection; **DNS** – domain name system; **GPRS** - General Packet Radio Service; **IWF** – interworking function; **LI** – lawful interception; **OCS** – online charging system; **OSS** – operational support system; **PCRF** – policy and charging rules function

https://builders.intel.com/blog/implementing-dynamic-service-function-chaining-for-gi-lan-uses/

Version 01 Updates (1/3)

- Service Function Chaining (SFC) for connecting compute
  - In general: SFC is flow (packet) steering
  - Forwarding encapsulated packets to IP hosts
  - Background: connecting VNFs (often in telco cloud)
  - RFC 8677: naming function & mapping to lower layer identifiers
  - Also: specify hop-by-hop transport between pairs of SFC nodes
  - Could be used to construct compute graph between application layer functions

Version 01 Updates (2/3)

- Multi-Access Edge Computing (MEC)
  - Added text on MEC as a platform
  - Mentioned possible combination with network slicing

Version 01 Updates (3/3)

Example:

Compute First Networking: Distributed Computing meets ICN

Michał Król[1], Spyridon Mastorakis[2], Dave Oran[3], Dirk Kutscher[4]

[1]University College London/UCLouvain
[2]University of Nebraska, Omaha
[3]Network Systems Research & Design
[4]University of Applied Sciences Emden/Leer

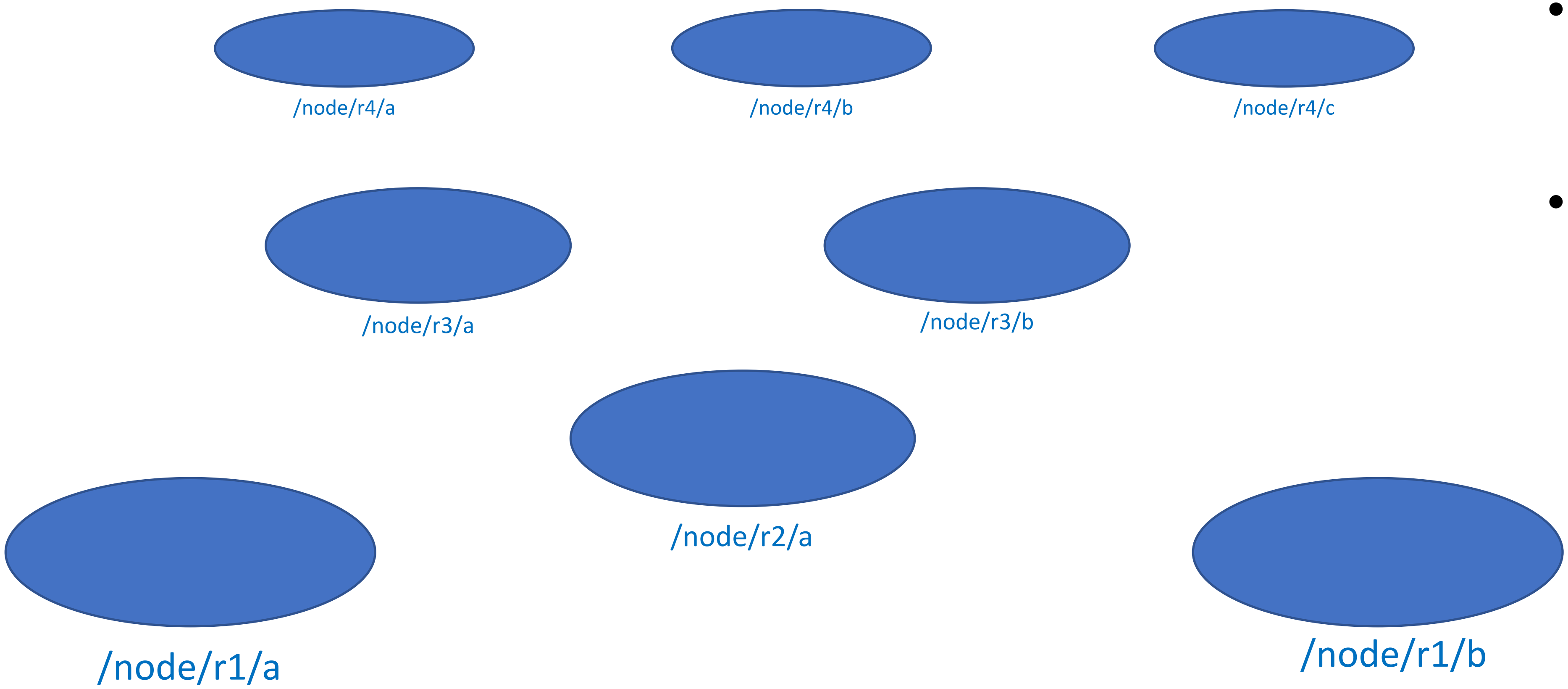
Motivation

- Computing in the Network is about treating computing as a first-class citizen in the system

- Reasoning about networked computation
  - Scalable
  - Secure
  - Reliable (congestion-controlled, fail-safe etc.)
  - Useful for application developers

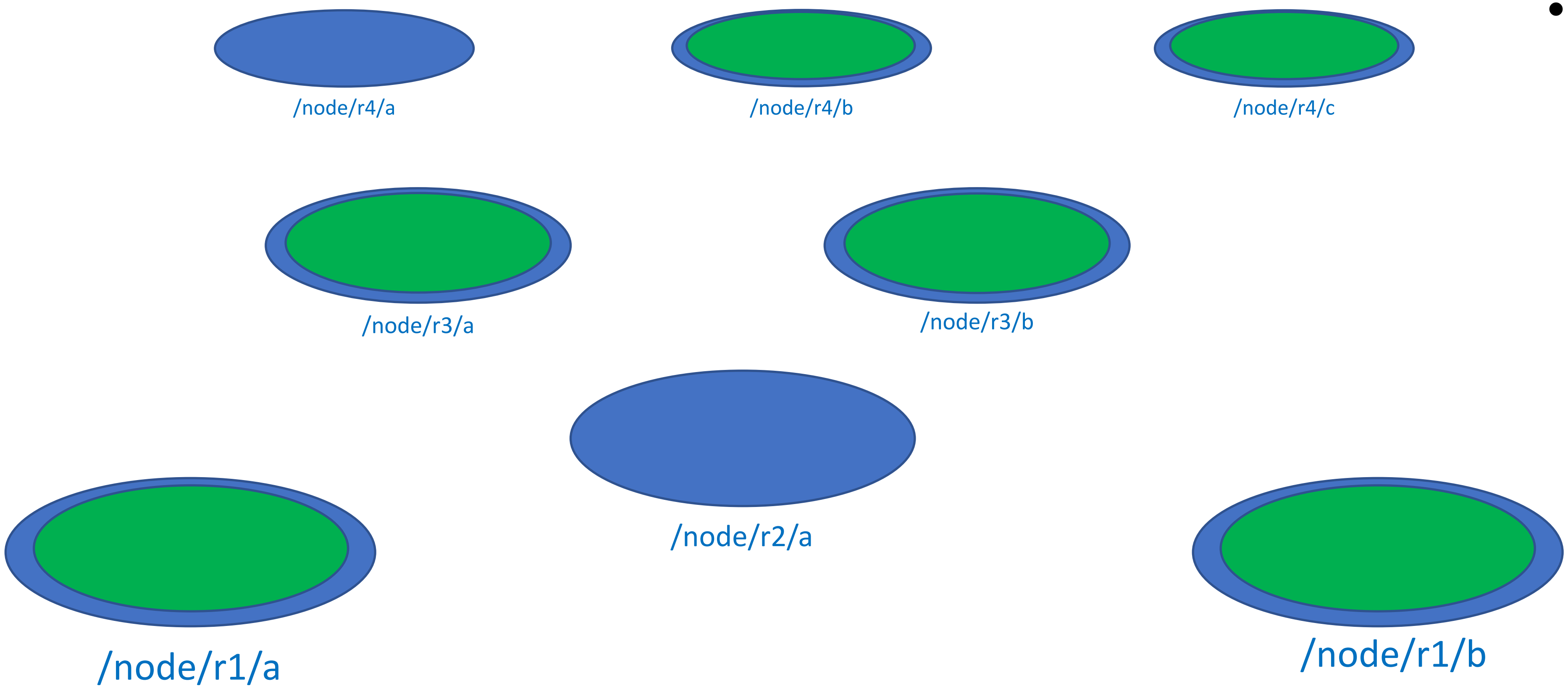
- Not just about controlling packet forwarding
  - Through tunnels, routing updates etc.

Concept



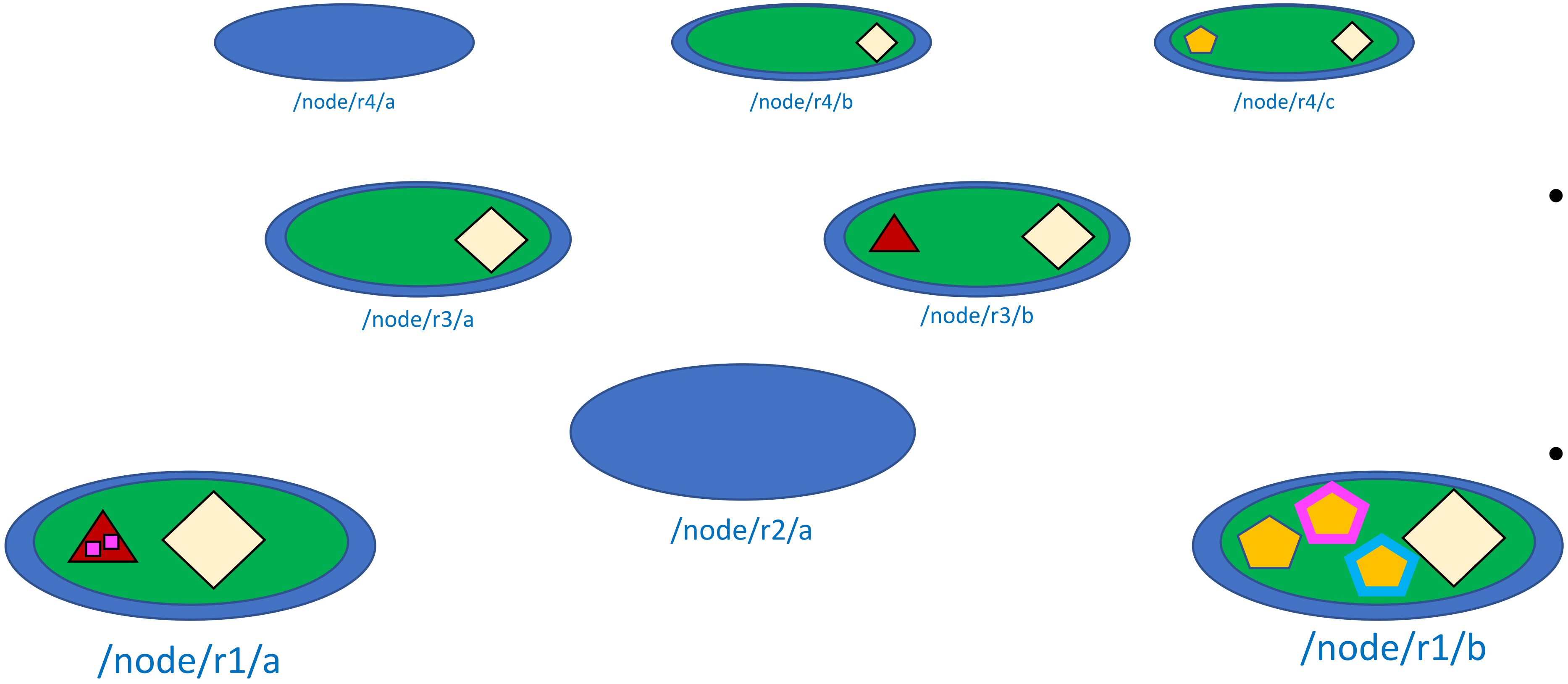
- Nodes in a network offering compute services
- Agnostic to specific execution environment
- But be able to leverage different platforms (GPUs, TEE) and select appropriate ones

Concept



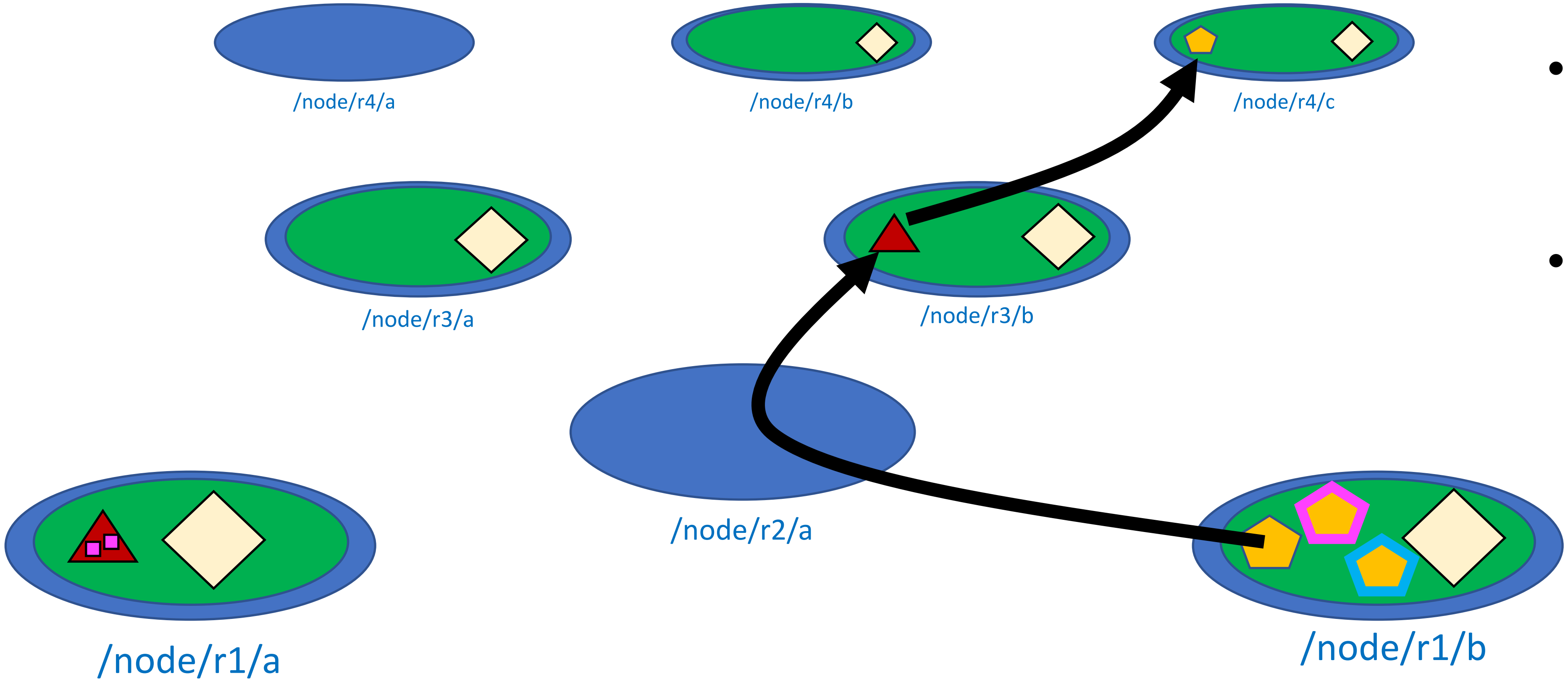
- Nodes could part of a distributed application context
- Nodes could be part of more than one context at a time

Concept



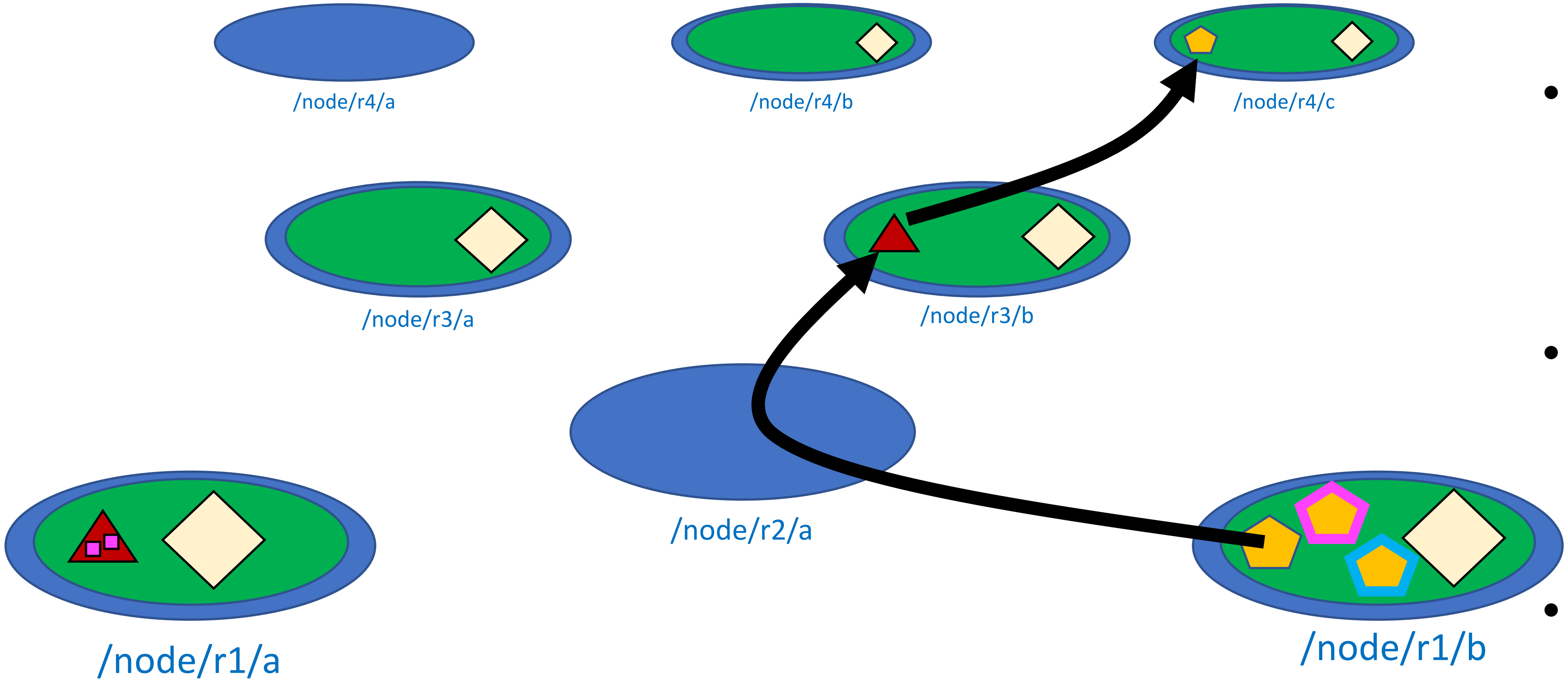
- In a distributed application session, the system can instantiate/invoke functions, actors as required
- 3 types:
  - Stateless functions
  - Stateless actors
  - Data
- Application semantics and resource allocation strategies determine where functions/actors reside

Concept



- RMI protocol for invoking stateless functions and actor member functions
- No assumption on function complexity, execution time
- Function calls can trigger other calls etc.

Concept



Information in the system

- „Where are functions“
- Resource utilization
- Performance

- Also: availability of unallocated resources (nodes)

- Info maintained by distributed data structures

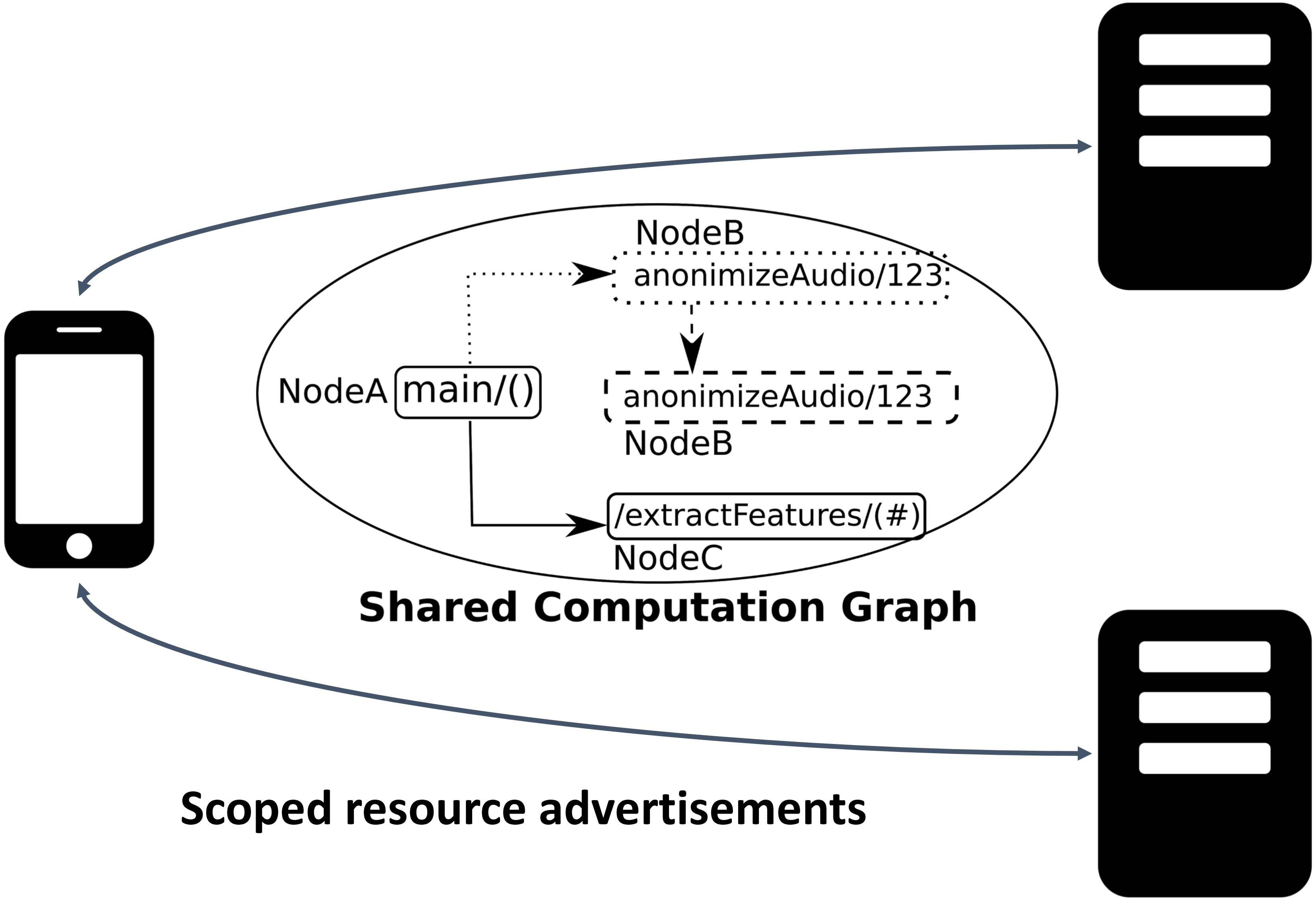
- Concept of using routing system to distribute some of this info

COIN Elements in CFN-ICN

Logical Function	Implementation in Current Design
Resource availability / load information dissemination	CRDTs (distributed data structure)
Transport and RMI Model	RICE (Remote Method Invocation in ICN)
RMI Steering	ICN Forwarding Hints
Programming & Execution Environment	Python (in this PoC)
Compute Classes	Stateless functions, stateful actors, data
Function Naming	ICN naming

Overview

Task Scheduler



Shared Computation Graph

Scoped resource advertisements

Terminology

- **Program** - a set of computations requested by a user.
- **Program Instance** - one currently executing instance of a program
- **Function** - a specific computation that can be invoked as part of a program.
- **Data** - represents function outputs and inputs or actor internal state.
- **Future** - objects representing the results of a computation that may not yet be computed.
- **Worker** - the execution locus of a function or actor of a program instance

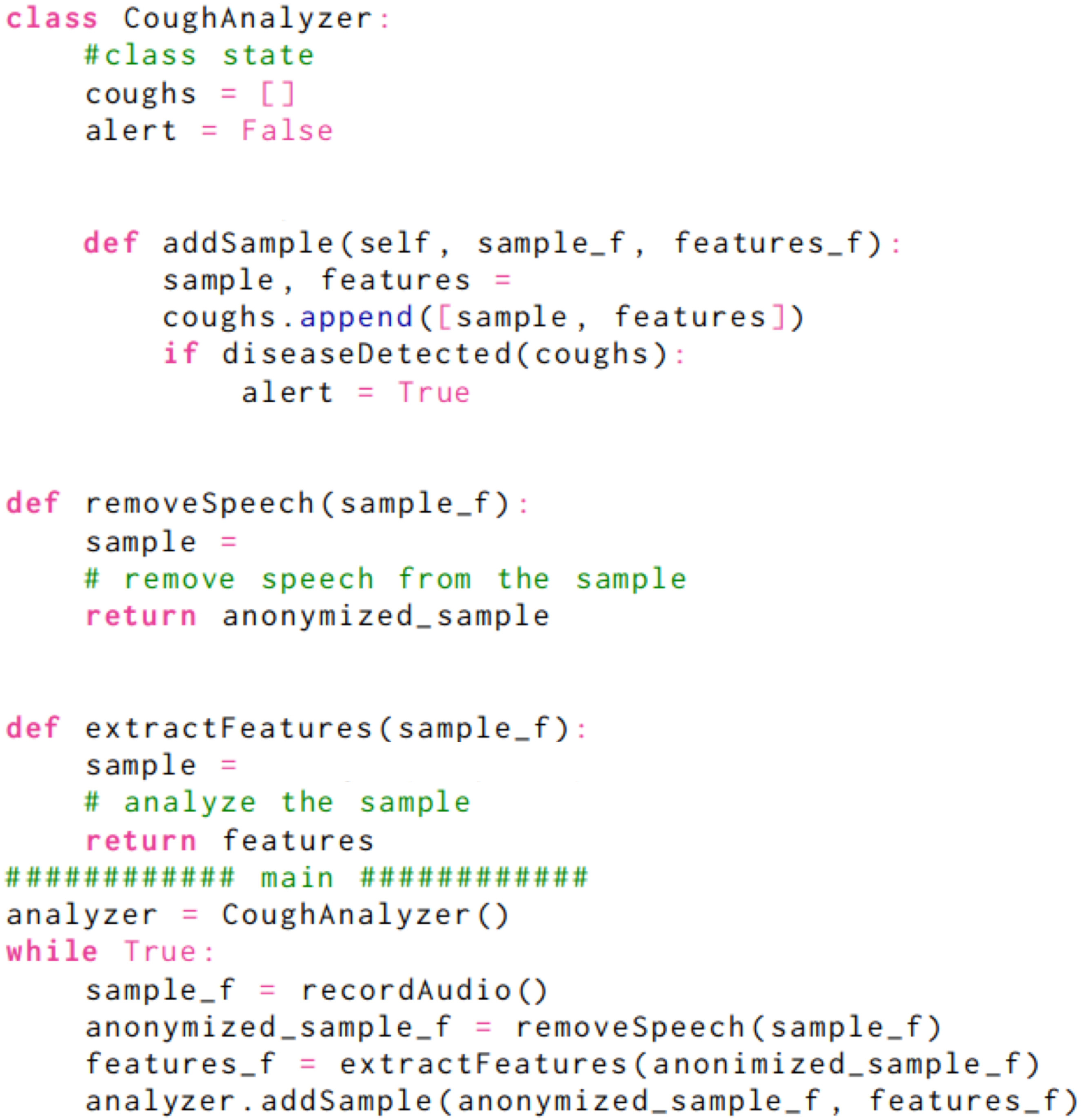
Code

Decorators:

- @cfn.transparent
- @cfn.opaque
- @cfn.actor

Methods:

- cfn.get(future)



```
class CoughAnalyzer:
    #class state
    coughs = []
    alert = False


    def addSample(self, sample_f, features_f):
        sample, features =
        coughs.append([sample, features])
        if diseaseDetected(coughs):
            alert = True


def removeSpeech(sample_f):
    sample =
    # remove speech from the sample
    return anonymized_sample


def extractFeatures(sample_f):
    sample =
    # analyze the sample
    return features
############ main ############
analyzer = CoughAnalyzer()
while True:
    sample_f = recordAudio()
    anonymized_sample_f = removeSpeech(sample_f)
    features_f = extractFeatures(anonimized_sample_f)
    analyzer.addSample(anonymized_sample_f, features_f)
```

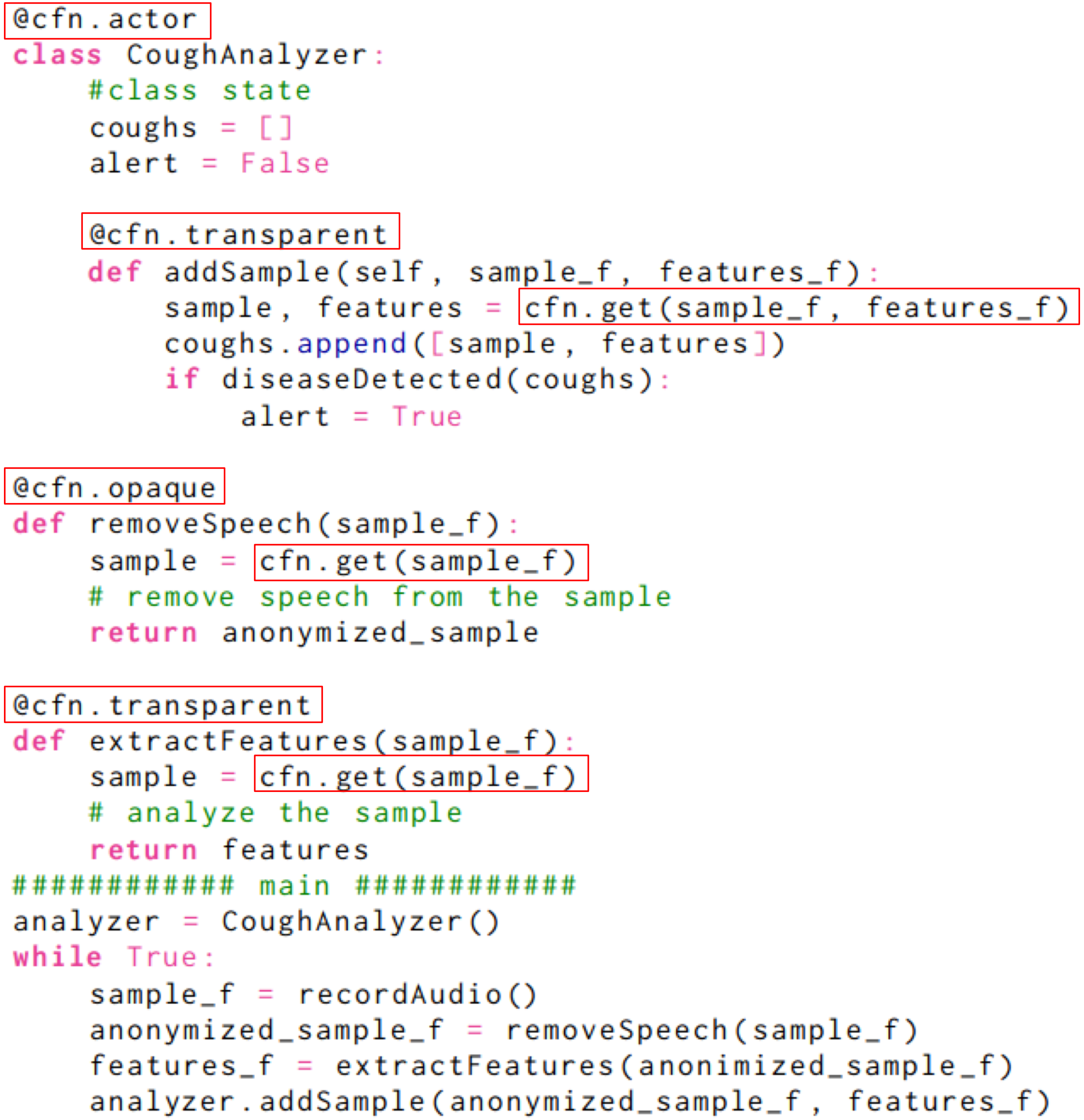
Code

Decorators:

- @cfn.transparent
- @cfn.opaque
- @cfn.actor

Methods:

- cfn.get(future)

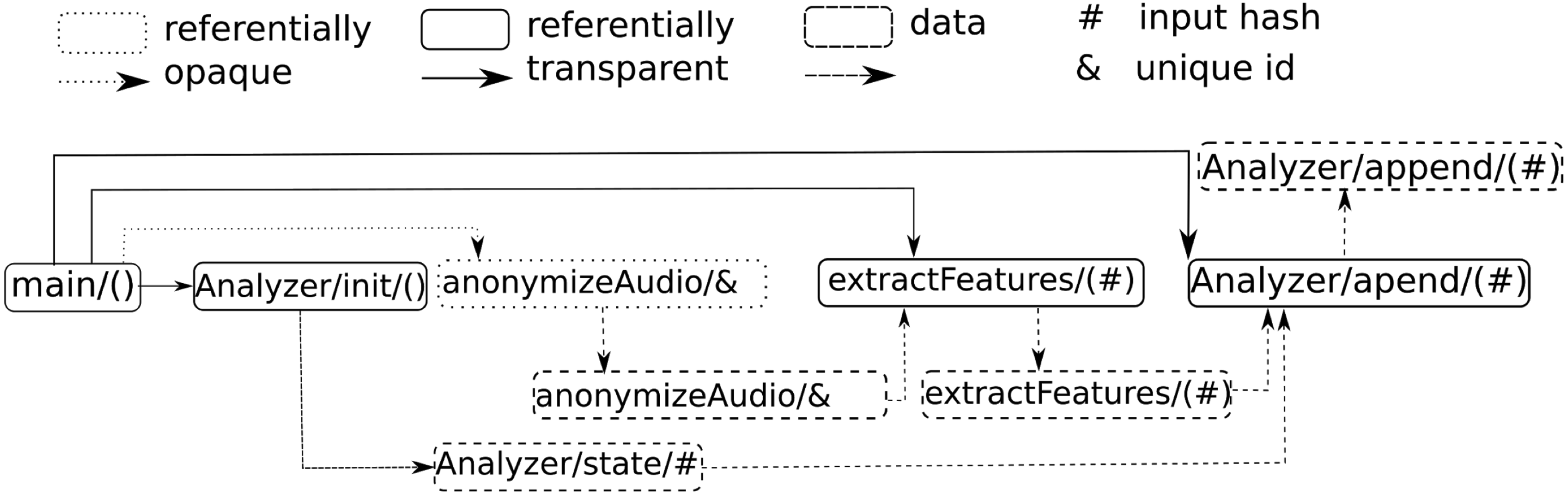


```
@cfn.actor
class CoughAnalyzer:
    #class state
    coughs = []
    alert = False

    @cfn.transparent
    def addSample(self, sample_f, features_f):
        sample, features = cfn.get(sample_f, features_f)
        coughs.append([sample, features])
        if diseaseDetected(coughs):
            alert = True

@cfn.opaque
def removeSpeech(sample_f):
    sample = cfn.get(sample_f)
    # remove speech from the sample
    return anonymized_sample

@cfn.transparent
def extractFeatures(sample_f):
    sample = cfn.get(sample_f)
    # analyze the sample
    return features
############ main ############
analyzer = CoughAnalyzer()
while True:
    sample_f = recordAudio()
    anonymized_sample_f = removeSpeech(sample_f)
    features_f = extractFeatures(anonimized_sample_f)
    analyzer.addSample(anonymized_sample_f, features_f)
```

Computation Graph

- Location of the data
- Chaining nodes using ICN names
- Different node types

- Graph is a CRDT
- Non-conflicting merge operations (set addition)



Computation Graph

In	**Name:** /extractFeatures/(#)	**Out**
/removeSpeech/(#)	**Type:** Referentially Transparent Function	/extractFeatures/(#)/r1
	Location: node1	/extractFeatures/(#)/r2
		/extractFeatures/(#)/r3

Computation Graph

In	**Name:** /extractFeatures/(#)	**Out**
/removeSpeech/(#)	**Type:** Referentially Transparent Function	/extractFeatures/(#)/r1
	Location: node1	/extractFeatures/(#)/r2
		/extractFeatures/(#)/r3

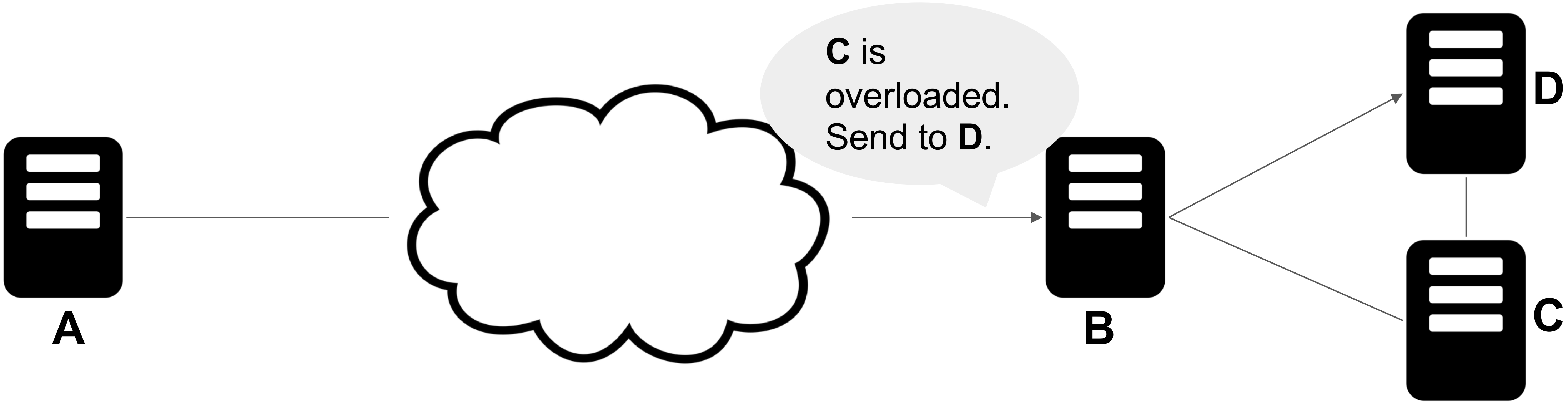
In	**Name:** /extractFeatures/(#)	**Out**
/removeSpeech/(#)	**Type:** Referentially Transparent Function	/extractFeatures/(#)/r1
	Location: node2	/extractFeatures/(#)/r2
		/extractFeatures/(#)/r3

Computation Graph

In	**Name:** /extractFeatures/(#)	**Out**
/removeSpeech/(#)	**Type:** Referentially Transparent Function	/extractFeatures/(#)/r1
	Location: node1, node2	/extractFeatures/(#)/r2
		/extractFeatures/(#)/r3

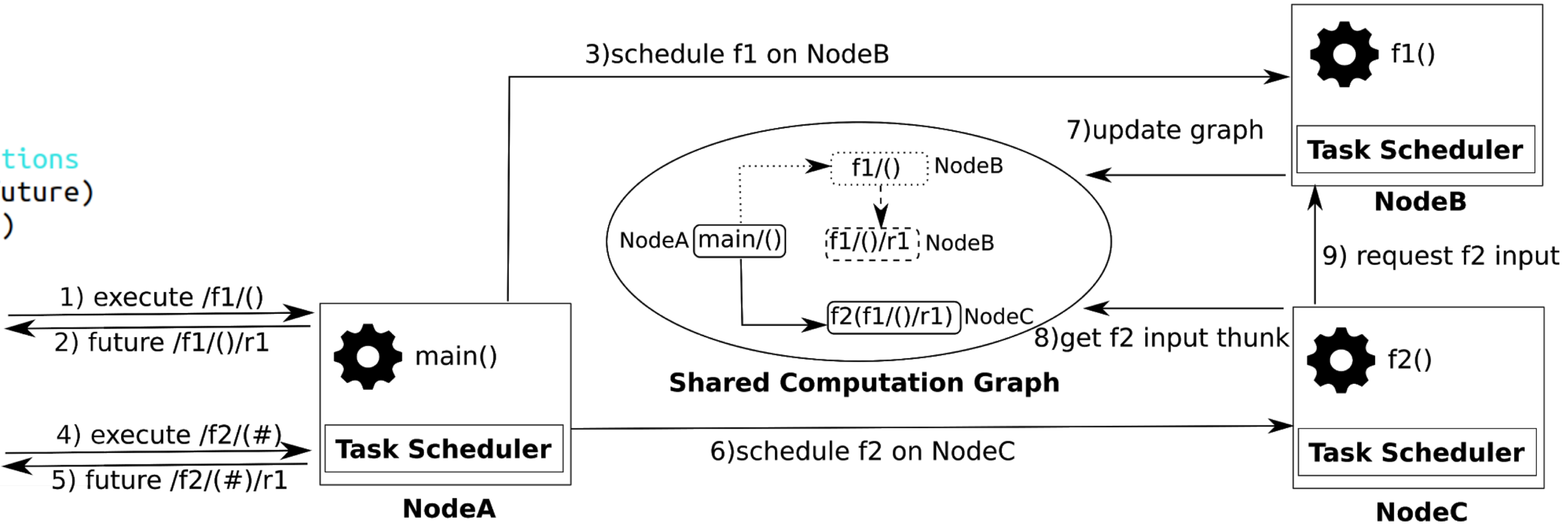
Task Scheduler

- Functions are invoked close to the data they rely on
- Forwarding hints to steer traffic
- Dependency information + data info are in the computation graph
- Each decision can be optimized by other forwarding nodes (late binding)
- The exact node is chosen using information from scoped resource advertisements



Example

```
@opaque
def f1():
    return random()

@transparent
def f2(future):
    #perform computations
    my_input = get(future)
    compute(my_input)

def main():
    f1_future = f1()


    f2(f1_future)
```



CFN-ICN Summary

- Distributed computation framework for general purpose computation
- Uses Computation Graph, Resource advertisement protocol and a scheduler
- Includes Transport and RMI functionality (RICE)
- Demonstrates feasibility of distributed approach
- Join optimization of network and computation resources
- Check paper for details (ACM ICN-2019)
- Code available at https://github.com/spirosmastorakis/CFN

Outlook

- Want to enable more decentralized decision-making in the network
- Consider dynamic network & platform load
- Think about QoS for computing and specific worker capabilities
- Soft-state approach: reduced coordination and state-keeping
- ICN to the rescue: late-binding, path steering

Suggestions

- Computing in the Network: More than just forwarding packets to nodes that happen host VMs or processes
    - Can be done today with various tools
- Embrace the idea of supporting distributed computing by leveraging networking concepts and mechanisms
    - Instead of building better pipes between processes

Next Steps for Draft

- Document more representative use cases
- Mention segment routing as another packet steering technology
- Some form of taxonomy to aid discussion in COINRG
- Overall goal: help us understand problem – not so much prescribing solutions