Directions for COIN

draft-kutscher-coinrg-dir

Dirk Kutscher, Jörg Ott, Teemu Kärkkäinen

25 July 2019 – COIN RG-to-be

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?

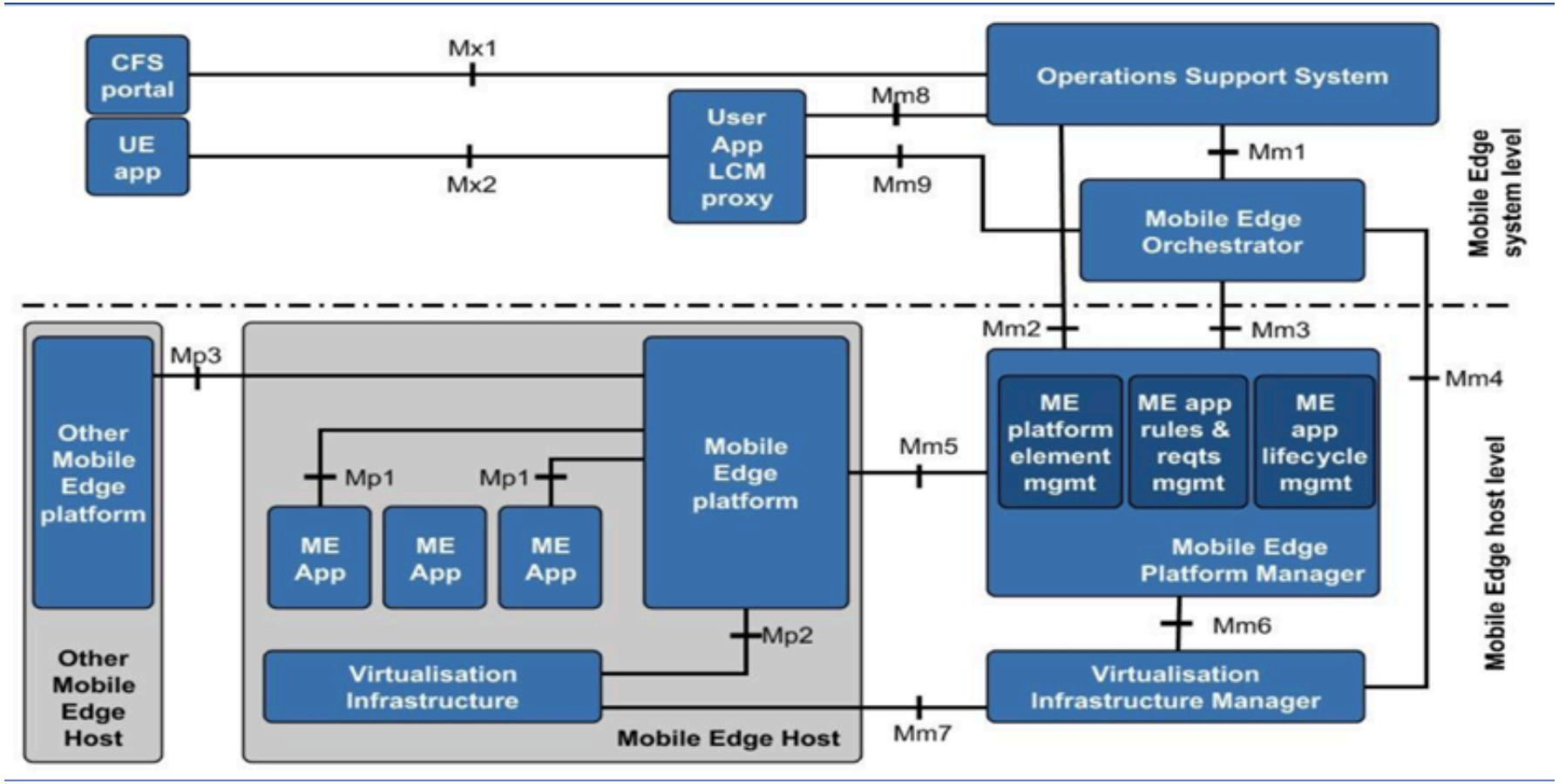
What does “in-network” really mean?

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

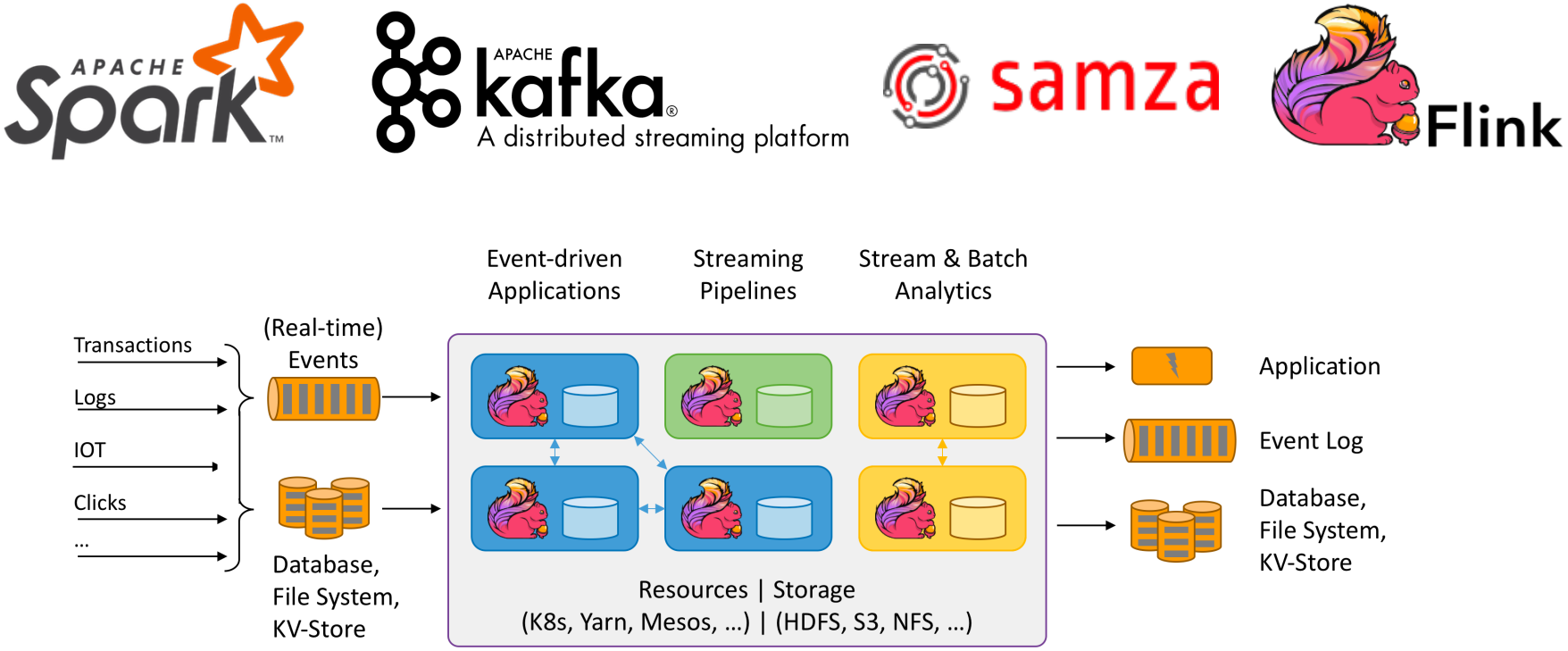
- These approaches are applied (more or less) successfully today and do not need COIN research…
  - ...but there is lots of engineering to be done in the IETF

Example: Mobile Edge Computing



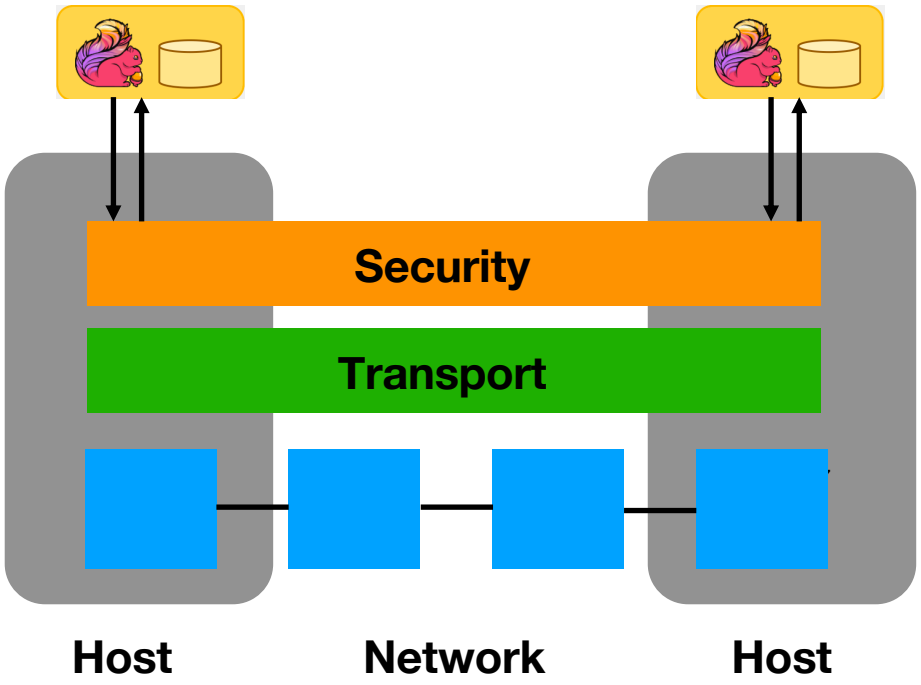
https://datatracker.ietf.org/meeting/98/materials/slides-98-nfvrg-sessb-12-multi-access-edge-computing-mec-applications-00

Example: Streaming Frameworks

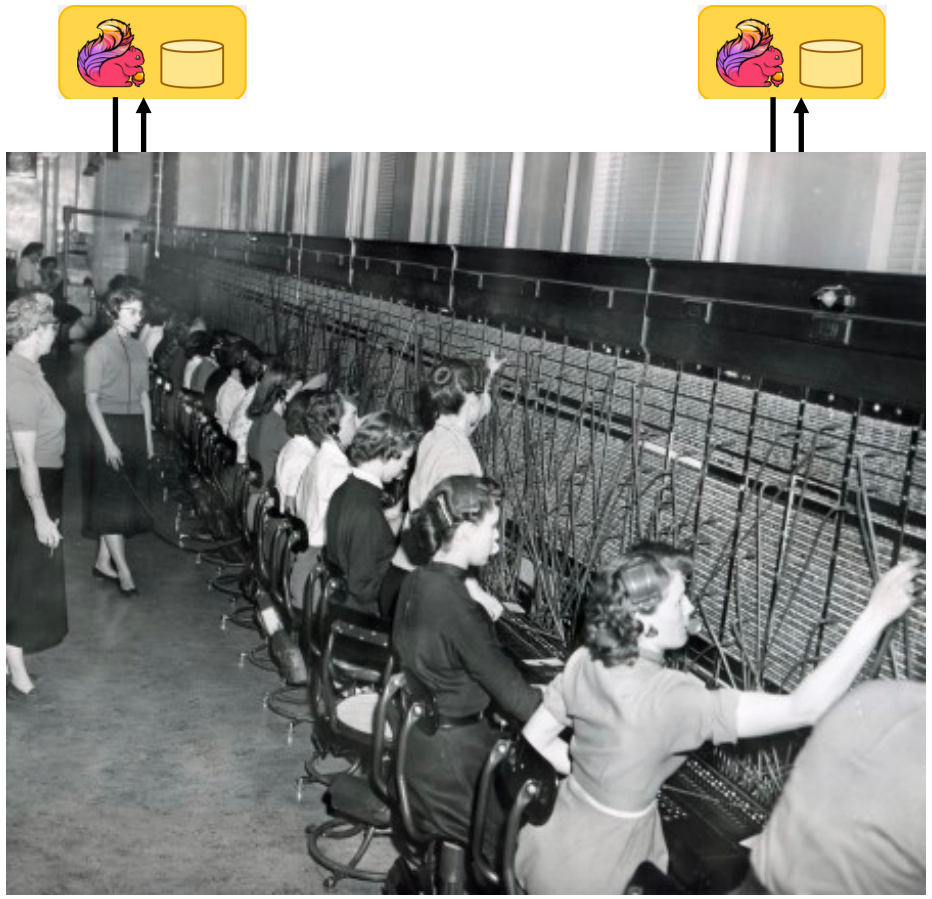


- Elaborate services and guarantees for different use cases
- Apache Flink: Different streaming connectors — but typically as network overlays

Decoupling Computing from the Network



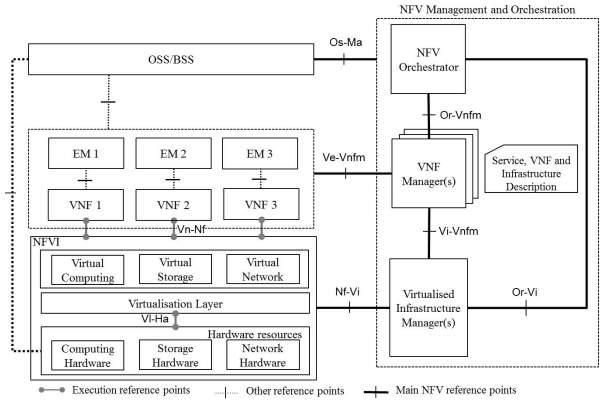
Decoupling Computing from the Network



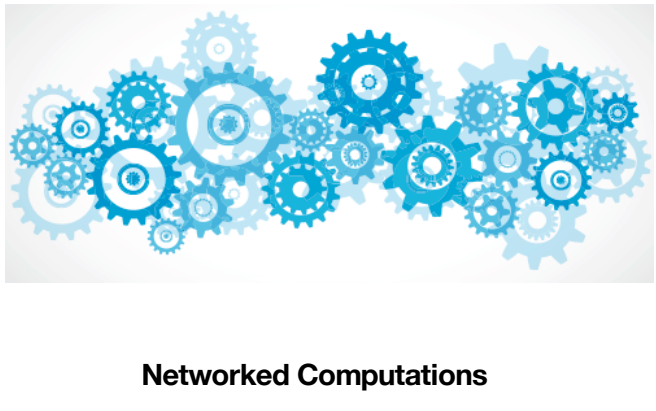
- Circuit-like connectivity
  - Limited visibility into network
- Different namespaces
  - DNS, discovery
- Trust often centralized
  - PKIs for TLS certificates etc.

Joint Optimization of Computing and Networking

- Holistic resource management
  - Network capacity
  - Compute resources
  - Storage
- Multi-dimensional requirements/preferences sets
  - App developer
  - User
  - Network operator



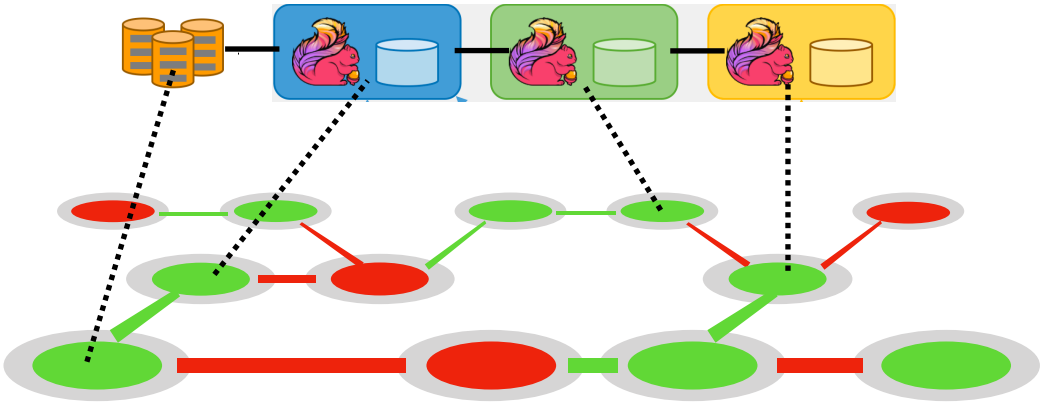
(Virtualized) Compute Servers in Networks



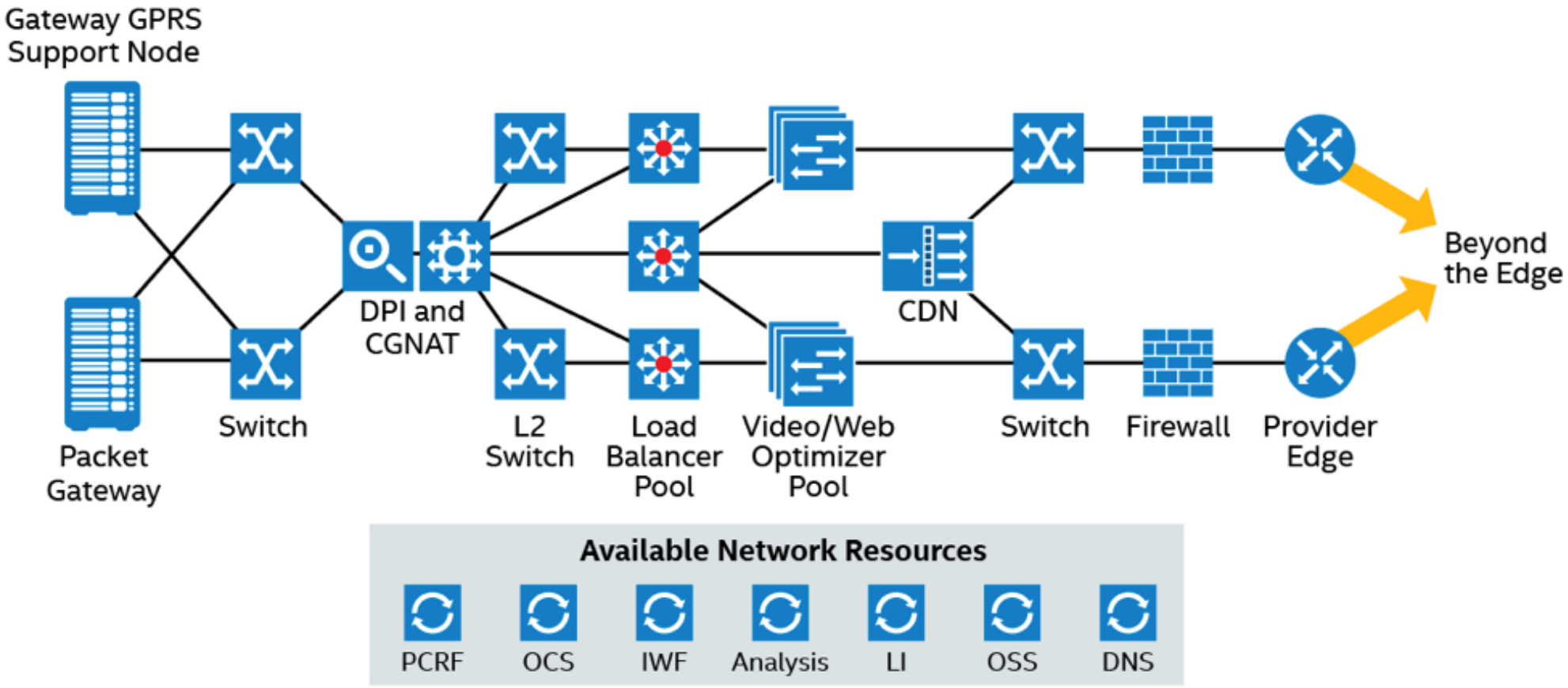
Networked Computations

Joint Optimization of Computing and Networking

- 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)
  - At others 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**



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/

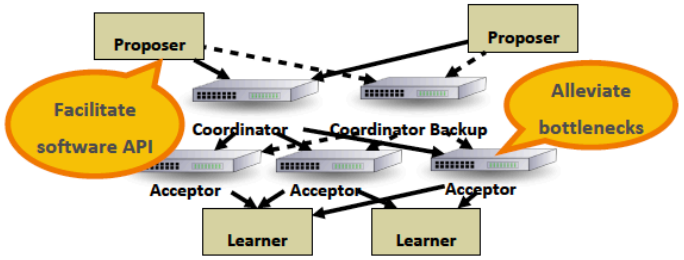
Service Function Chaining

- Flow/Packet-based abstraction
  - Think DPI, Firewalls
- Assumes function is within security perimiter and can access all packets in a flow
- Intended for operator „Gi-LAN“ scenarios – not a platform for distributed computing
- General remarks also apply to name-based service chaining and segment routing

Programmable Data Plane

Example: NetPaxos

Consensus is a fundamental problem for fault-tolerant systems



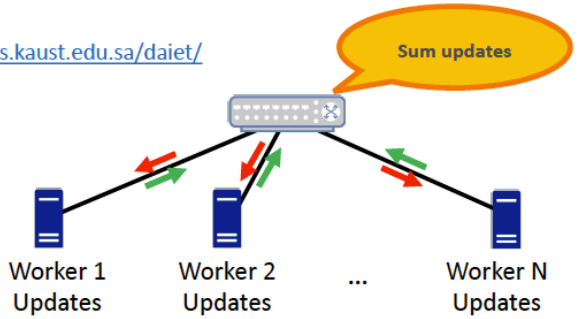
- Offering consensus as a network service has significant performance benefits
- Implement Paxos logic in network devices
- Demonstrate consensus at 9 M msgs / s (4.3x improvement) and low latency (80% reduction)

µs	P4FPGA	SDNet	Netronome
Forwarding	0.37	0.73	-
Coordinator	0.72	1.21	0.33±0.01
Acceptor	0.79	1.44	0.81±0.01

Example: DAIET

https://sands.kaust.edu.sa/daiet/

Data aggregation is a common task in many DC apps; high potential for ML



- Offload aggregation task to switches to alleviate communication bottlenecks and improve overall training time
- Exploit full network bandwidth of workers

Aggregation micro-benchmark:
- 1.28GB, 320M-element tensor
- Tofino switch
- 2 to 8 workers at 10Gbps

Results:
Transfer time 1.9 s (1.56s optimal limit)

Marco Canini: In-Network Computation is a Dumb Idea Whose Time Has Come

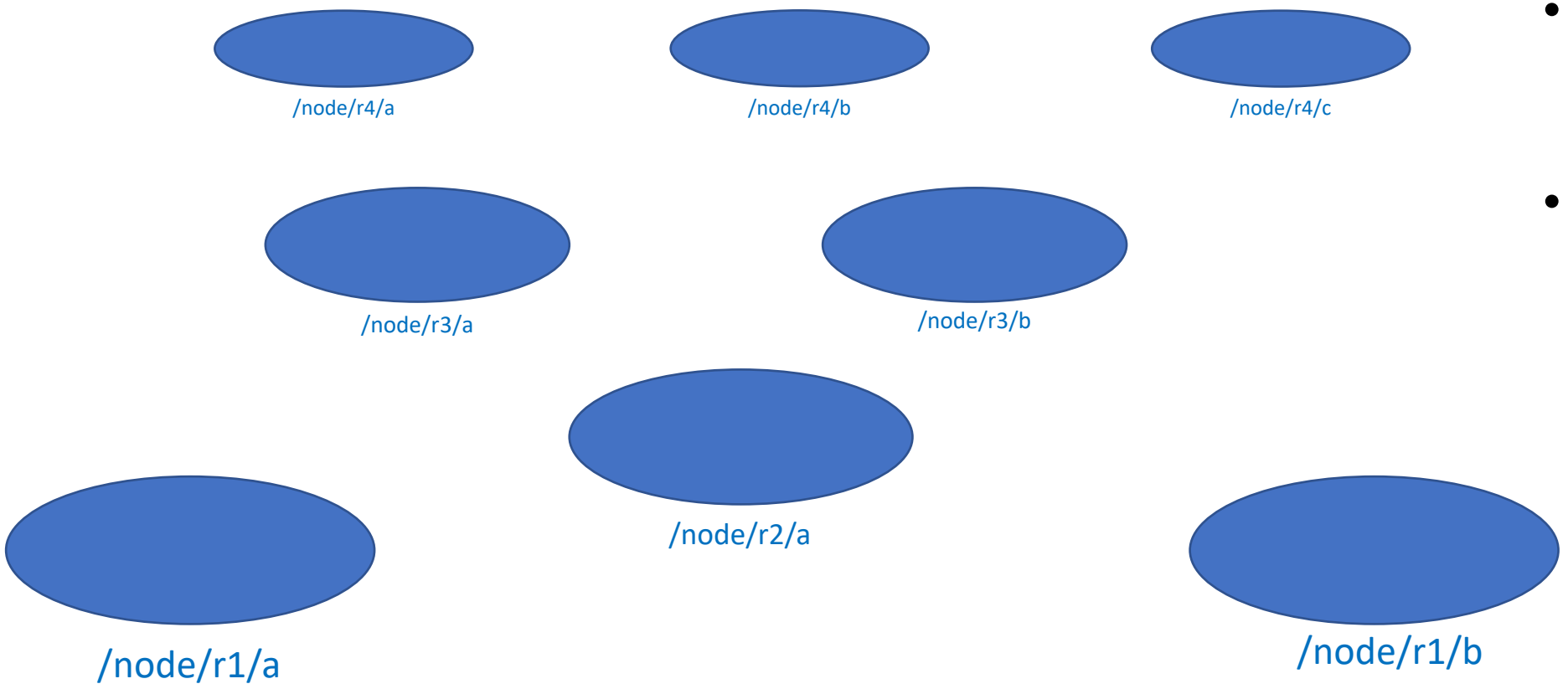
Programmable Data Plane

- Offloading certain tasks in a distributed computing system to programmable switches
  - Programming application-specific switch behavior (for example, with P4)
- Good example that highlights the potential of a particular execution environment
- Effectively similar assumptions as service chaining approach
  - Operate on packets
  - Does not need/provide transport/security functionality

Two directions

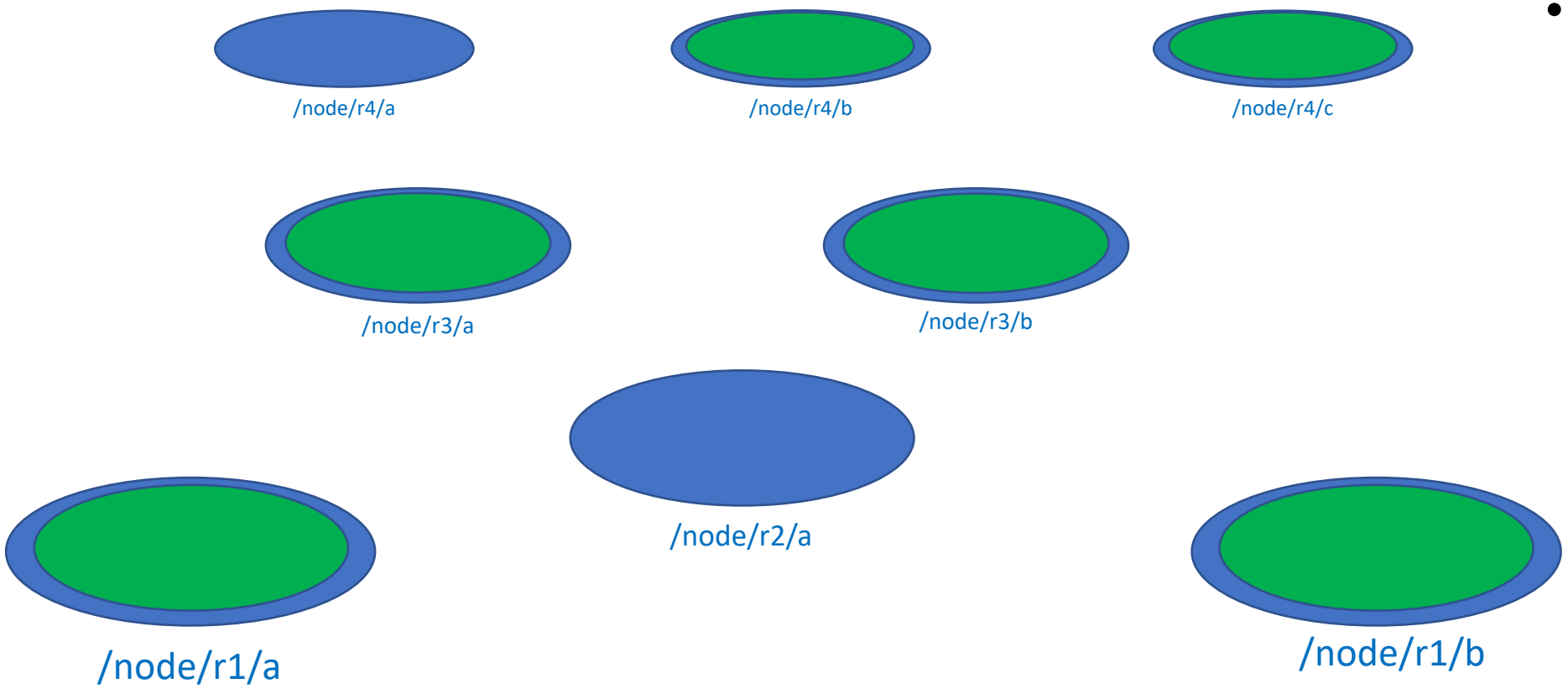
- From application layer overlays to native support
  - Pushing down the stack
- From per-packet matching and decisions to larger application data units
  - Moving up the stack

COIN Example



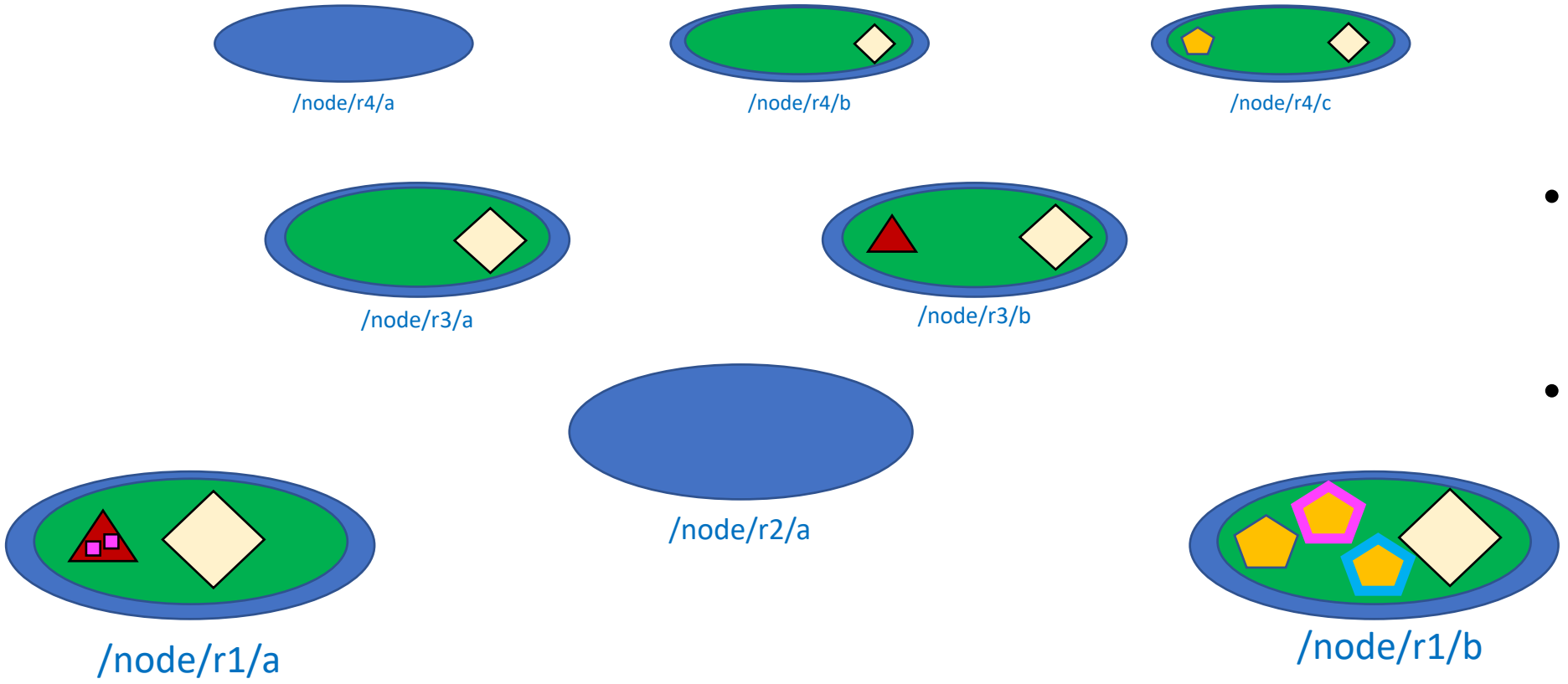
- Nodes in a network offering compute services
- Agnostic to specific execution environment
- But be able to leverage different platforms (GPUs, TEE) and select appropriate ones

COIN Example



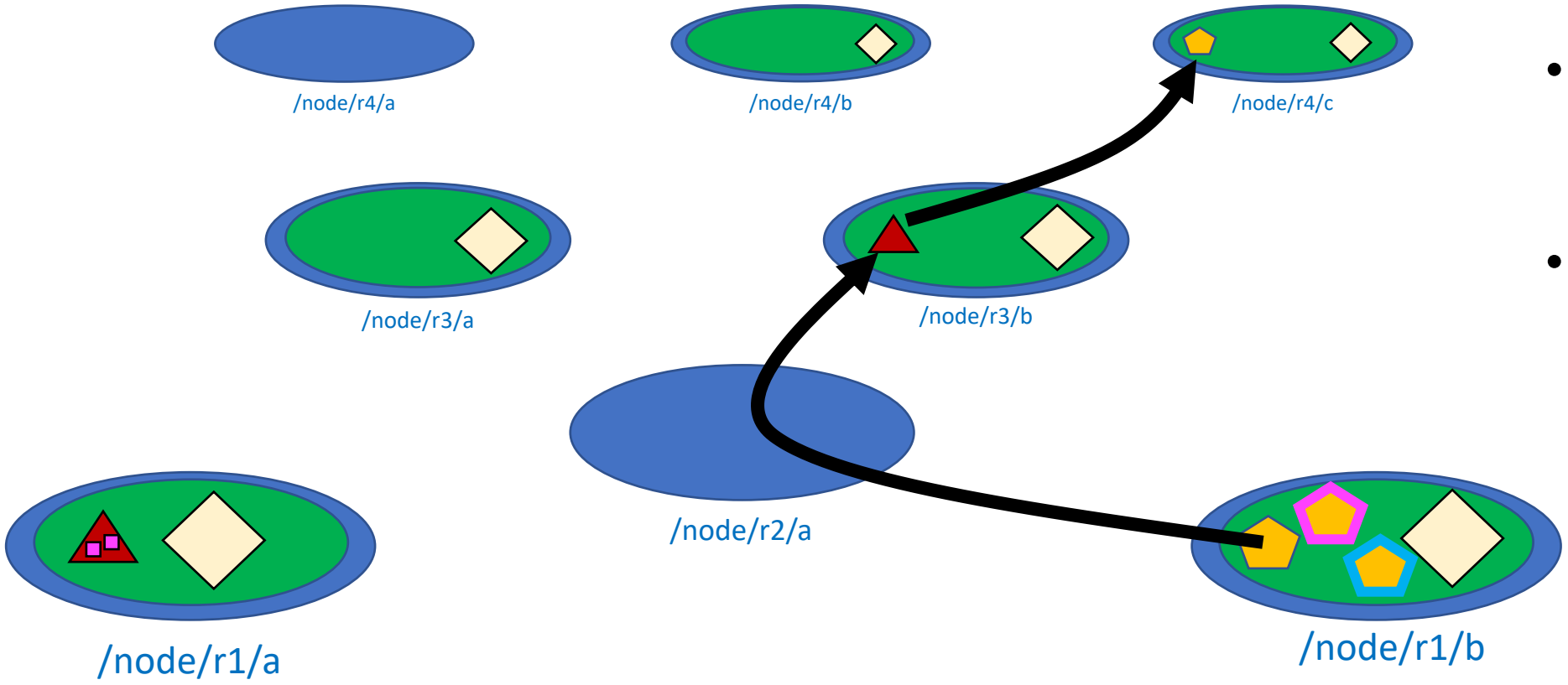
- Nodes could part of a distributed application context
- Nodes could be part of more than one context at a time

COIN Example



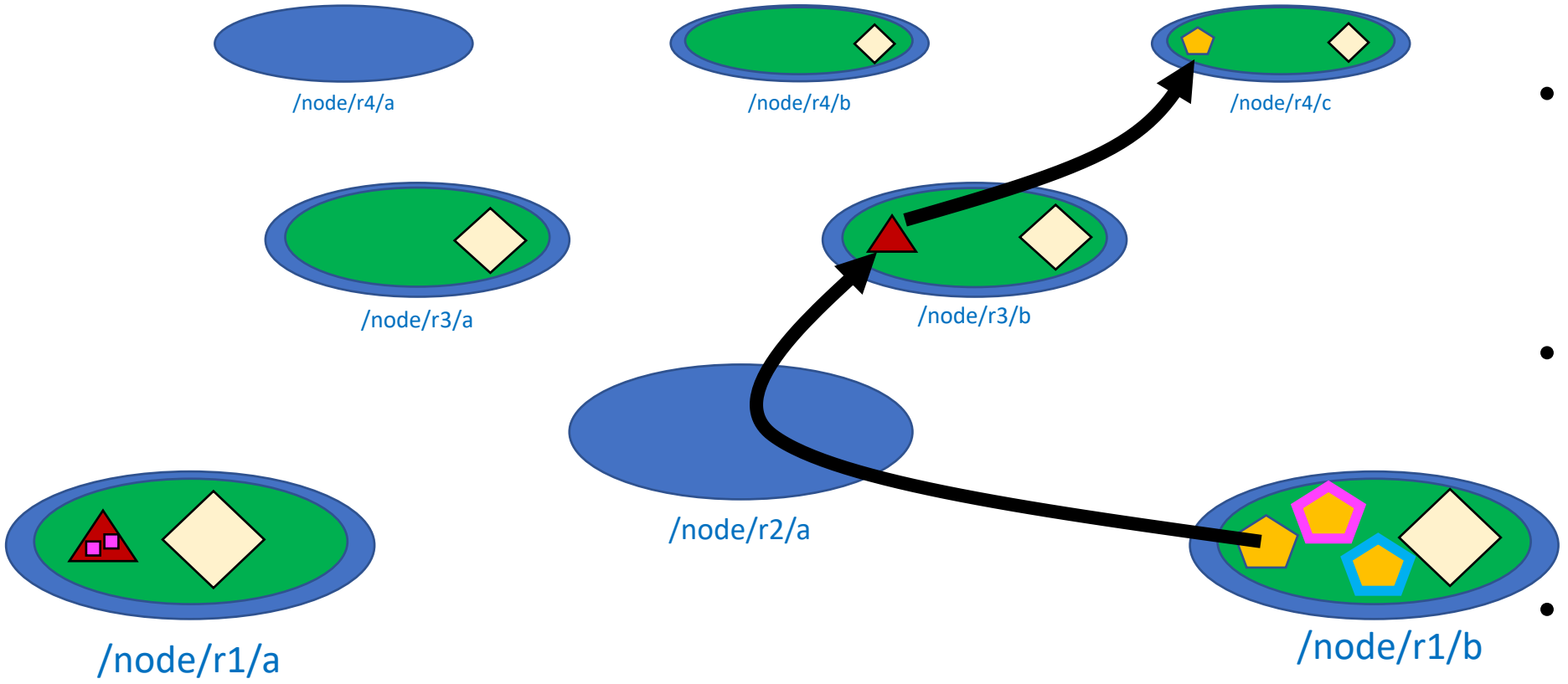
- In a distributed application session, the system can instantiate/invoke functions, actors as required
- 2 types:
  - Stateless functions
  - Stateless actors
- Application semantics and resource allocation strategies determine where functions/actors reside

COIN Example



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

COIN Example



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

Some thoughts on “computing”

Granularity of functions

- ADD $42, %eax
  - Maybe not

… lots of options in-between …

- find_faces_and_identify_people_in_photo ()
  - Possibly

Functions need data – where from?

- Parameters of a function call
  - E.g., an image to process
- Operational context
  - E.g., the trained ML parameters
- Background data
  - E.g., the large key value store or database for lookups
- Function calls need to provide parameters and identify context
  - In-packet / in payload

What’s a sensible data unit?

- Per-packet processing can be a useful tool
- But may not be the ultimate goal

- Need a sensible notion of Application Data Units (ADUs)
- Transport layer (termination)
- Need an idea of how to apply security properties

Side effects of functions?

- Persistent
  - Updates to background data and/or operational context
  - Need to propagate or store
- Temporary
  - Stack when in-network functions call other in-network functions
  - State management
  - Failure handling and garbage collection
  - Where to keep?
    - In the node? In a (growing) ADU?
- In-between
  - Computed results (interim or final)
  - Sharable?

Pull processing vs. push processing

- Pull
  - RPC-style interaction driven by the “calling” application
  - On-demand
- Push (data-driven, triggered)
  - Data flows towards aggregation points (e.g., smart cities)
  - Pre setup
    - Line switching: Node Red-style dedicated setup
    - Packet switching: Rules govern dynamic instantiation
- Distinguish application semantics from network forwarding primitives

Performance considerations

- Passing / fetching values
  - On-demand vs. prefetching
  - Pipelining
  - Handles
- Data reuse
  - Caching
  - Controlled sharing
    - Naming data + semantics (+ scoping)

Life cycle… (of a function or a process)

- Provisioning
- Instantiation
- Running || waiting
- Replicating
- Terminating + garbage collecting

- Extremes
  - Dedicated hardware box for quantum crypto
  - Generic execution platform that fetches bytecode

And COIN…?

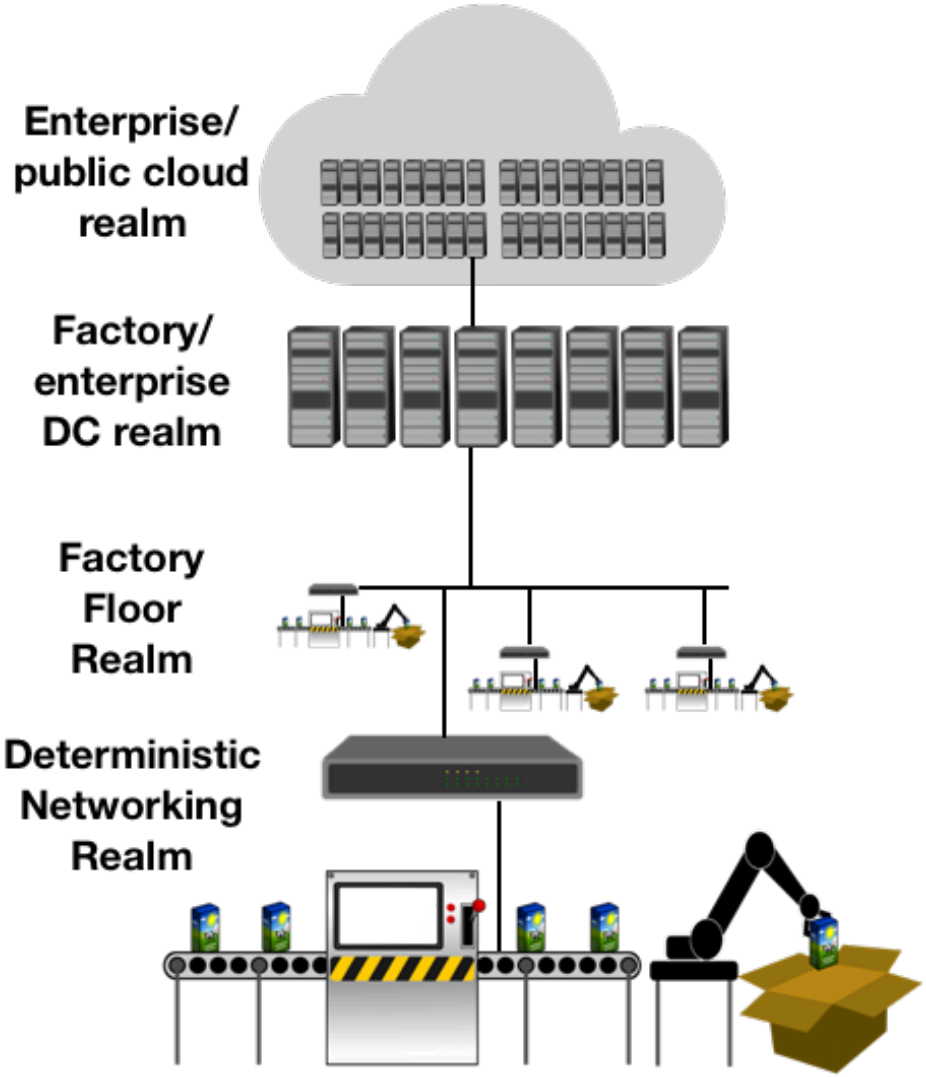
Target environment

- We are the **Internet** Research Task Force

- **Open** networking environments
- Do not make (too m)any assumptions on trustworthiness of peers, code, …
- Assume vast heterogeneity
  - Provisioning
  - Capabilities and performance of compute nodes
  - …

Structuring use **classes**

- Use cases are a good to provide motivation
- But we need to also understand the programming characteristics
- Not by domain but by functional properties



Things to Agree On

- RMI model and protocol
- Types of function and semantics (stateless functions vs. Stateful actors)
- Programming model (not language bindings)
- Resource description and semantics
- Resource allocation mechanisms

Things to Agree On

- RMI model and protocol
- Types of function and semantics (stateless functions vs. Stateful actors)
- Programming model (not language bindings)
- Resource description and semantics
- Resource allocation mechanisms

Implementation Specifics

- Execution environment architectures
- Programs and programming abstractions for platforms
- What else?

Wait, There is More…

- Discovery: resources, functions, results
- Programming models, APIs, bindings
- Versioning
- Resilience against failure, bugs (loops), DOS attacks
- Orchestration
- Management & operations
- Policies
- …