Beyond Micro-Services

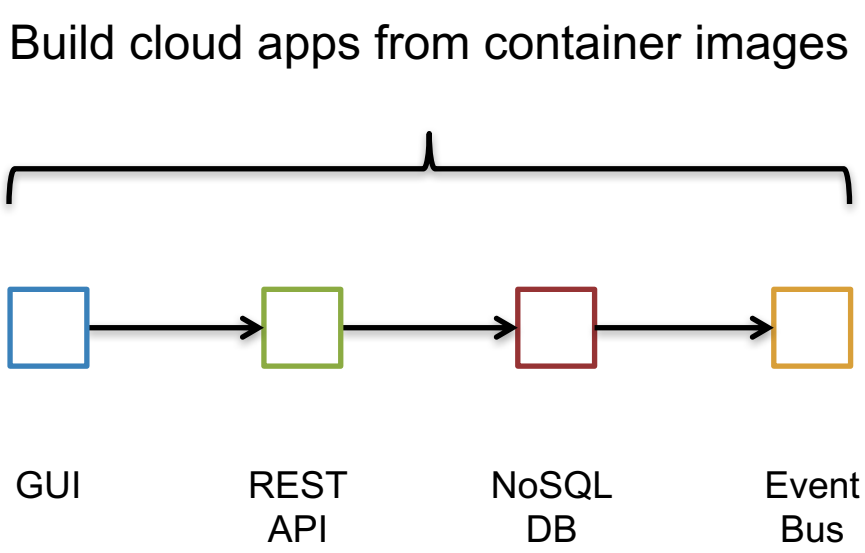
– CORD's Model-Driven Architecture –

Larry Peterson
Open Networking Lab

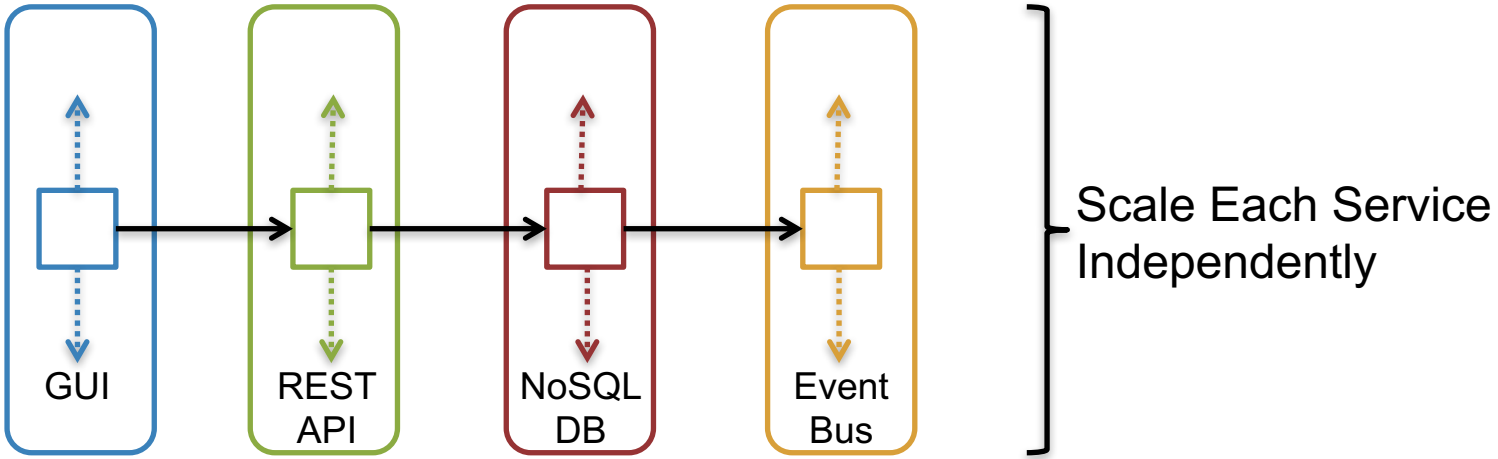
CORD
Central Office Re-architected as a Datacenter

#OpenCORD

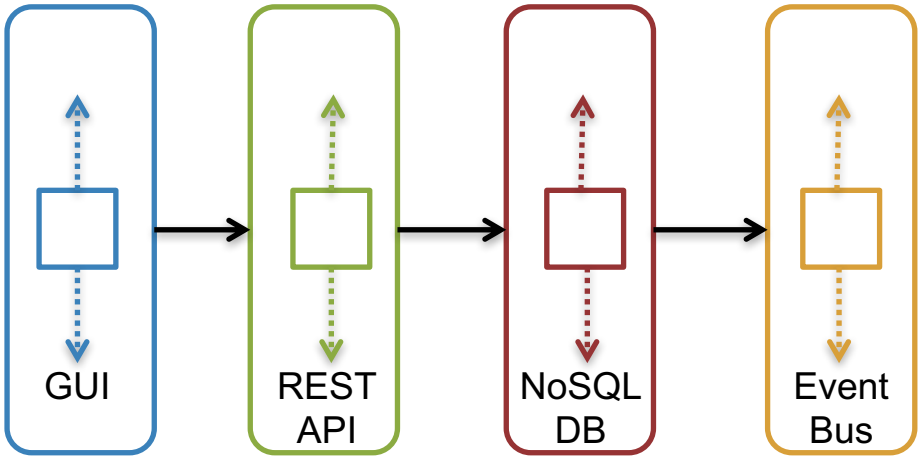
Micro-Services



Micro-Services



Micro-Services

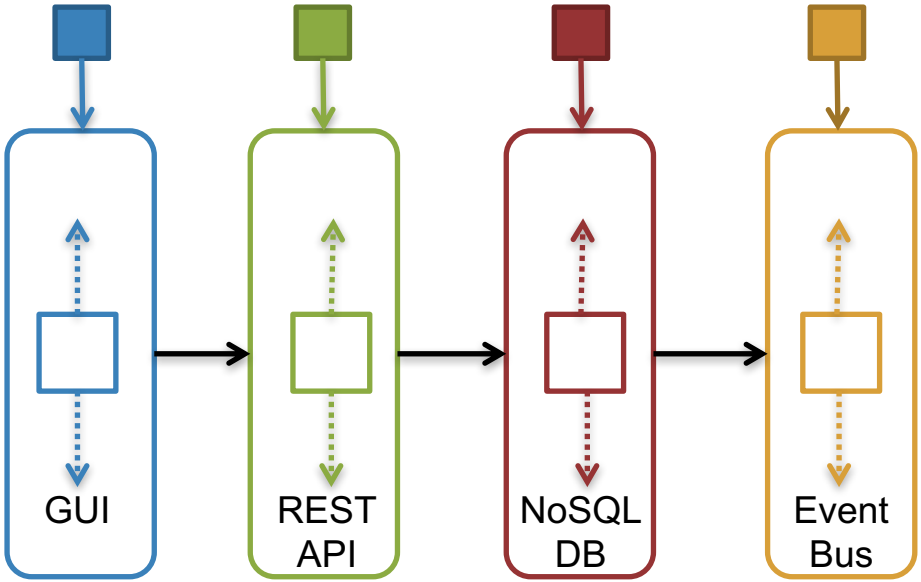


Load Balance Requests Across a Set of Containers

Load Balance Containers Across a Set of Servers

(Kubernetes, Docker Swarm)

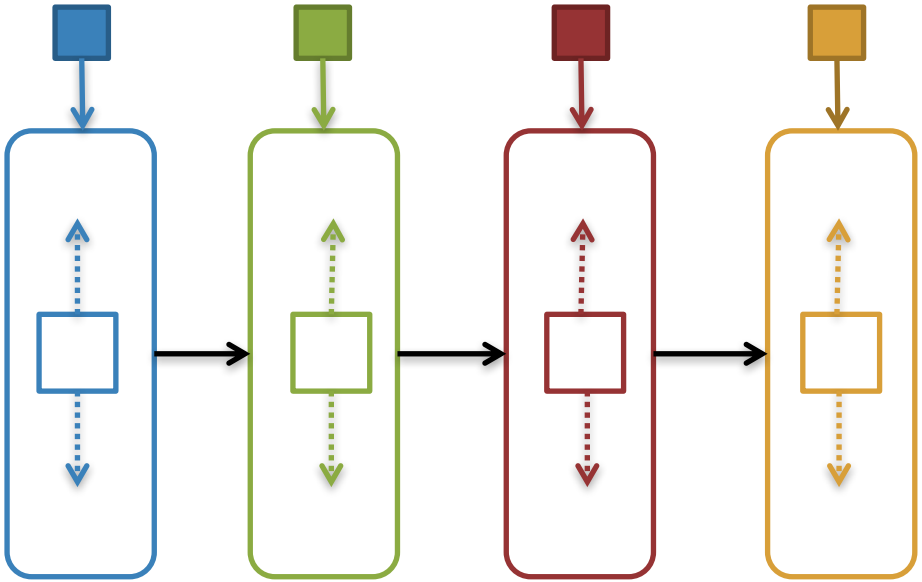
Micro-Services



Configure with a DevOps Playbook

(Ansible)

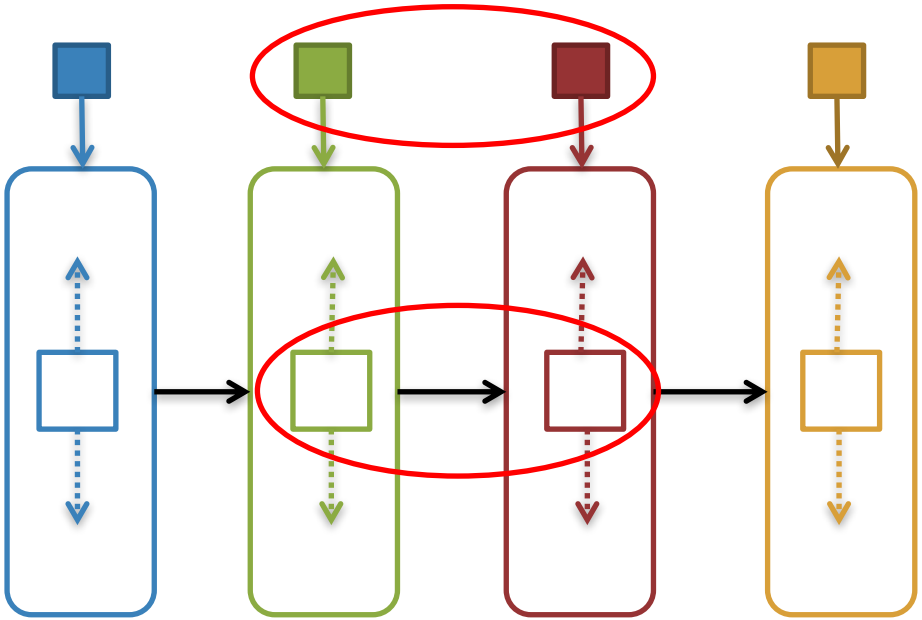
Micro-Services



Assumptions

- Single Application
  - Single Trust Domain
  - Fixed Set of Services
- Fixed Infrastructure
  - Virtualization Technology
  - Network Functionality

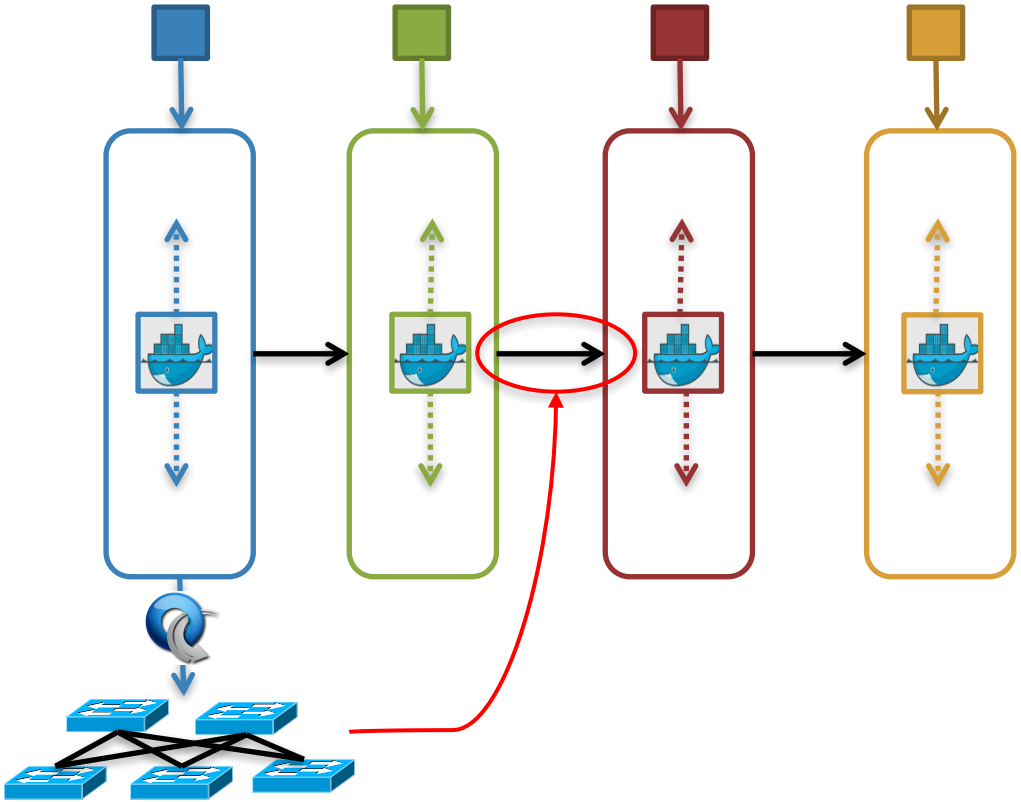
Micro-Services



Assumptions

- Single Application
  - Single Trust Domain
  - Fixed Set of Services
- Fixed Infrastructure
  - Virtualization Technology
  - Network Functionality

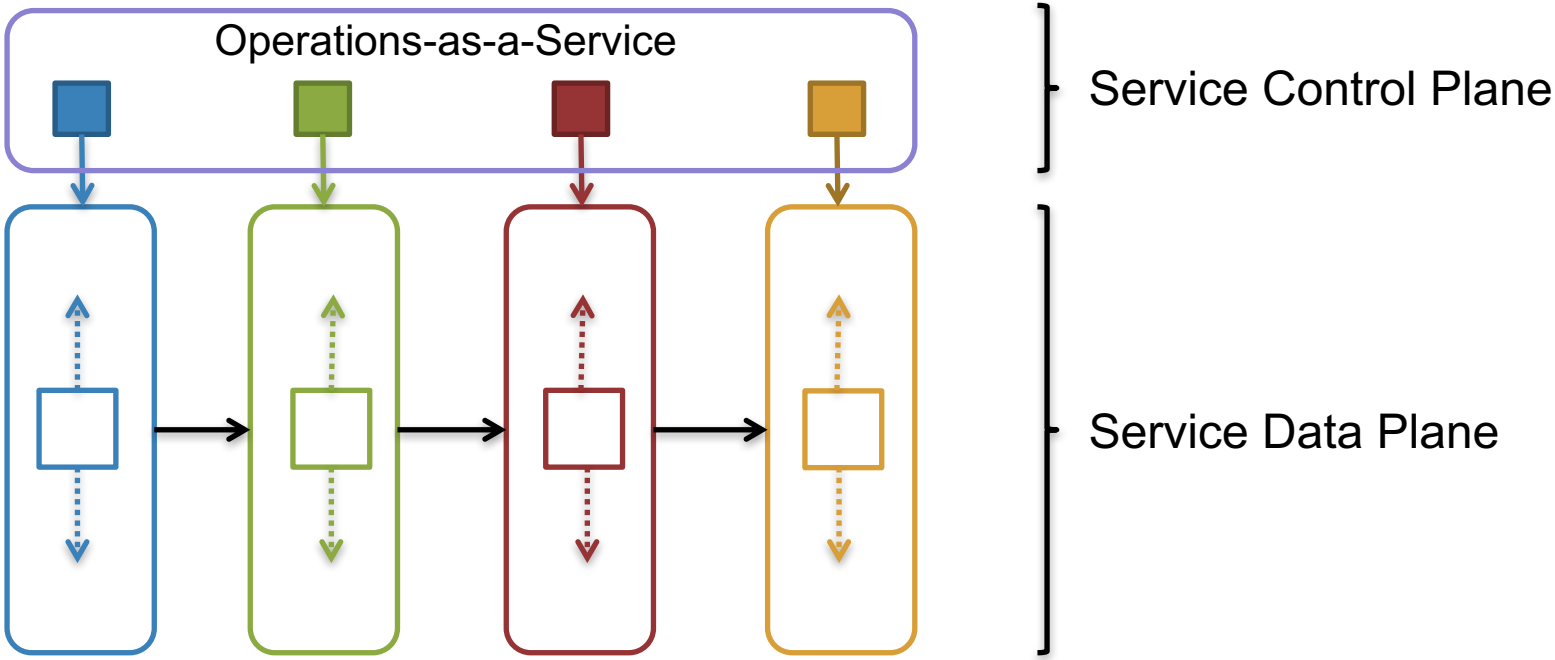
Micro-Services



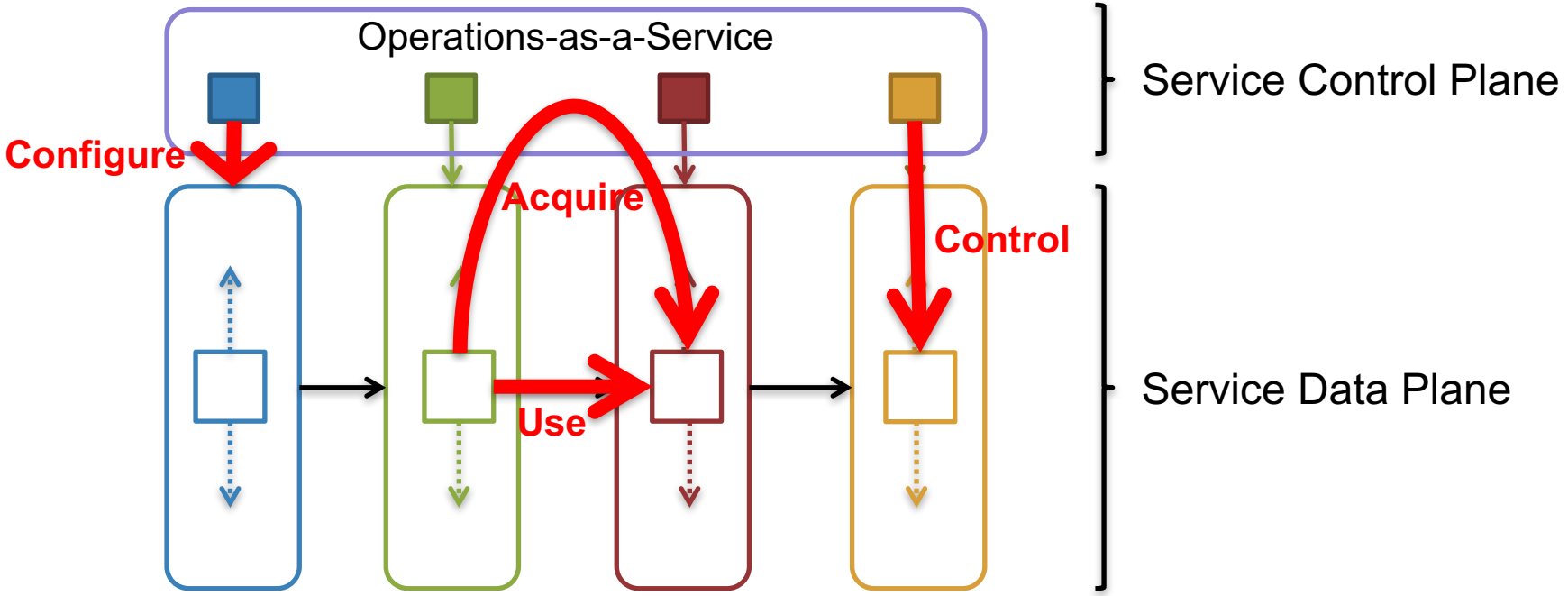
Assumptions

- Single Application
  - Single Trust Domain
  - Fixed Set of Services
- Fixed Infrastructure
  - Virtualization Technology
  - Network Functionality

Beyond Micro-Services



Beyond Micro-Services



Beyond Micro-Services

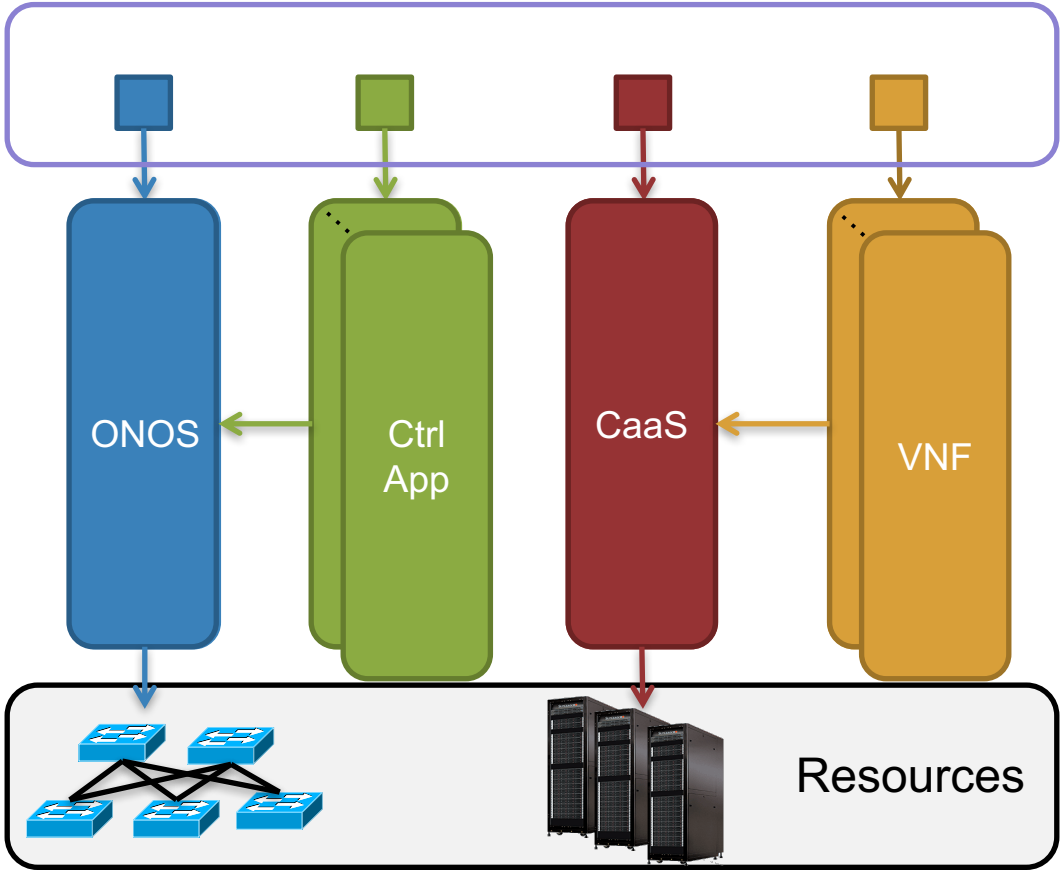
Assumptions

- Single Application
  - Single Trust Domain
  - Fixed Set of Services
- Fixed Infrastructure
  - Virtualization Technology
  - Network Functionality

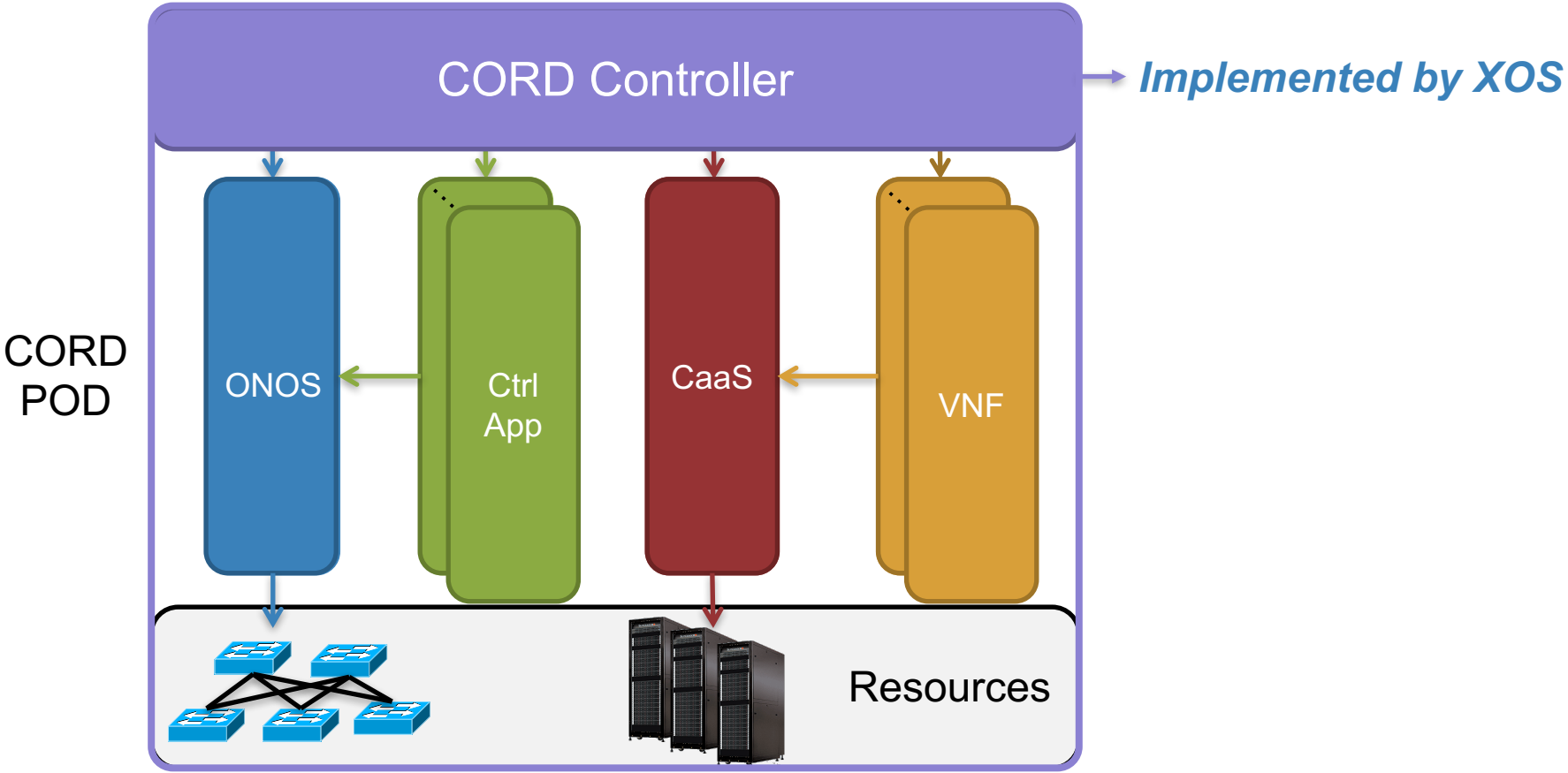
Service Control Plane

- Multi-Tenant Platform
  - Mediate Trust Across Domains
  - Configurable Set of Services
- Programmable Infrastructure
  - Multiple Virtualization Technologies
  - SDN Control Apps as Services

Beyond Micro-Services



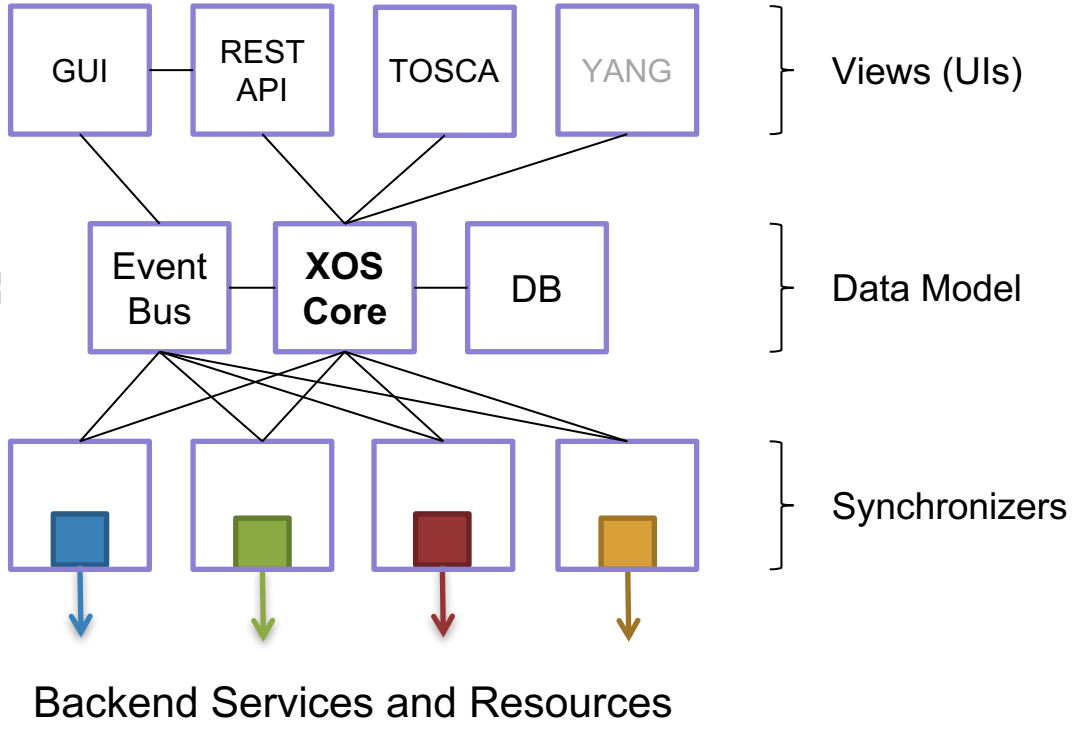
Beyond Micro-Services



XOS Constructed from Micro-Services

CORD Controller (XOS)

=



XOS Core = Data Model Framework

XOS Core =

Models, Policies, Invariants, Constraints

Language to Define & Operate on Models

- *xproto* (protobufs + syntactic sugar)
- define predicates on the models

Generative Tool Chain

- *xosgen*

Auto-generate code to…

implement interfaces, execute synchronizers, enforce security, validate consistency, distribute state

XOS Core

Data Model

- Initializes with a set of “core” models (e.g., Service, Instance,…)
- Extended by on-boarding service-specific models
- Internal representation is based on *xproto* (protobufs + extensions)
- Includes both “state” and “polices”
- Includes both “declarative” and “runtime” state

Interface

- Represented using gRPC/protobufs
- Auto-generated from data model (YANG, TOSCA, REST)
- Includes on-boarding operation (Model + Synchronizer)

Synchronizers

Per-Service Container

- Built from base image that includes *Synchronizer Framework*
- Includes a *Sync_Step( )* that is invoked when model changes
- Typically includes an *Ansible Template*

Synchronizer Framework

- Dependency management
- Error recovery
- Work partitioning
- Parallelization
- Logging

Model Definition

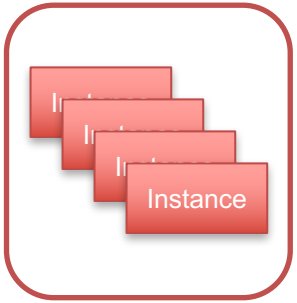
```
message VTRService {
    optional string description = 1 [max_length = 254, null = True, blank = True];
    required bool enabled = 2 [default = True, null = False, db_index = False, blank = True];
    required string kind = 3 [default = "generic", max_length = 30, blank = False,  null = False];
    required string name = 4 [max_length = 30, blank = False, null = False];
    optional string versionNumber = 5 [max_length = 30, blank = True, null = True];
    required bool published = 6 [default = True, null = False, blank = True];
    optional string view_url = 7 [max_length = 1024, null = True, blank = True];
    optional string icon_url = 8 [max_length = 1024, null = True, blank = True];
    optional string public_key = 9 [max_length = 1024, null = True, blank = True];
    optional string private_key_fn = 10 [max_length = 1024, null = True, blank = True];
    optional string service_specific_id = 11 [max_length = 30, null = True, blank = True];
    optional string service_specific_attribute = 12 [null = True, blank = True];
    optional manytoone controller->ServiceController:services = 13 [null = True, blank = True];
}
```

Model Definition

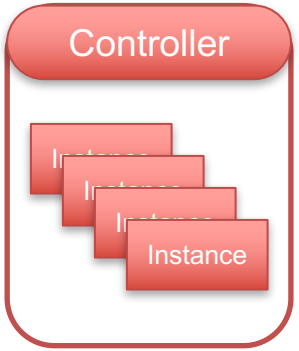
```
policy slice_policy
    < context.user = object.creator
      or exists Privilege:Privilege.site = object.site
                and Privilege.user = object.user
                and Privilege.permission='pi'
                or Privilege.permission='admin' >;

message Slice::slice_policy (XOSBase) {
    required string name = 1 [max_length = 80, blank = False, null = False, db_index = False];
    required bool enabled = 2 [ default = True, null = False, db_index = False, blank = True];
    required int32 max_instances = 3 [default = 10, null = False, db_index = False, blank = False];
    optional manytoone service->Service:slices = 4 [db_index = True, null = True, blank = True];
    optional string exposed_ports = 5 [db_index = False, max_length = 256, null = True, blank = True];
    optional manytoone creator->User:slices = 6 [db_index = True, null = True, blank = True];
    optional manytoone default_flavor->Flavor:slices = 7 [db_index = True, null = True, blank = True];
    optional manytoone default_image->Image:slices = 8 [db_index = True, null = True, blank = True];
    optional manytoone default_node->Node:slices = 9 [db_index = True, null = True, blank = True];
    …
}
```

Models

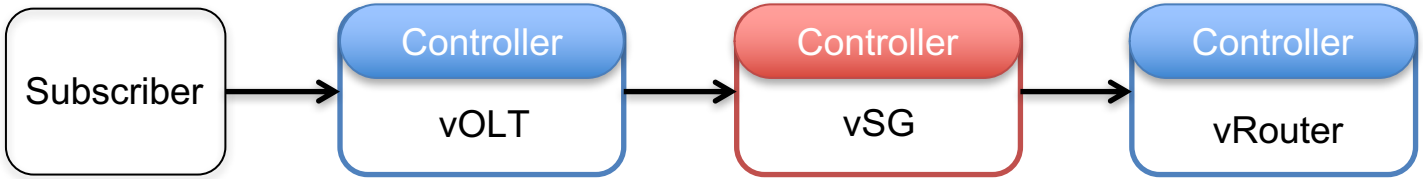
Instance =
(VM | Container |
Container-in-VM)



Slice =
(Instances[ ] + Networks[ ])



Service =
(Controller + Slices[ ])



Service Graph =
(Tenants[ ]+ Services[ ] + Dependencies[ ])

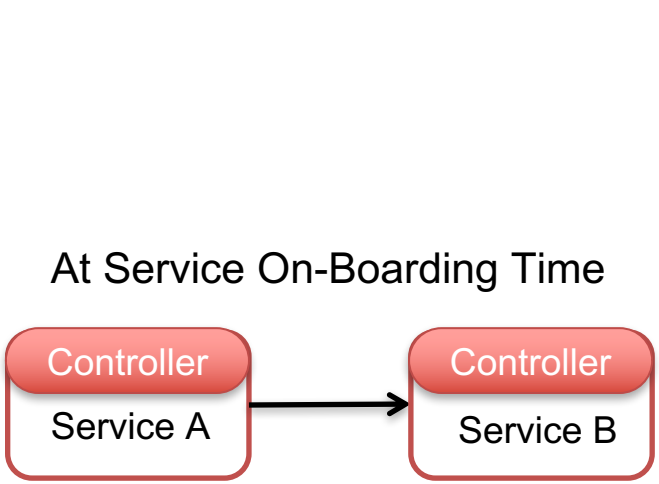
Model-Driven Design

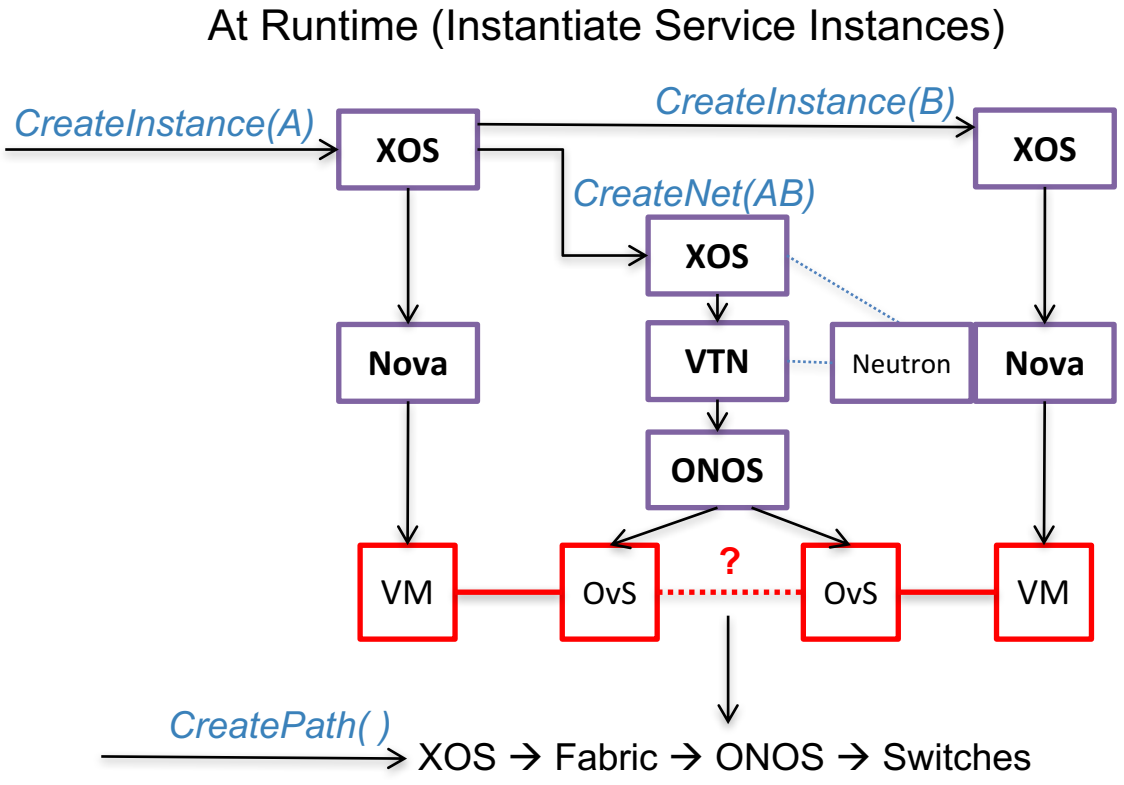
- Models are the definitive specification of the architecture
  - Defines the abstract objects and the relationships among them
  - Predicates (first order logic) defines actions on models
- Architecture is “executed” to operationalize the system
  - Represents the system’s authoritative state
  - Auto-generates all Northbound APIs
  - Enforces security policies and engineering invariants
  - Activates the data plane (backend components)

Model-Driven Design

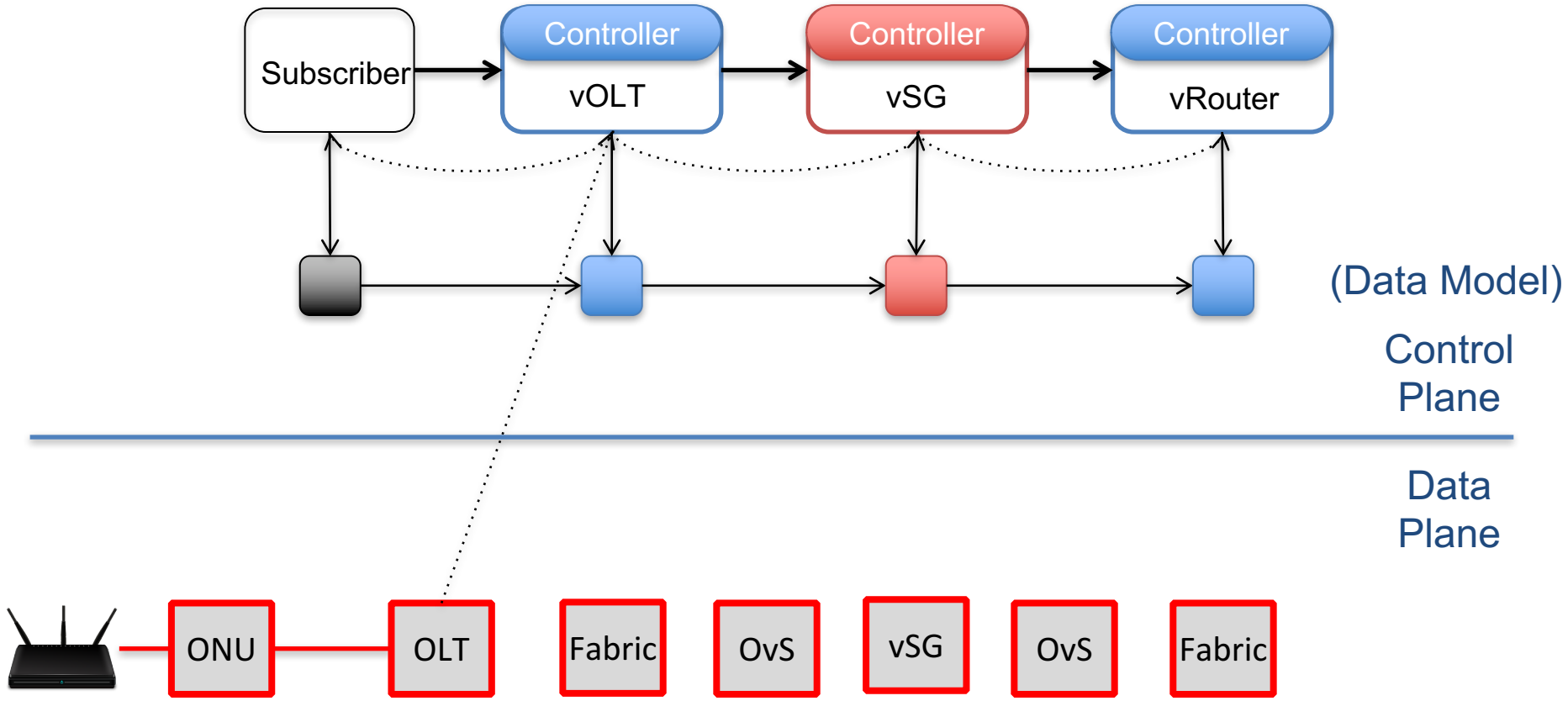
- Architecture evolves over time
  - On-board new models to extend the architecture
  - Add invariants (predicates) to reflect experience
    - New user requirements (from operators)
    - New engineering constraints (from developers)

Activating the Data Plane

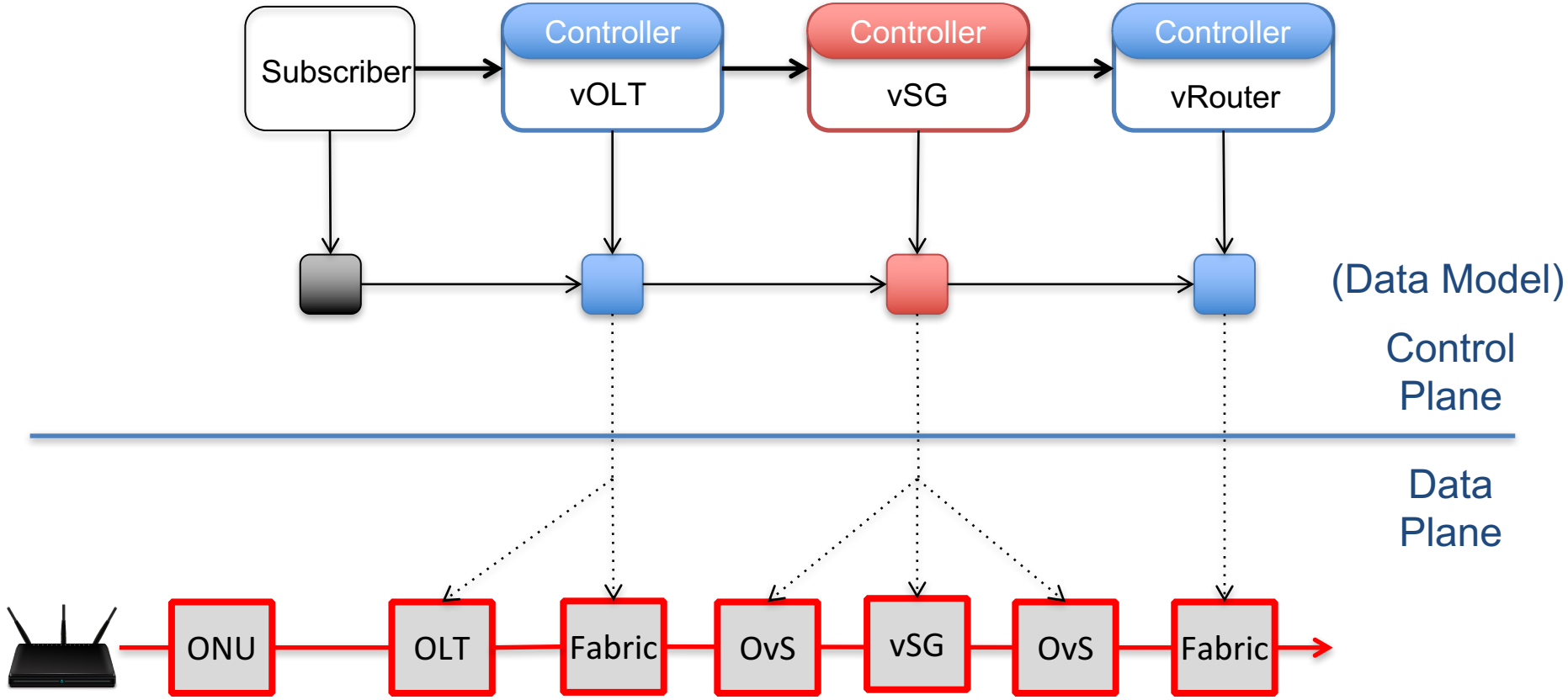




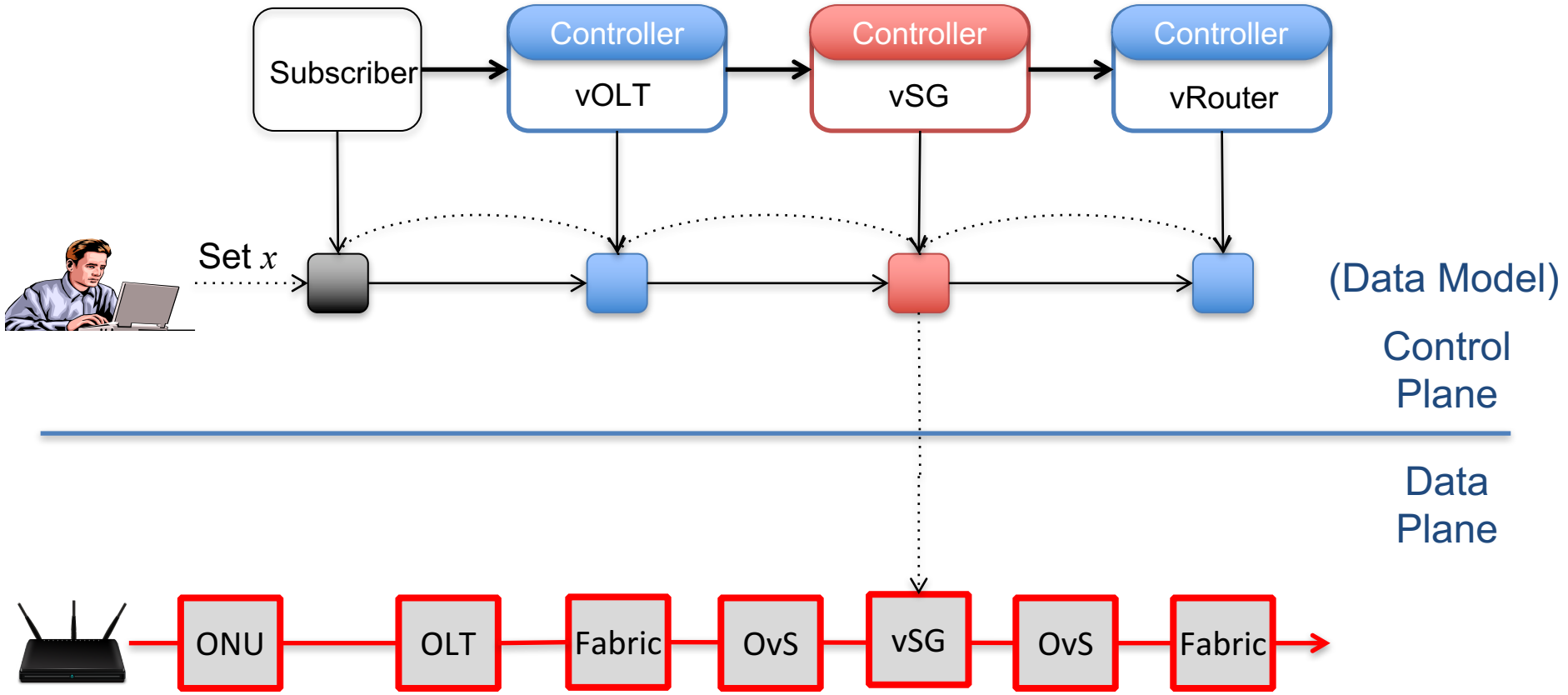
Activating the Data Plane



Activating the Data Plane



Controlling Service Instances



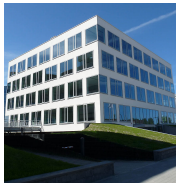
Runtime Interface (REST)

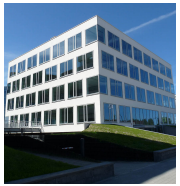
GET http://portal.cord.net:9999/api/tenant/cord/subscriber/1/

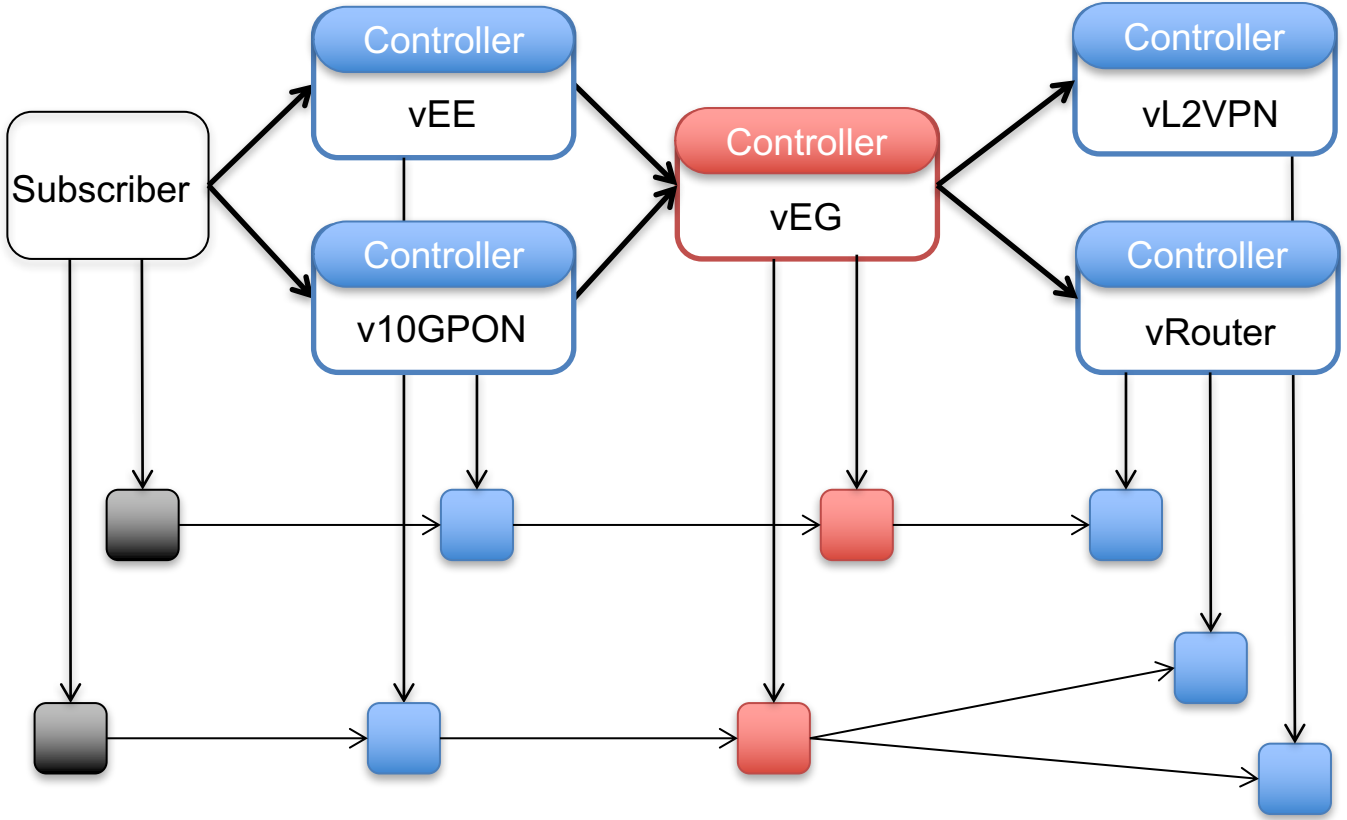
Response:


```
{
    "humanReadableName": ”John Doe"
    "id": 1
    "features": {
        "cdn": true
        "uplink_speed": 4000000000
        "downlink_speed": 1000000000
        "uverse": true
        "status": "enabled"
    }
    "identity": {
        "account_num": "123"
    }
    "related": {
        "instance_name": "mysite_vcpe"
        "vsg_id": 4
        "c_tag": "432"
        "instance_id": 1
        "wan_container_ip": null
        "volt_id": 3
        "s_tag": "222"
    }
-
}
```

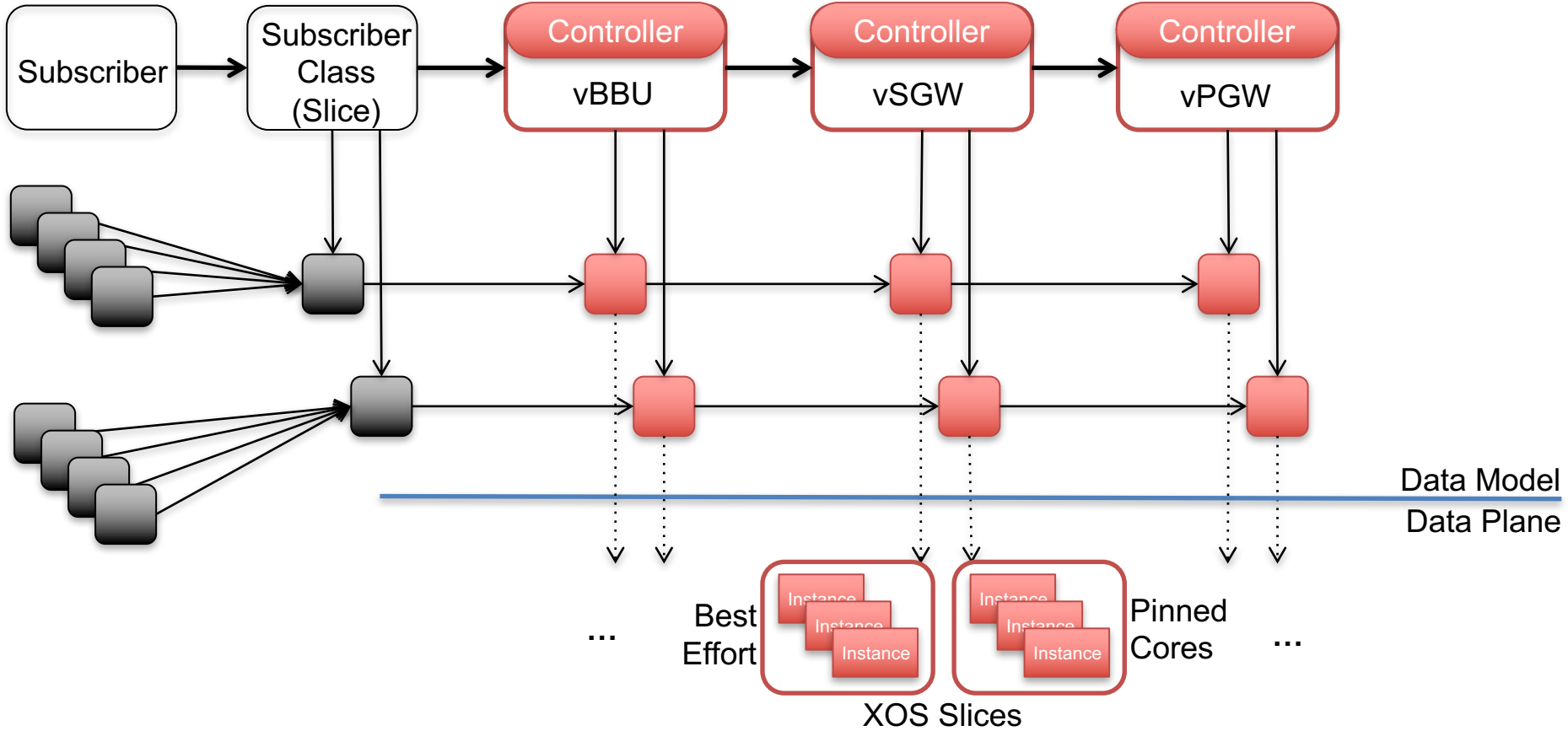
Service Graph vs Service Chain (E-CORD)



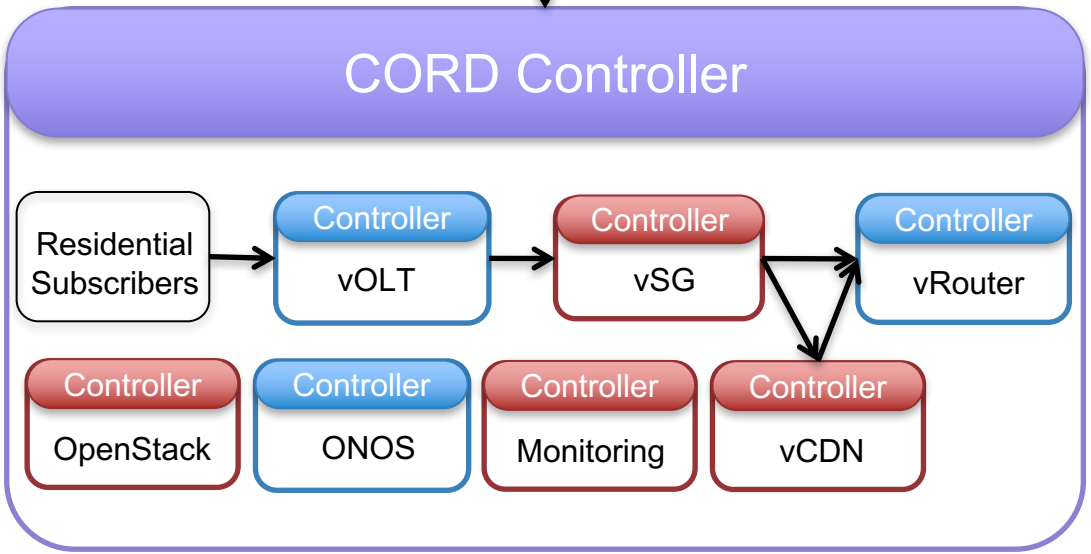




Granularity of Service Instances (M-CORD)



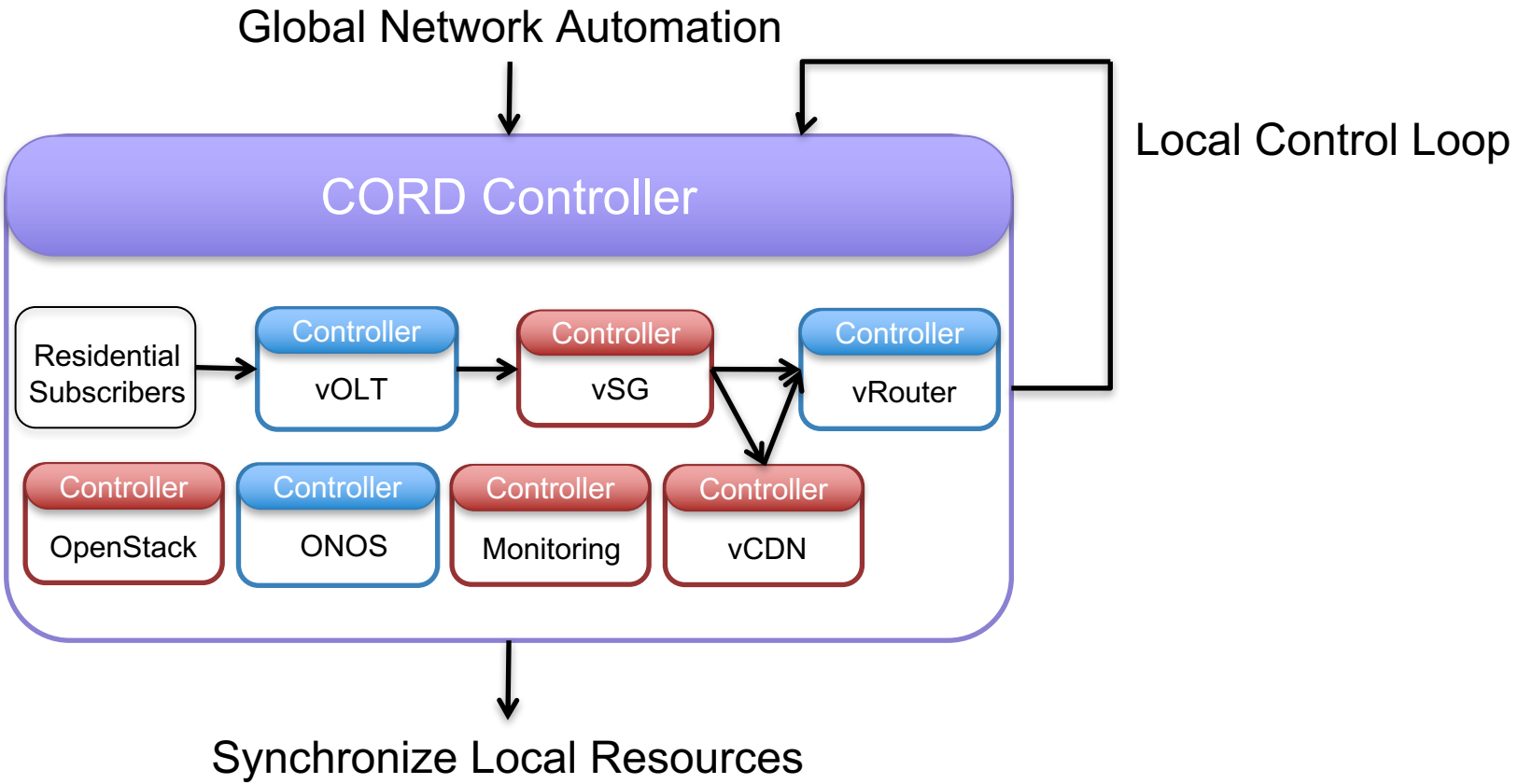
Runtime Interface



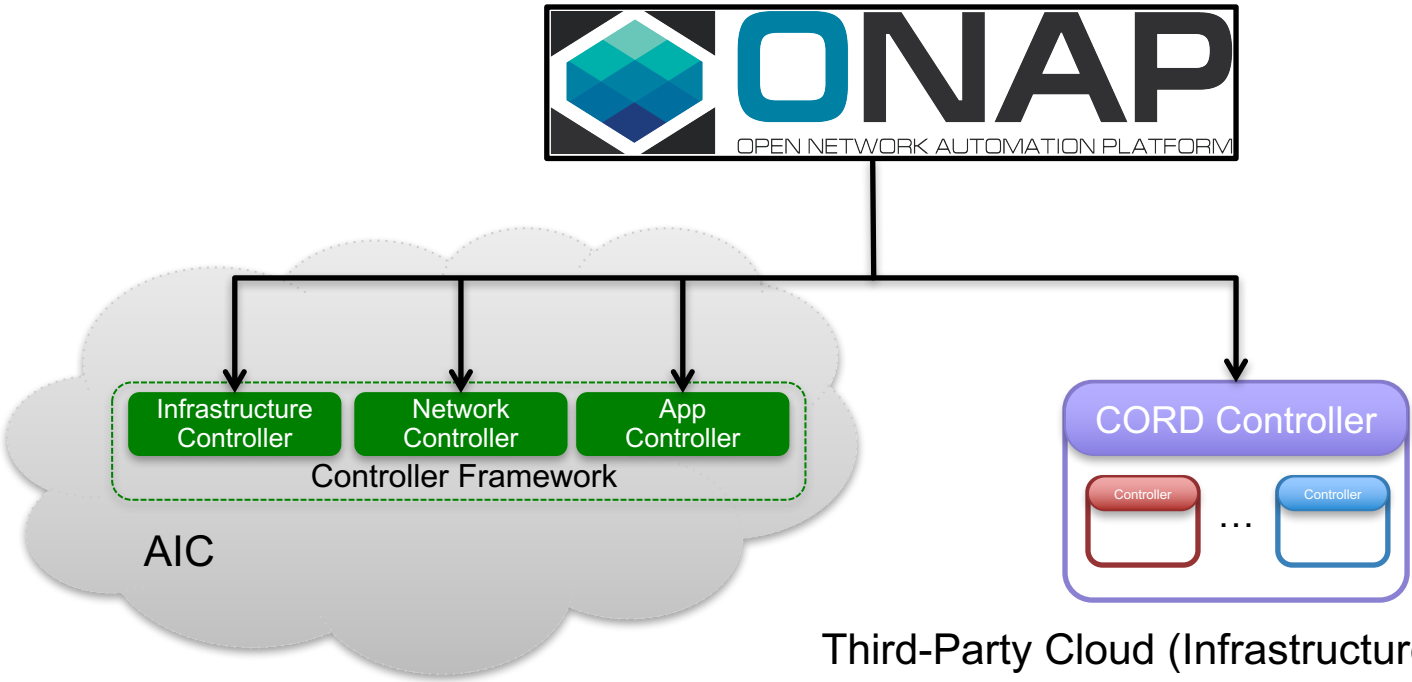
Runtime Interface

- On-board Services
- Synchronize State
- Provision Services
- Instantiate Instances
- Control Instances
- Report Inventory
- Report Analytics

CORD Controller



Big Picture



Third-Party Cloud (Infrastructure) Controller
- – Peer of AIC, Amazon, Azure
- – Unique Capabilities
  - → SDN-based Services
  - → Access Devices (East/West Traffic)

Summary

Micro-Services are a tried-and-true way to build scalable apps

DevOps is an agile way to manage and control scalable services

But…

- Limited security model → Single trust domain
- Limited flexibility → A solution, not a platform
- Limited use of SDN → Plumbing, not a source of services

Solution…

- Layer Operations-as-a-Service on top of Micro-Services
- Leverage centralized Data Model to “drive” DevOps tools
- Leverage SDN as a source of innovative services