ONAP

OPEN NETWORK AUTOMATION PLATFORM

ONAP Micro-service Design Improvement

Manoj Nair, NetCracker Technologies

Micro Service Definition

Micro service architectural style is an approach to developing a single application as a **suite of small services**, each **running in its own process** and communicating with lightweight mechanisms, often an HTTP resource API. These services are **built around business capabilities** and **independently deployable** by **fully automated deployment machinery**. There is a **bare minimum of centralized management** of these services, which may be written in **different programming languages** and use **different data storage technologies**.

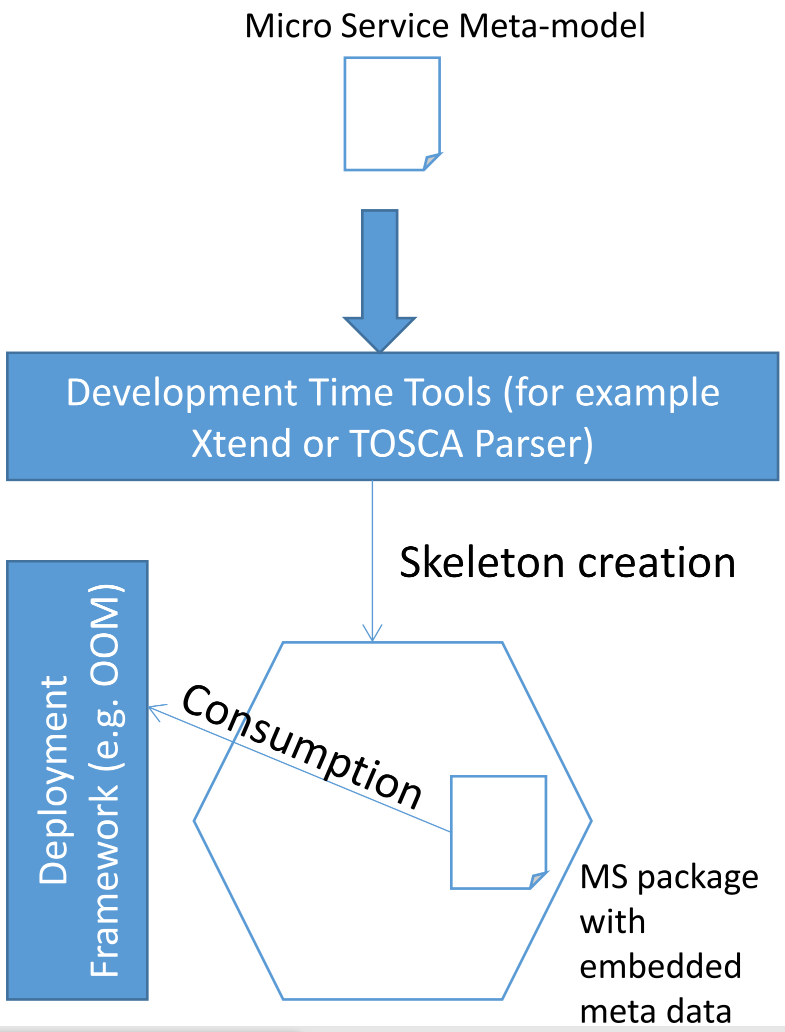
-- James Lewis and Martin Fowler(link)

Problem Statement

1. Micro Service Meta Model : Majority of micro services in ONAP are created around functional boundaries without any well-set guideline
   - Here guideline means the expression of structure and artifacts micro services use in ONAP to ease development, integration and deployment– like capability/endpoint registration, deployment configuration, dependencies, resources requirements, non-functional characteristics etc.
2. Micro Service Boundaries : Boundaries of micro services based on high cohesion, low coupling principle
3. Micro-Service Shared Concerns: How cross-cutting concerns of micro services are handled
   - Such as external dependency configurations like IP address, URL, Authentication tokens etc, Logging, Health Check, Monitoring metrics, Tracing, data persistence, distributed messaging etc.

1. Micro Service Meta Model

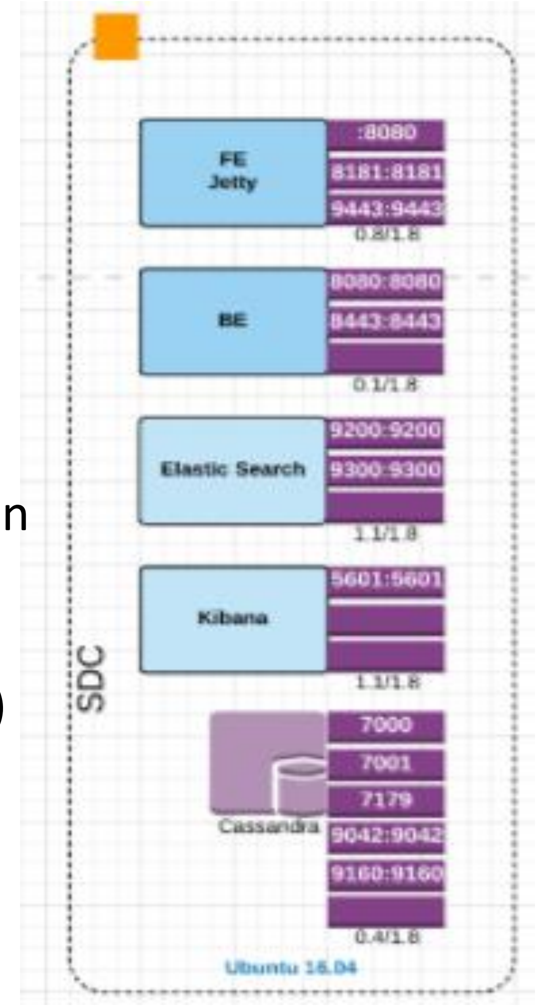
- Defines the key characteristics of a micro service
  - Which can be used at development time tools for development of micro service structure
  - Which can be used at run time to discover the dependencies, capabilities
- Development time tools such as Xtend, Maven Archetypes etc. and DM Languages like EMF,TOSCA, YANG (or general purpose languages like XML,YAML): To be developed
  - To generate structure of micro service
  - Docker file for deploying micro service
  - Registration configuration for registering with DMaaP (for topics) , MSB (for end points)
  - Kubernetes deployment artifacts
- Meta data placed in the deployment package of micro service is used by the deployment/runtime frameworks : Partially available
  - To deploy the micro service based on the deployment artifact
  - To allocate resources based on the micro service requirements
  - Resolve dependencies
  - Register micro service capabilities and end points
  - Register with DMaaP Topics


Micro Service Meta-model
Development Time Tools (for example Xtend or TOSCA Parser)
Skeleton creation
Deployment Framework (e.g. OOM)
Consumption
MS package with embedded meta data

2. Micro Service Boundaries

- Boundary of a micro service is a subjective decision and varies
- In ONAP Micro Services are created based on functional boundaries for e.g. SDC is built with 4 different micro service each focusing on a specific functional capability – with tight dependency between each other
- Too much logic is built into BE – Titan Graph, SDC Processing , Catalog Management etc. – Models, Design, Catalog, Onboarding/Distribution, Health Check (Cassandra connection) Functionalities are built into same micro service (BE) forming a monolith

Example SDC Micro service View



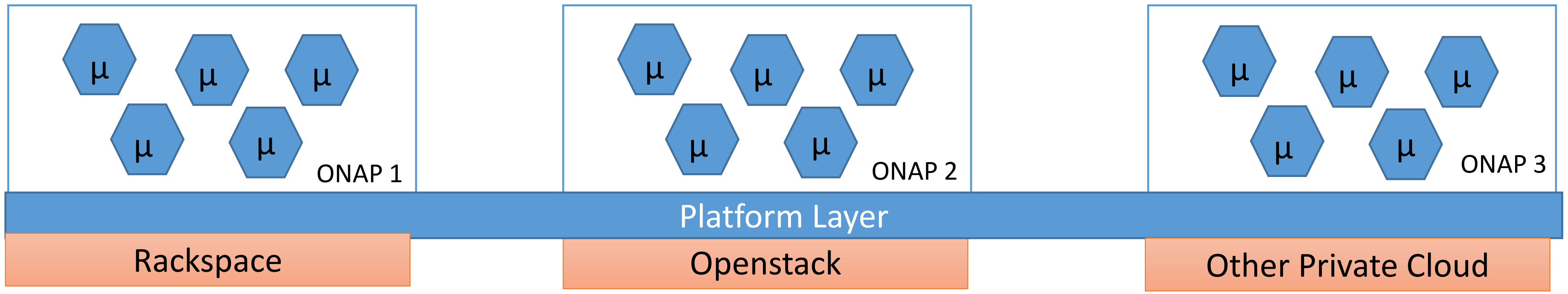
Typical models for fixing micro service boundaries –

- Moving parts that change frequently (e.g. SDC Model, Policy Configuration, Thresholds in DCAE/Clamp)
- Technology specific Polyglots (e.g. VFC)
- Functions/Features with similar non-functional requirements (e.g Caches, Graph, States)
- Transactions across micro services and consistency requirement (e.g. A&AI updates and SDC Catalog updates)
- Security based isolation requirements (e.g. customer, service, resource and topology information stored together in A&AI)

Key Message : Re-Structure Micro Services based on S3P and Business Context

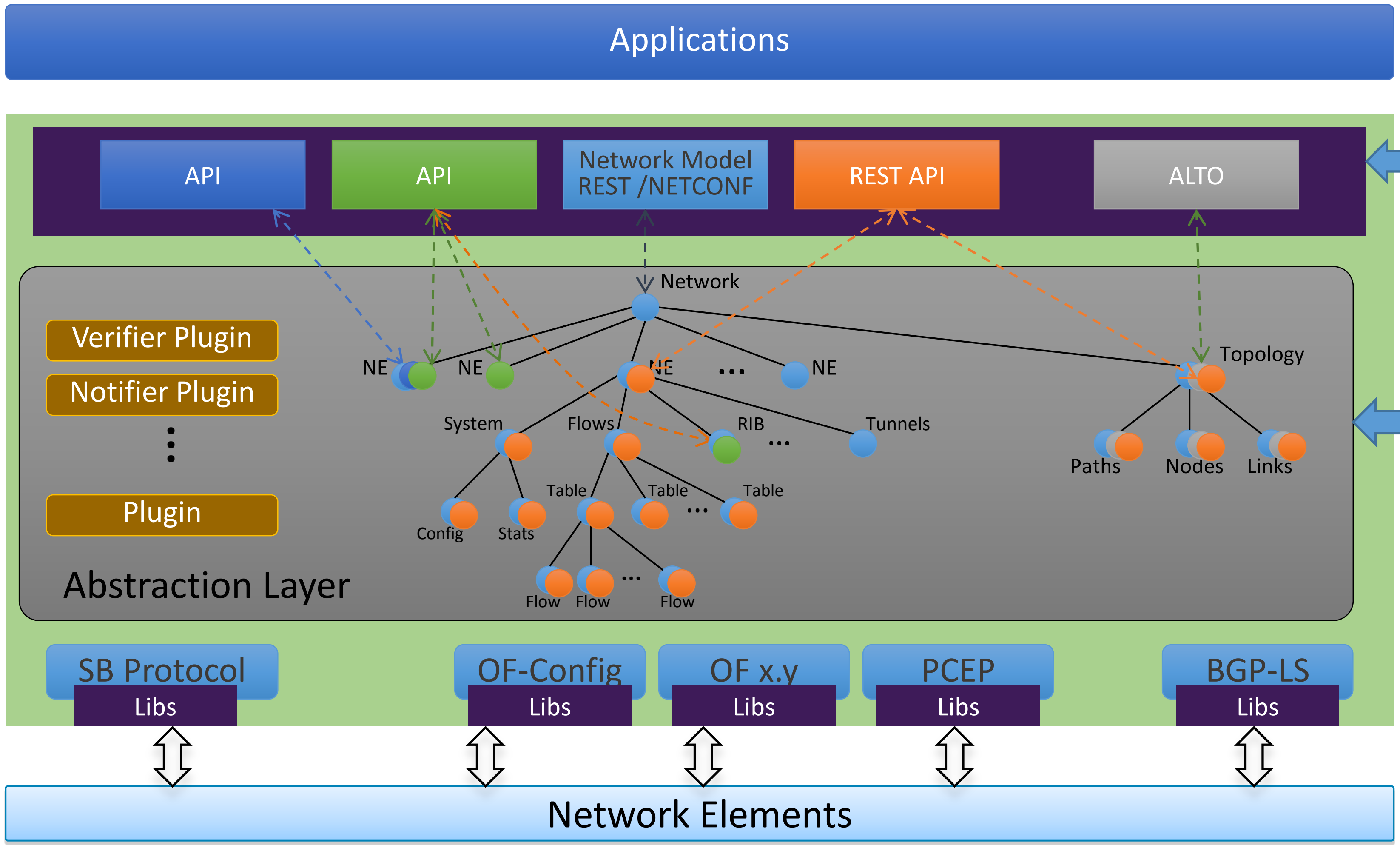
3. Micro Services Shared Concerns

- ONAP Common Services layer consists of services that are shared across micro services like Logging, AAF, DMaaP, MSB etc. which are set of micro services in itself.
- Shared concerns can further extended to be a platform layer for micro services. Platform layer for micro services can host set of common needs of micro services which can be technology specific or neutral.
- In addition to the container orchestration/scheduling and health check/monitoring , it will be convenient to use a shared platform layer so that
  - Distributed concerns of micro services are hidden in the platform layer which gives location transparency
  - Can centrally control the S3P aspect of micro services rather than each micro service handling it independently
  - Work transparently across container orchestration technologies like Swarm, Kubernetes or VM Orchestration Technologies like open stack
  - Isolate micro services from infrastructure layer intricacies
- Note : Platform layer may have implication on the deployment model (Greenfield or Brownfield) in customer premise where in ONAP need to interwork with existing micro services. Also platform layer make sense if there are many micro services that need to be managed



Key Message : Portability of micro services and MS focus on core domain (can we use a PaaS)

Learning from Opendaylight 1/2



Reference: Opendaylight MD-SAL Architecture

Projects developed as multiple micro services register with MD-SAL platform – about capabilities, interfaces, entities etc. Each project is modelled using YANG and compiled to create skeleton code

OpenDaylight MD-SAL – Platform logic enabling the Development and Run time Framework

MD-SAL takes care of data storage , cluster data consistency, sharding, interaction between micro services, set of tools for project skeleton

Learning from Opendaylight 2/2

- Model Driven Service Abstraction Layer (MD-SAL) is base framework used in Opendaylight
- SAL is modeled by YANG
- SAL supports two formats of code generation based on YANG Models
  - **Binding-independent format:**
    - Data and functions calls independent of generated Java language bindings
    - Data and functions calls expressed in neutral DOM-like model
    - Typically used in framework tools, South bound plugins
  - **Programmatic (Binding-aware) format:**
    - APIs generated from YANG models
    - APIs represented in Java as generated classes
    - Typically used for developing applications or services
- SAL has built in brokers to read/write data from data store, invoke rpc calls, publish/subscribe notifications.
- SAL also contains a Configuration Sub system for micro services to register and search capabilities, apply initial configuration.

Our Proposal

- Bring in consistency in expressing micro service configuration while allowing freedom of choice. – Several reference implementation exists for example yang based model for ODL , Xtend, TOSCA etc.
- Be more prudent in defining micro service boundaries so that refactoring of code becomes easy for meeting S3P and carrier grade performance
- Enable portability of micro services by separating out the infrastructure requirements and configurations from core MS logic – through development of a platform layer) – PaaS Solutions as one option

ONAP
OPEN NETWORK AUTOMATION PLATFORM

Thank You