Scalable Software-Defined Monitoring for Service Provider DevOps

W. John¹, C. Meirosu¹, B. Pechenot², P. Sköldström², P. Kreuger³, R. Steinert³

¹ – Ericsson Research; ² – ACREO Swedish ICT; ³ – SICS Swedish ICT;
IN A NUTSHELL

UNIFYing cloud and carrier networks

› Increased velocity of service introduction

› Unified network-cloud programming abstraction and orchestration

› Novel, integrated management features to coping with high service velocity
DevOps in the IT and DC world

An agile approach to deliver software, with:

- Increased deployment frequency
- Lower failure rate
- Faster mean time to recovery
- Maximum predictability, efficiency, and maintainability of operational processes
- Programmable and automatic processes

https://en.wikipedia.org/wiki/DevOps#Goals
SP environment vs. Datacenter environments

• Higher spatial distribution- telecom resources are spread over wide areas for coverage
• Heterogeneous hardware and software
• Lower levels of redundancy in access and aggregation networks compared to typical data centers
• Carrier-grade requirements on high availability and latency
• Likely to have business boundaries between Dev and Op roles, keeping some “silos” intact
OBSERVABILITY: MAIN CHALLENGES

Centralized and manual = scalability limitations

Existing monitoring solutions

• Counter based approaches (OpenFlow, SNMP, REST) scale badly for frequent and fine-grained observability
• Common interfaces between various data sources/tools and management components needed to support automation
Scalable software define monitoring

1. A software defined Monitoring function (MF) concept introducing a control/data plane split
   • Allows to perform some management function distributed
   • Requires advanced programmability and processing capabilities

2. A scalable and flexible communication channel between MF components and higher control/management layers
   • DoubleDecker (DD): messaging system based on ZeroMQ
   • Transport/address agnostic, flexible, lightweight, feature-rich
Components of Monitoring functions (MF)
- MF control app
- Observability Points (OP)
  - Local control plane (LCP)
  - Local data plane (LDP)

Components of DoubleDecker (DD)
- DD brokers
- DD clients
  (ie. any other component)
- Aggregators
Example MF: Distributed link utilization monitor

The RateMon MF is implemented in terms of two components:

- Monitoring specific local control component (LCP) for estimation and analysis of observed rates, and variation of measurement intensity
- Local data plane (LDP) component for accessing counters

Example MF: Distributed link utilization monitor

- Use two counters (first and second statistical moments for link utilization)
- Transmit only the estimate to the controller, instead of high-speed data flow
- Estimating congestion risk: a simple threshold method gives a success rate of >98% when updating the estimates at 0.3 s intervals.

DoubleDecker messaging channel

Requirement on messaging channel (for SP Scenario)

- Propagating configuration, signalling, and results from monitoring functions (MFs) to control/mgmt entities in higher layers
- Distributed architecture, keeping messages as local as possible
- Lightweight and easy (e.g. no IP termination in MF, single channel)
- Topology independent and network address agnostic, supporting dynamicity and migration
- Flexibility of transport mechanisms
- Support for multiple programming language for flexible usage
DoubleDecker messaging channel

Message system on top of existing connectivity

- Existing Message systems: RabbitMQ, ActiveMQ, and ZeroMQ
- RabbitMQ and ActiveMQ have advanced features (e.g. reliability), but require TCP/IP transport and use centralized brokers
- ZeroMQ is more lightweight, and supports diverse transport methods (shared memory, IPC, TCP/IP, etc.) and programming languages (C, C++, Python, Erlang, ....)
DoubleDecker* messaging based on ZeroMQ

- Clients and Brokers based on ZeroMQ socket types
- Clients use no address, but generic identifiers
- Clients connect to brokers using one of the supported protocols
- Clients can be OF agents, daemons, containers, VMs, controllers ... 
- Additional features: connectivity heartbeat, authentication, tenant isolation, direct messaging and pub/sub, topological scoping

* Demo at EWSDN: Monitoring Transport and Cloud for Network Functions Virtualization
DoubleDecker with local aggregator

- A local aggregator can be used to pre-process (or filter) monitoring data from generic MFs
- Supports aggregation for all types of (legacy) MFs and OAM tools
- Allows aggregation across various metrics
Numerical analysis: Scenario

- All physical and virtual switch ports are monitored: 823k ports on 193k switches
- Packet overhead: VXLAN, IPv4, UDP headers (72 bytes)
  - SNMP: SNMPv3 header (18 byte)
  - OpenFlow: OFv1.5 header (16 bytes)
- Counters (packet data) are 64-bit long
- Read and message frequencies corresponding to RateMon [1]
- OpenFlow uses multipart replies and port statistic replies to OF ANY (aggregate counters from all ports of a switch in one message)

Numerical analysis: Results

- **RateMon:**
  - Bandwidth savings: ~4000 times (SNMP and OF)
  - Message rate reduction: ~3000 times (SNMP) and 700 (OF)

- **DoubleDecker aggregation at Rack-level**
  - Additional message rate reduction: ~230 times

- **Resulting controller/NMS requirements**
  - 1072 messages/s and 49Mb/s
  - (Compared to 820 million messages/s and 627 Gb/s)
Conclusion

• Scalability of monitoring can be improved by
  • Adopting split-architecture principles for certain monitoring functions
  • Using distributed monitoring channel with local aggregation features
• Proposed system enables distributed pre-processing of results (e.g. node-local)
• A tradeoff between observability granularity and resource consumption
• Numerical analysis shows savings of several orders of magnitude

• Open issues:
  • More expressive programming capabilities for better integrating monitoring with orchestration and controllers (ongoing work)
  • Aggregation strategies for different metrics and purposes
  • More flexible dataplane programming capabilities required for advanced monitoring
Backup
UNIFY SP-DEVOPS CONCEPT

Troubleshooting

Operate

Define

Deploy

Verification

Observability

Operator

Service developer

VNF Developer

VNF developer support
Example MF: Distributed link utilization monitor

- From the first and second moments, the parameters of the lognormally distributed rate observed at ms intervals are estimated at longer intervals (min, sec or sub-sec).

- By inspecting the percentiles (e.g. > 95%) of the resulting cumulative density function, the risk of congestion can be assessed.

- Compared to assessing the average rate during 5 minutes (which is common practice today), the probabilistic approach provides a more informative metric of the congestion risk.
Example MF: Distributed link utilization monitor

- Inspection of the percentiles and a suitable threshold allows for detecting congestion episodes autonomously.

- A naive detector (simple thresholding at a certain level of risk) gives for example a success rate of over 98% when detecting episodes of high congestion risk when updating the estimates at 300 ms intervals.

- The approach is highly efficient – running the MF at varying time scales reduces the monitoring overhead by 3000 times compared to implementing a similar MF using SNMP.