IETF 96 – IRTF – SDNRG

Berlin, Germany
July 22, 2016

Control as LCD for future networking

Artur Hecker and Zoran Despotovic

European Research Center, Munich
Huawei Technologies

Programmable networks: change in paradigm

- Legacy: design a network to provide a specific (set of) service(s)
  - Protocols/ exchanges (management, control, data) + stacks/ logic
  - Deployment topology, configuration to bind the pieces
  - Operational traffic steering/traffic distribution
- Programmable networks:
  - Infrastructure with basic capabilities and open interfaces
  - Services: several logics programmed on the latter
- Change: design the network now, program the service later
- Software brittleness (*): cyclomatic complexity

Code from vendors tested on their HW

Untested code from 3rd parties

(*) Steve Bellovin

Example: OF SDN

- **Chicken-Egg Problem** in SDN
  - Current SDN *promises* a “software-defined networking”, yet it actually *requires* an existing, well-configured and well-working TCP/IP network
    - Note that this is independent of in-band / out-of-band discussions
  - A pre-set, fixed CP in SDN cannot suit all use cases that SDN promises
    - Non-functional requirements: QoS, scalability, reliability, resilience
- **Self-inflicted errors** in SDN
  - Insufficient protection: the programming model is comparable to DOS
    - You can write a control app to disconnect the controller from switches
    - Hard to protect against this w/o limiting programmability

Limits programmability

General Problem Statement

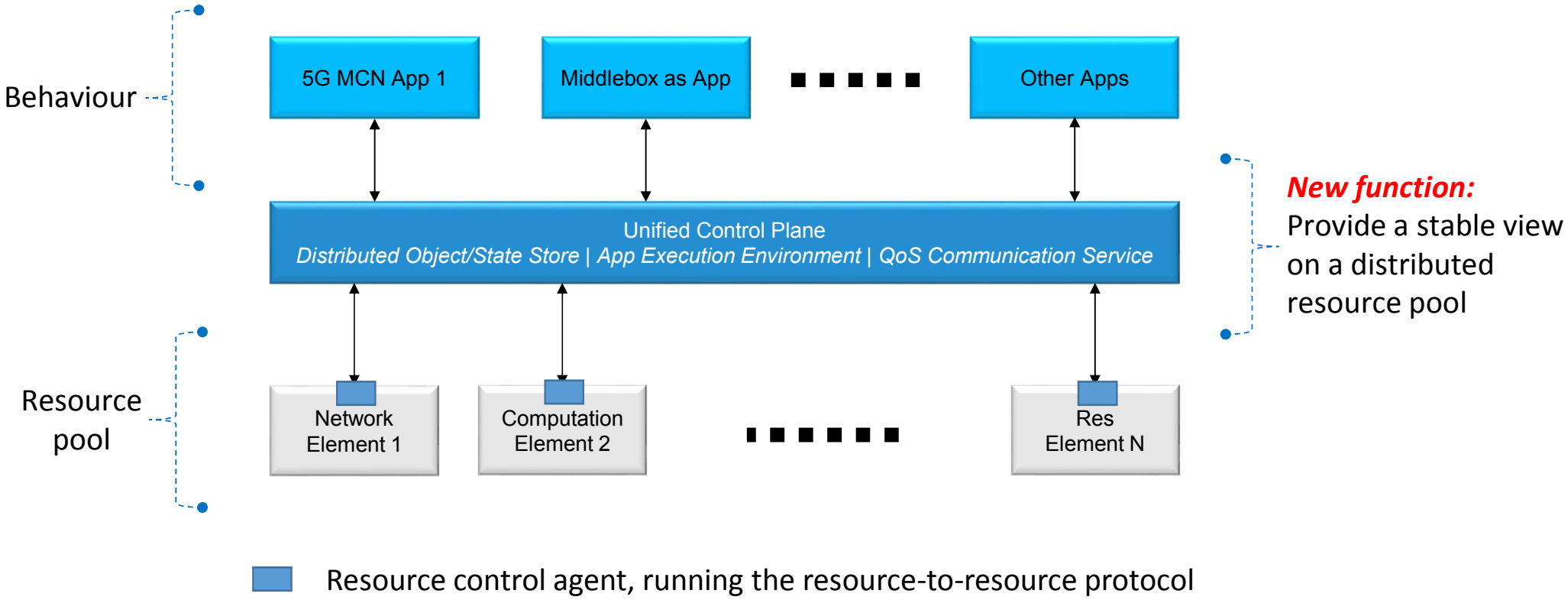
- Context
  - Many components (HW/SW; remote/local; short-/long-lived)
  - Need to be able to bind them to working services operationally

- What is the common minimal requirement on all those components?
  - How to make them programmable?
    - Without making the components too complex
    - Without having to manually deploy things
  - How to make such programmability simple / usable?
    - What do you need to know to start? Does trial-and-error model work?

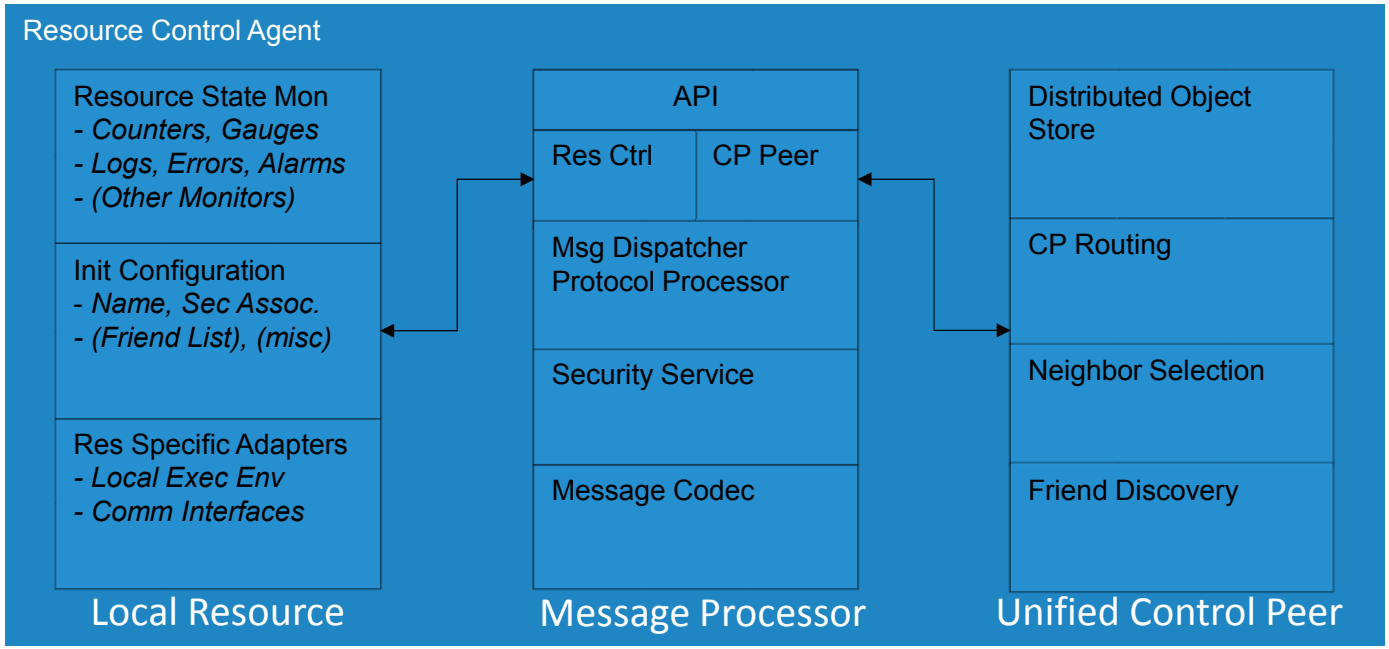
Our Proposal: Unified control

- Resource-to-resource protocol suite, **dedicated to establishing and maintaining control**
  - Akin to BGP establishing and maintaining IP routing
  - ... But without presuming a specific usage
- Two dimensions of unification
  - Horizontal: span different types of components
  - Vertical: span both executing and executed modules

Our model


Behaviour
5G MCN App 1
Middlebox as App
Other Apps
Unified Control Plane
Distributed Object/State Store | App Execution Environment | QoS Communication Service
Resource pool
Network Element 1
Computation Element 2
Res Element N
New function:
Provide a stable view on a distributed resource pool
Resource control agent, running the resource-to-resource protocol

RCA implementation



Two faces of the RCA: the RCA acts both as an interface to the local resource that it controls, and as a building block of the control plane spanning all resources within the control domain (unified control peer).

Phases

- All resource elements (RE) have an RCA
- Phases, repeated (on event / periodically)
  - All REs bootstrap (find all visible friends)
    - Friends: RCAs from the same control domain
  - All RCAs choose from the friend list some neighbors
  - RCAs run routing over neighbors only
  - The controller capacity and placement is decided autonomously
    - E.g. the topologically most important RE with compute capacity becomes Controller
  - Using distributed storage, RCAs eventually discover a new abstract function “controller”

Conclusions

- We propose a new resource to resource protocol suite
- Capable of producing self-* control planes
  - Need to produce a resilient common functionality to be able to control the resources and the modules
    - Network OS “Kernel”
- Could be a possible extension to ANIMA
  - Extend to other resource types and modules
  - Extend from control channel to control plane
  - Fundamental: infrastructure control through the controlled infrastructure
    - Conflict modeling

Appendix: Cmp. Unified Control to ANIMA

Criterion	ANIMA	Unified Control
Zero preconfiguration ready	Yes (for networking resources)	Yes (for resources)
Discovery	“All nodes”	Only friends
Autonomic Control Plane	Yes, interconnect nodes	Yes, establishes control
Routing	On all nodes	On neighbors only
Compute Nodes	No	Yes
Overlay structure	As it emerges	Use neighbor selection criteria
Distributed storage	No?	Yes
Secure bootstrap	Yes	Not considered so far
Support for topology dynamics	Yes	Yes
Religion/ paradigm	Autonomic networking	Controlled networking Only autonomic in its own implementation

Appendix: what's wrong with orchestration?

- Orchestration is a management function ☺
  - Requires signaling channels and control
  - Is too far away
  - Cannot efficiently react to faults, local events, etc

State of the art: