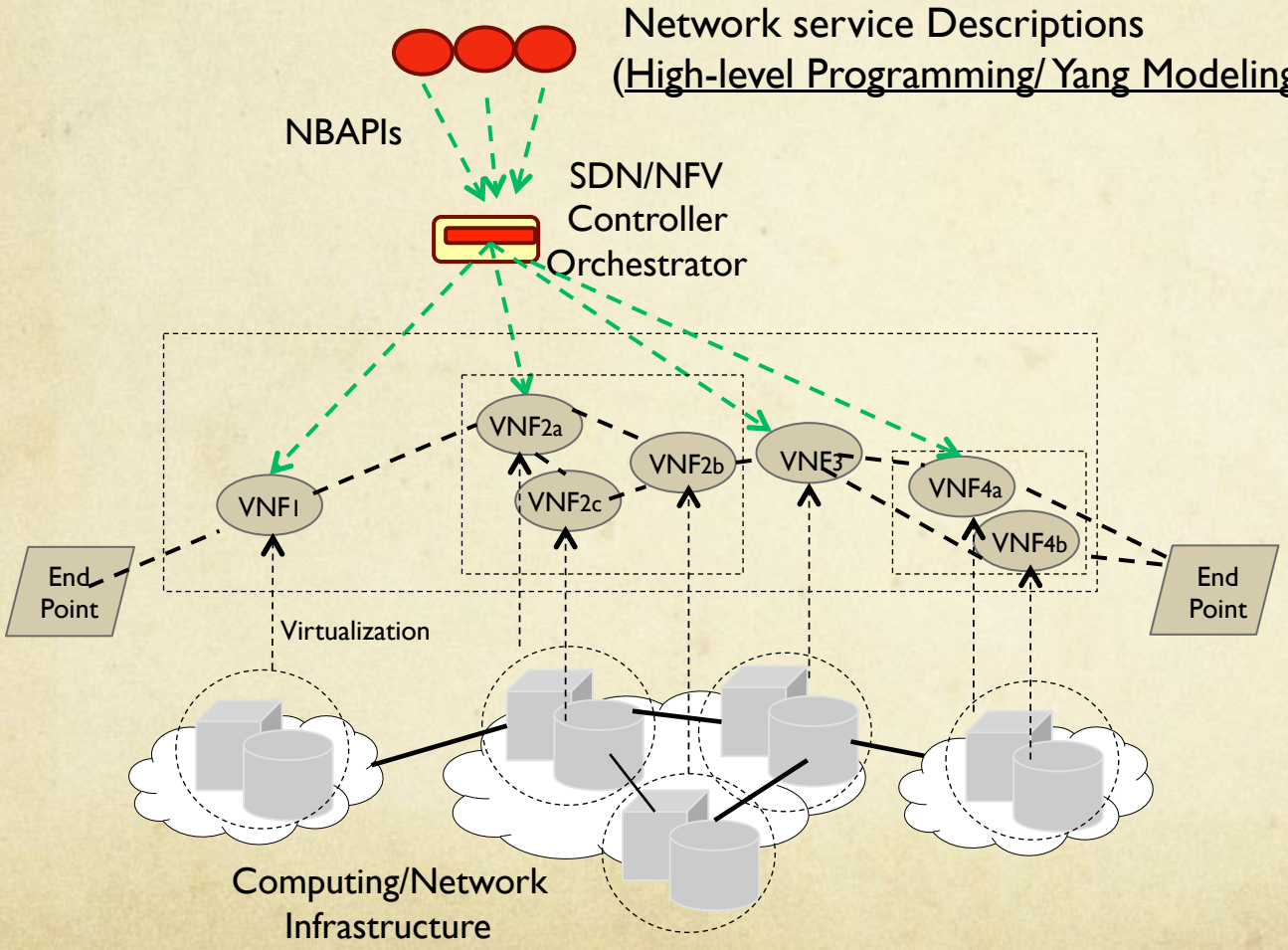
Verification of NFV Services : Problem Statement and Architecture

draft-shin-nfvrg-service-verification-00

M-K. Shin, ETRI
K. Nam, Friesty
S. Pack, Korea Univ.
S. Lee, J. Yi, ETRI

Proposed NFVRG Meeting@IETF91, Honolulu, Hawaii

Programmable Infrastructure

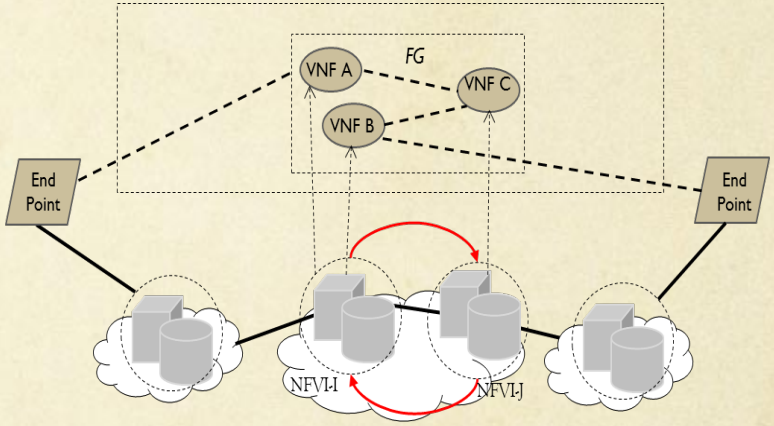


A Network Compiler that translates network service description in which 3rd Parties as well as Operators define what they want from the infrastructure and compiles it into low level instructions (e.g., Network Controller/OpenStack primitives) for network service components

Problem Statement (1/2)
(Key Properties to be checked)

1. **Dependencies of Network Service Components**
   - NFV components (e.g., NFVI, VNFs, Orchestrator, Network Controller, etc.) have intricate dependencies that make operation incorrect.

2. **Loop-Free in VNF FGs**
   - In VNF FGs, an infinite loop construction should be avoided and verified.



3. **Policy and State Consistency**
   - When the policy is changed, the policy should be reconfigured in VNF service nodes as soon as possible. If the reconfiguration procedure is delayed, inconsistent policies may exist in service nodes.

Problem Statement (2/2)
(Key Properties to be checked)

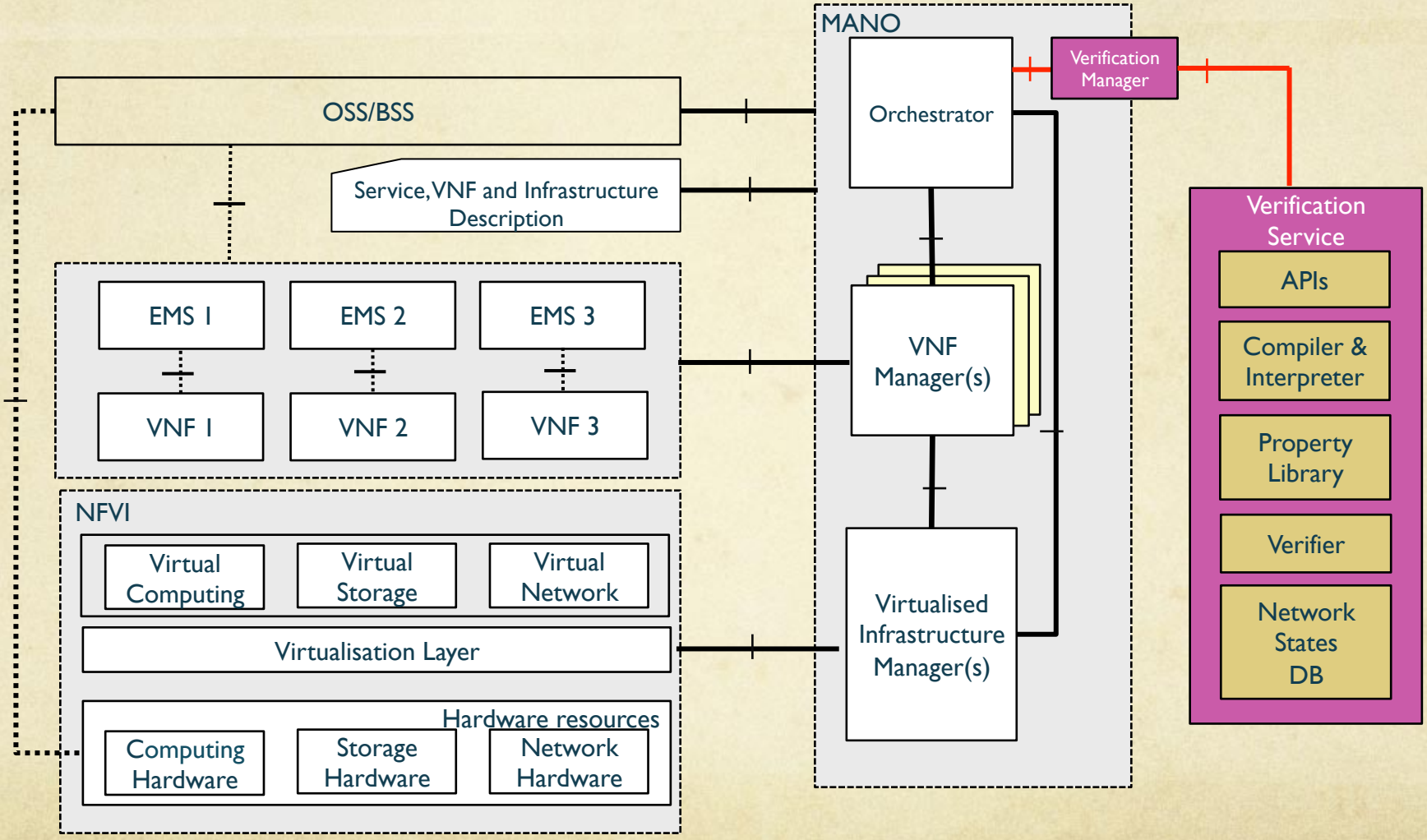
4. **Load Balancing and Optimization among VNF Instances**
5. **Performance Bottleneck**
6. **Security Hole**
7. More general requirements (esp., related to service functions chains) can be also found at draft-boucadair-sfc-requirements-05

Note that some of properties above, such as load balancing, optimization, performance, etc. should be monitored and checked together by MANO operations.

Minimal Requirements

- R1 : It should be able to check global and local properties and invariants. (E.g., Loop-freeness and resource isolation between VNFs can be regarded as global. The policies that are related only to the specific network controllers or devices are local.)
- R2 : It should be able to access to the entire resource DBs as well as network states whenever verification tasks are started.
- R3 : It should be independent from specific solutions and frameworks, and APIs.
- R4 : It should process standard protocols such as Netconf, YANG, OpenFlow, I2RS, etc. and northbound and southbound interfaces that are related network configurations, and used by OSS.

Verification Framework



Note that Verification Service and Verification Manager in the NFV MANO should communicate using APIs to accomplish the verification tasks.

Verification APIs Example

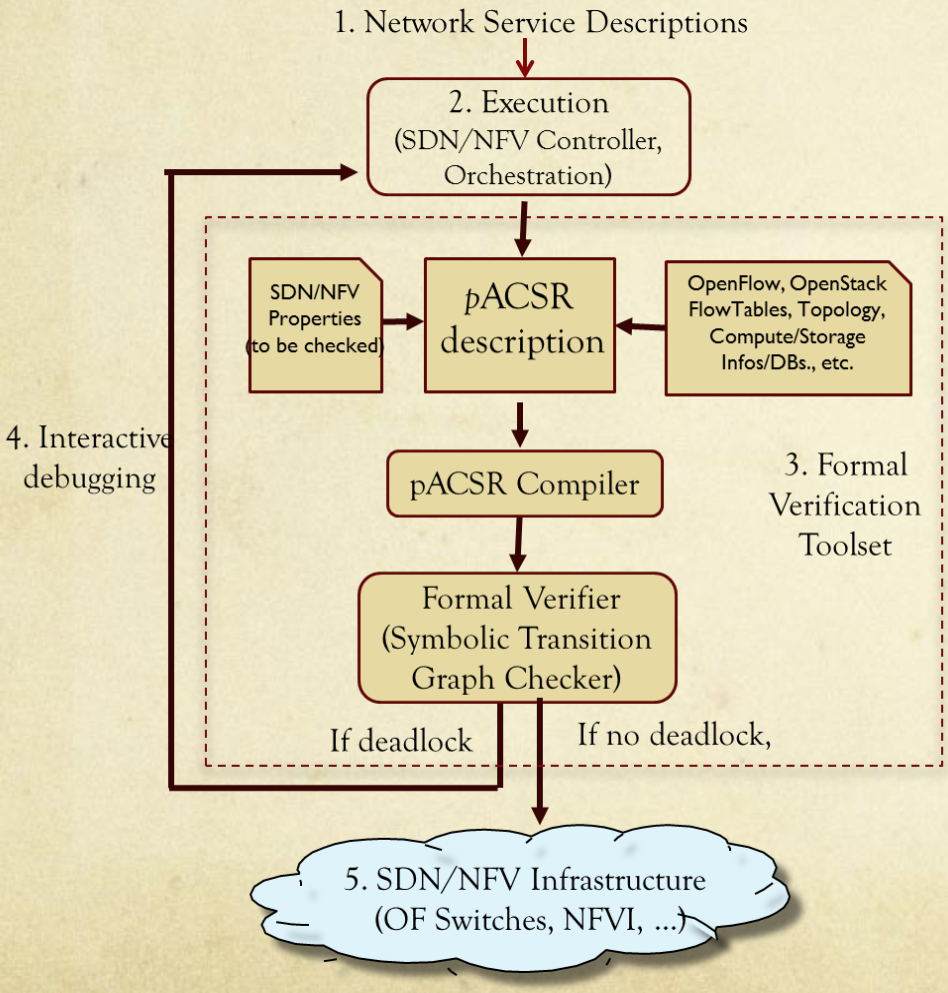
- Verification Manager (in NFV MANO)
  - set_properties(properties or invariants)
    - It sets a list of properties to be checked. After calling this API, verification tas ks are automatically started whenever configurations are changed
    - It returns a structure consists of the result (true or false) and explanation in ca se of false (e.g. properties are already set, conflictions)
  - get_properties() returns a list of properties set in the NFV MANO
  - get/set_network_configuration gets or sets global network configurati ons required for VNFs
- Verification Service
  - verify (property, virtual_network, vnf)
    - parameters:
    - returns a structure consists of verification result(true or false), and explanatio n or counter example in case of false

Challenging Issues

- Finding infinite loops
  - General solutions for the infinite loop can lead to intractable problem (e.g. the halting problem). To make the verification practical and minimize the complexity, some of the restrictions are required.
- Real-time verification
  - A few invariants can be checked in real-time but it would be impossible if the size of VNFs increases or properties checked are complex.
- Languages and their semantics
  - Network service descriptions in NFV need to be precisely expressed using appropriate semantics (e.g., formal method). Languages and semantic models optimized to the verification framework need to selected or newly developed.

Back-up slides

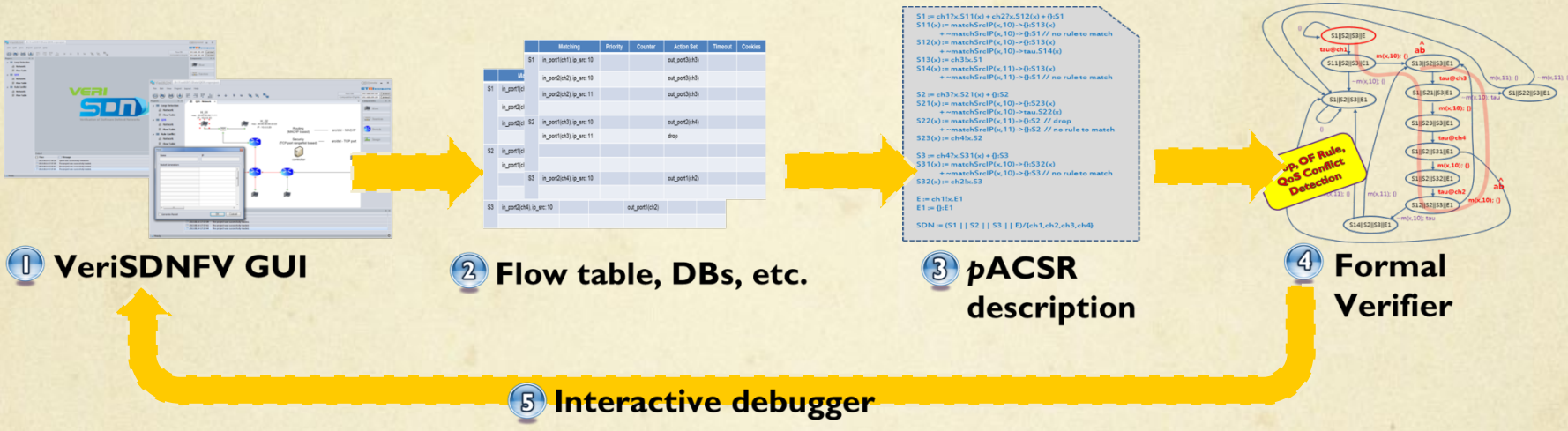
Our Approach – Formalism



Network Formalism A formal description is a specification expressed in a language whose semantics are formally defined, as well as vocabulary and syntax. The need for a formal semantic definition means that the specification language must be based on logic, mathematics, etc., not natural languages.

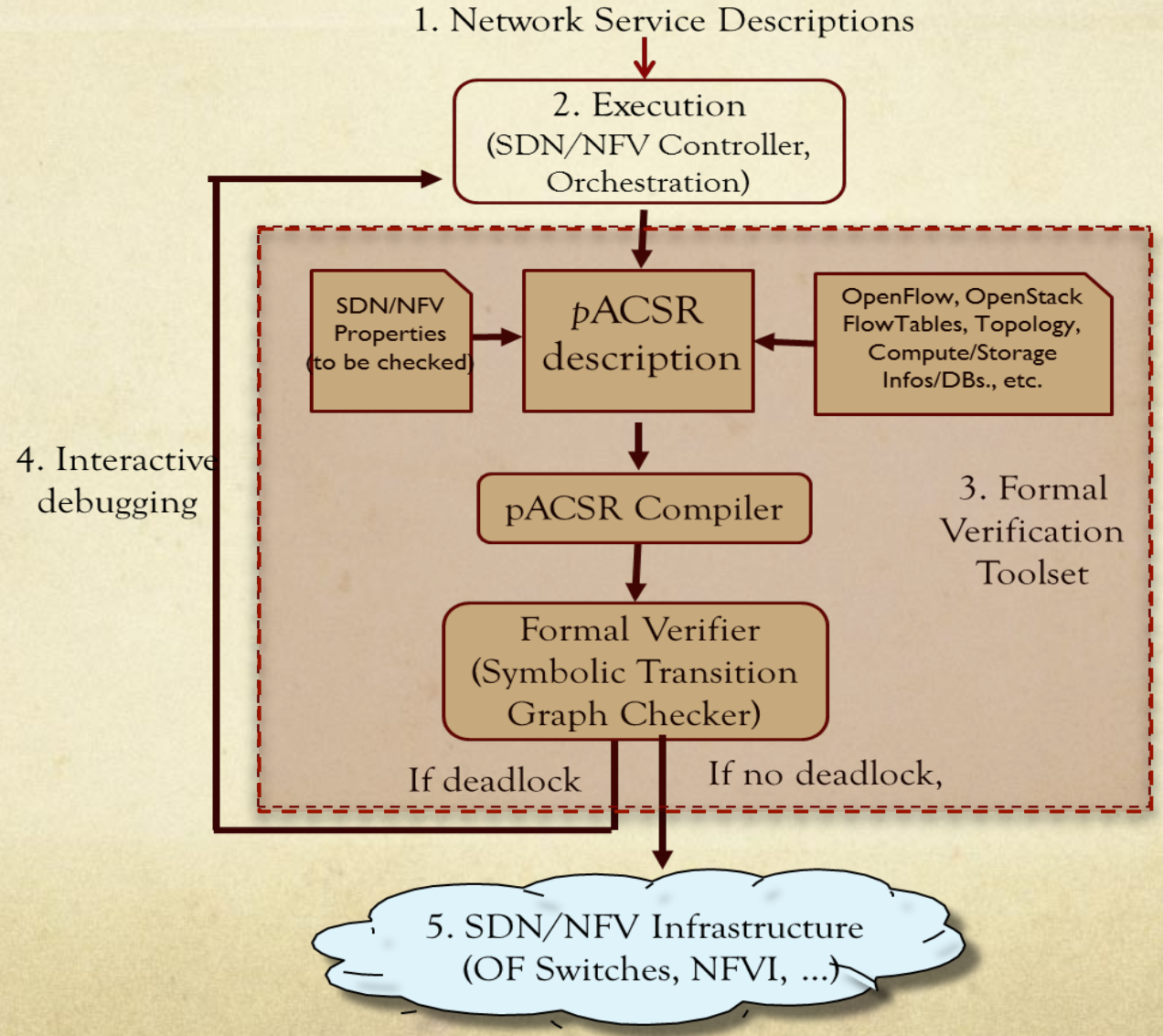
Formal verification The act of proving or disproving the correctness of designs or implementations with respect to requirements and properties with which they must satisfy, using the formal methods or techniques

Our Verification Theory and Tool Set (VeriSDNFV)



The goal is to provide a formal foundation for verification of SDN/NFV-enabled services.

VeriSDNFV Tool Set



pACSR (Process Algebra Lang.)

$$P \coloneqq NIL \mid A{:}P \mid e.P \mid be \rightarrow P \mid P_1 + P_2 \mid P_1 \| P_2 \mid [P]_I \mid P\backslash F \mid P\backslash\backslash I \mid C(\vec{x})$$

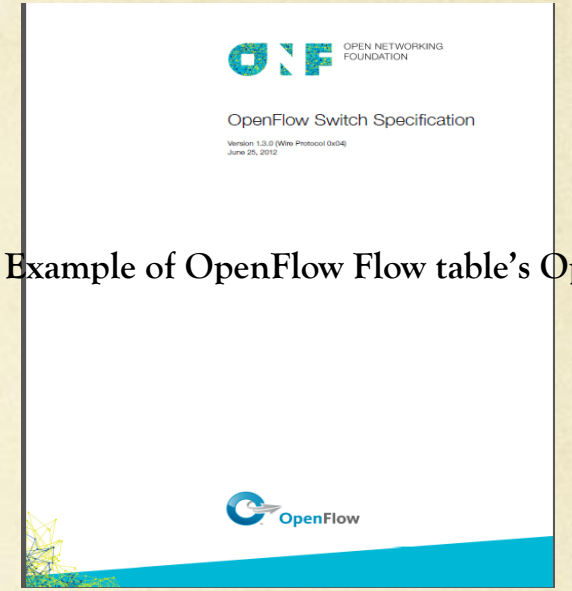
$$e \coloneqq (\tau, ve) \mid (l?, ve) \mid (l!, ve) \mid (l?x, ve) \mid (l!ve_1, ve)$$

$$A \coloneqq \varnothing \mid \{S\}$$

$$S \coloneqq (r, ve) \mid (r, ve), S$$

Syntax	Description
NIL	A process that executes no action(i.e., it is deadlocked)
A:P	To execute a timed, resource-consuming action A, consume one ti me unit, and proceed to the process P.
e,P	To execute the instantaneous event e, and proceed to P.
P1+P2	A non-deterministic choice between P1 and P2 which is subject t o priority arbitration.
P1||P2	The parallel composition of P1 and P2 in which the events from P 1 and P2 synchronize or interleave while the timed actions from P 1 and P2, if they do not share any resource, reflect the synchrono us passage of time.
[P]I	The close operator, to produce a process that uses the resources in I exclusively.
P\F	The restriction operator, to restricts events with labels in F from executing.
P\\I	To represent the behavior of P in which the identities of resource s in I are concealed.
C	A process constant with a certain arity. Each process constant C with arity n is associated with a process definition of the form $C(x) \stackrel{\text{def}}{=} P$, where x is a vector of n variables.
be->P	A conditional process term which behaves like P if the boolean ex pression be evaluates to true, or NIL if be is false.

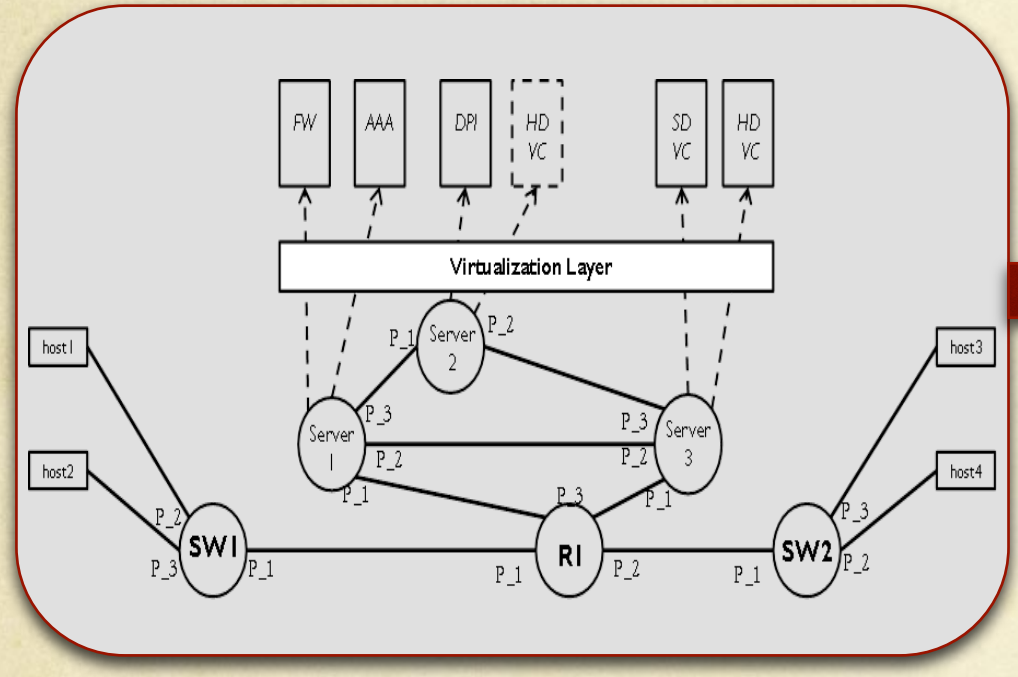
An Example of OpenFlow Flow table's Operations



P(x) := if matchSrcIP(x,source_IP)→ch!x.nil
S := ch?y.nil
Sys := (P(x) || S)/{ch}
….

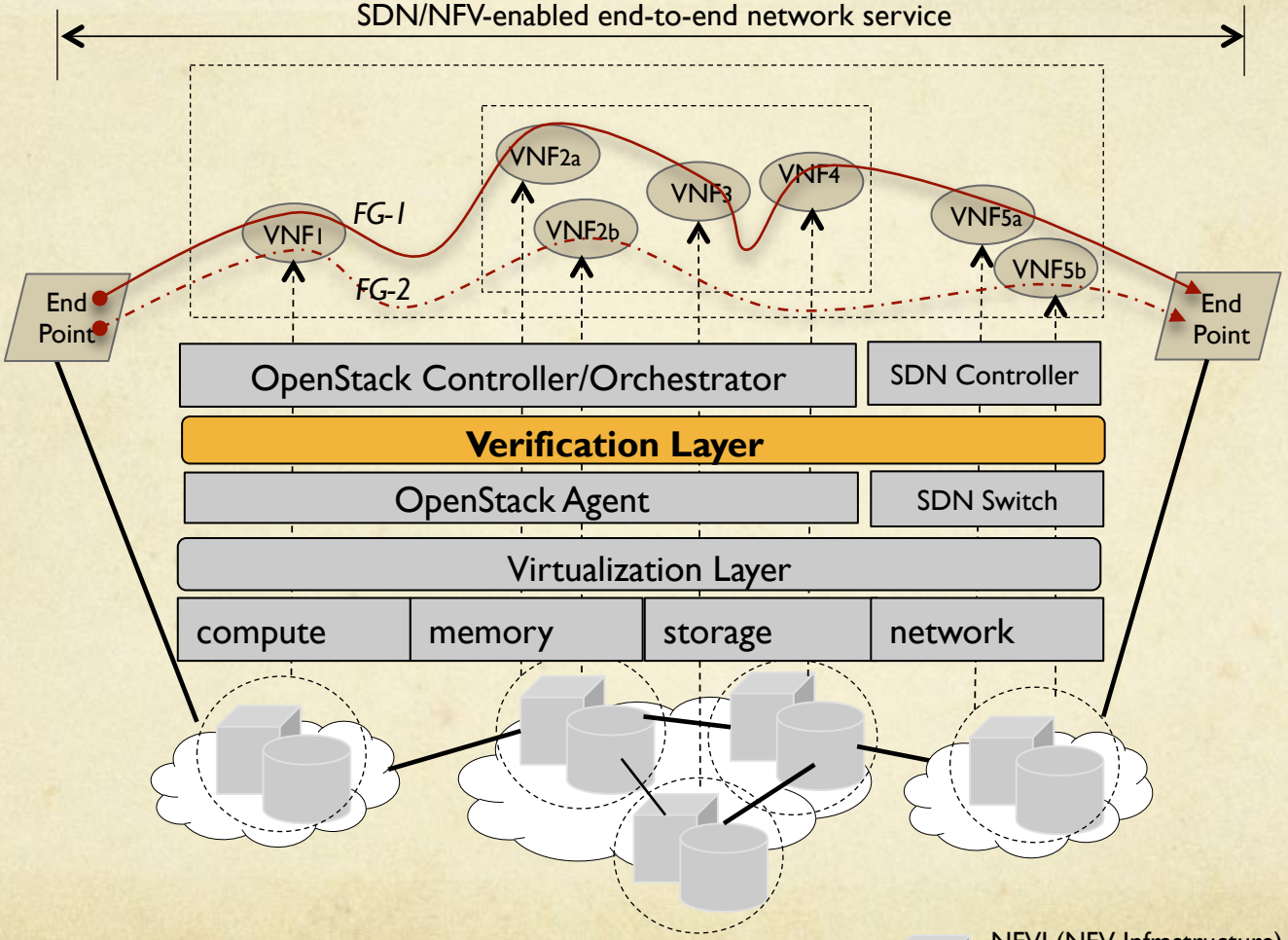
pACSR (Process Algebra Lang.)

An Example of Service Functions Chains
(End-to-end Virtual Content Delivery Service)



NFV(x) := ServiceChain || PacketFlow(x)
ServiceChain := {}:HDVC_S3?.ServiceChain +*
replicated?.ServiceChain_R
ServiceChain_R := {}:HDVC_R_S2?.ServiceChain_R +*
replicated?.ServiceChain_R
PacketFlow(x) := {}:{PACKETFLOW}:SW1_P3(x)*
SW1_P3(x) := {}:if SW1_P3_C1(x)*
→ {SW1_P1}:R1_P1(x)
else IDLE
SW2_P1(x) := {}:if SW2_P1_C1(x)*
→ {SW2_P2}:Host3(x)
else if SW2_P1_C2(x)
→ {SW2_P3}:Host4(x)
else IDLE
SW2_P2(x) := {}:if SW2_P2_C1(x)*
→ {SW2_P1}:R1_P1(x)
else IDLE
SW2_P3(x) := {}:if SW2_P3_C1(x)*
→ {SW2_P1}:R1_P1(x)
else IDLE
.....

End-to-end SDN/NFV Services



Related Works

Product/Project	Goals
VeriFlow	To verify network-wide invariants in real time (Data plane verification, no formal technique)
NICE	To test/debug OpenFlow applications (Debugger for simplified OF-based models)
Frenetic	To develop network programming languages with Rigoro us semantic foundations
Abstraction for Network Update	To develops a formal model of OpenFlow networks
Header Space Analysis (HSA)	To allows to statically check network specifications and co nfigurations (e.g., reachability failures, forwarding loops a nd traffic isolation and leakage problems)