Verifying Properties of SDN

Vasileios Klimis, Bernhard Reus, and George Parisis

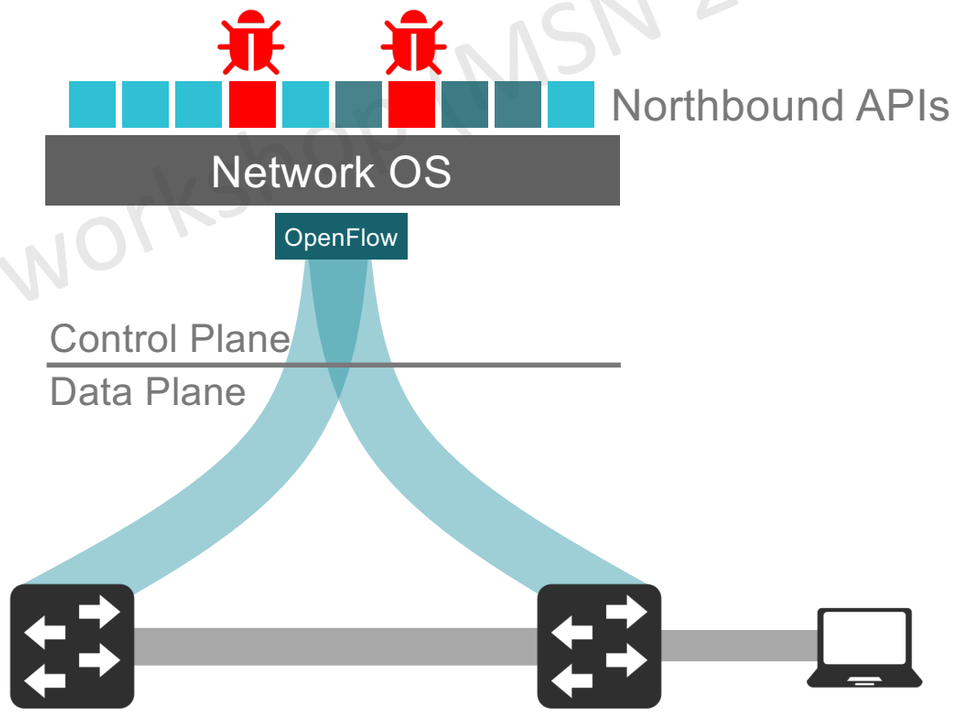
UNIVERSITY OF SUSSEX

Art vs. Science

- Debugging a network remains a manual, largely unsystematic process
- Expensive bugs that might show up rarely are becoming increasingly hard to debug

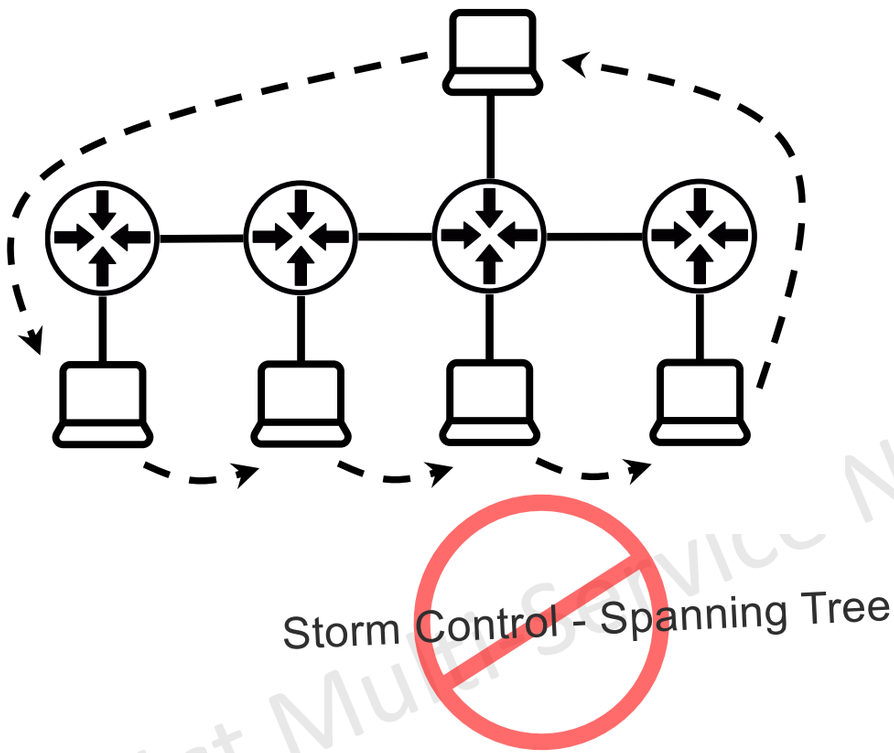
Why SDN Verification?

- Complex, Concurrent, Async, Distributed
- Softwarisation
- Open-Sourcing
- Third-party APIs



Art vs. Science

Example: Loops in an acyclic topology



No ‘magic bullet’ to debug/verify loops conventionally

Property:

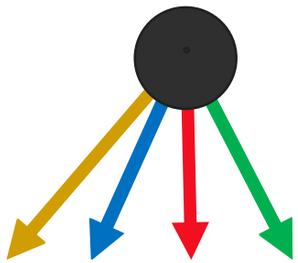
“No packet should loop in my network”

Algorithmic Verification & Debugging

reactive, incomplete, manual

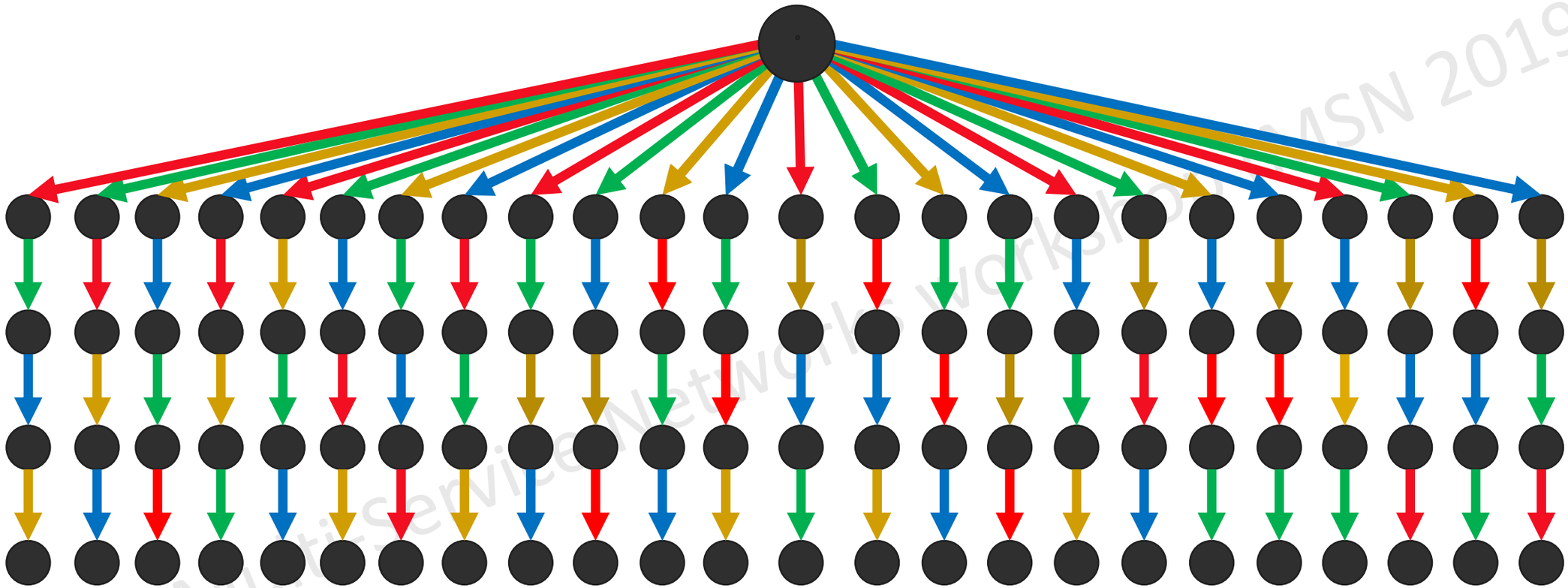
proactive, exhaustive, automatic

Scalability Challenges of verifying SDN



```
match(pkt1)
nomatch(pkt2)
install(rule1)
match(pkt3)
```

Scalability Challenges of verifying SDN



4! interleaved orderings

Considering all possible event orderings leads to exponential explosion in the state space.

Scalability Challenges of verifying SDN

Applying exhaustive analysis to networks even worsen the situation

Significant amount of state	Unboundedly many packets
	Unbounded number of events due to highly dynamic network state changes
	Unbounded interleaved orderings over events

Contributions

- Synthesise the independency and invisibility constraints
- Soundness proofs of the reduction
- A unified/compact data structure:
  - all buffers are merged into two central views (Packet/Rule view), encoded as metadata
- Canonicalise the form of Data Structure
- Data packed in bit-vectors

Previous Work

Kuai [1]

threads hard-wirings

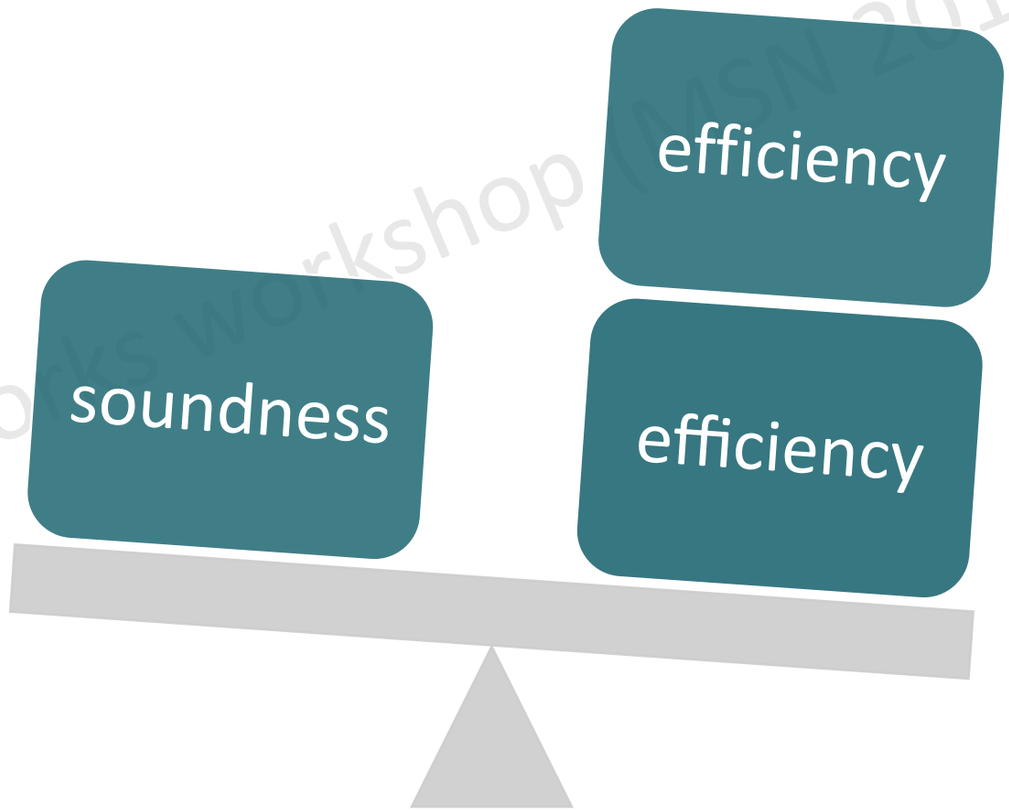
reducing nondeterminism

restrictive Spec Lang

imposing restrictions on what a property is allowed to assert about

strong independencies

context-unaware

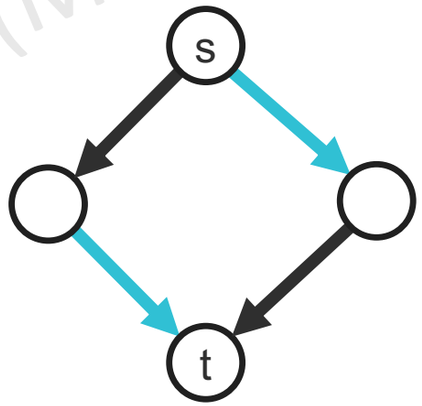


[1] R. Majumdar, S. Deep Tetali, and Z. Wang, “Kuai: A model checker for software-defined networks,” in 2014 Formal Methods in Computer-Aided Design (FMCAD). IEEE, oct 2014,

Context-Sensitive POR & SDN

Our Proposal

This work considers the problem of verifying SDN using:



Pragmatic Representation for SDN

↳ adding realism and representativeness to the model

Partial Order Reduction technique

relying on a context-sensitive notion of independence between transitions

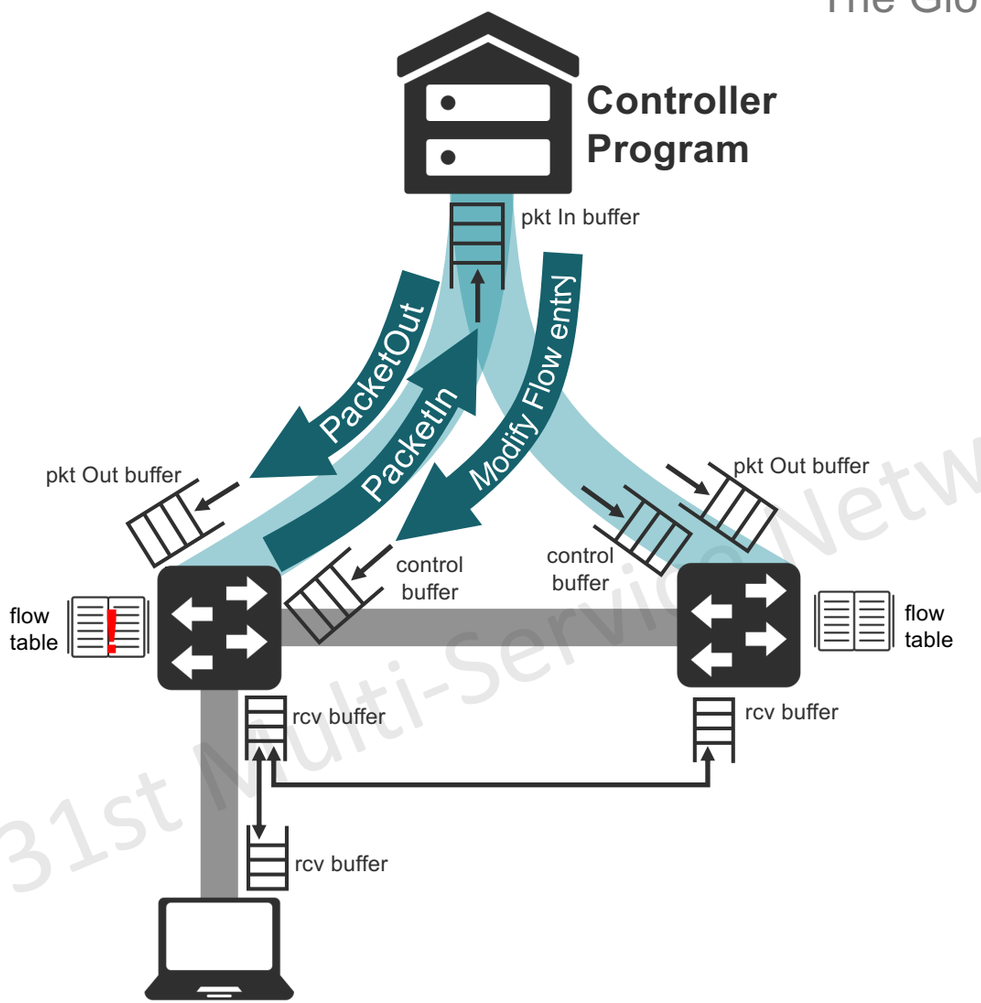
↳ pruning the state-space

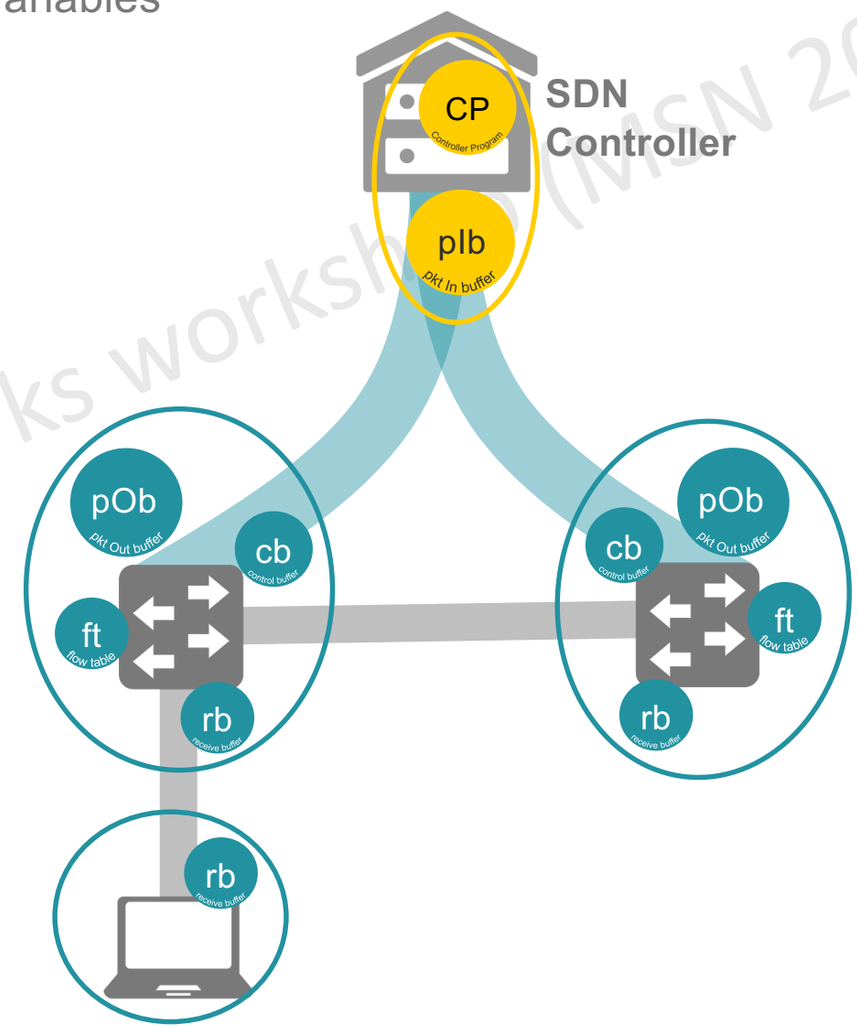
Two actions are Independent (commutable) == the combined execution of them has the same effect under different interleaved orderings.

POR reduces the size of the state-space to be checked by a model checker, exploiting the commutativity of concurrently executed transitions, which result in the same state when executed in different orders.

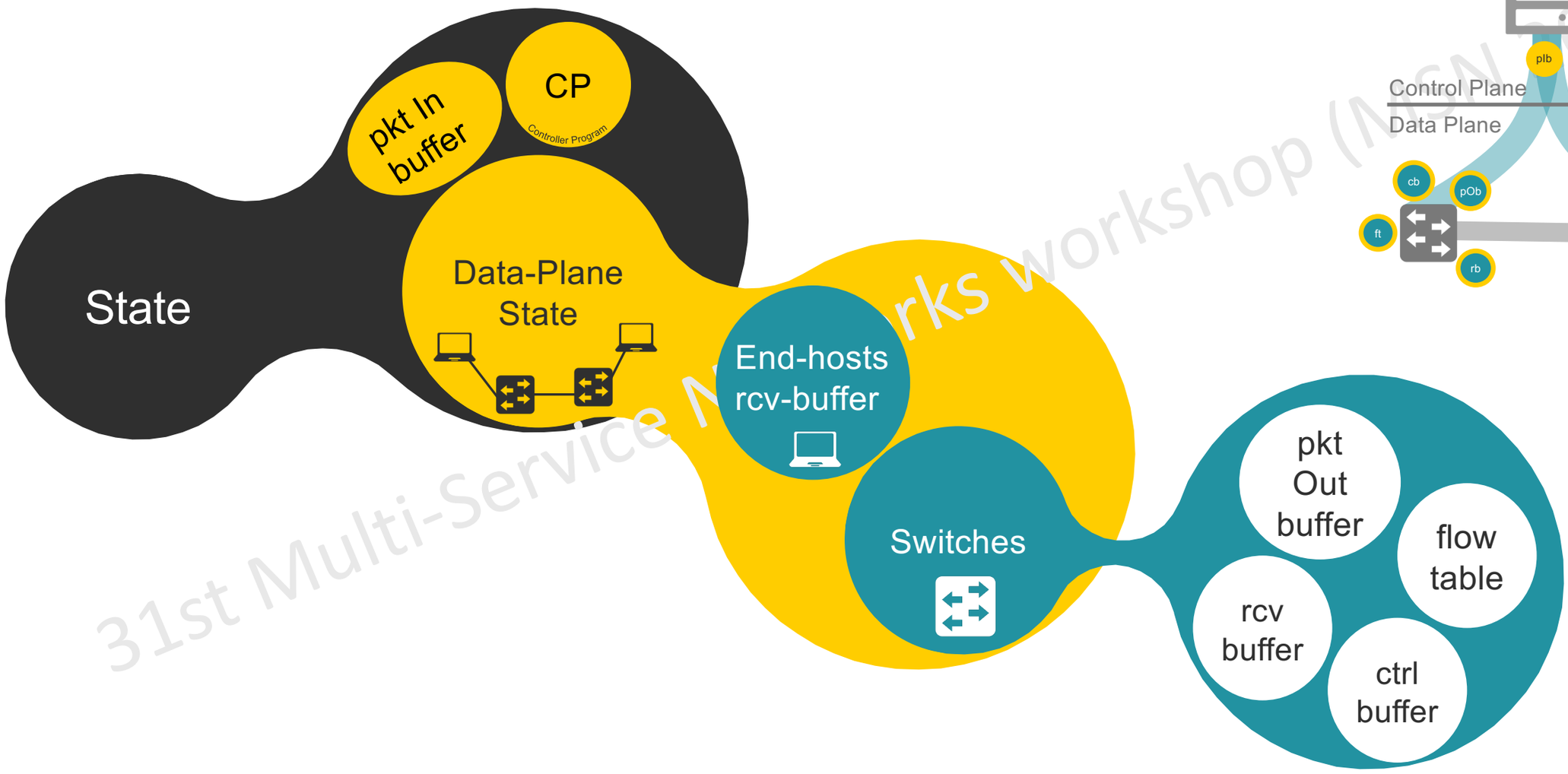
SDN as a state machine

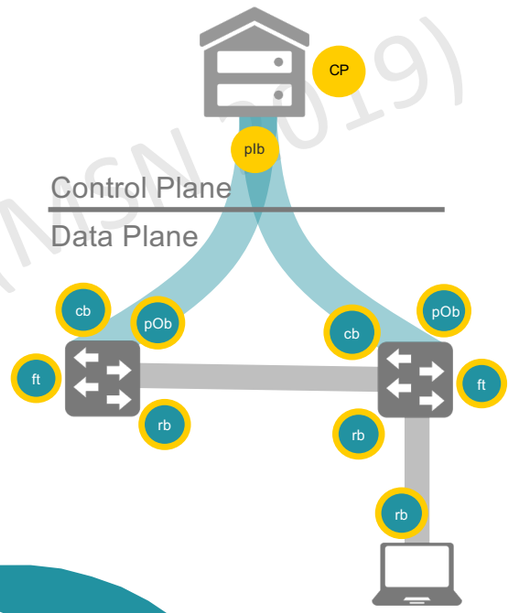
The Global Variables



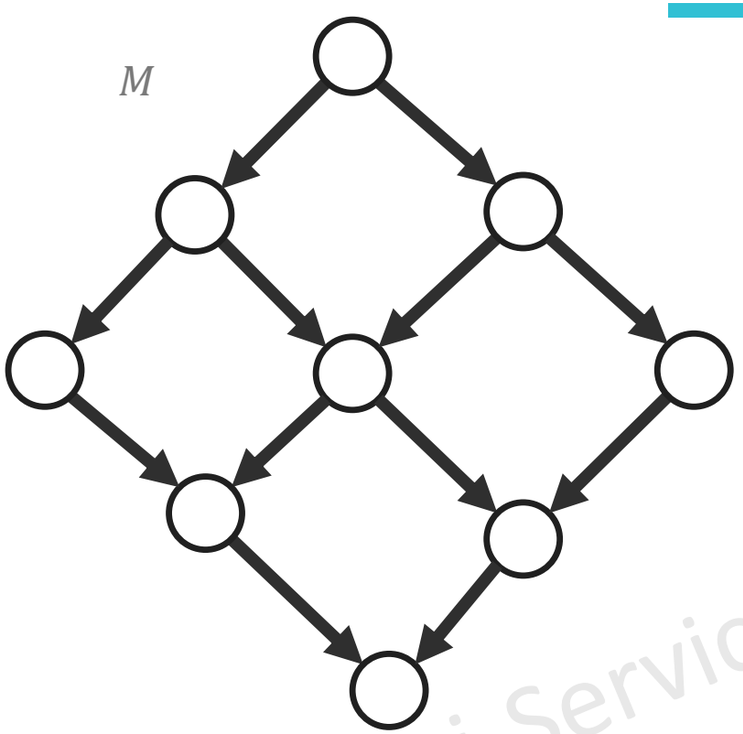


The State of the System





Decision Procedure



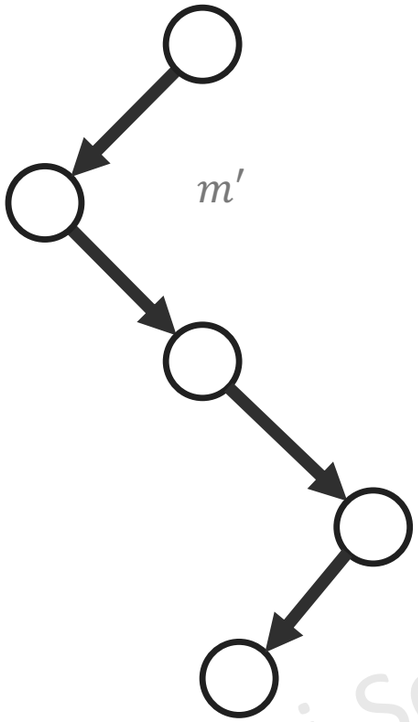
Property Specification

LTL - Formula $\neg\varphi$

STATIC PRE – ANALYSIS

- Abstract away the details irrelevant to property
- Precompute the set of the silent actions against the contexts
- Canonicalise the form DS by automatically pre-computing the minimum covering data set

Decision Procedure

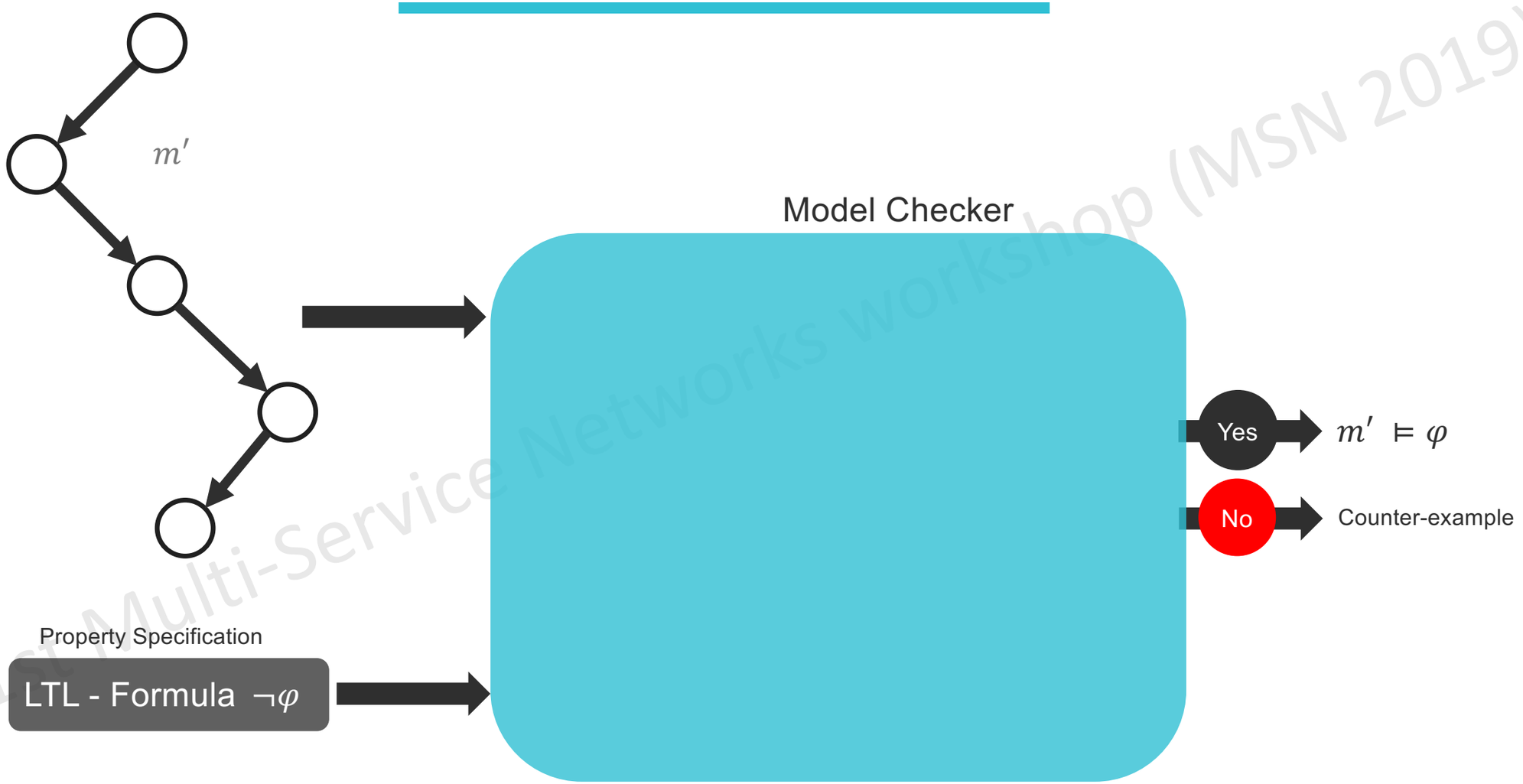


A reduced, more abstract graph, which is a homomorphic image of the initial one:
Both models fulfil the same observable property (Proofs)

Property Specification

LTL - Formula $\neg\varphi$

Decision Procedure



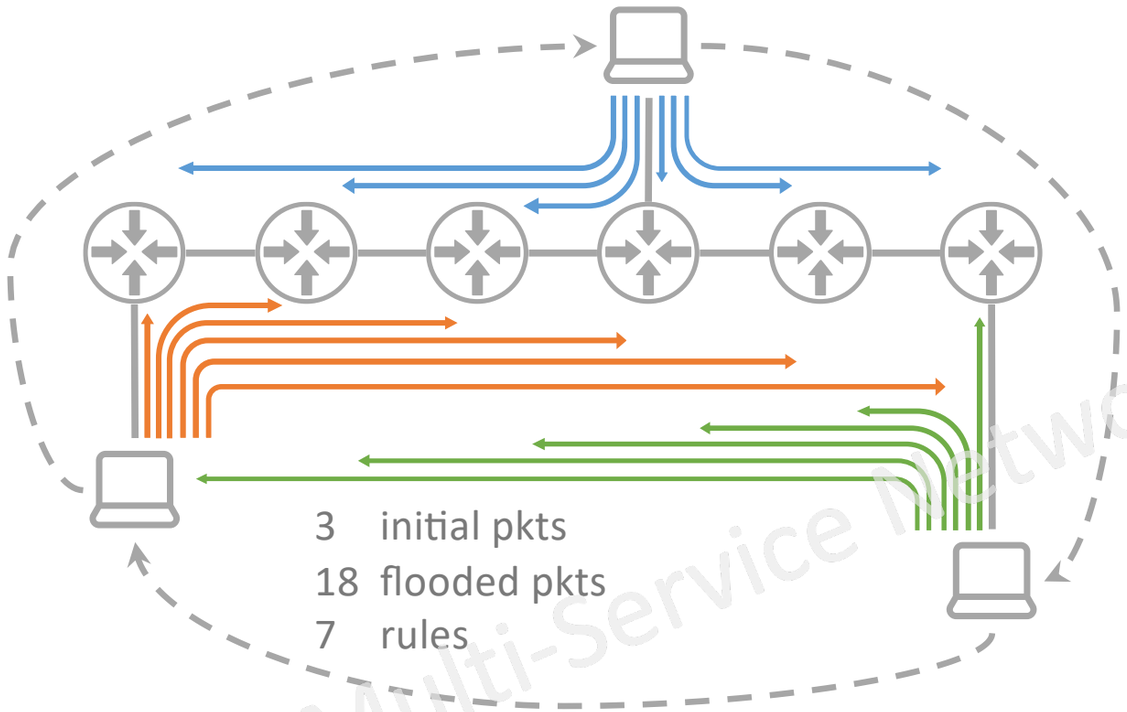
Implementation

CP and buffers modelled in UPPAAL as concurrent processes

The basic underlying semantic domain of UPPAAL: timed automata.

MAC-Learning switch

Evaluation Highlights



	w/o POR	w/ POR
States	Did not terminate within	2,978,131
Time	2 days	13 min

	struct type	bit-packing
Memory	7.9 GiB	0.6 GiB
State size	2.8 KiB	0.2 KiB

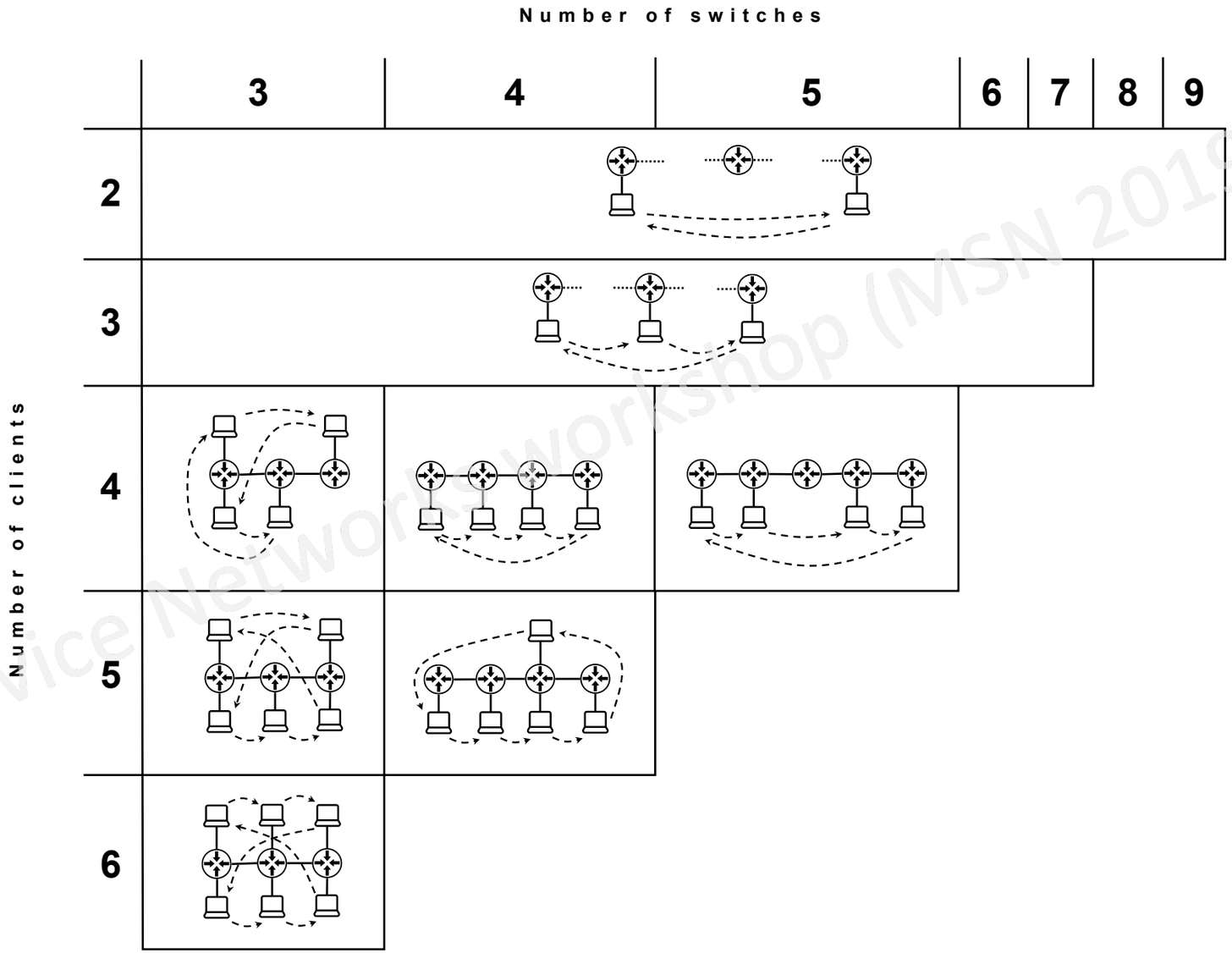
Property:

"No packet should loop in my network"

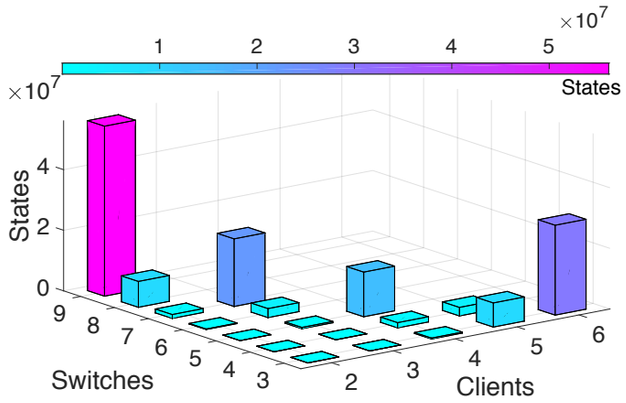
$\Box \forall s \in SW \ \forall p \in Packet \ . \ \neg(p.reached[s] \ \wedge \ p \in s.pb)$

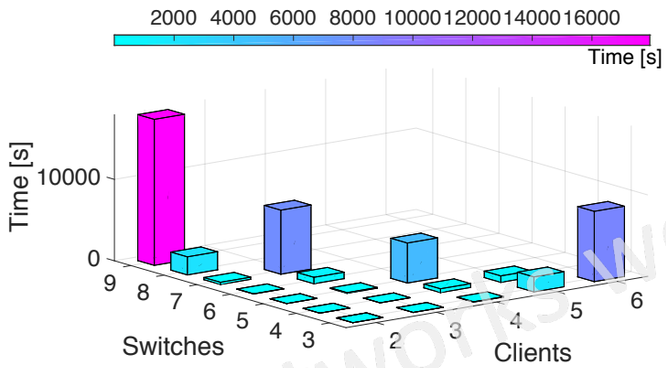
MAC-Learning switch

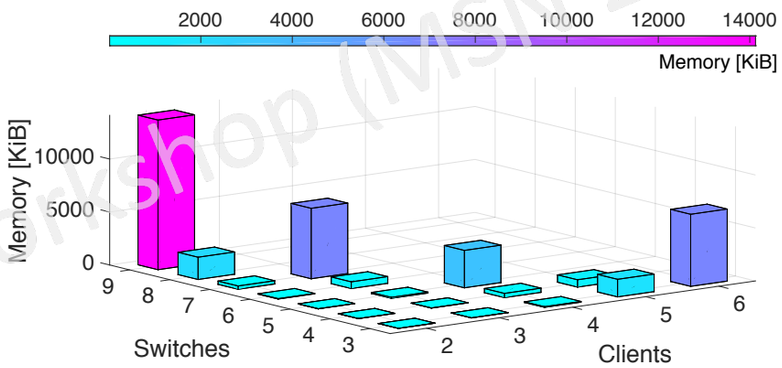
Different acyclic topology settings for verifying absence of loops using the Pyswitch MAC address learning switch application.

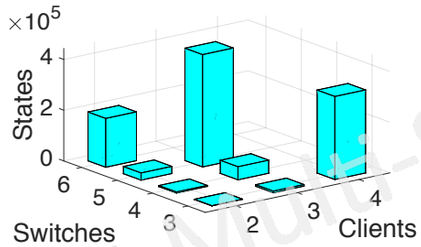


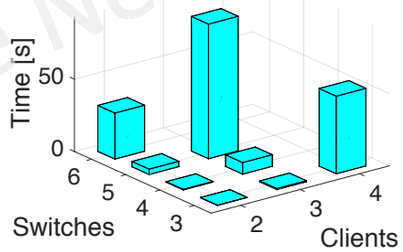
MAC-Learning switch

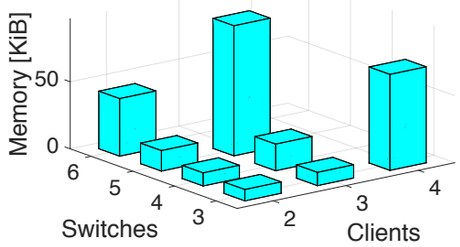












Thanks