# Security automation using Traffic Flow modeling

**Simone Bussa**, Riccardo Sisto, Fulvio Valenza

**Politecnico di Torino**

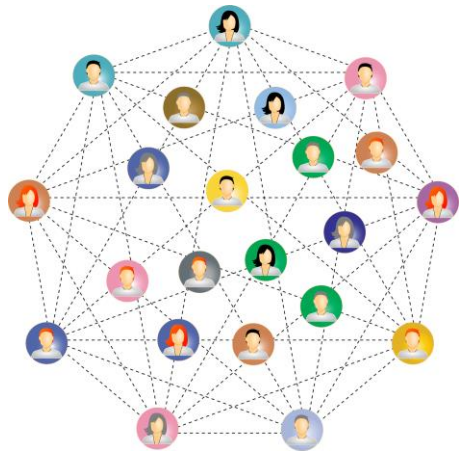
July 01, 2022 – IEEE NetSoft: International Conference on Network Softwarization

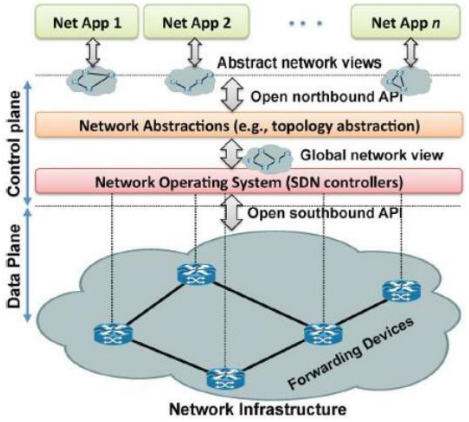
netgroup

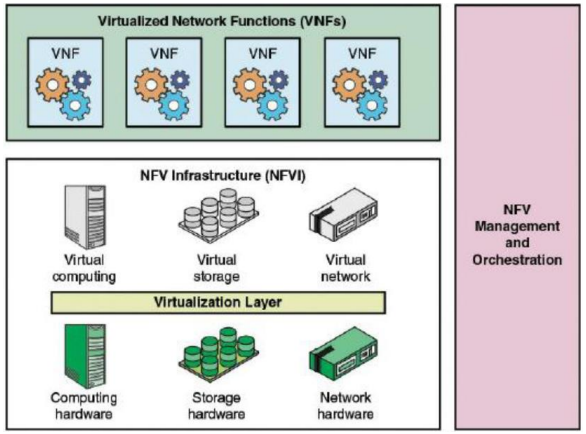
1859

# Scenario

- **Computer networks** nowadays:
  - Constantly increasing in size and importance
  - Flexibility and dynamism in network creation and configuration
  - Manual configuration time consuming and error-prone
  - Cyber-threats






# Scenario

- In literature, effort to find automatic tools to verify and configure the network security functions
- **Policy Verification**: checking whether a set of security policies is correctly implemented in our system
- **Policy Refinement**: automatically configure the security functions of our network, based on a set of high-level policies
- Our work aims to be a contribution to solve these problems

# Scenario

- Problems:
  - Solutions MUST scale and consider the complexity of real networks
  - Complexity of solving these problems is strictly dependent on how the traffic of the network is modeled
  - Forecast a priori the behavior of the network

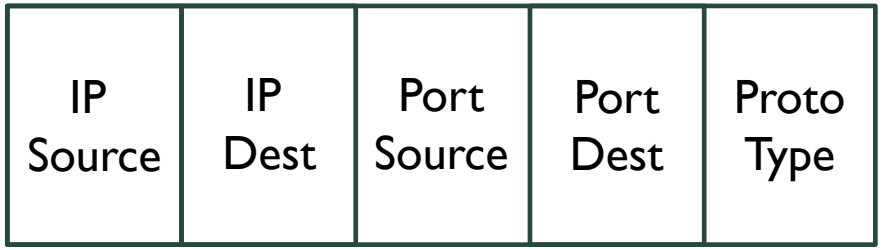
Efficient modeling of network traffic and network functions

# Paper objectives

- Two novel (and different) approaches to model the network traffic
  - Atomic Flows and Maximal Flows
  - Two ways to represent, identify and aggregate all the packets of the network

# Predicates

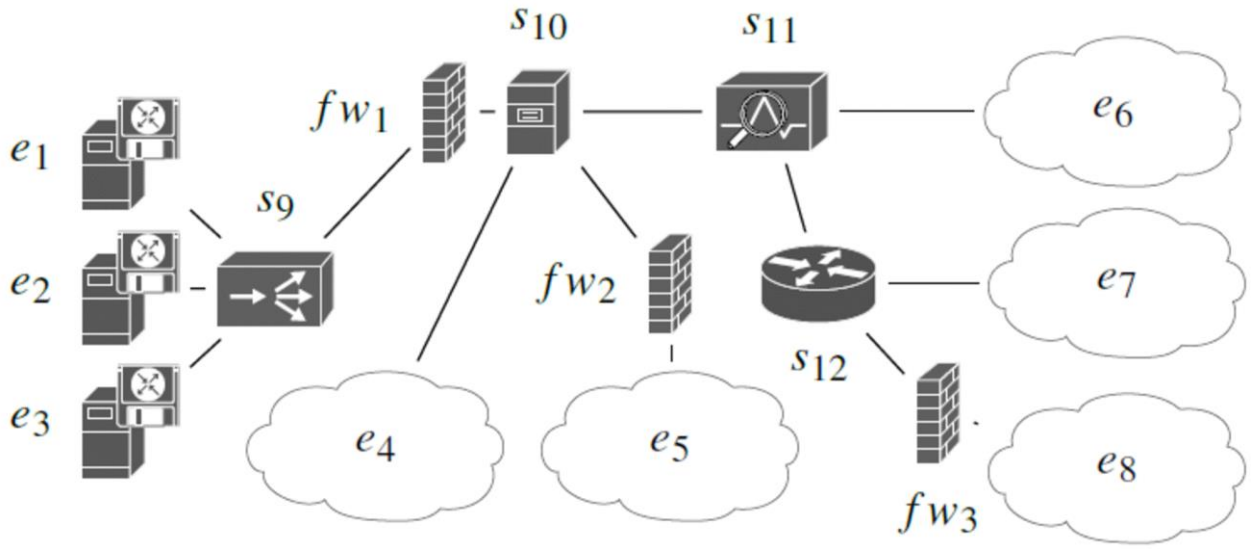
- Packets are forwarded and transformed based on the content of their header
- We represent the header of a packet as a **Predicate**
- Packets represented by the same Predicate behave in the same way
- BDD, Tuple Representation, Wildcard expressions




IP quintuple

# Traffic Flows

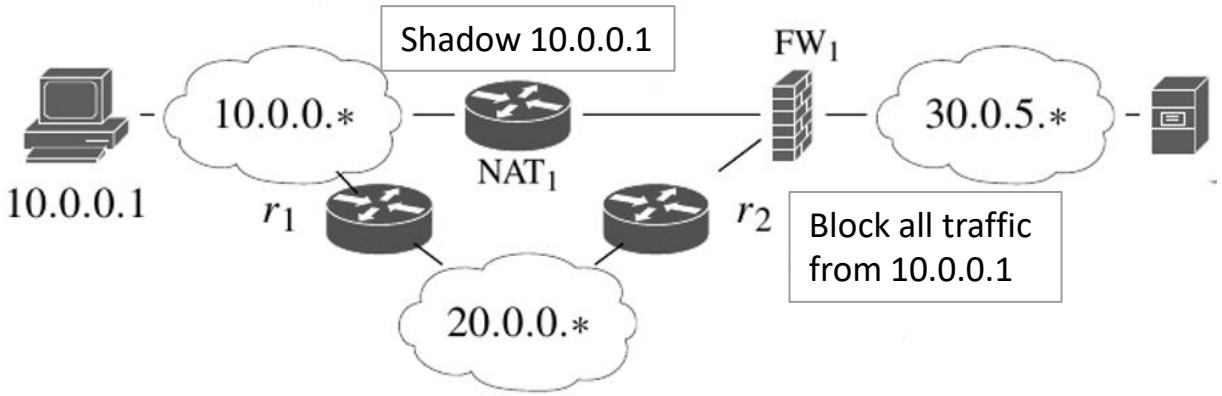
- Lists of alternating nodes and Predicates
- Describe the evolution of a packet along a certain path in the network




Problem: how to characterize and compute the flows

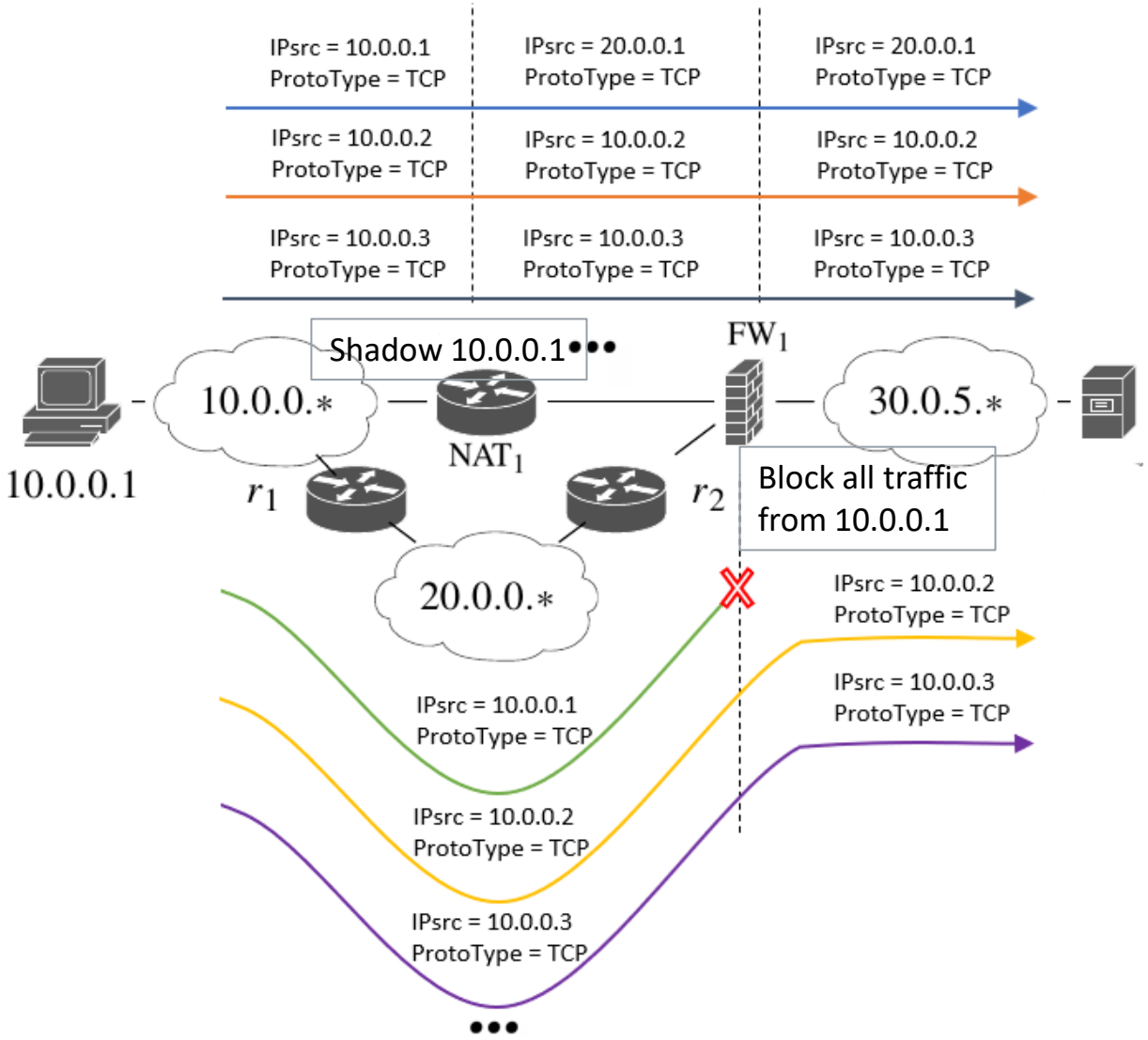
Two proposed solutions

# Network Security Policies




- **Security Policies** to limit the cardinality of Traffic Flows set
- Sources and destinations of the Flows chosen according to **Security Policies**
- Example, “Block all TCP traffic from 10.0.0.0/24 to 30.0.5.1”

# Network Security Policies



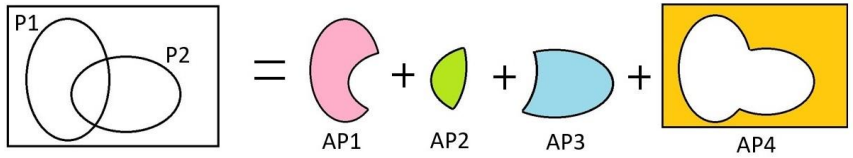

- **Security Policies** to limit the cardinality of Traffic Flows set
- Sources and destinations of the Flows chosen according to **Security Policies**
- Example, “Block all TCP traffic from 10.0.0.0/24 to 30.0.5.1”
- Complexity is in the number of generated flows and Predicate representation

# Approach

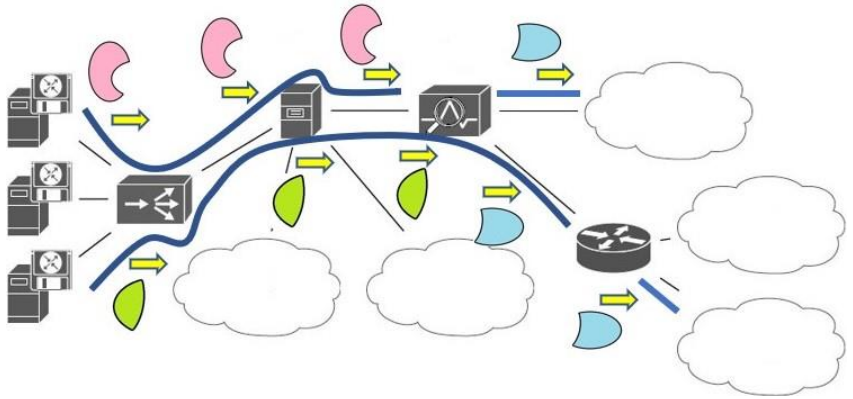
- **Atomic Flows**: simplify Predicate representation (atomic predicates), but lead to a greater number of “simple” flows
- **Maximal Flows**: reduce the number of flows (by aggregating them together), but have complex Predicate representation

# Atomic Flows

- Make use of the concept of **Atomic Predicates**, formalized in 2015 by two researchers (Yang and Lam)
- *«Atomic predicates are the smallest set of disjunct predicates such that each predicate, of the set over which they are computed, can be expressed as a disjunction of a subset of them»*




First phase: Atomic predicates computation

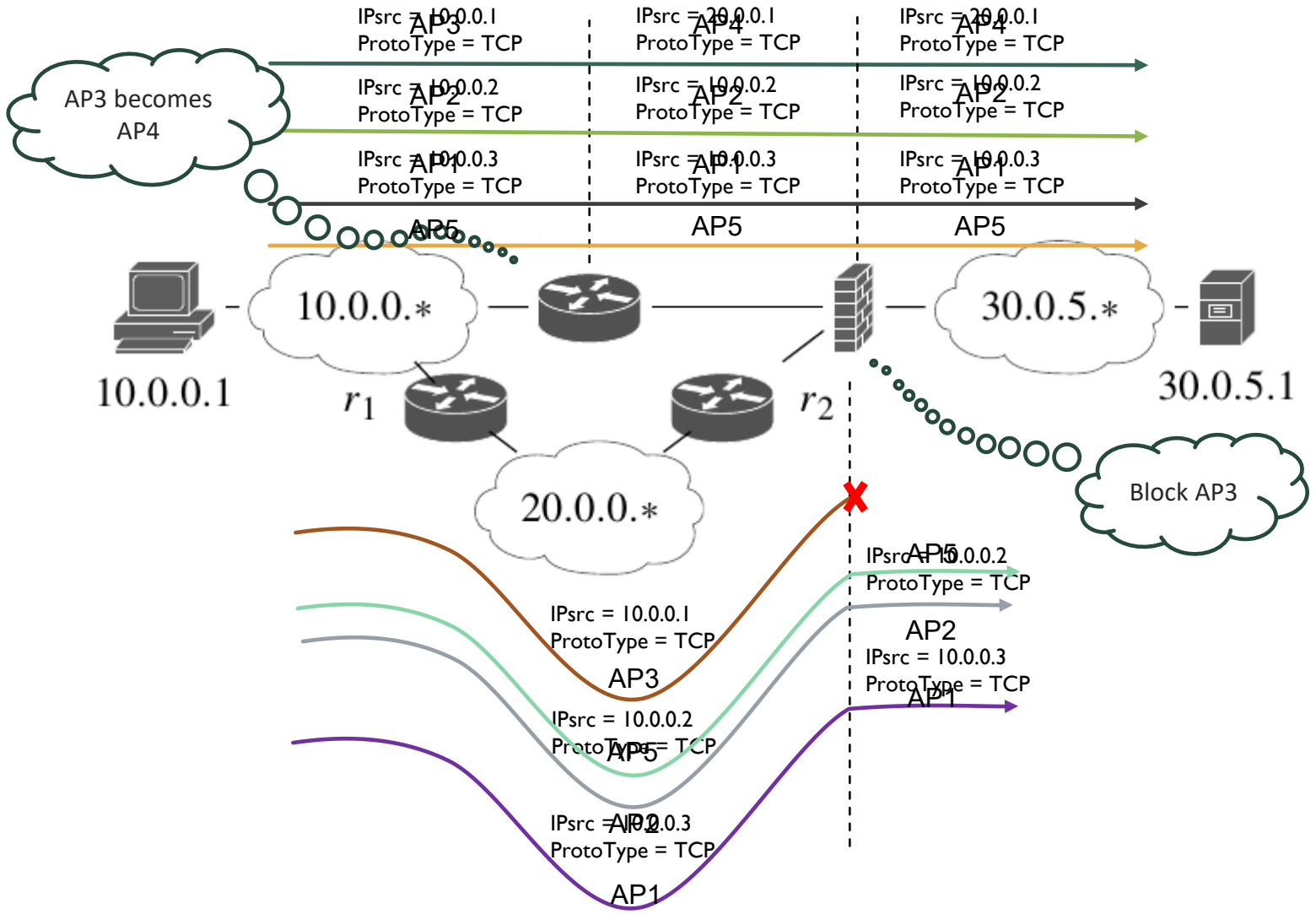


Second phase: Atomic flows computation

# Atomic Flows: main ideas

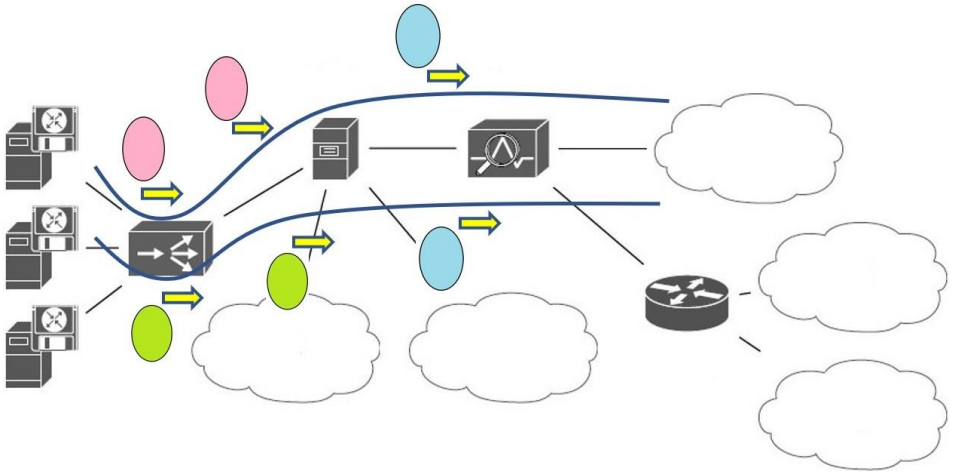
- The goal is to split each Traffic Flows into flows that are as simple as possible and totally disjointed
- Since Atomic Predicates are disjoint and unique, it is possible to represent them with simple integer identifiers
- Atomic Flows are, then, lists of alternating nodes and simple identifiers
- Also, Network Functions become functions working on integers

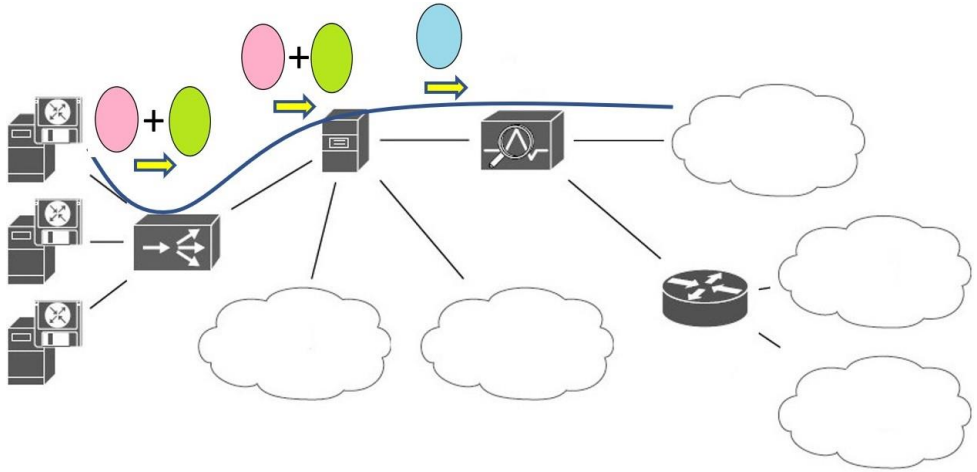
# Atomic Flows: main ideas



# Maximal Flows

- Try to aggregate as many flows as possible into **Maximal Flows**
- All flows represented by the same Maximal flow behave in the same way while crossing the network
- It is sufficient to consider only the Maximal Flow and not each single flow it represents

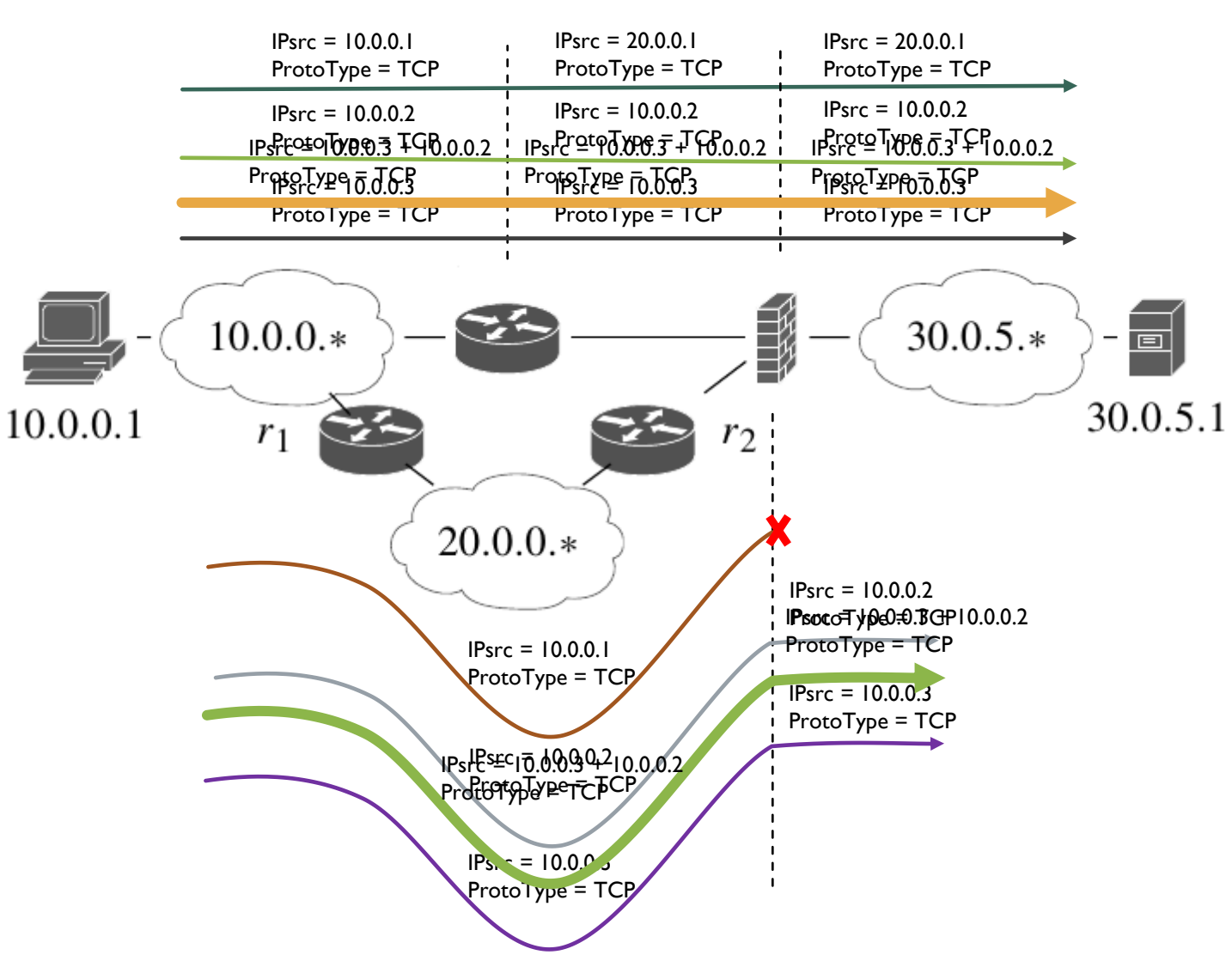




# Maximal Flows: main ideas

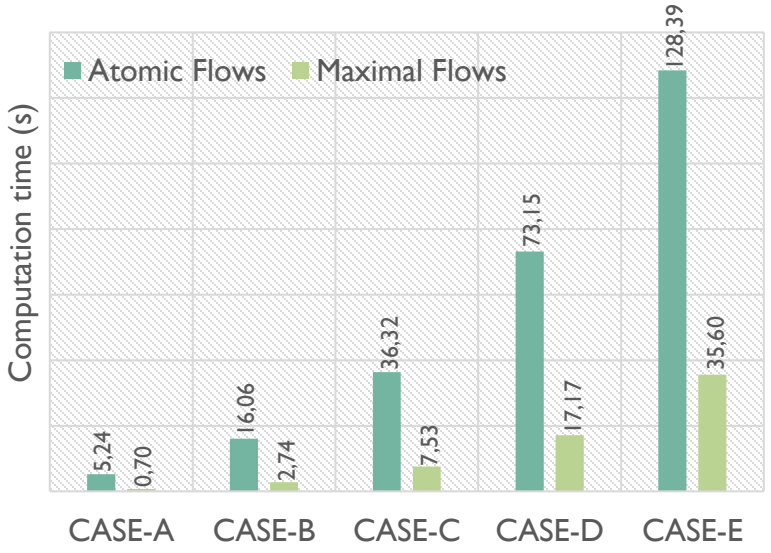
- Predicates of the flow are disjunctions of quintuples
- Predicates, being no longer unique and atomic, cannot be replaced by integer identifiers, but they need an explicit representation (BDD, Wildcards etc.)
- We can consider only the set of Maximal Flows, which is smaller than the set representing all the flows of the network

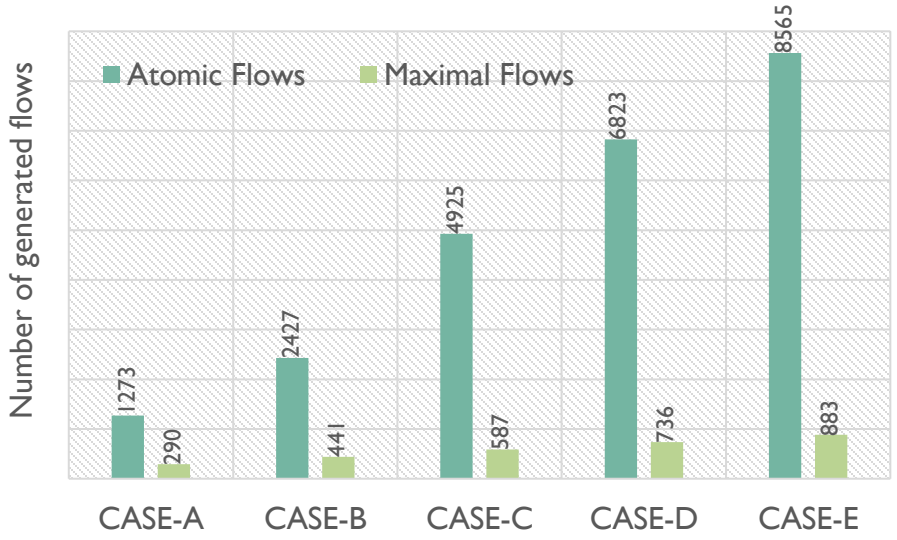
# Maximal Flows: main ideas



# Comparison: Traffic Flows computation

1. Atomic Flows computation is **slower** than the Maximal Flows one
2. Atomic Flows approach generates a **greater number** of flows
3. Any tool using Atomic Flows can work with simple **integers** instead of the explicit representation of each Predicate

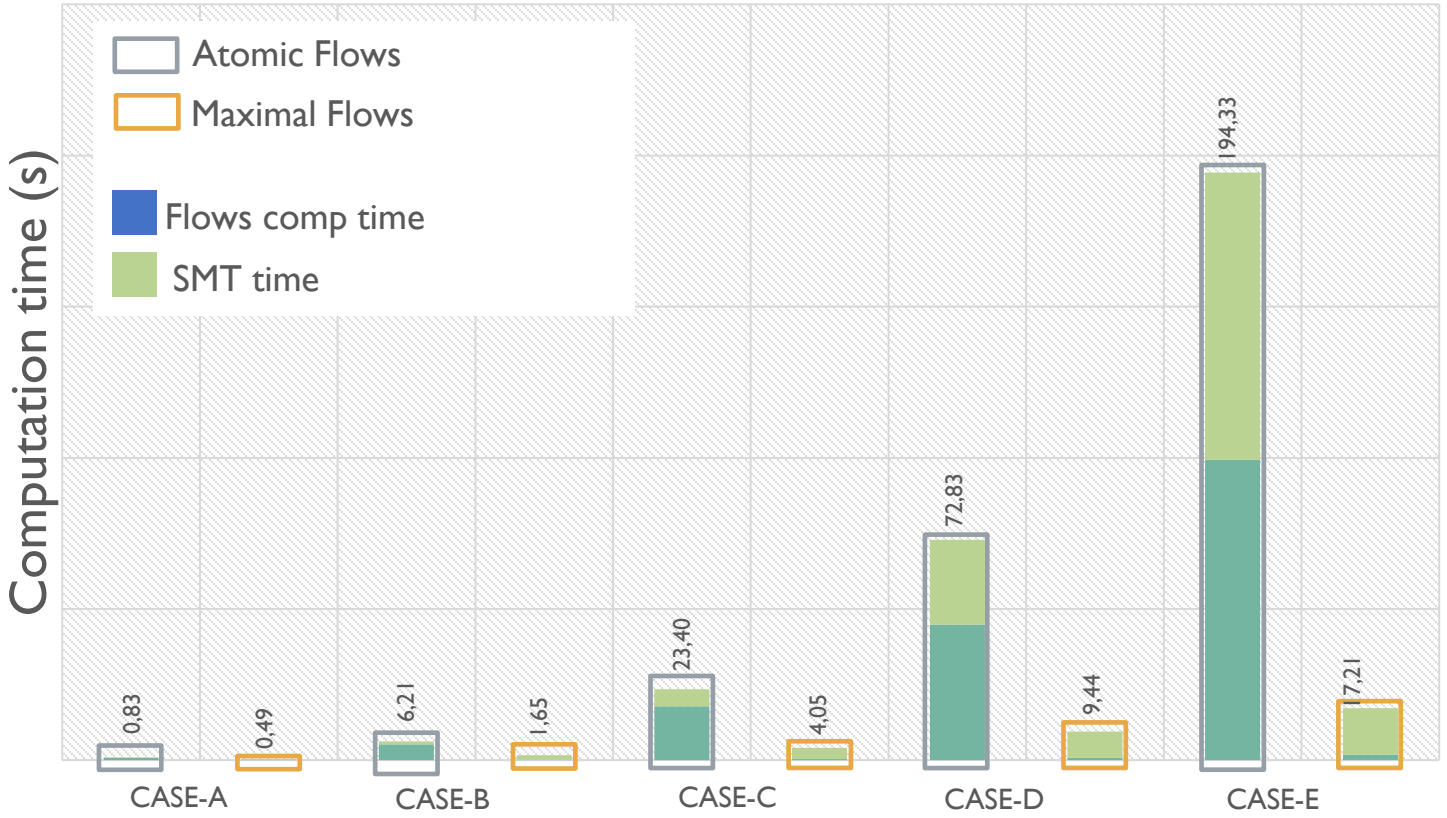


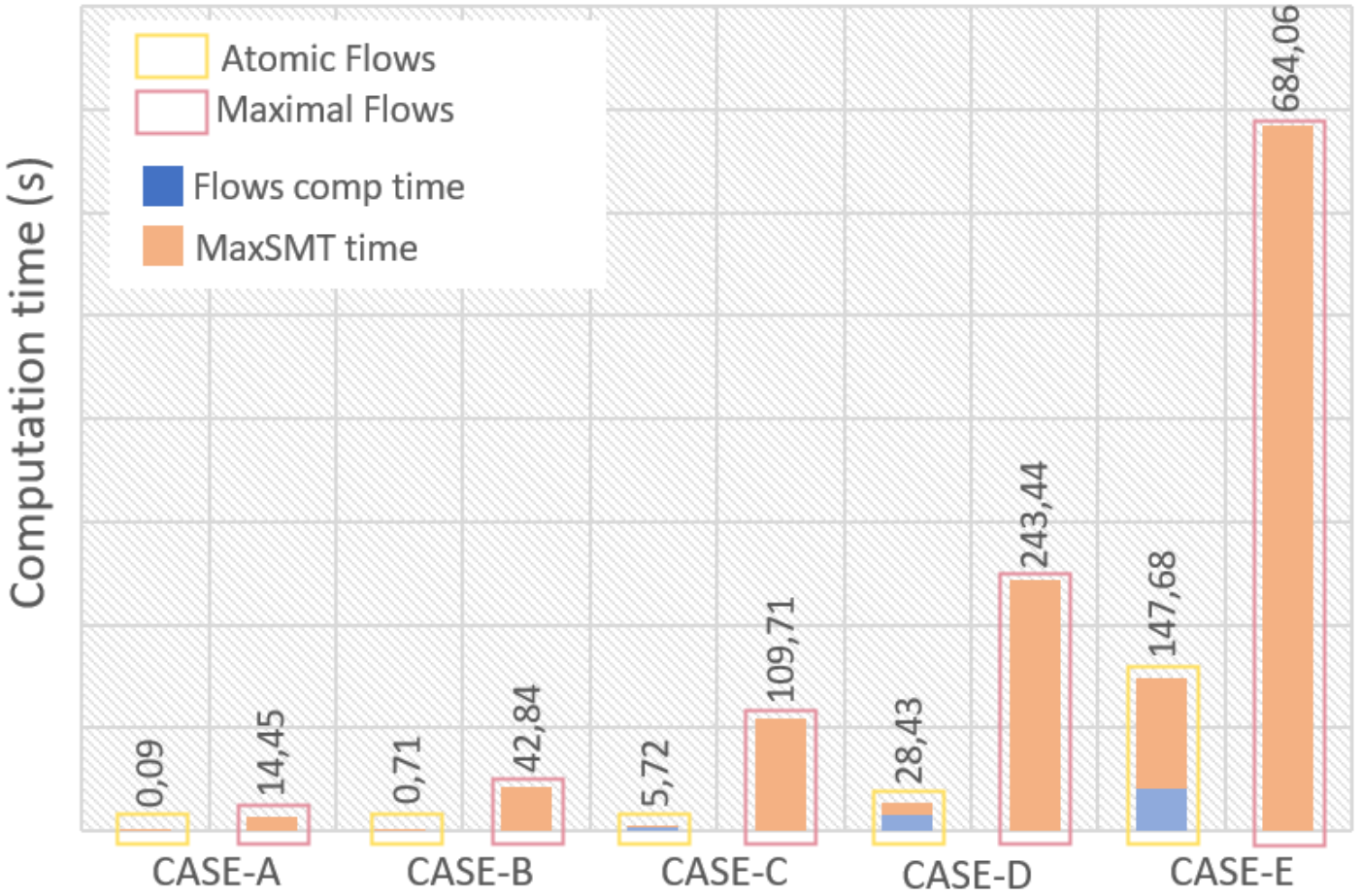
# Comparison: network security management

- Overall efficiency depends on the problem the flows are used to solve
- We consider two Network Problems: Reachability (Verification) and Refinement
- Solved using **Verigraph2.0** and **Verefoo**
- Both frameworks use an SMT solver
- The entire network management process consists of two phases: Traffic Flows computation and SMT resolution

# Results using Verigraph2.0



# Results using Verefoo



# Conclusions

- Achieved results:
  - Definition of **Traffic Flows**
  - **Two approaches** to identify and characterize the flows of the network
  - How computed flows can be used to solve **network management problems**
  - **Comparison** between the two approaches
    - Maximal flows better for verification, atomic flows better for refinement
- Future work
  - **Extend** our analysis to other types of verification/configuration problems


# Thanks for your attention!

Simone Bussa

simone.bussa@polito.it


netgroup