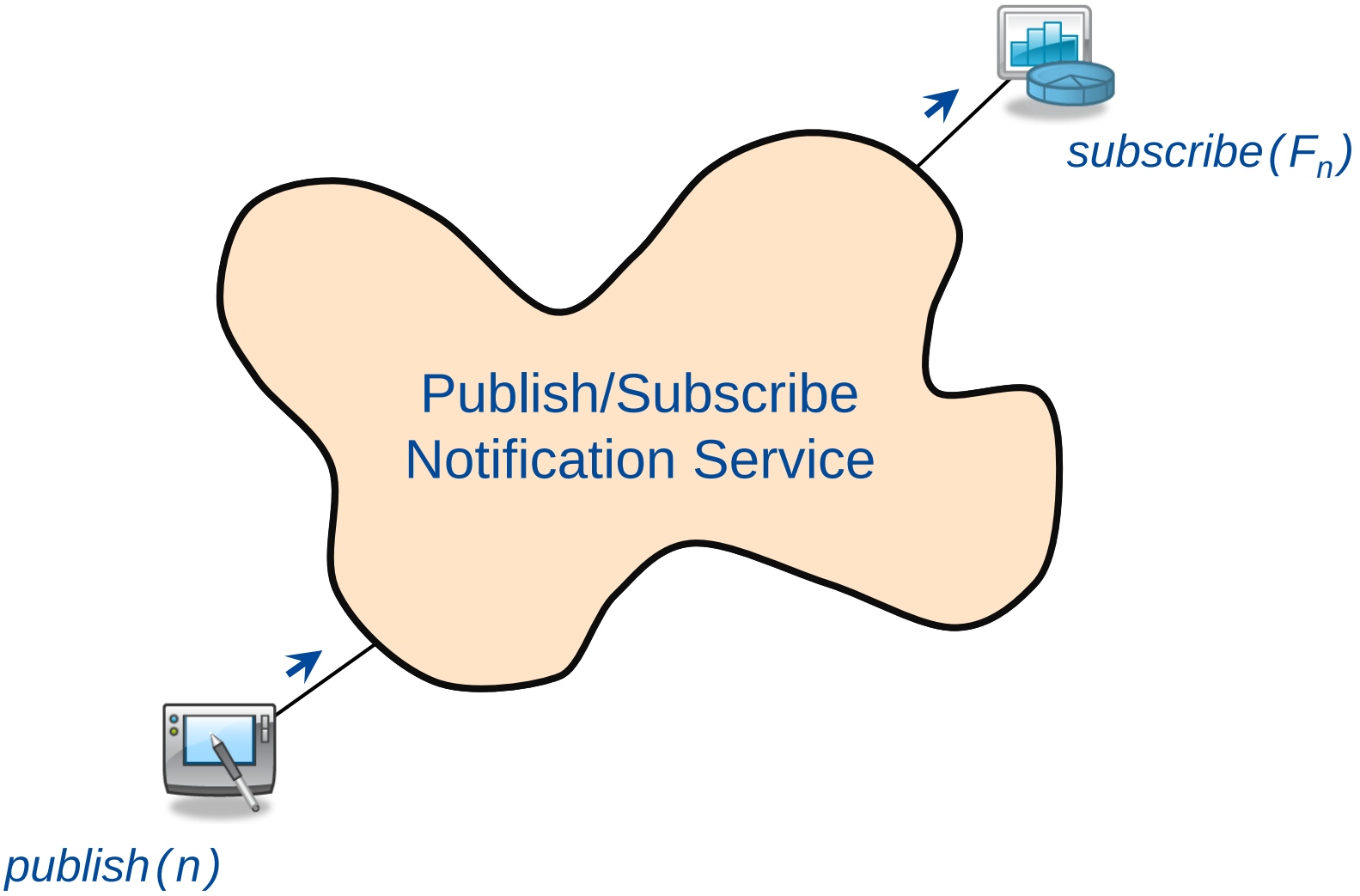
Universität Rostock

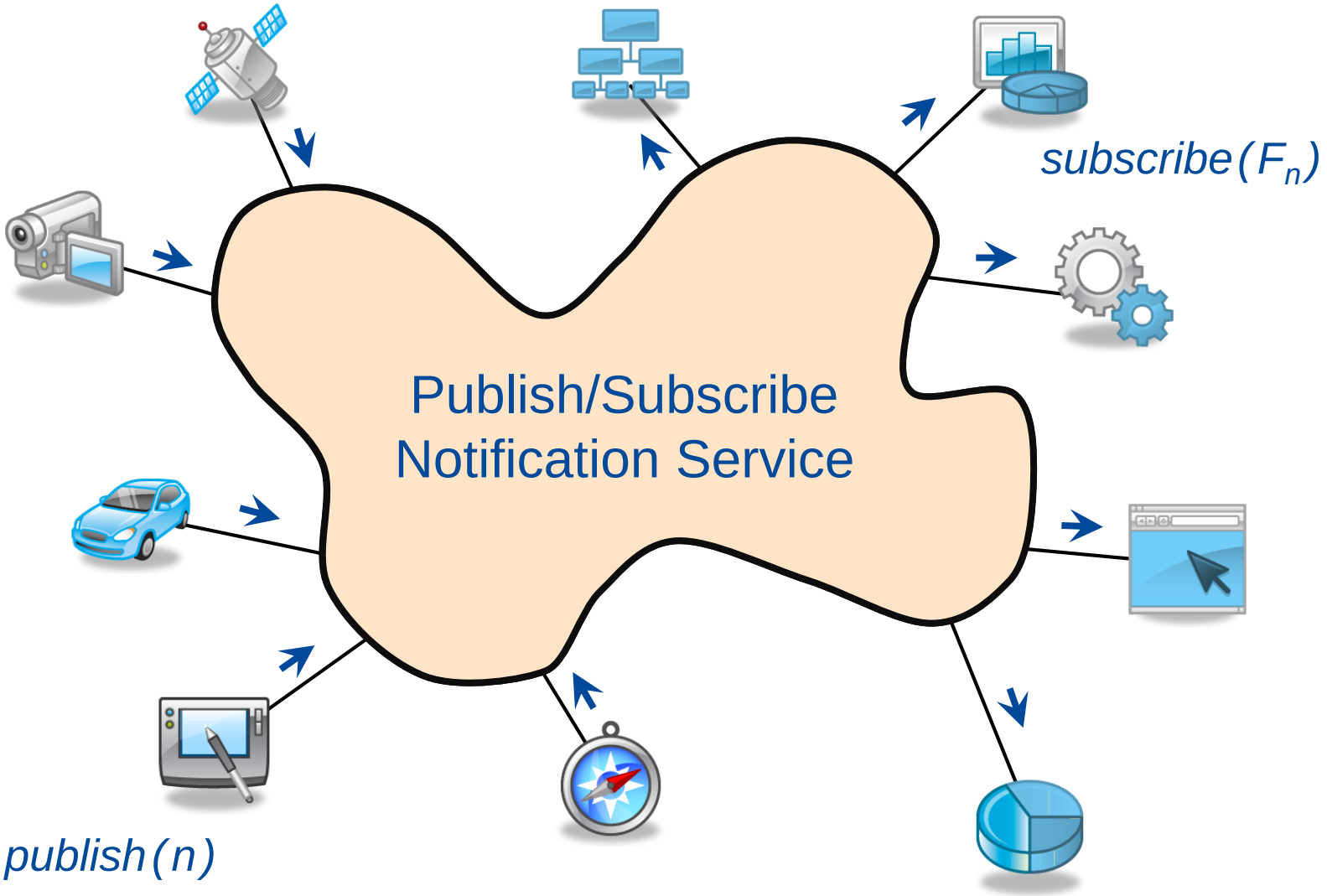
Traditio et Innovatio

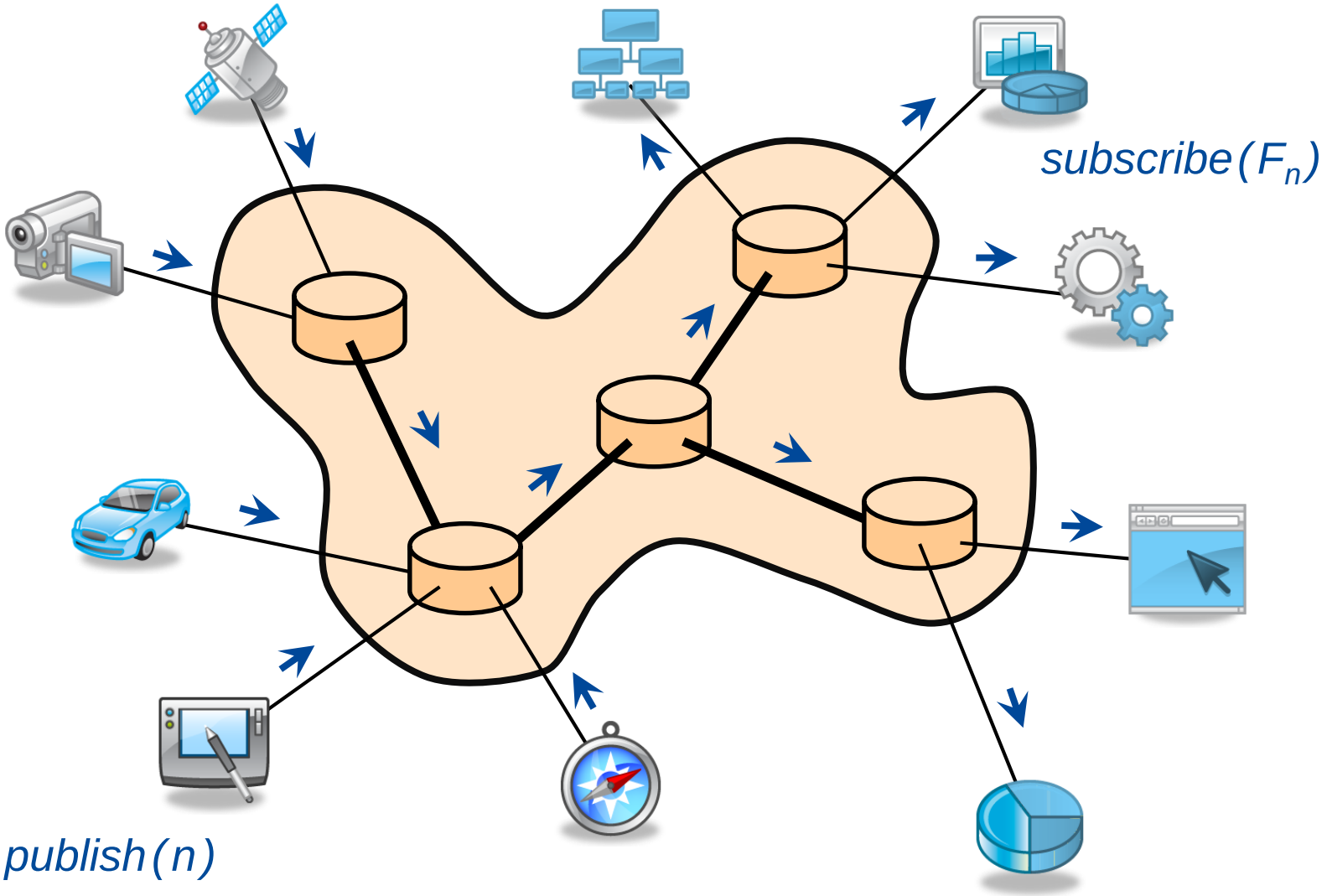
Content-based Publish/Subscribe in Software-defined Networks

Helge Parzyjegla, Christian Wernecke, and Gero Mühl

Architecture of Application Systems (AVA)
Faculty of Computer Science and Electrical Engineering
University of Rostock, 18051 Rostock, Germany


subscribe(F_n)
Publish/Subscribe
Notification Service
publish(n)


Publish/Subscribe
Notification Service
subscribe (F_n)
publish (n)


subscribe(F_n)
publish(n)

Publish/Subscribe Broker Network

- Broker implemented on application layer
  - Subscriptions with arbitrary filter expressions
  - Easy implementation of advanced features
  - → Flexibility
- Forwarding similar to application layer multicast (ALM)
  - Forwarding latency between switch and broker host
  - Delay for traversing the OS network stack
  - Scheduling delay on broker host
  - Costs for serialization, deserialization, and filtering
  - → Latency

Does SDN enable us to combine
flexibility with efficiency?

Agenda

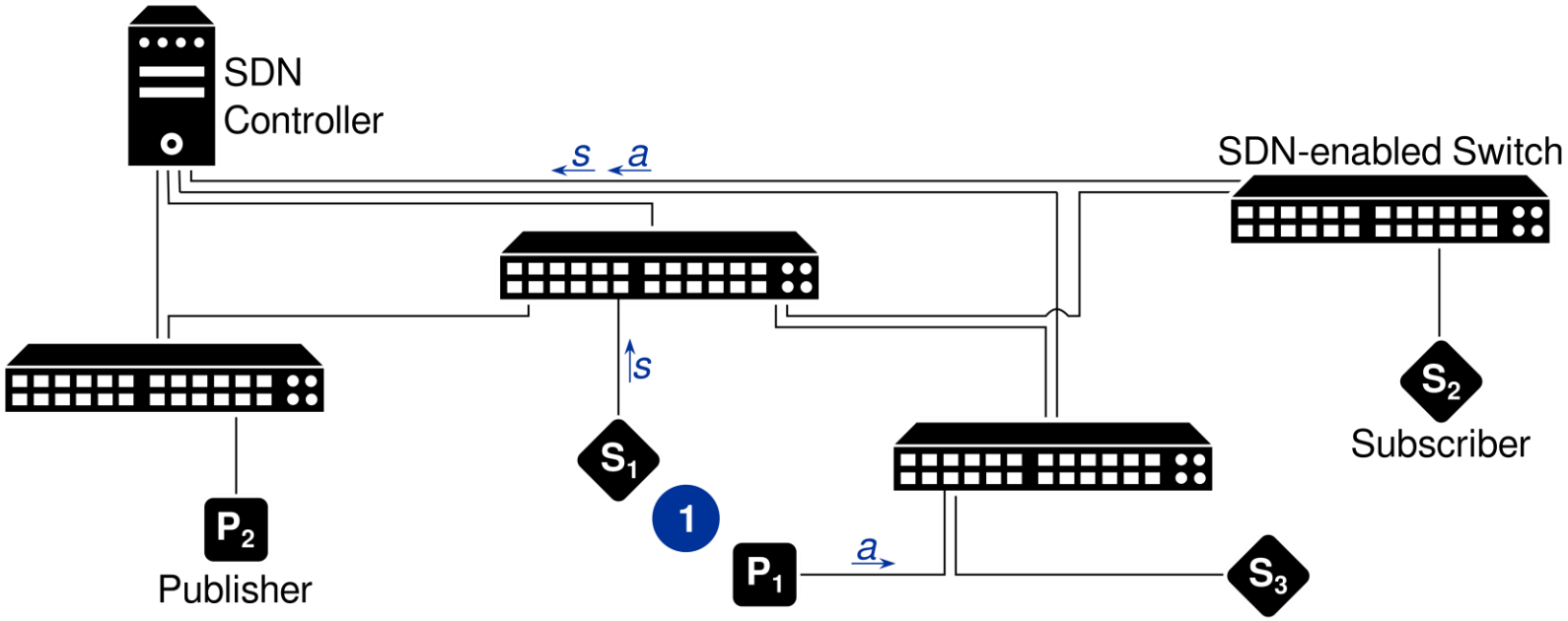
- Motivation
- SDN-based publish/subscribe lifecycle
- Notification forwarding with OpenFlow
  - Bloom filter and forwarding tables
  - Reducing the number of false positives
- Notification forwarding with P4
  - Scheme for multicast source routing
  - Hybrid scheme with preinstalled forwarding trees
- Conclusions

Software-defined Networks and Content-based Publish/Subscribe

- Software-defined Networking (SDN)
  - Separation of network control plane and data plane
  - Control logic operated on a global view → logically centralized
  - Programmable switches capable of individually handling packets according to flow specification → intelligent devices

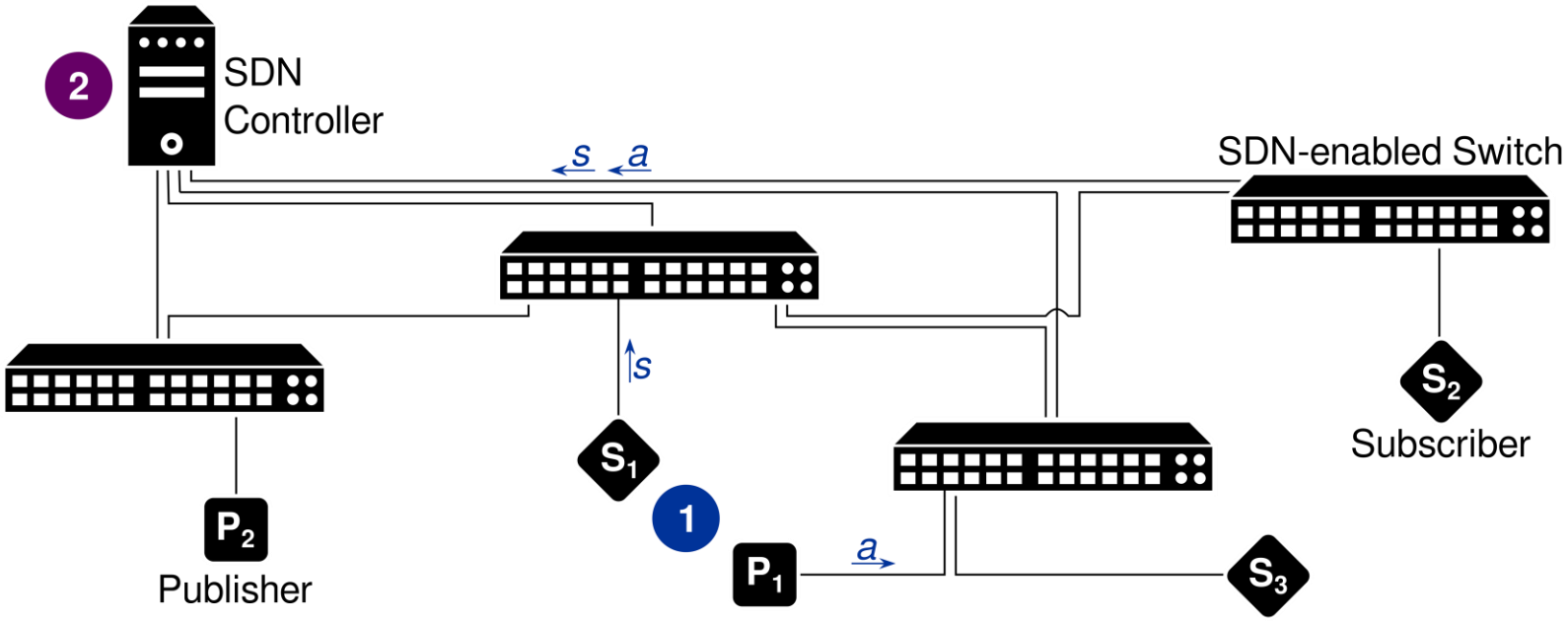
- Content-based publish/subscribe
  - Filtering based on the notification's content
  - But no switch will analyze the payload of a notification packet
  - Requires labeling of notification packets in order to make them processable by SDN-capable switches
  - Requires corresponding adaptations/changes to the publish/subscribe interaction scheme

SDN-based Publish/Subscribe Lifecycle



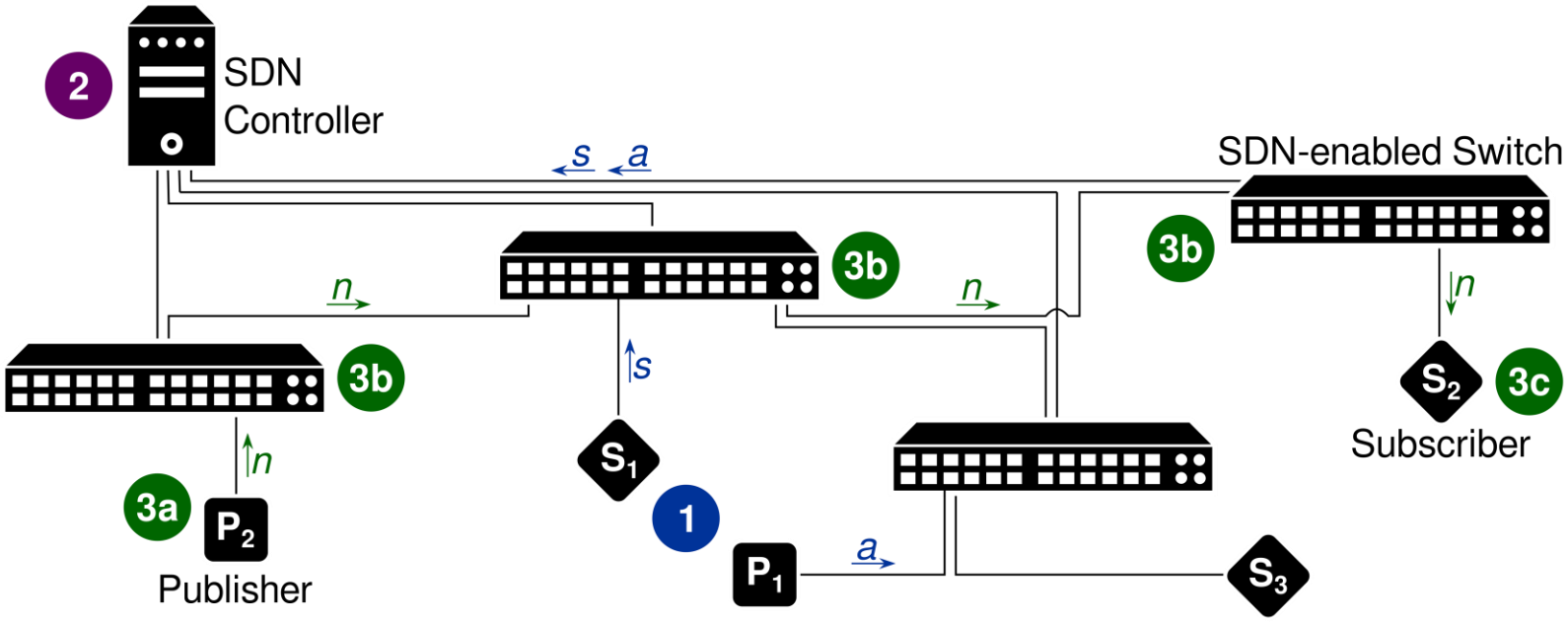
1 Advertising and subscribing

SDN-based Publish/Subscribe Lifecycle



1. Advertising and subscribing
2. Installing the distribution rules

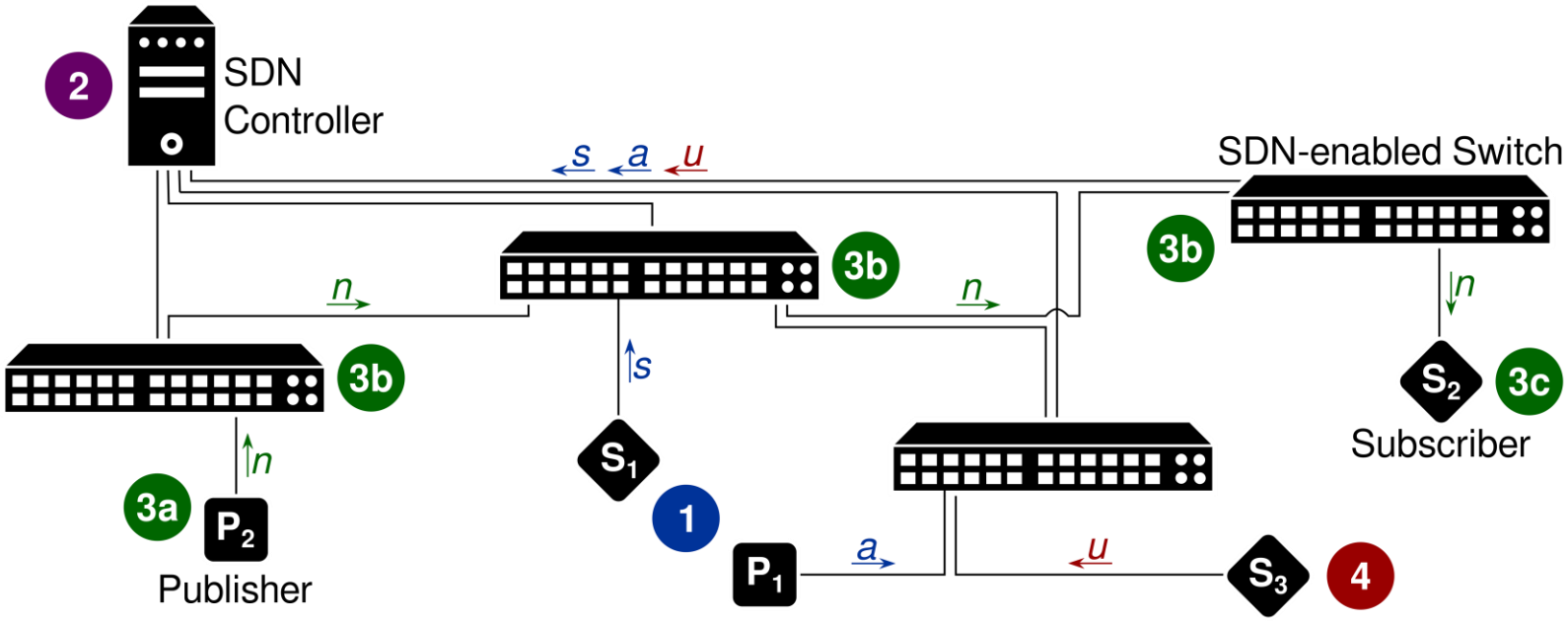
SDN-based Publish/Subscribe Lifecycle



1. Advertising and subscribing
2. Installing the distribution rules
3. Notification distribution

   (a) preprocessing, (b) forwarding, (c) postprocessing

SDN-based Publish/Subscribe Lifecycle



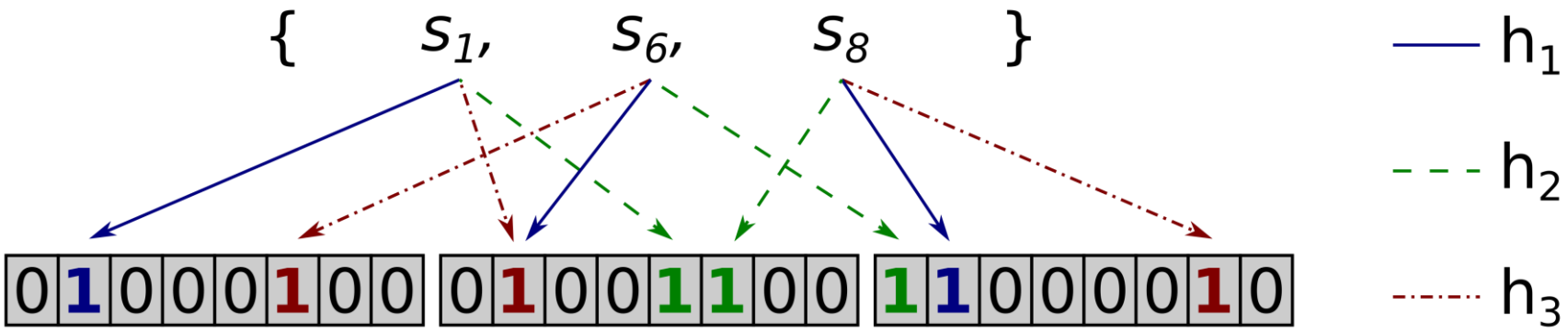
1. Advertising and subscribing
2. Installing the distribution rules
3. Notification distribution
   (a) preprocessing, (b) forwarding, (c) postprocessing
4. Unadvertising and unsubscribing

Labeling of Notifications

- Content-based filtering
  - Most versatile and dynamic form of publish/subscribe
  - Boolean filter is evaluated against the notification's content
  - Active filter expressions and published content dynamically determine the set of receivers for each notification anew

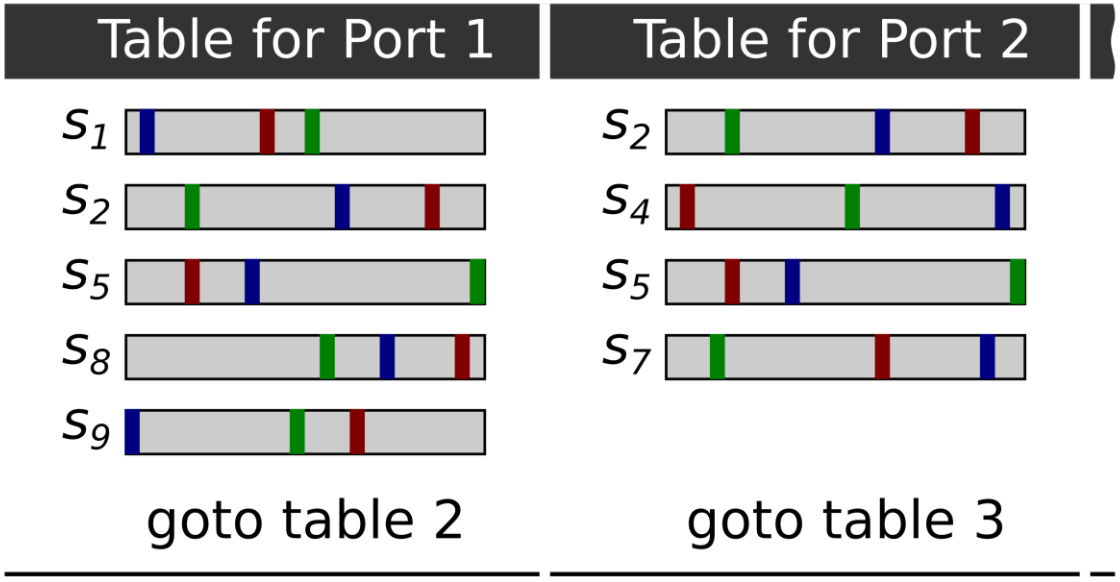
- Challenges for labeling notifications
  - Set of receivers for two consecutive publications may differ substantially (e.g., up to no common receiver)
  - Exponential number of receiver combinations
  - Limited processing capabilities of SDN-enabled switches
    - OpenFlow knows only fixed set of protocols and not all operations are available for every header field

Bloom Filter



- Probabilistic data structure for representation of sets
  - Bit array of length *m* and *k* hash functions
- Adding elements to the Bloom filter
  - Hash the element to get *k* array positions and set those bits to 1
- Testing elements for membership in the Bloom filter
  - hash element and check all *k* array positions
  - At least one position is 0 → not a member of the set
  - All positions are 1 → probably a member of the set
  - Possibility of false positives if bits are set by other elements

Flow Table Organization (OpenFlow)



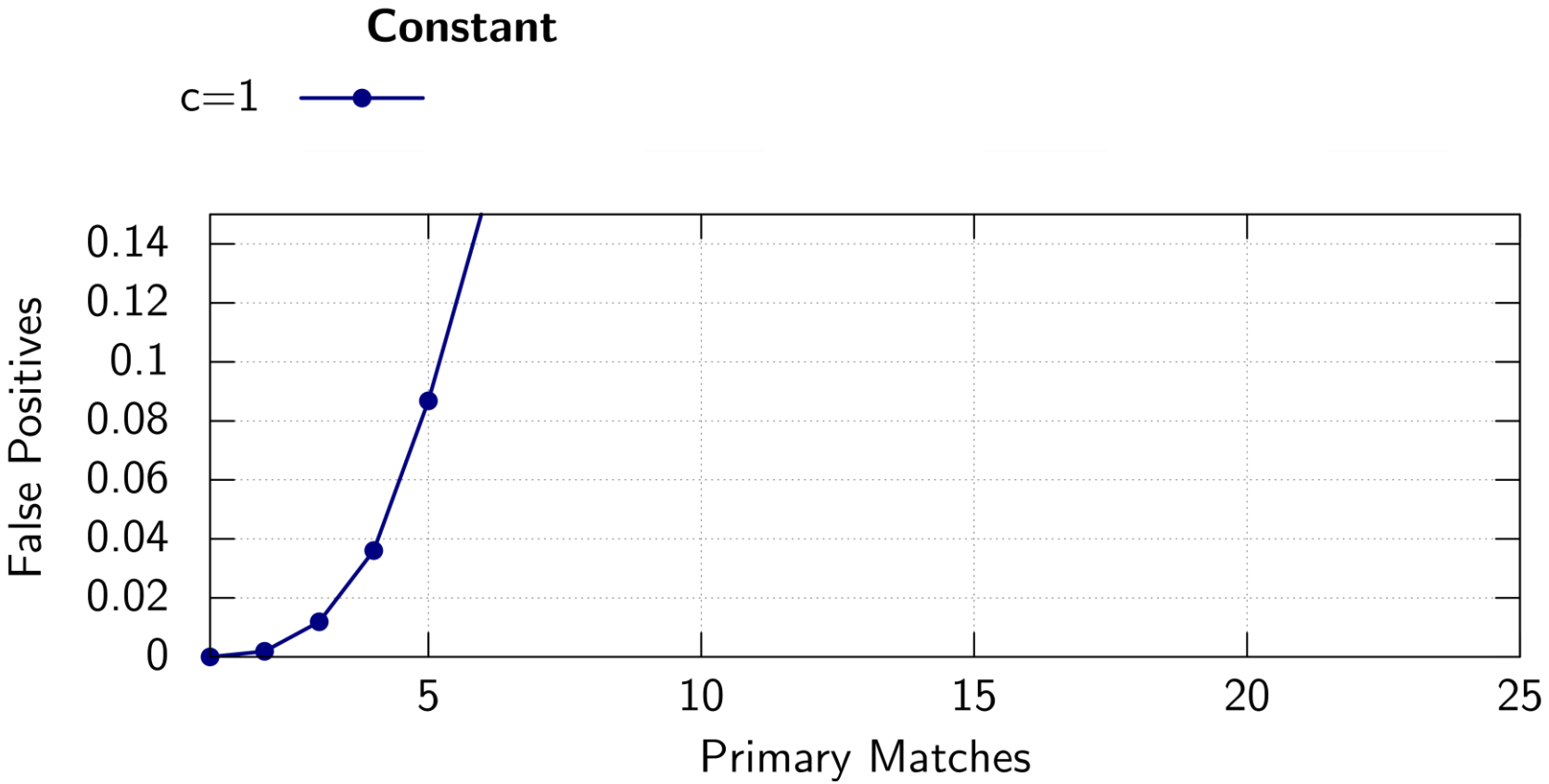
- Bloom filter in IPv6 source/destination address (→ 256 bits)
  - Test membership using bitmask operations
- Flow table for each port with entries for subscriptions
  - Entries for subscribers reachable over the respective port
  - If (at least) one rule matches, packet is forwarded

Evaluation

- Implementation
  - Open vSwitch (version 2.5.2) as software switch
  - Mininet (version 2.3.0) for network emulation
- Setup
  - Topology with 21 switches and 250 clients
  - 1250 different content-based filter expression available
  - Measurements repeated 25 times and averaged
- Analysis
  - Probability of false positives

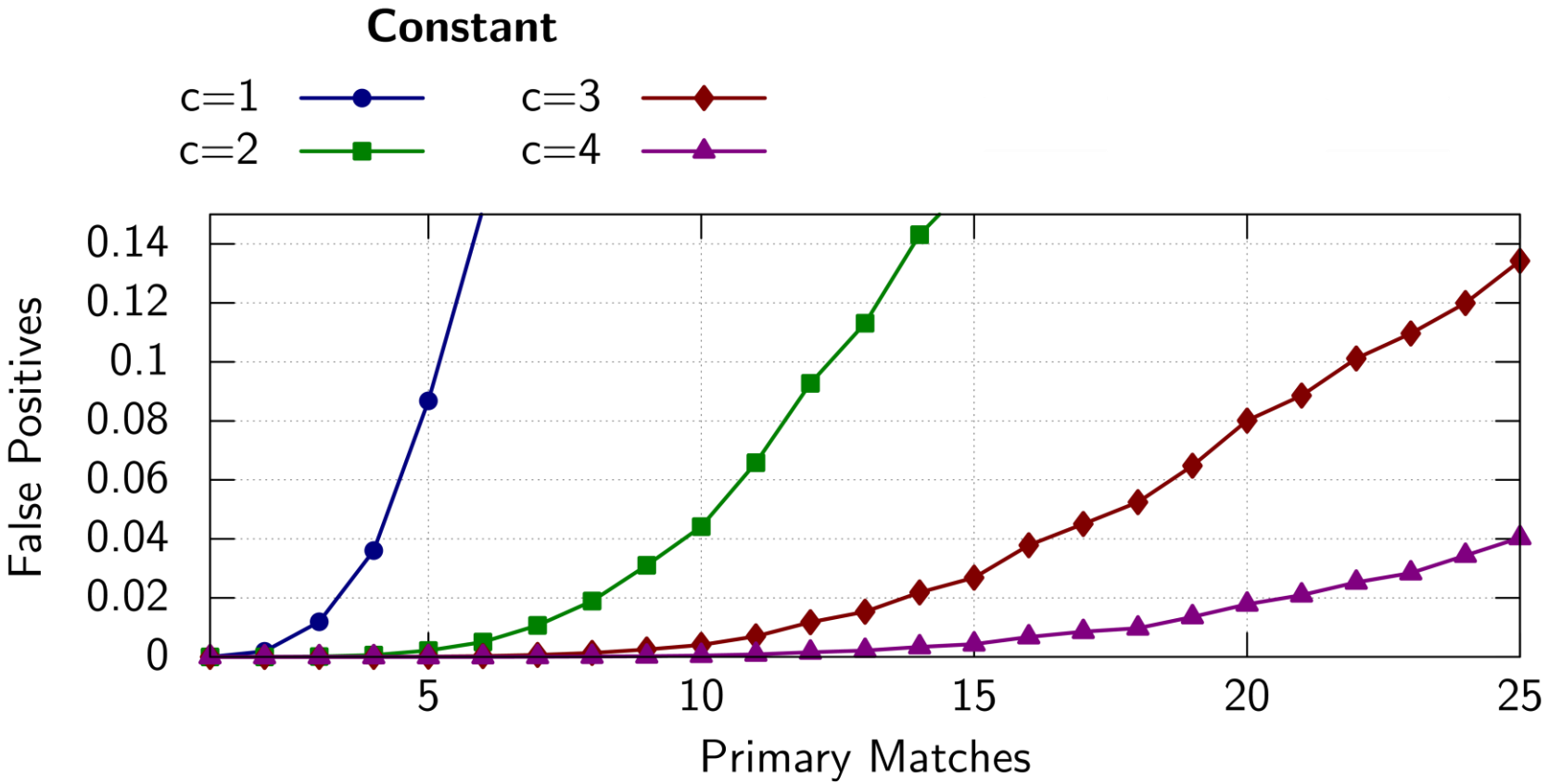
    → How to keep the number of false positives low?

Limiting False Positives (i)



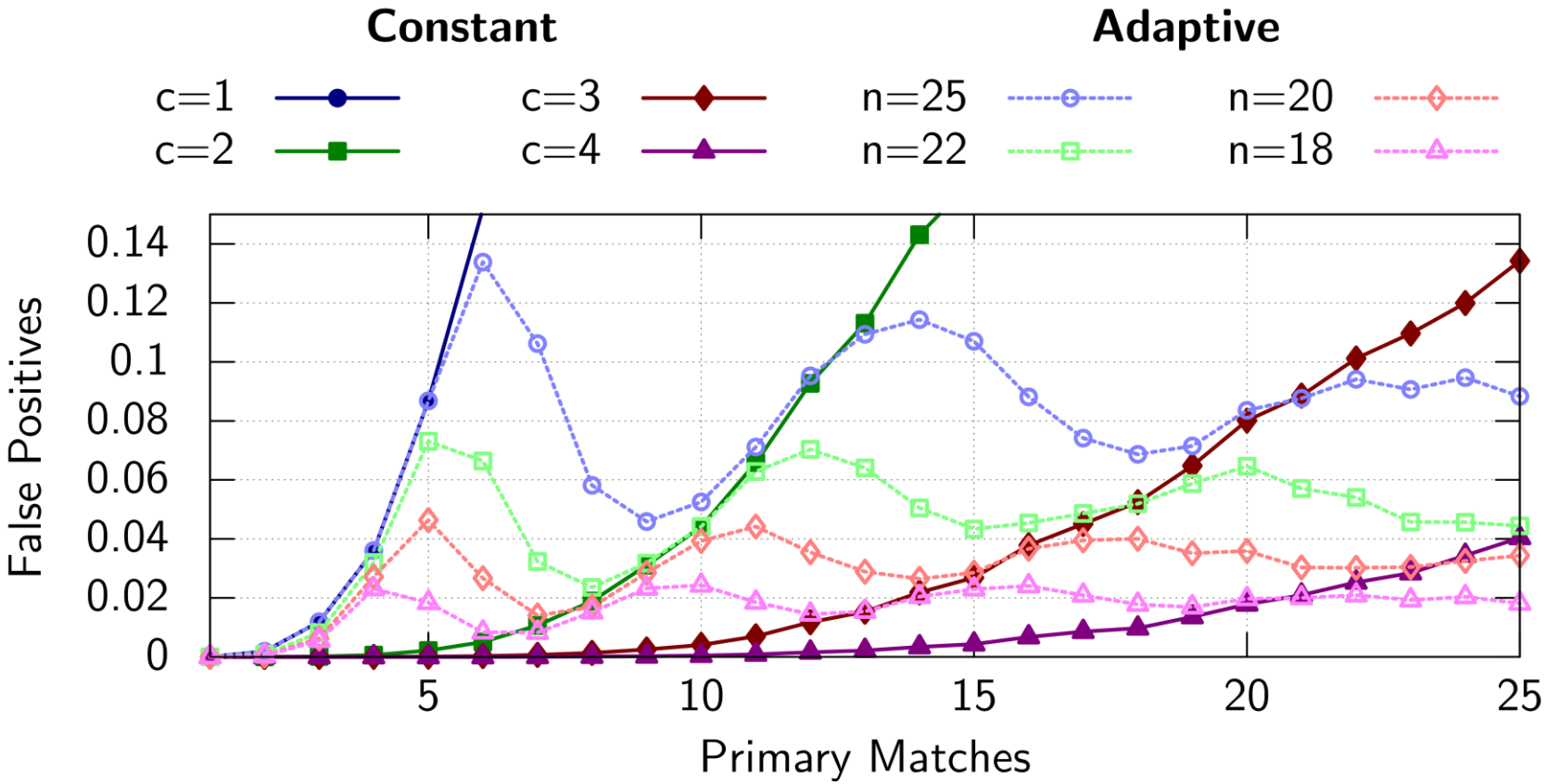
- More elements in Bloom filter → more bits are set 1
- Higher probability of falsely deliveries → more false positives

Limiting False Positives (ii)



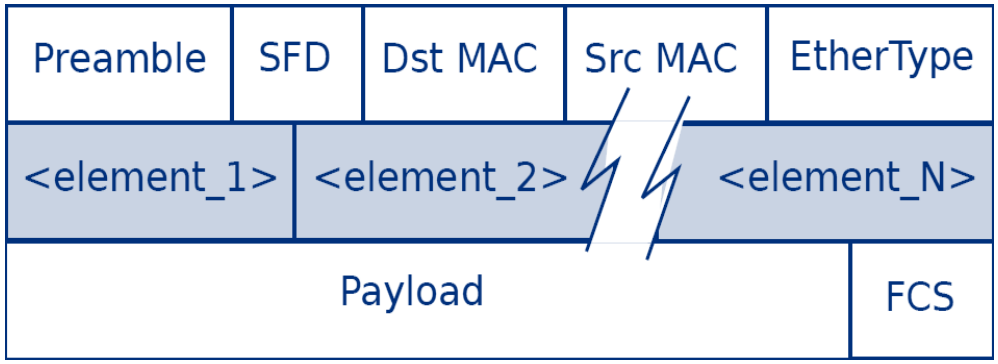
- Sending multiple notification copies each having a Bloom filter
- Equally distribute matching subscriptions among the filters

Limiting False Positives (iii)



> Equally fill up the Bloom filters to certain threshold before sending an additional notification copy → adaptive approach

P4 – Programming Language



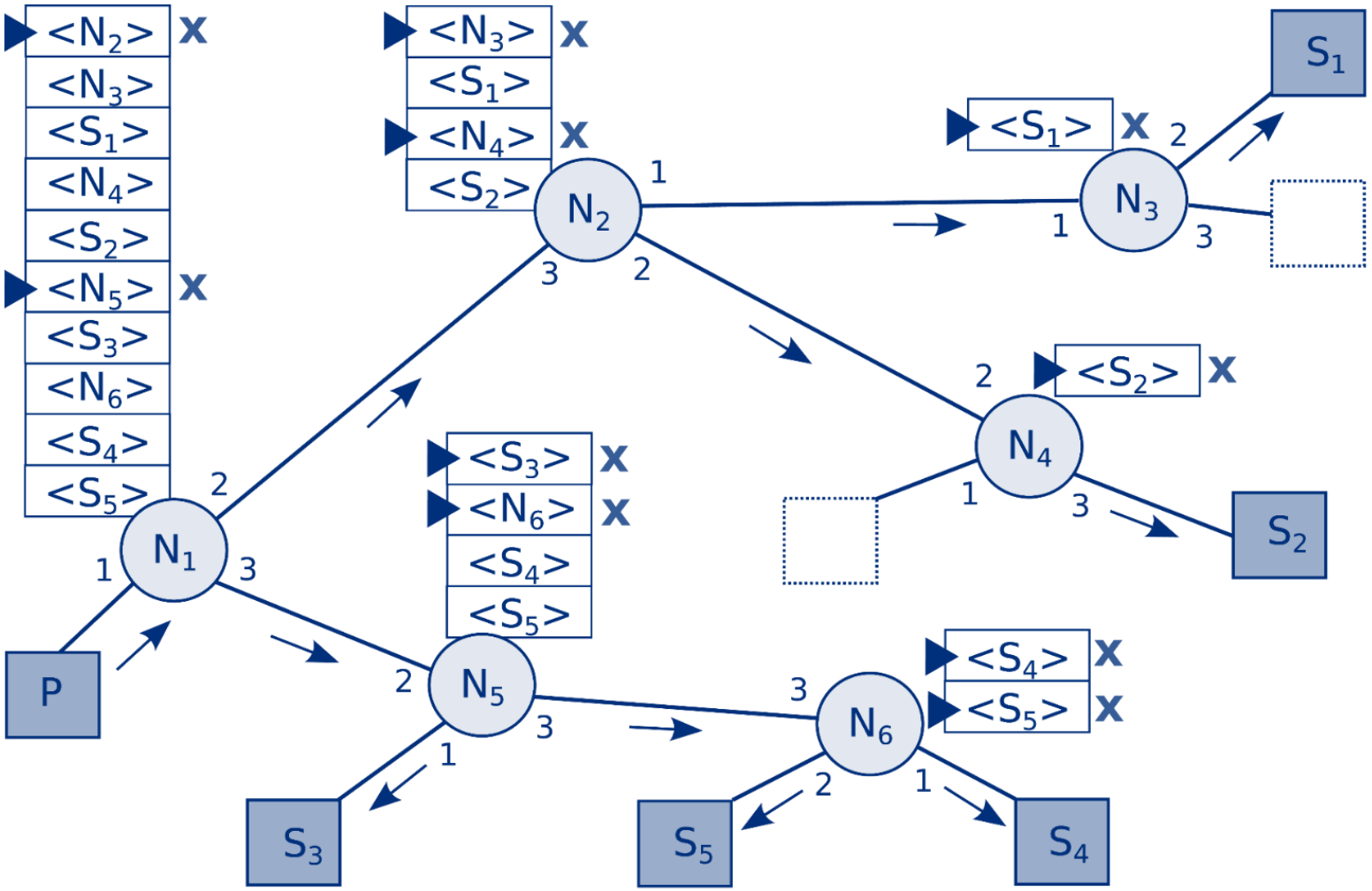
P4 eases embedding and processing custom information between layers 2 and 3.

- Imperative, domain-specific language targeting at packet forwarding applications in the data plane
- Designed to be protocol-independent, target-independent and reconfigurable → "Next-generation OpenFlow"
- Specification of own parsers allows for rapid prototyping, implementation and evaluation of custom protocols
- Support for repeated header fields → header stacks

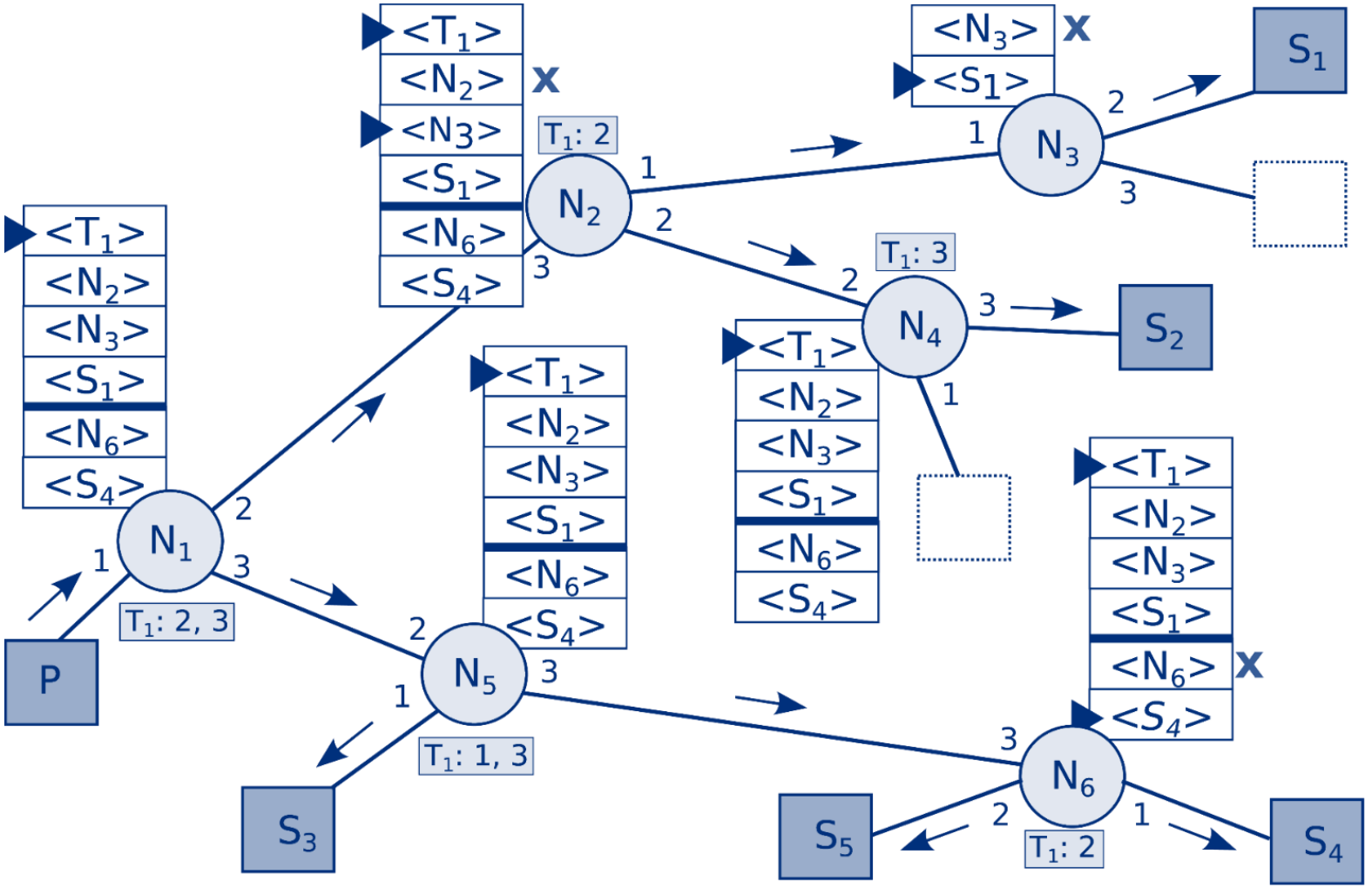
Management of Network State

- Traditional multicast
  - Stored forwarding trees within the network switches
  - Updates required whenever destinations are added or removed
- SDN-based multicast
  - SDN eases construction and management of forwarding trees
  - Update of flow tables are still necessary → possible bottleneck
- (Multicast) source routing without network state
  - Encode whole forwarding tree in the packet header
  - No stored forwarding information and no updates
- Hybrid scheme (multicast and source routing)
  - Extend a stored distribution tree with additional subscribers
  - Source routing to additional subscriber

Encoding Forwarding Trees in the Header

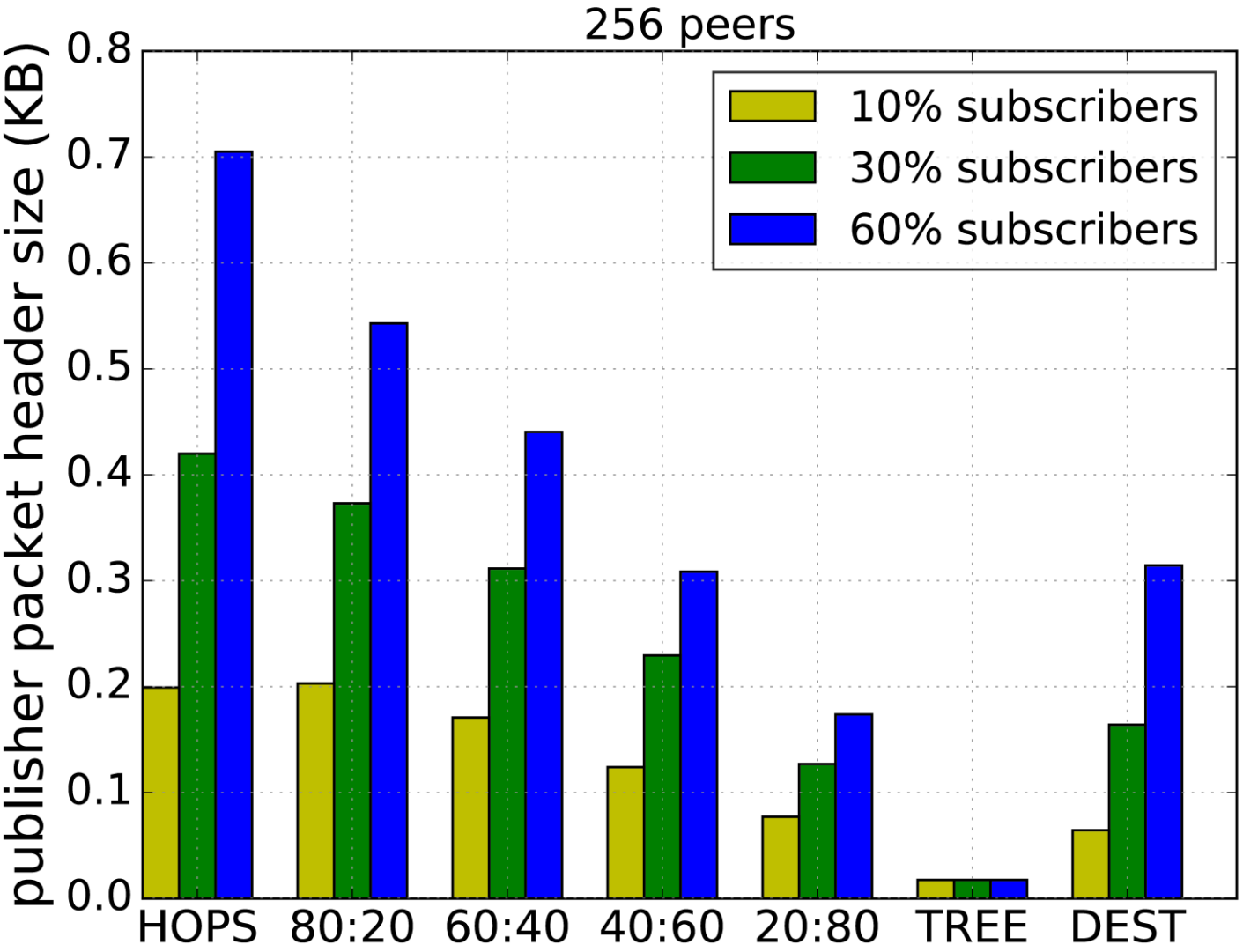


Hybrid Encoding of Forwarding Trees

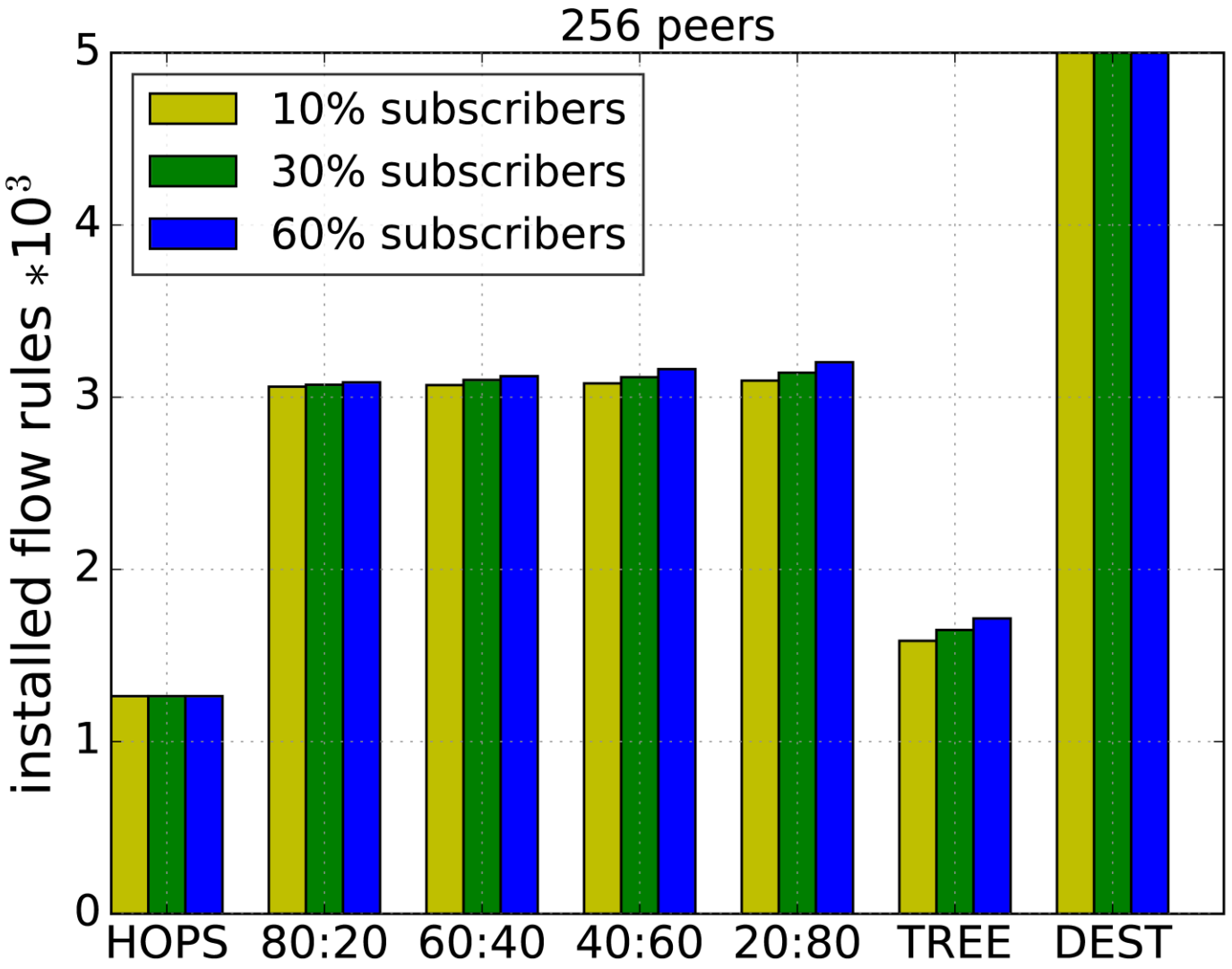


▶...entry used for forwarding x...entry removed from stack

Evaluation: Initial Header Size



Evaluation: Flow Rules / Flow Entries



Conclusions

- Content-based publish/subscribe with OpenFlow
  - Publish/subscribe lifecycle with SDN controller
  - Bloom filters to flexibly label and forward notifications
  - Means for limiting the number of false positives
- Notification forwarding with P4
  - Source routing frees from managing network state
  - Hybrid approach flexibly extends forwarding trees
- Outlook
  - Advanced encoding schemes for OpenFlow
  - Stitching/cutting of multicast trees in P4
  - Analysis of real world application scenarios

Q & A

Thank you for your kind attention.

Dr.-Ing. Helge Parzyjegla
helge.parzyjegla@uni-rostock.de
https://www.ava.uni-rostock.de