Distributed services across the network from edge to core

PhD defense

Amedeo Sapio

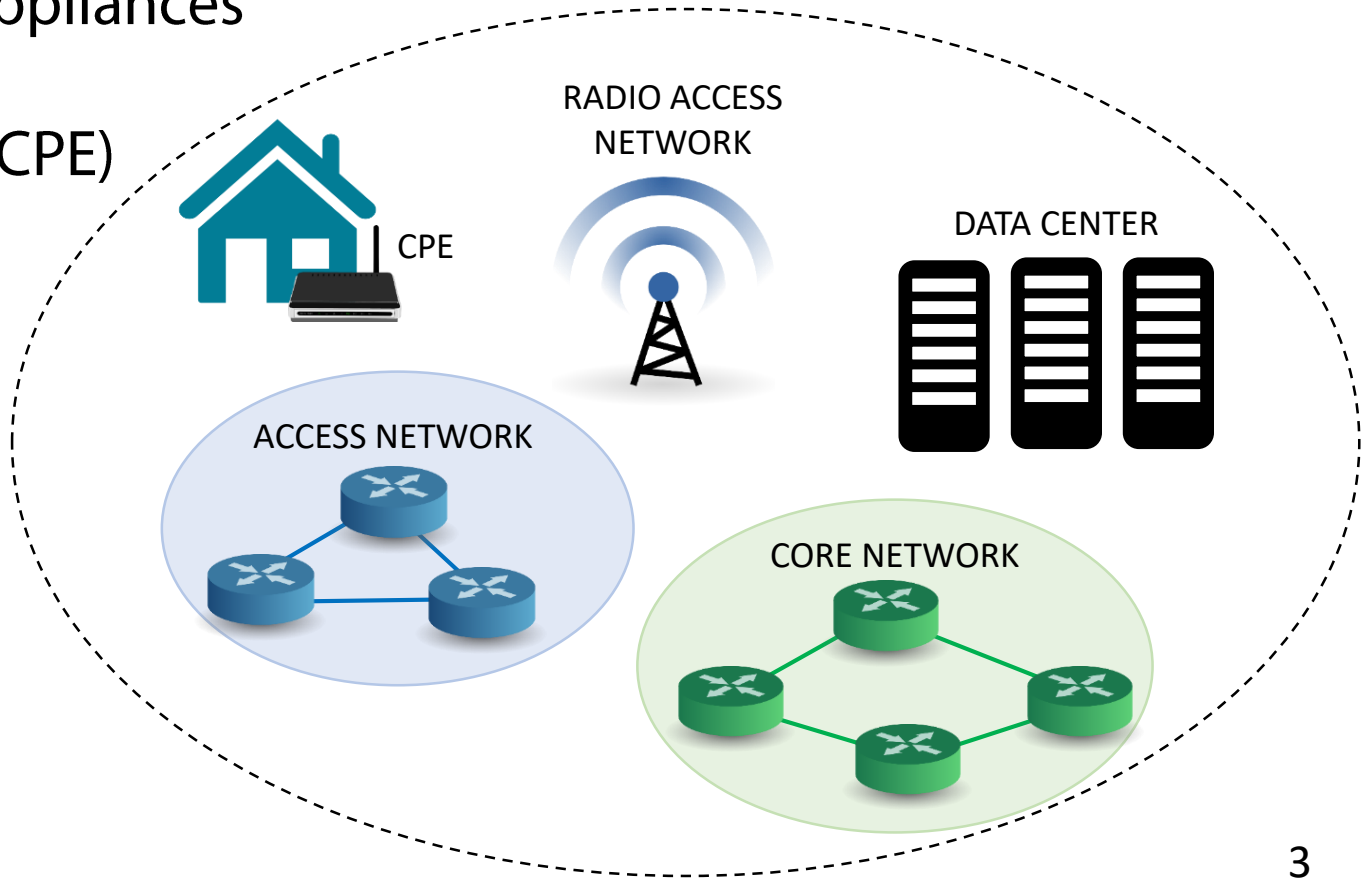
Politecnico di Torino
May 14th, 2018

Scenario

- Telecom companies are increasingly relying on selling value-added services to boost their revenues
  - Music and video streaming
  - Safe-browsing, anti-malware and parental control
- Network "softwarization" is paving the way to the commoditization of telecommunications infrastructure
  - Fixed-function middleboxes can be replaced by software network functions
    - General-purpose hosts
    - Flexibility
    - Lower time to market

Scenario

- Network Service Providers (NSPs) rely on a distributed infrastructure consisting of heterogeneous devices:
  - High speed, special purpose, appliances
  - Low-cost, resource-limited Customer Premise Equipment (CPE)
  - Data-centers



Thesis goal

- The work described in this Ph.D. thesis aims at showing how all these different devices can be used to provide additional services
  - suited to their specific constraints and limitations

- Leveraging new network paradigms:
  - Network Functions Virtualization
  - Fog and edge computing
  - Data plane programmability

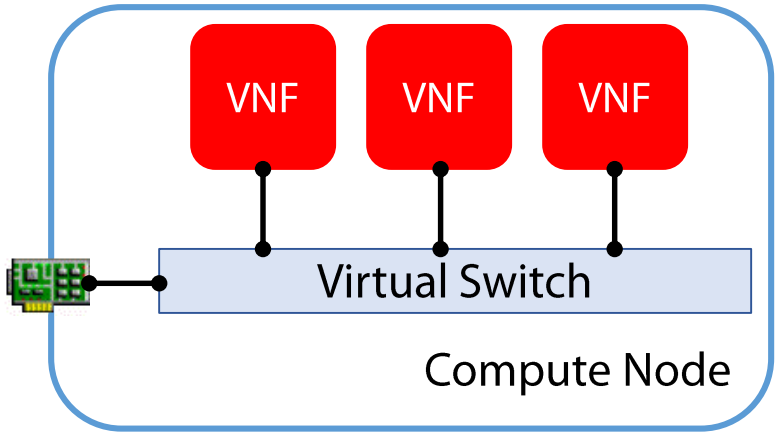
Network services in the data center:

Network Functions Virtualization

Network Functions Virtualization (NFV)

- NFV targets the execution of software network functions (NFs) in isolated Virtual Machines (VMs), rather than on dedicated hardware.

- Advantages:
  - Faster provisioning
  - Dynamic resources allocation
  - Centralized management
  - Dynamic traffic steering



- NFV requires the ability to:
  - Effectively assign compute nodes to virtual NFs (VNFs)
  - Allocate the appropriate amount of resources, such as CPU quota, RAM, virtual interfaces, etc.

Orchestration and scheduling decisions require an estimation of expected NFs performance vs. resource consumption.

NF modeling[1]

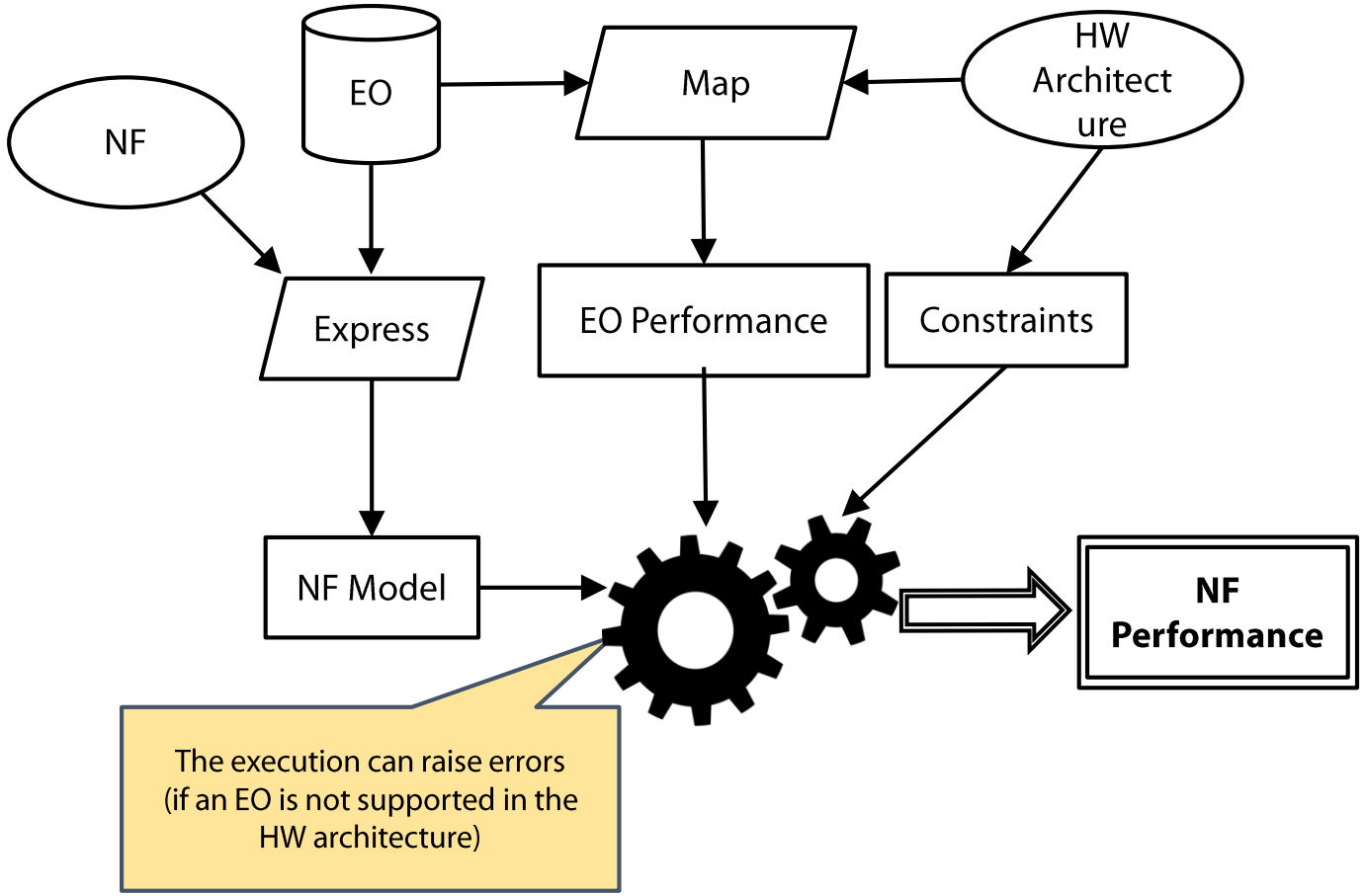
- Generally, most NFs perform a rather small set of recurring operations when processing the average packet
  - A well-defined alteration of packet headers, coupled with data structure lookup

- Elementary NF Operations (EOs):
  - Informally defined as the longest sequence of elementary steps (e.g., CPU instructions or ASIC transactions) that is common among the processing tasks of multiple NFs

[1] Mario Baldi and Amedeo Sapio. Network Function Modeling and Performance Estimation. International Journal of Electrical and Computer Engineering, 2018.

The process

- A NF can be modeled by splitting its functionality in EOs
  - **Hardware independent**
- Each EO is mapped on the hardware component(s)/function(s) involved in its execution
  - **Hardware platform specific**

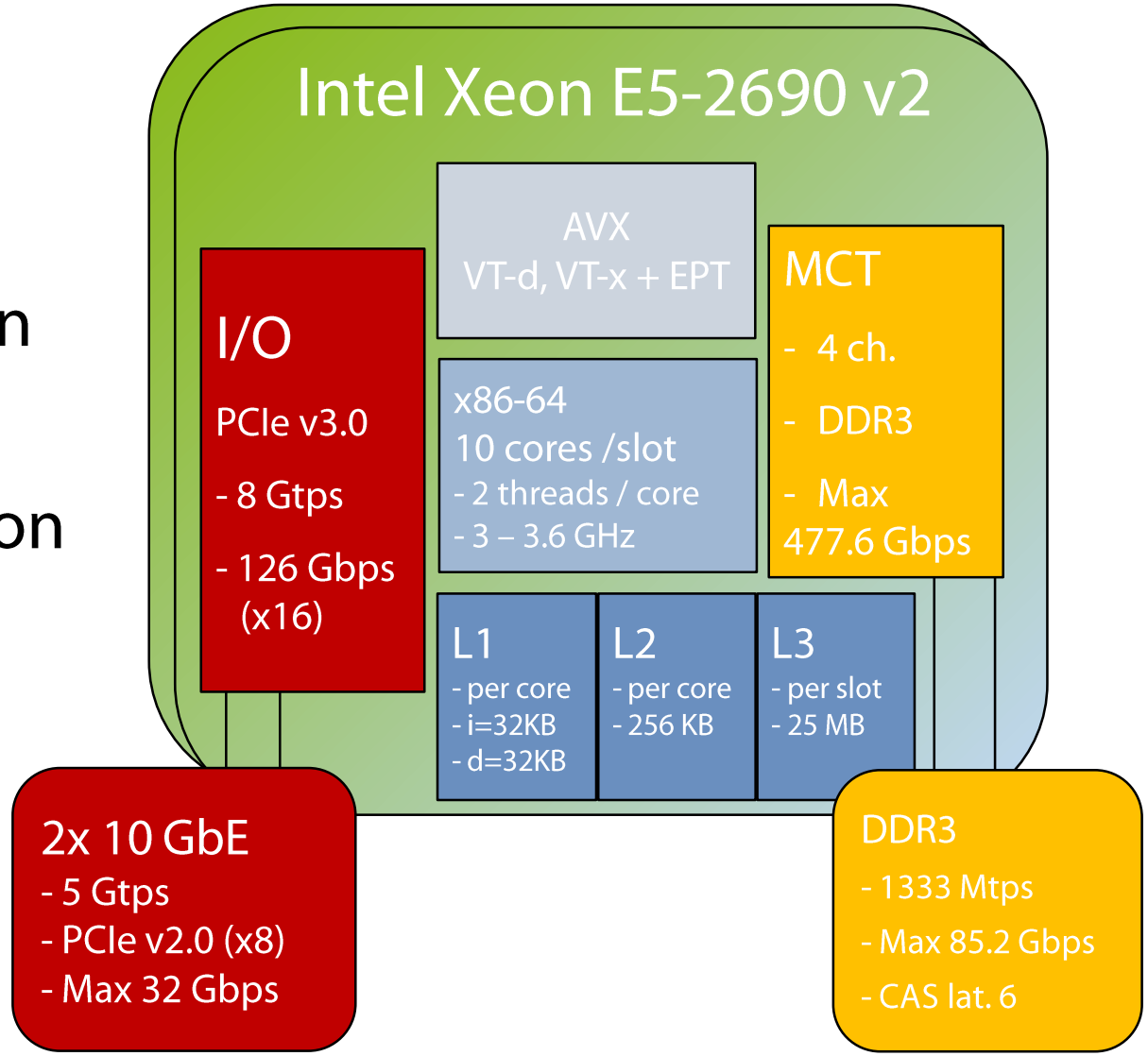


List of sample EOs

	EO	Parameters	Description
1	`I/O_mem / mem_I/O`	hdr, data	Packet copy between I/O and (cache) memory
2	`parse / deparse`	b	Parse or encapsulate a data field
3	`increase / decrease`	b	Increase/decrease a field
4	`sum`	b	Sum 2 operands
5	`checksum / inc_checksum`	b	Compute IP checksum
6	`array_access`	es, max	Direct access to a byte array in memory
7	`ht_lookup`	N, HE, max, p	Simple hash table lookup
8	`lpm_lookup`	b, es	Longest prefix match lookup
9	`ct_insertion`	N, HE, max, p	Cache table insertion

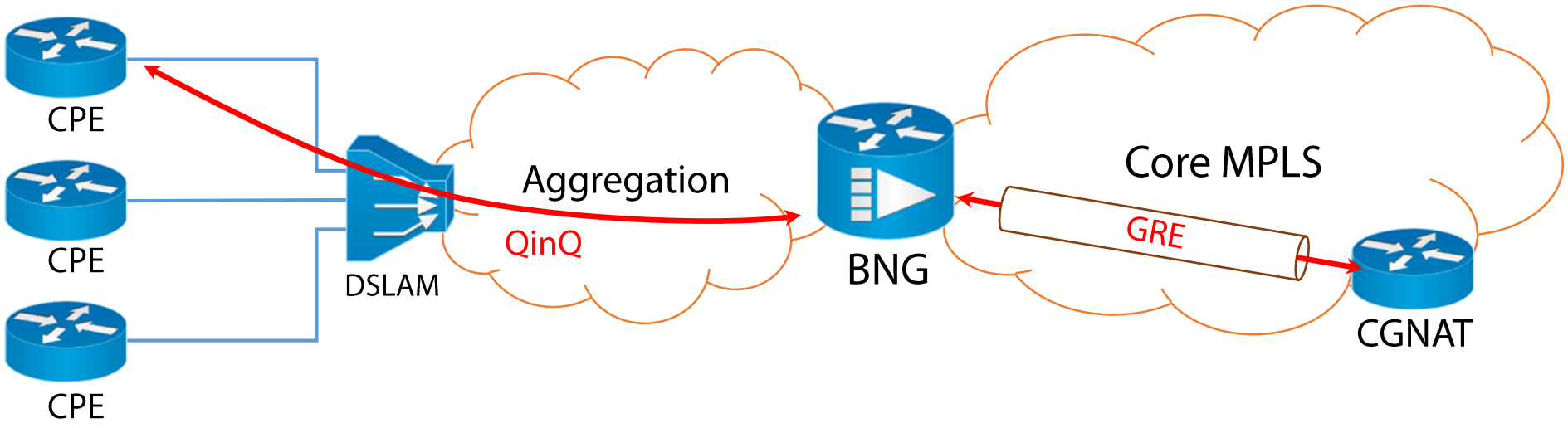
Mapping to hardware

- Each EO is mapped on the hardware components involved in its execution
- This mapping takes into consideration the limits of the involved hardware components (e.g., clock frequency).



Modeling use cases

- L2 Switch
  - Basic Forwarding
  - Learning Switch
  - MPLS Switch
- Broadband Network Gateway
  - QinQ ↔ MPLS+GRE



HW independent models

Basic Forwarding

I/O_mem(30,ps)
parse(6)
ht_lookup(1,12,2M,0)
deparse(6)
mem_I/O(30,ps)

MPLS Switch

I/O_mem(34,ps-4)
parse(3)
ht_lookup(1,12,1M,0)
parse(1)
decrease(1)
deparse(10)
mem_I/O(34,ps-4)

Learning Switch

I/O_mem(30,ps)
parse(8)
ht_lookup(1,14,2M,0)
parse(12)
ct_insertion(2,14,2M,0)
deparse(6)
mem_I/O(30,ps)

Broadband Network Gateway

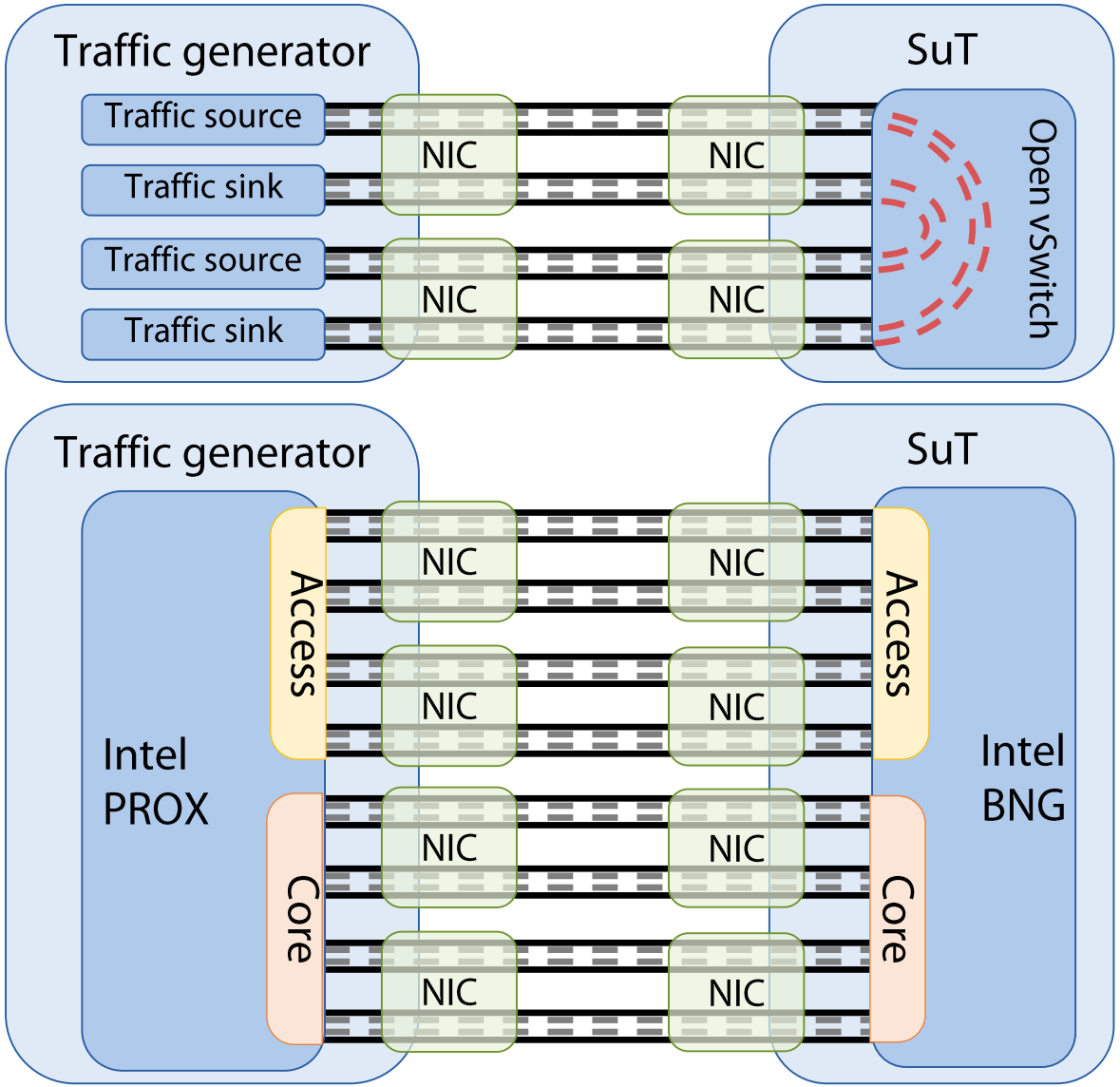
Packet from access network
I/O_mem(42,ps-20)
parse(8)
ht_lookup(1,7,16M,0)
parse(4)
lpm_lookup(2,23)
parse(1)
decrease(1)
parse(2)
inc_checksum(1)
checksum(ps-14)
sum(2)
checksum(20)
parse(16)
ct_insertion(2,23,64K,0)
deparse(70)
mem_I/O(70,ps-20)

Packet from core network
I/O_mem(70,ps-56)
parse(8)
ht_lookup(2,23,64K,0)
parse(1)
decrease(1)
parse(2)
inc_checksum(1)
deparse(42)
mem_I/O(42,ps-56)

Experimental evaluation

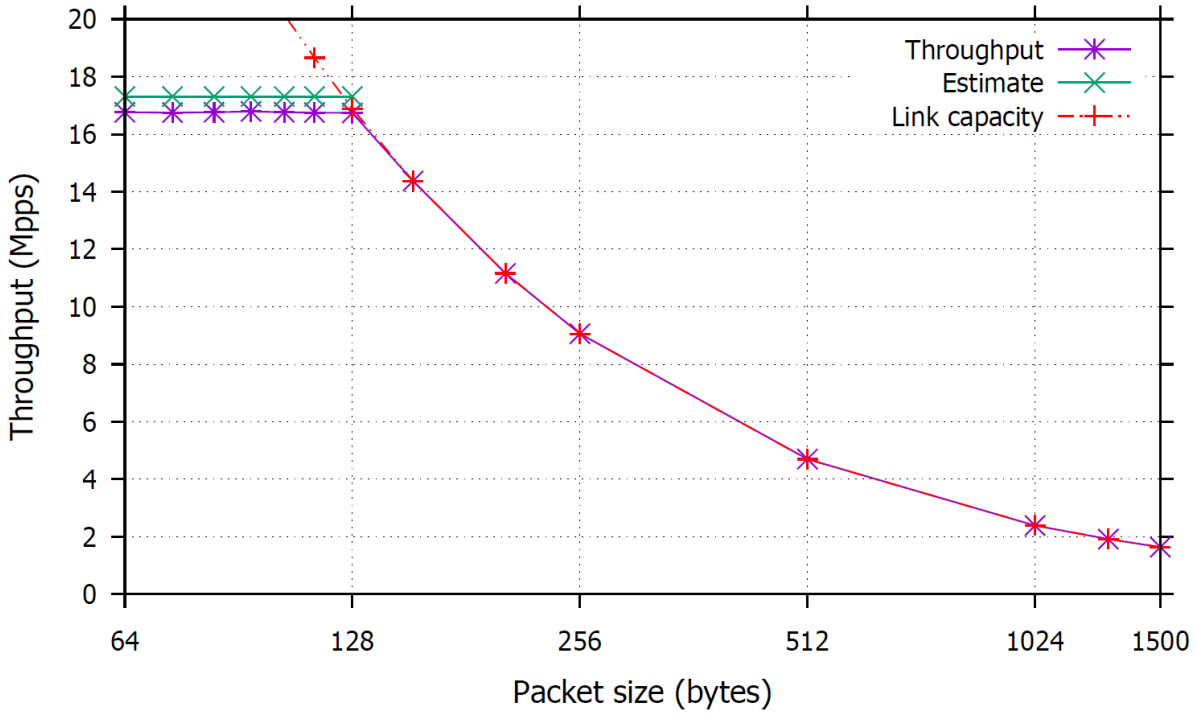
Testbed setup:

- System under test (SuT):
  - Intel DPDK drivers
    - DDIO to load packets directly in the L3 cache
  - Open vSwitch
  - Intel software BNG
- Traffic generator:
  - PF_RING/DNA drivers
  - Intel Packet pROcessing eXecution Engine (PROX)

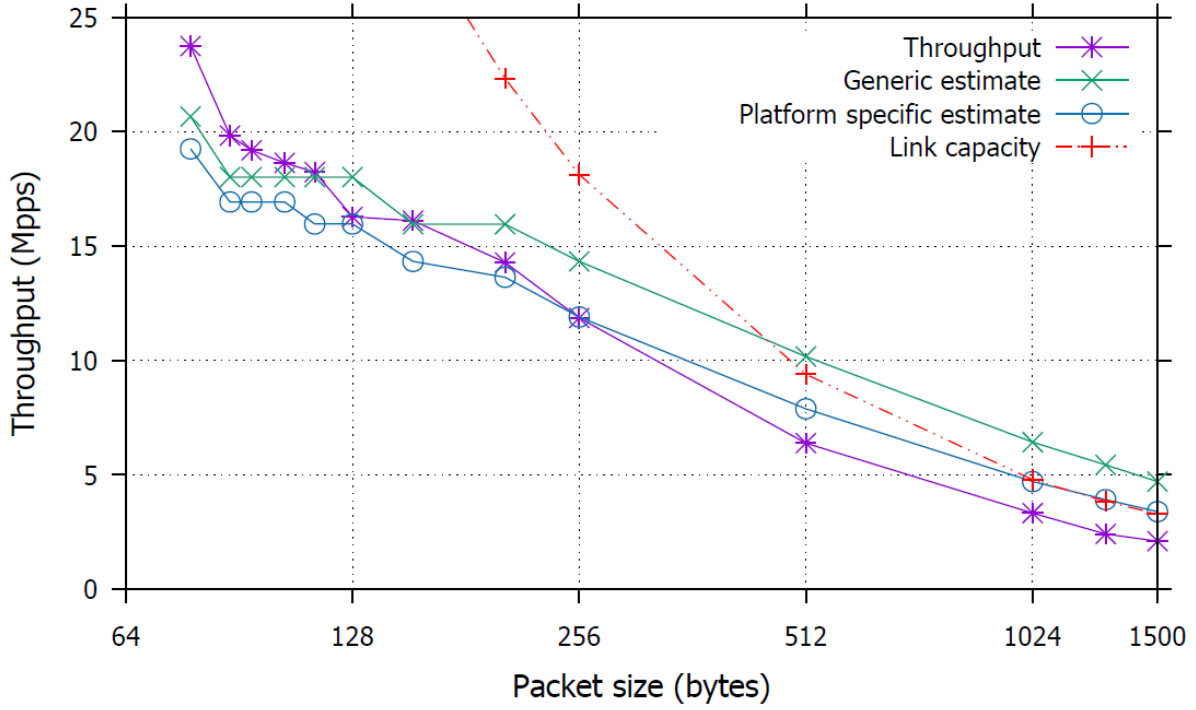


Experimental evaluation

Basic Forwarding



Broadband Network Gateway



Concluding remarks

- Experimental results show that software NF's performance are heavily affected by specific factors:
  - The effectiveness of HW and SW caching mechanisms
  - Traffic runtime characteristics
  - The implementation of the NF
  - Parallel execution of operations

- Our performance estimation approach is well suited to NFs designed to perform a well-defined packet processing operation at high speed
  - General purpose implementations based on a generic, configurable pipeline are not well modeled by our approach

Network services on-premises:

Resource-constrained residential gateways

Residential gateways or CPEs

- Modern residential gateways are widely deployed to provide broadband Internet access to families, small and medium-sized enterprises
- Customers can benefit from services deployed on CPEs:
  - Low latency
  - Security, protection and privacy
- They usually have limited computing and memory resources

Goal: combine the benefits of the cloud with the locality of services running on local CPEs

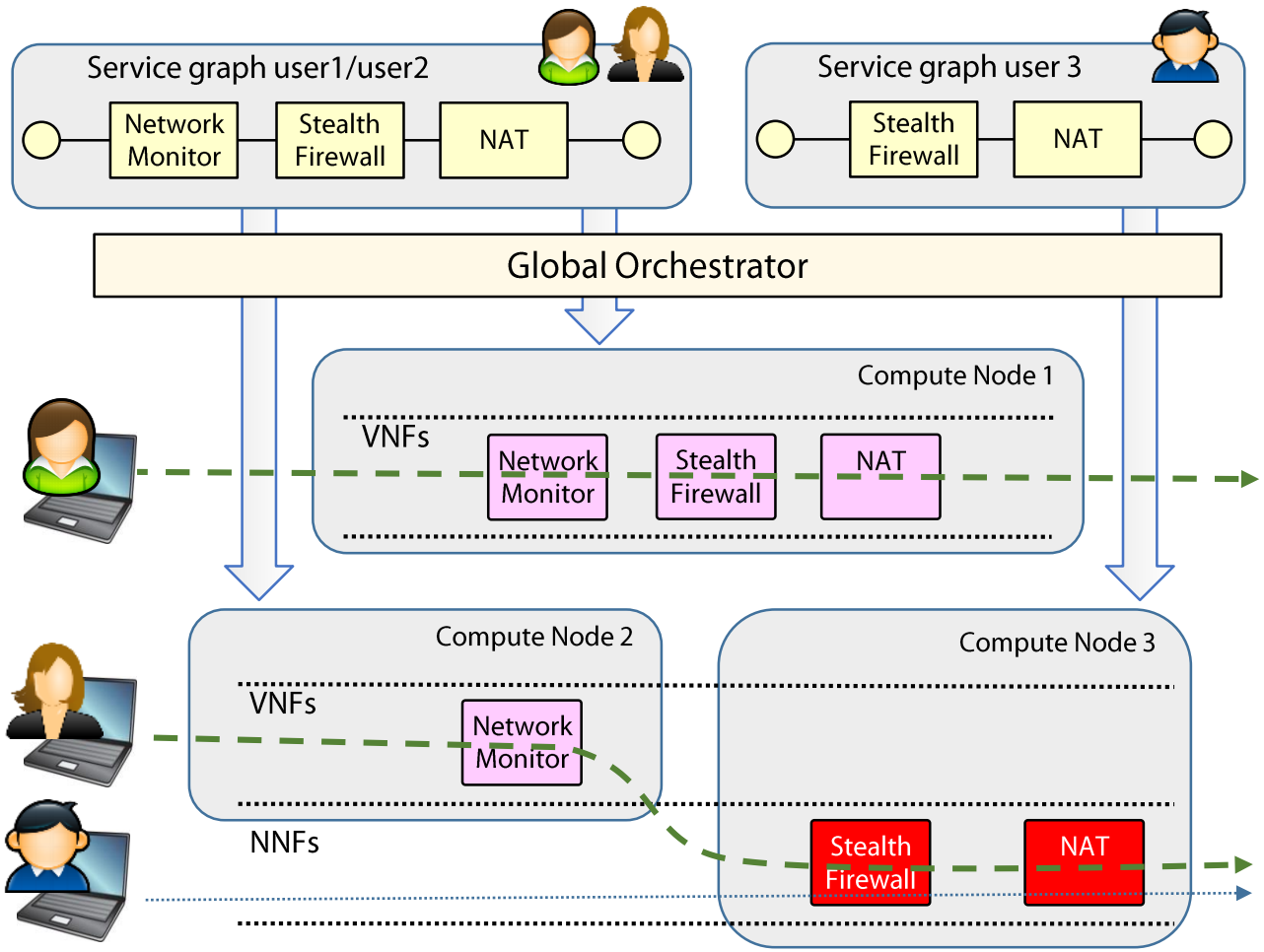
CPEs in the NFV domain[2]

- CPEs are usually based on low-cost hardware that cannot run virtual machines
- However, most CPEs are based on Linux, which includes a broad set of existing software NFs
  - Firewall, NAT, virtual switch, etc.
- We propose to integrate existing CPEs in an NFV domain
  - Complex VNFs in the data center
  - Simple *Native Network Functions* (NNFs) are executed in the CPE
- NNFs can exploit hardware components already available in CPEs
  - Crypto hardware accelerator, integrated L2 switch, etc.

[2] R. Bonafiglia, S. Miano, S. Nuccio, F. Risso, and A. Sapio. Enabling NFV services on resource-constrained CPEs. IEEE CloudNet, 2016

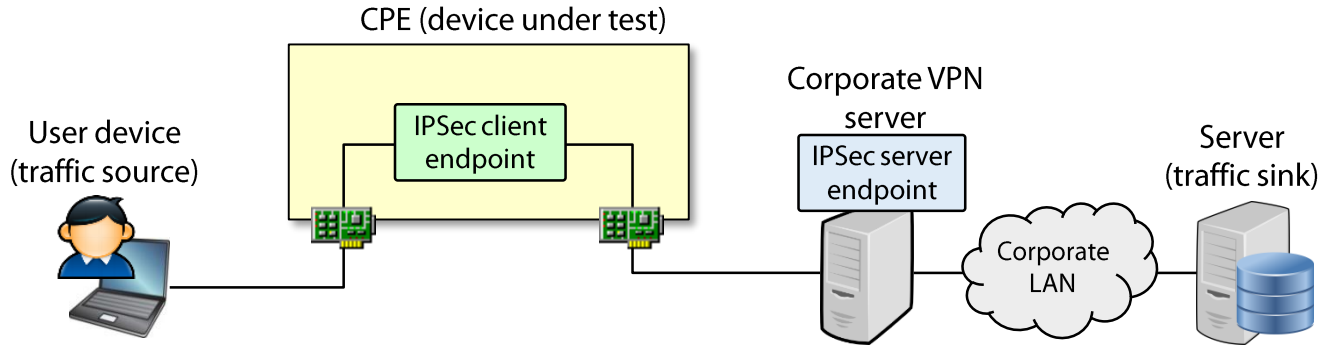
NF deployment

- The orchestrator can optimize the scheduling of NFs:
  - NNF: services that require proximity to the end user
  - VNF: services that require powerful hardware
- Common northbound interface:
  - Export platform capabilities
  - Manage the lifecycle of the NF
  - Set up the service chain
- Isolation of NNFs is limited



Preliminary evaluation

- NNFs and Docker bring significant performance improvements



- NNF require less storage:
  - Smaller image
  - Fewer additional libraries
- Less time required to download the NNF image from a remote location

IPsec client implementation	Thr./CPU (Mbps/load)	RAM (MB)	NF image (MB)
Server CPE - KVM/QEMU	796 / 100%	390.6	522
Server CPE - Docker	1095 / 80%	24.2	240
Server CPE - NNF	1094 / 80%	19.4	5
Domestic CPE - NNF	57.2 / 100%	5	2
Business CPE - NNF	617 / 90%	1.9	3.7

Enforcement of dynamic HTTP policies

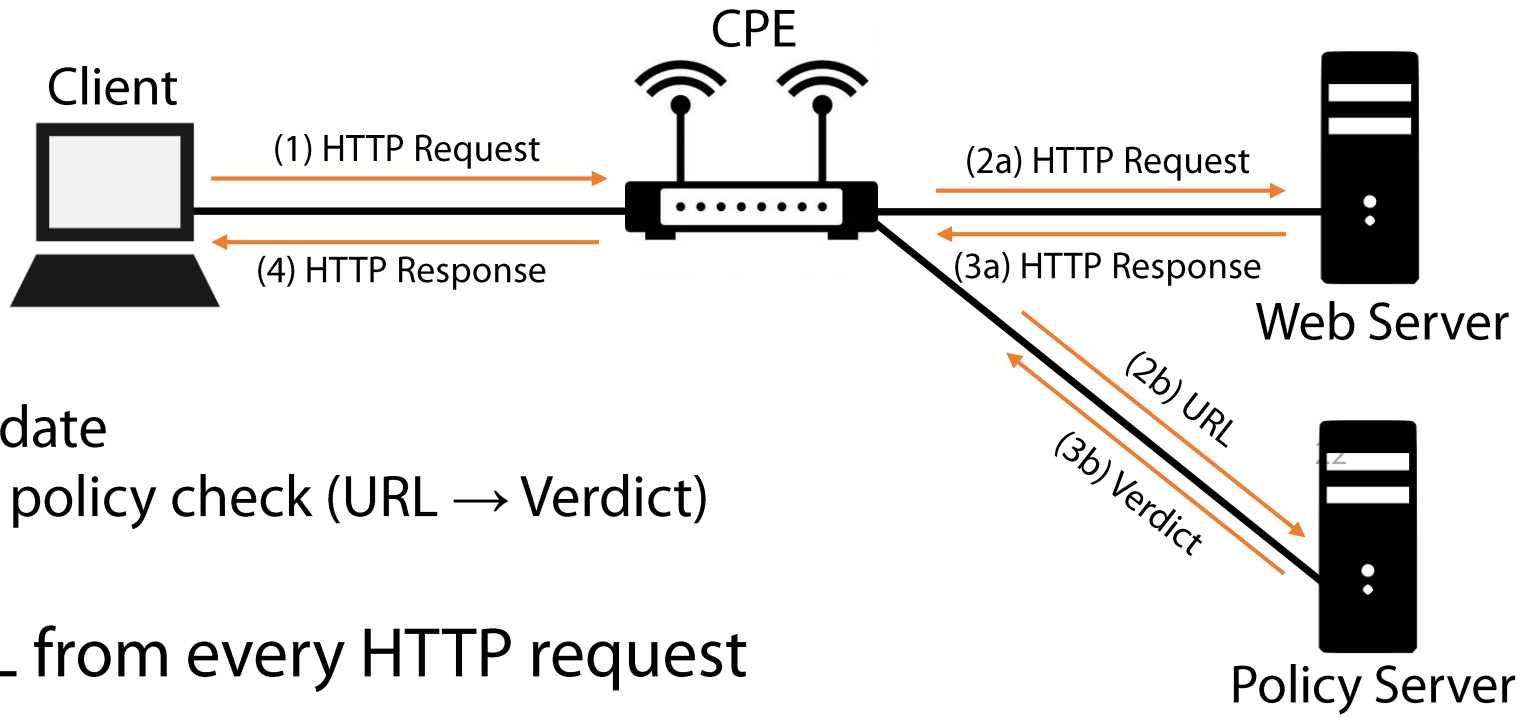
- The residential gateway is the ideal appliance to apply traffic filtering
  - All the outbound traffic must pass through it

- Enforcement of HTTP policies requires to parse packets up to the application layer
- URL blacklists are often very large, and must be frequently updated

- Resource-constrained CPEs cannot perform complex operation at wire speed, neither can store large amount of data

U-Filter is an efficient solution to integrate a URL filtering service in a CPE leveraging a distributed architecture

U-Filter [3]

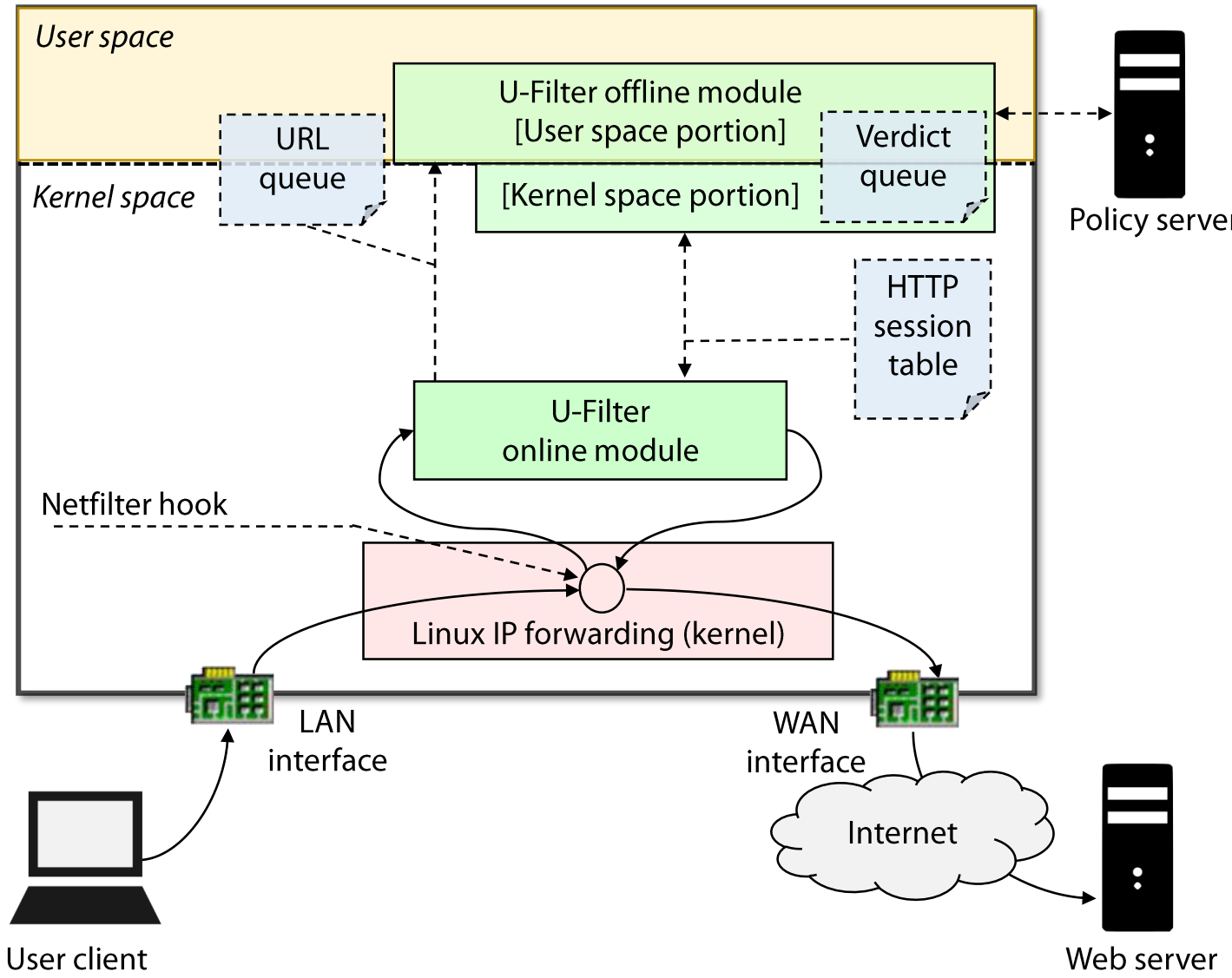


- A remote policy server:
  - keeps the URL database up-to-date
  - provides a fast API to request a policy check (URL → Verdict)
- Limited DPI to extract the URL from every HTTP request
  - No regex
- Policy compliance is verified without holding outgoing packets
- U-Filter holds at most one packet.
  - It can support a large number of sessions
- Additional latency reduced to the minimum

[3] R. Bonafiglia, A. Sapio, M. Baldi, F. Risso, and P. C. Pomi. Enforcement of dynamic HTTP policies on resource-constrained residential gateways. Elsevier Computer Networks, 2017.

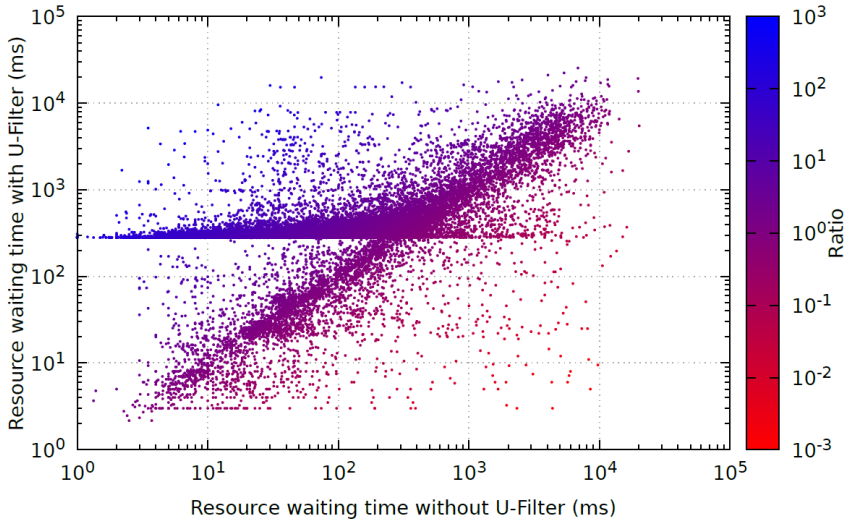
U-Filter architecture

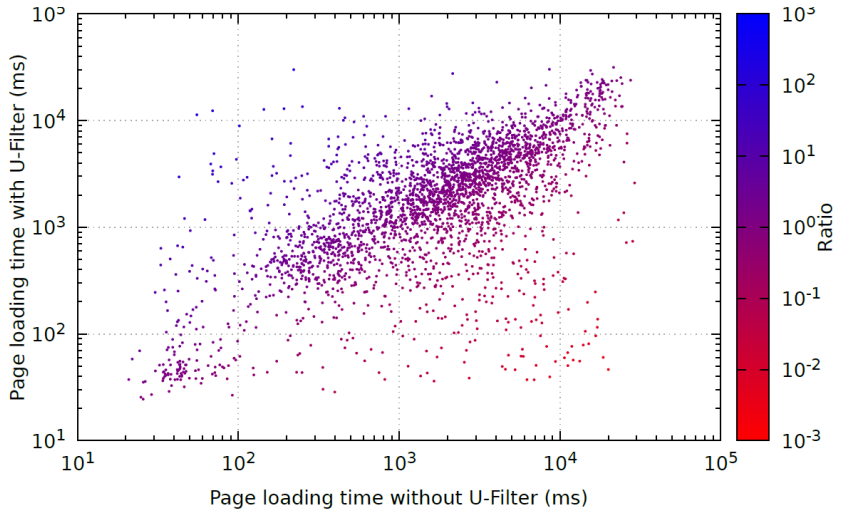
- Online module in the data plane:
  - Extracting requested URLs
  - Apply the policy decisions on the return traffic
- Offline module:
  - Queries the remote policy server
- Shared, efficient data structures.
  - The HTTP session table stores the first packet of the HTTP response, if the verdict is not yet available
  - The following packets are forwarded to the client



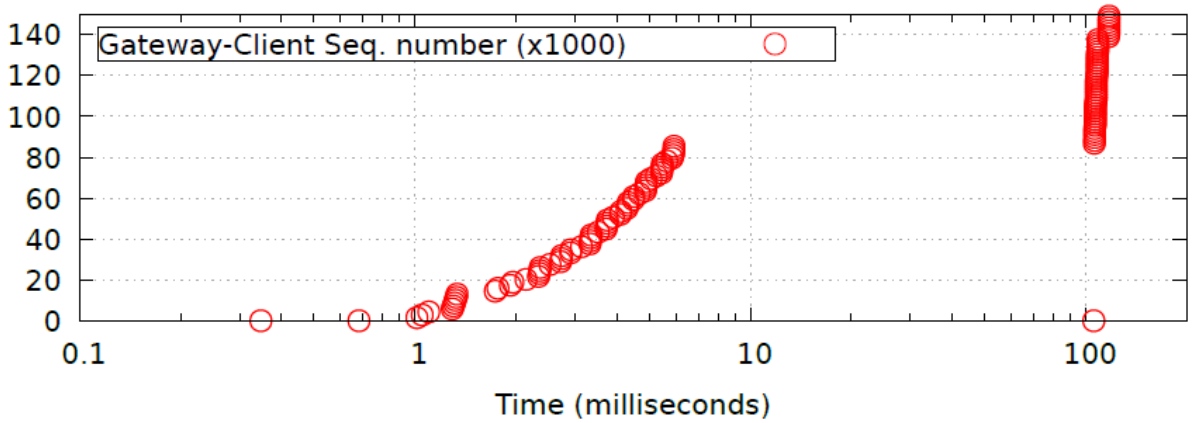
Experimental validation

Browsing experience

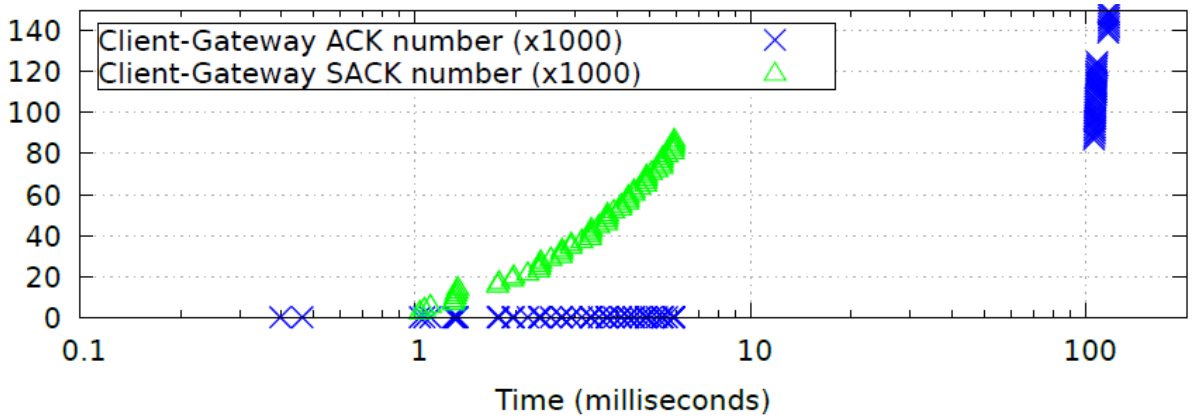




Interaction with TCP



Response packets timing on the LAN link



Acknowledgement packets timing on the LAN link

Conclusions

- Residential gateways can contribute to the execution value-added services:
  - Offloading complex operations to the cloud
  - Low overhead coordination is required

- Native Network Functions are a powerful abstraction to extend NFV to support heterogeneous devices

- U-Filter combines:
  - the advantageous location of the CPE with
  - the computational power of the cloud

  to reduce the overhead of complex services and the impact on the user experience

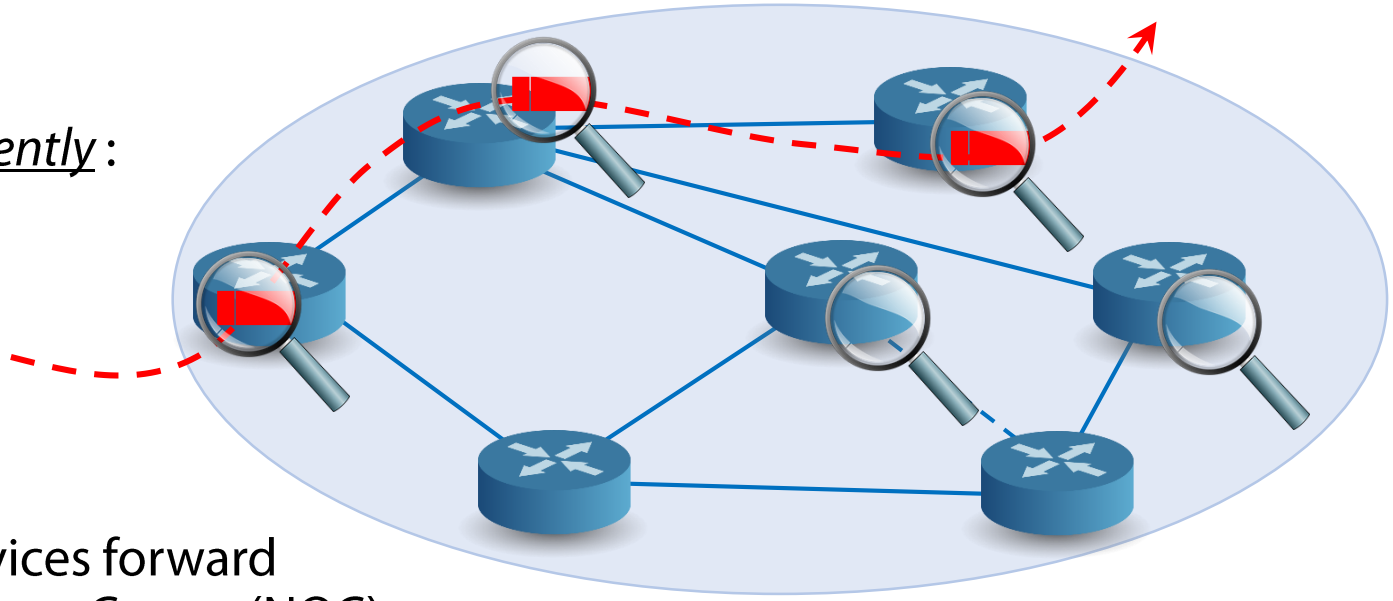
Network services in the core:

Distributed and coordinated packet processing

Traffic analysis: state of the art

- NSPs deploy multiple middleboxes in various locations of their network to obtain a comprehensive perspective of traffic and activities behind it

- These devices monitor packets *independently* :
  - Redundant processing
  - Duplicated reports
  - Inefficient use of resources



- To correlate distributed events, these devices forward the captured traffic to a Network Operations Center (NOC)
  - Significantly increases the amount of traffic in the network
  - High resource requirements for the NOC
  - De-duplication overhead

Massively Distributed Network Data Caching Platform[4]

- To improve the efficiency of network-wide traffic monitoring, we propose:

  MEDINA: a highly distributed and decentralized traffic processing platform

- Enhances traffic forwarding devices with the capability to process packets along a path in the network
- Significantly reduces the storage and processing requirements at the NOC and traffic overhead
- Proposes a limited overhead coordination and self-adaptation algorithm to distribute tasks across multiple devices

[4] A. Sapio, M. Baldi, F. Risso, N. Anand, and A. Nucci. *Packet Capture and Analysis on MEDINA, A Massively Distributed Network Data Caching Platform.* Parallel Processing Letters, 2017.

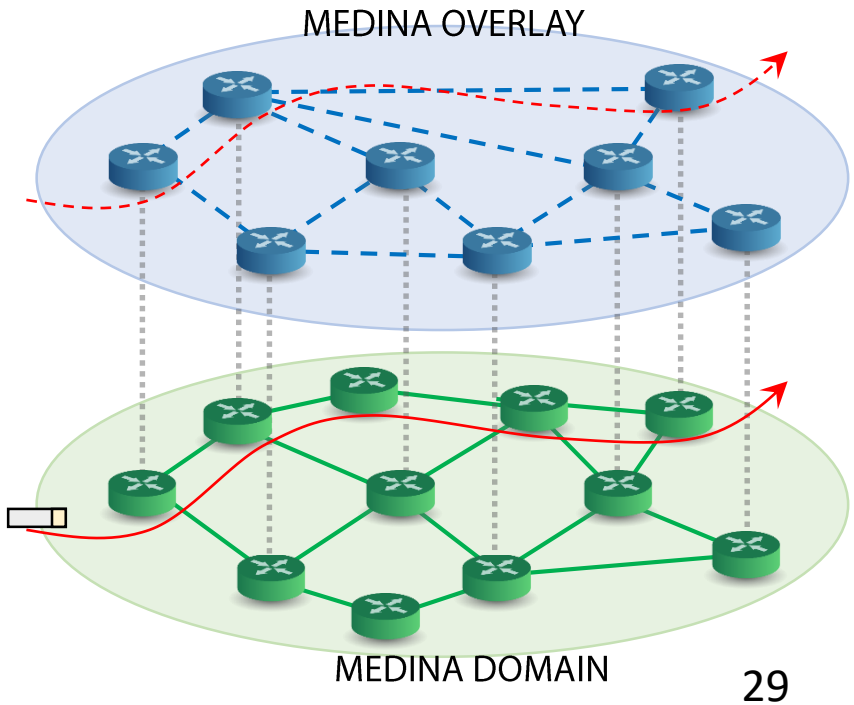
Distributed and coordinated traffic analysis

- Decentralized approach

- All MEDINA nodes are part of a Peer-to-Peer (P2P) network:
  - Share their capabilities and constraints
  - Update their constraints, throughout their operation

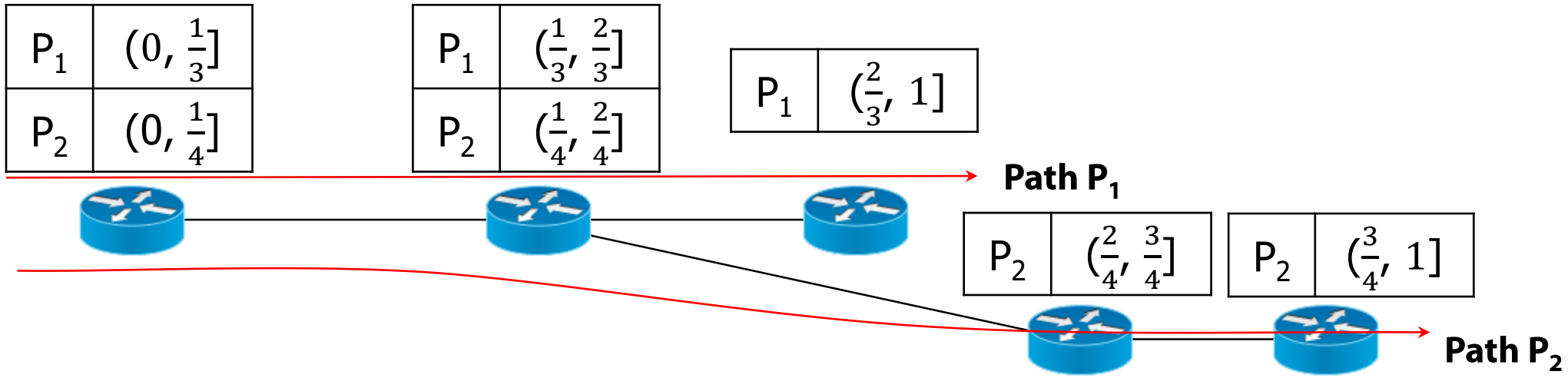
- They converge to a shared load distribution plan such that each packet is always captured:
  - Precisely *n* times
  - By different nodes along the route to its destination

- The load is distributed *fairly* considering:
  - The requirements of all the nodes
  - The path of each packet across the domain
  - The amount of traffic processed by each node



Hash-based selection mechanism

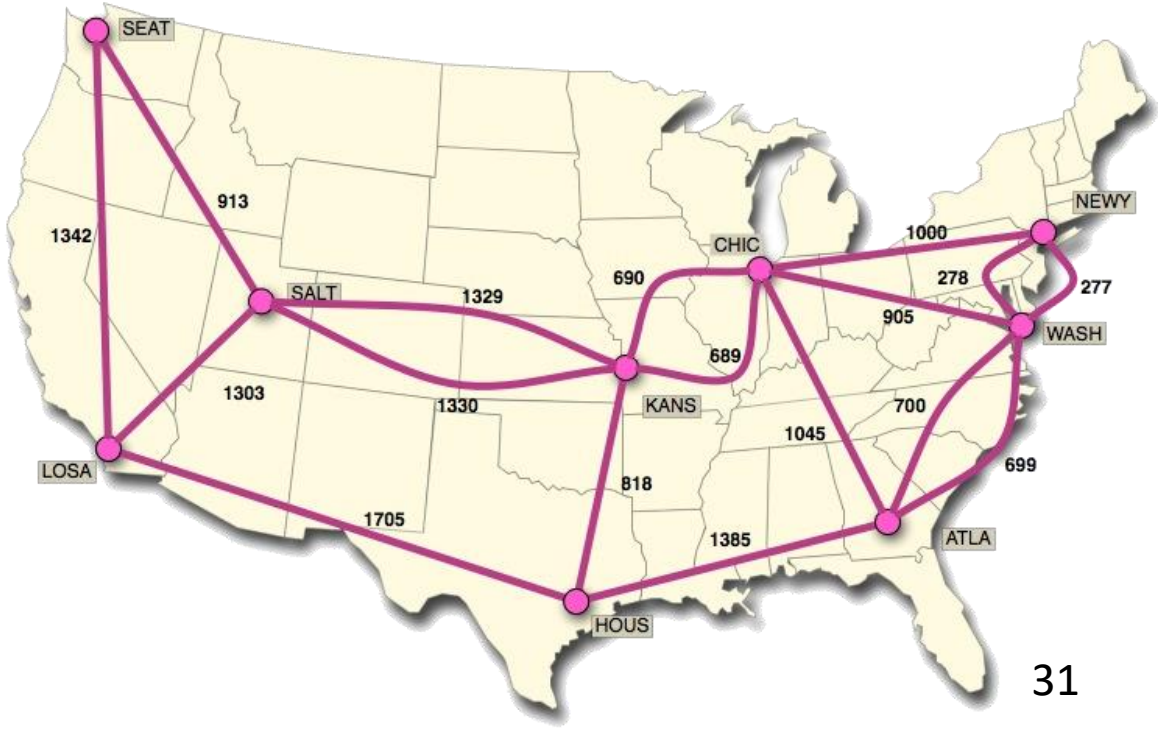
- Nodes compute an hash on part of each packet (*hash key*)
- The hash space is divided among the nodes sharing a path
- Every node is in charge of a fraction of the hash space
  - chosen considering its capabilities and the (variable) amount of traffic
- All the path-invariant, high entropy, fields of the IP header and all the bytes of the packet payload can be used as hash key



Experimental evaluation

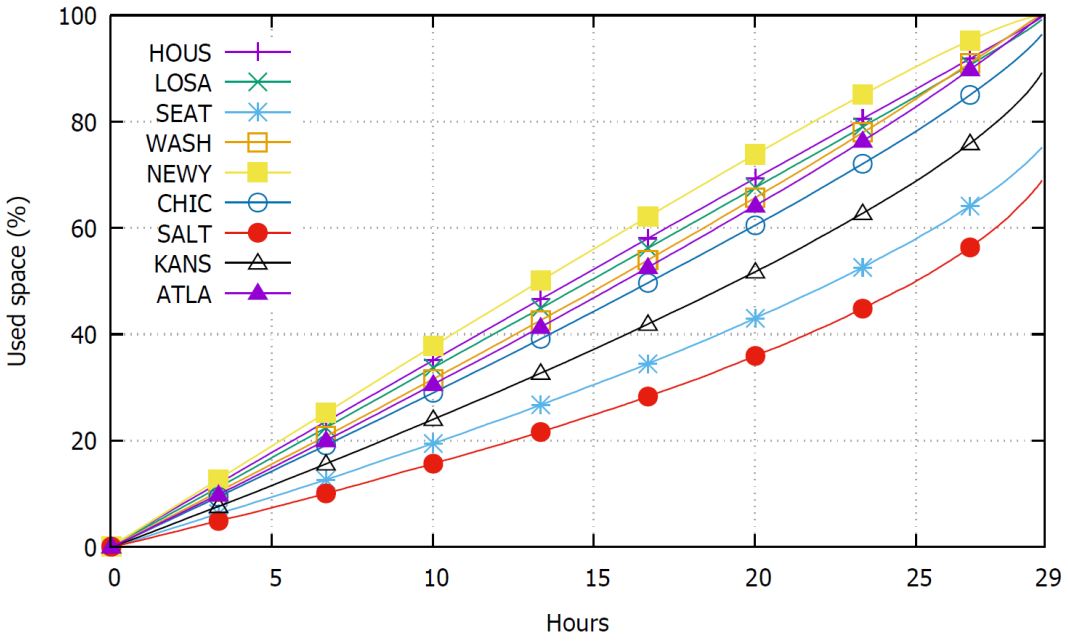
- We simulate a deployment on the *Internet2* network
- The amount of storage per node is proportional to the population of the city that it serves
  - with a ratio of 1 GB for each 100 people (Up to 84 TB for New York)
- We apply a gravity model to a baseline traffic volume T_b
  - 8 million IP flows per 5-minute interval
- Traffic volume $TV_{i,e}$ for each IE-pair:

$$TV_{i,e} = T_b * \frac{P_i * P_e}{\sum_{j,k \in B} P_j * P_k}$$

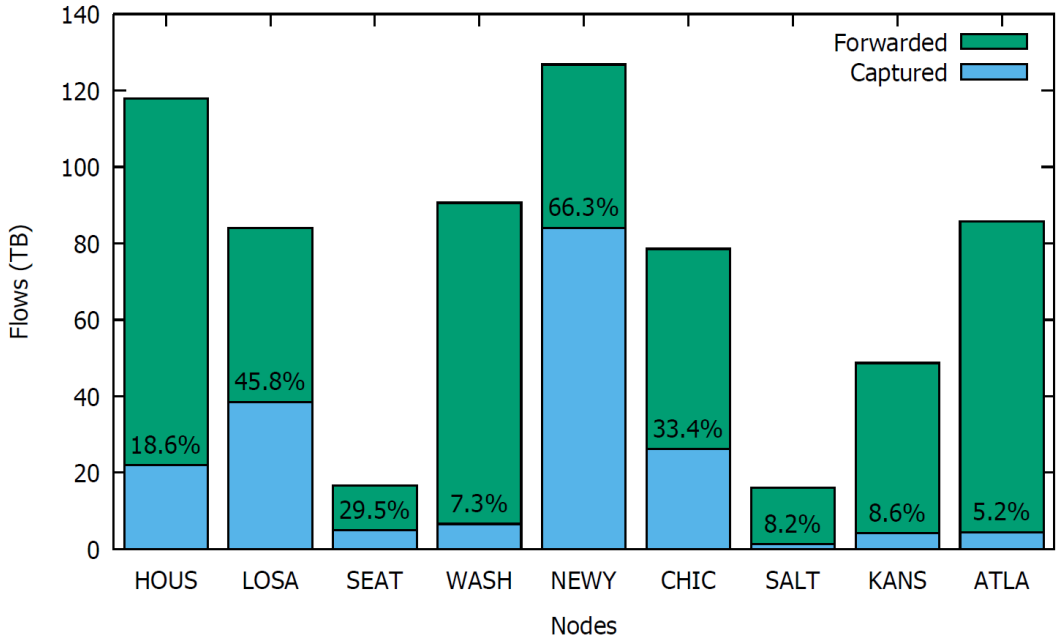


Experimental evaluation

- Used space over time
  - All the nodes in a path are completely full in 29 hours



- Processed traffic
  - On average, a node captures around 25% of the traffic it forwards



Conclusions

- MEDINA provides decentralized coordination among packet processing nodes

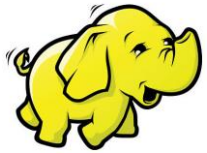
- The load distribution is autonomously adapted to changing traffic conditions, leveraging data shared by all the nodes in a path

The principles underlying MEDINA could be applied to generic processing and storage of data units forwarded through a network

Network services in the data center:

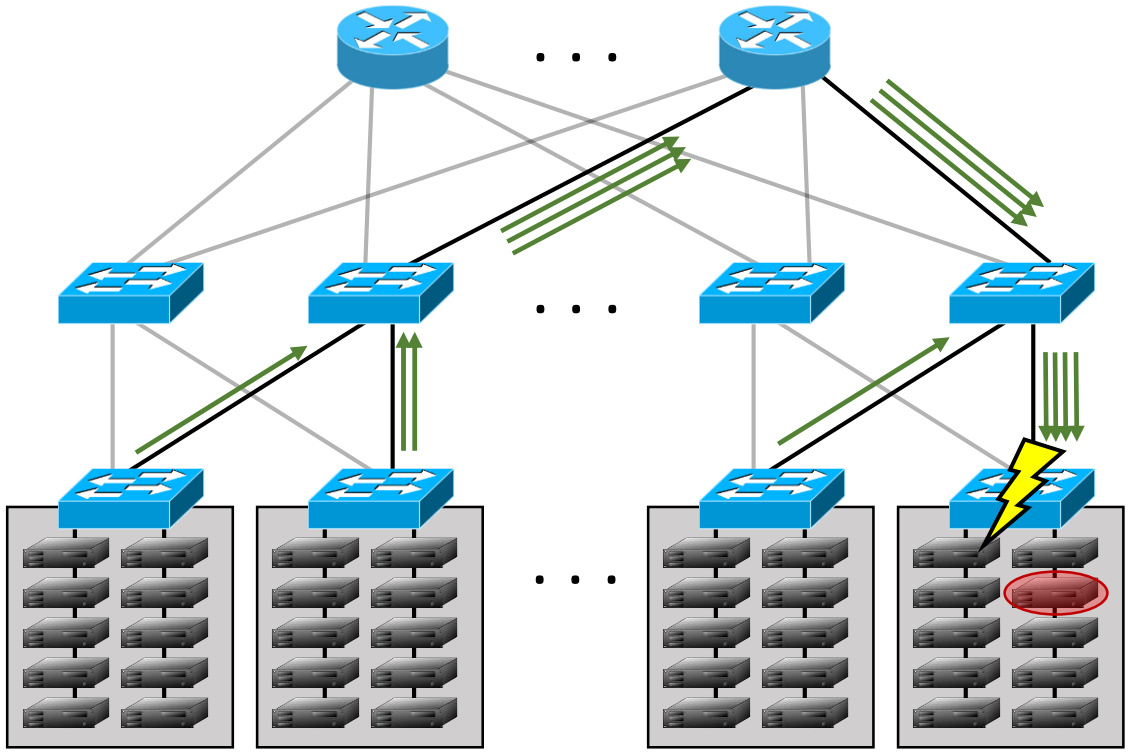
Programmable data plane

The problem



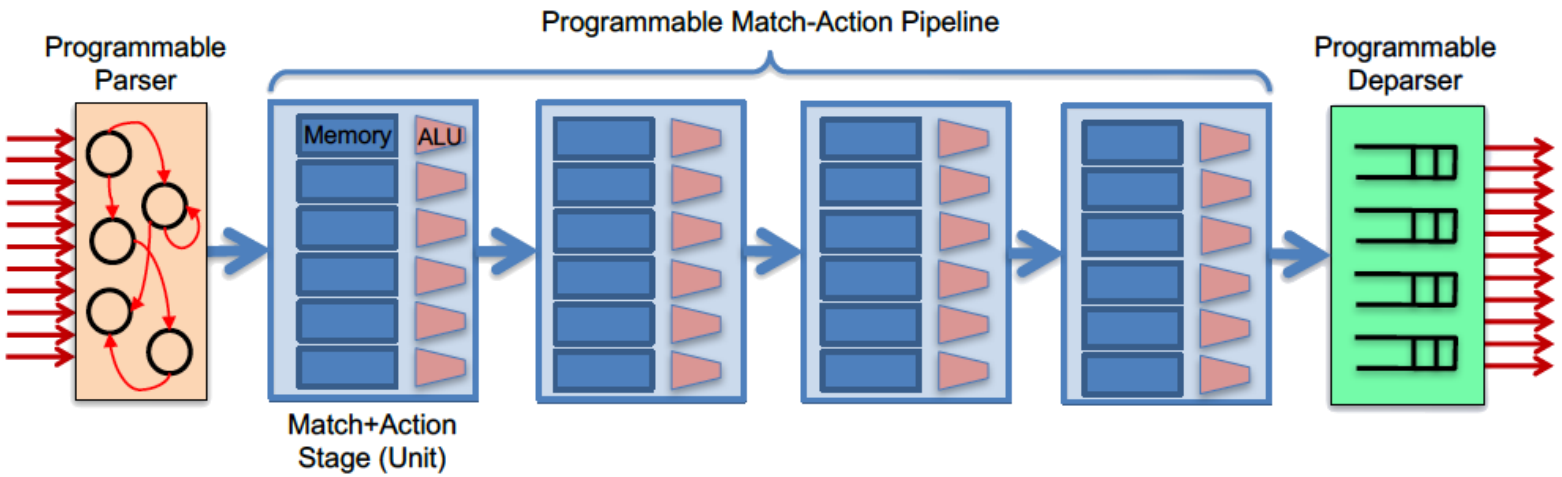
TensorFlow

- Data center applications scale by distributing data and computation across many servers
  - Massive amount of data exchanged through the fabric
- Given the sheer amount of traffic, the network becomes the bottleneck
- The communication cost can
  - increase the job completion time
  - reduce the accuracy of the result (for a fixed training time)



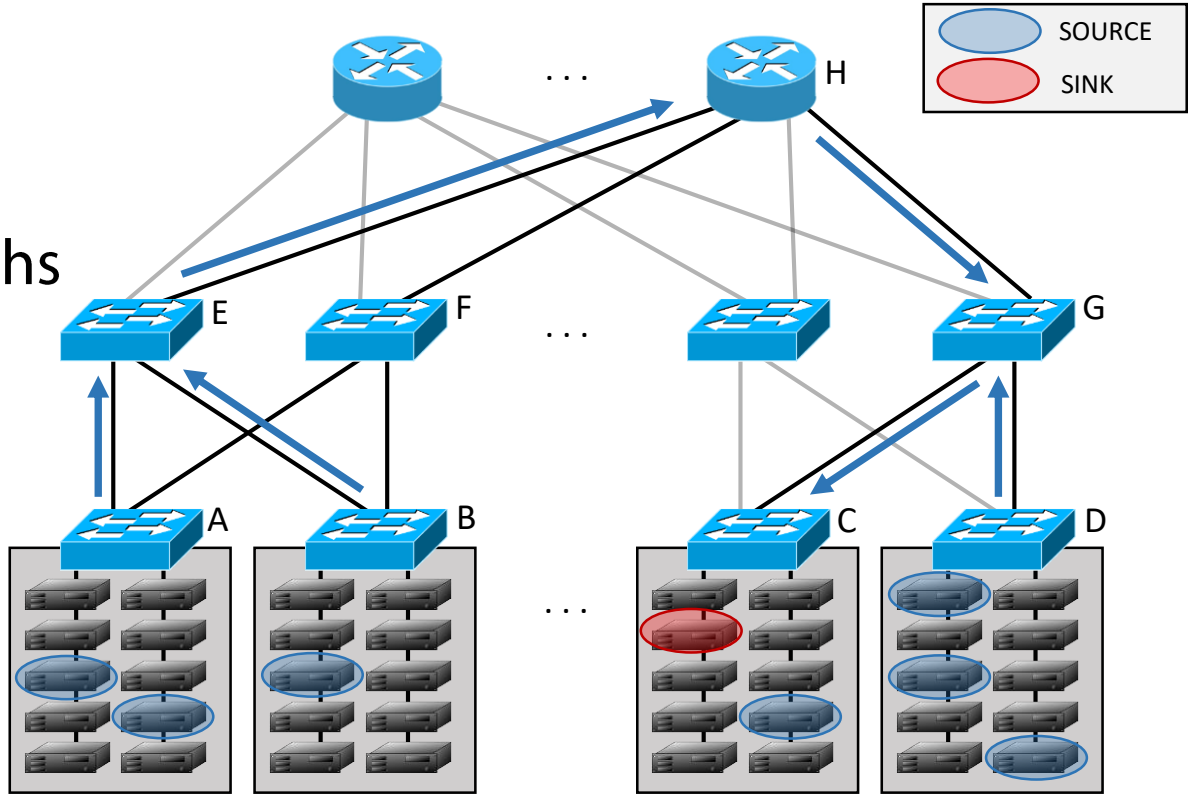
Programmable dataplane

- Switches with programmable pipelines.
  - Define your own parser and choose the set of possible actions
    - Packet modifications
    - Logic / arithmetic operations
    - State management
  - The actions are performed **at line rate (Tbps)**!
- Programmable SmartNICs
  - Hardware acceleration of flow processing



DAIET: Data Aggregation In nETwork[5]

- Offload part of the computation to switches and smartNICs
- Leverage programmable network devices to perform data aggregation along network paths



Benefits:

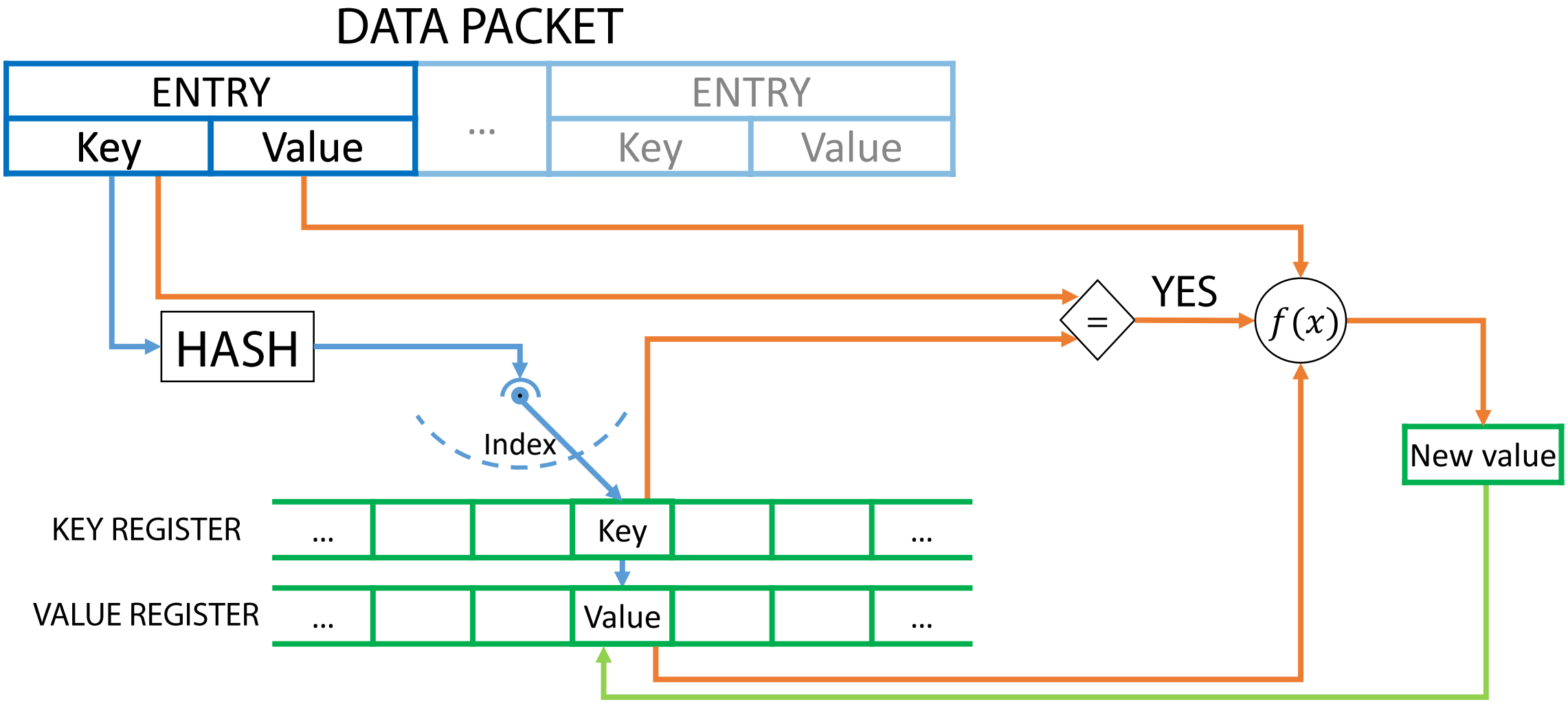
- Reduced traffic
- Lower bandwidth utilization
- Lower pressure on switch buffers
- Less work required by the CPUs/GPUs

[5] A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini, and P. Kalnis. *In-Network Computation is a Dumb Idea Whose Time Has Come.* ACM HotNets, 2017.

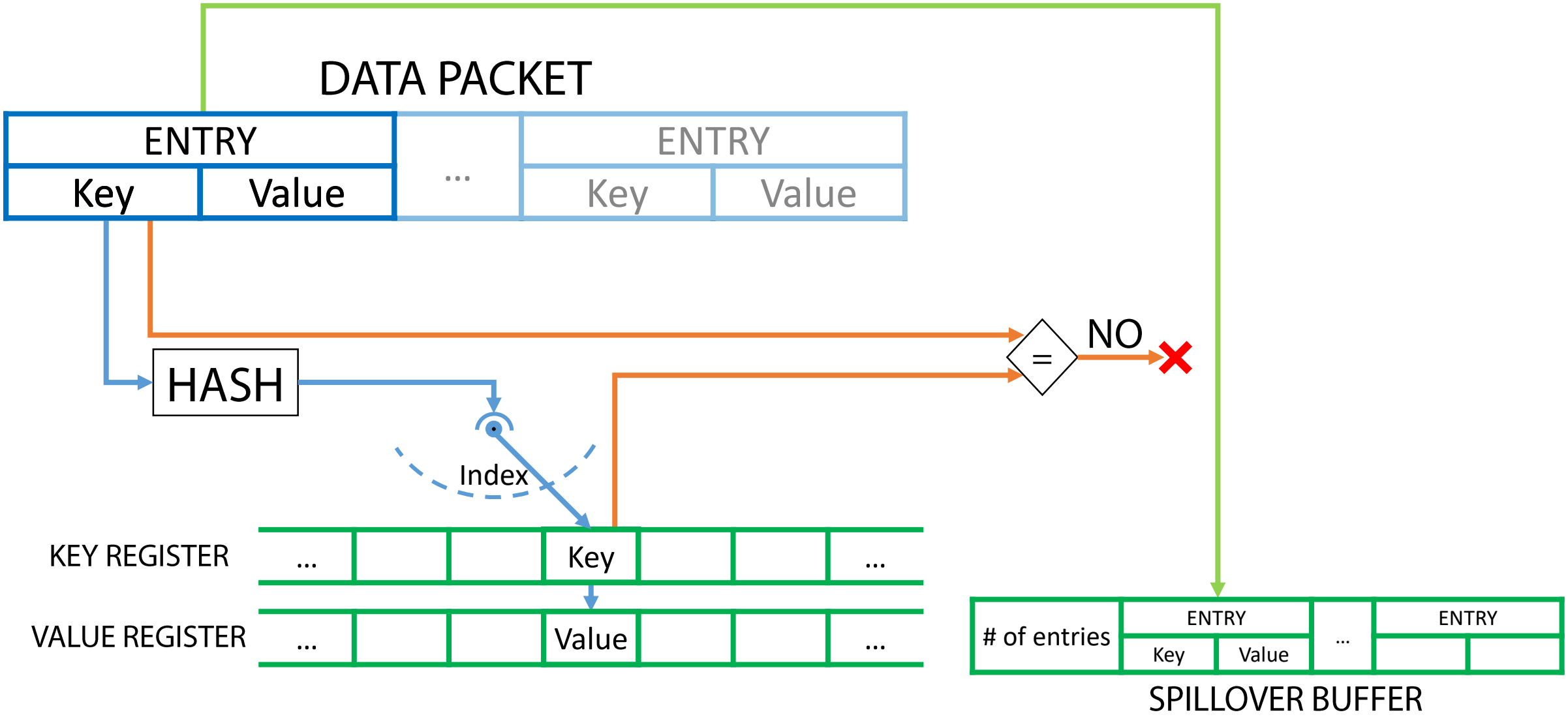
Challenges

- Limited available memory
  - Small TCAM; 20-30 MBs of SRAM
- Line rate processing
  - 5.12 ns per packet
  - Switches can process only 200-300 bytes per packet
- Small set of possible actions
  - Simple arithmetic operations (+, -, hash, but NO *, /, ^, sqrt)
  - No floating point (on most platforms)

Switch design in DAIET



Switch design in DAIET



Preliminary Evaluation

DAIET prototype in P4

Evaluate with a Word Count application in MapReduce

Questions:

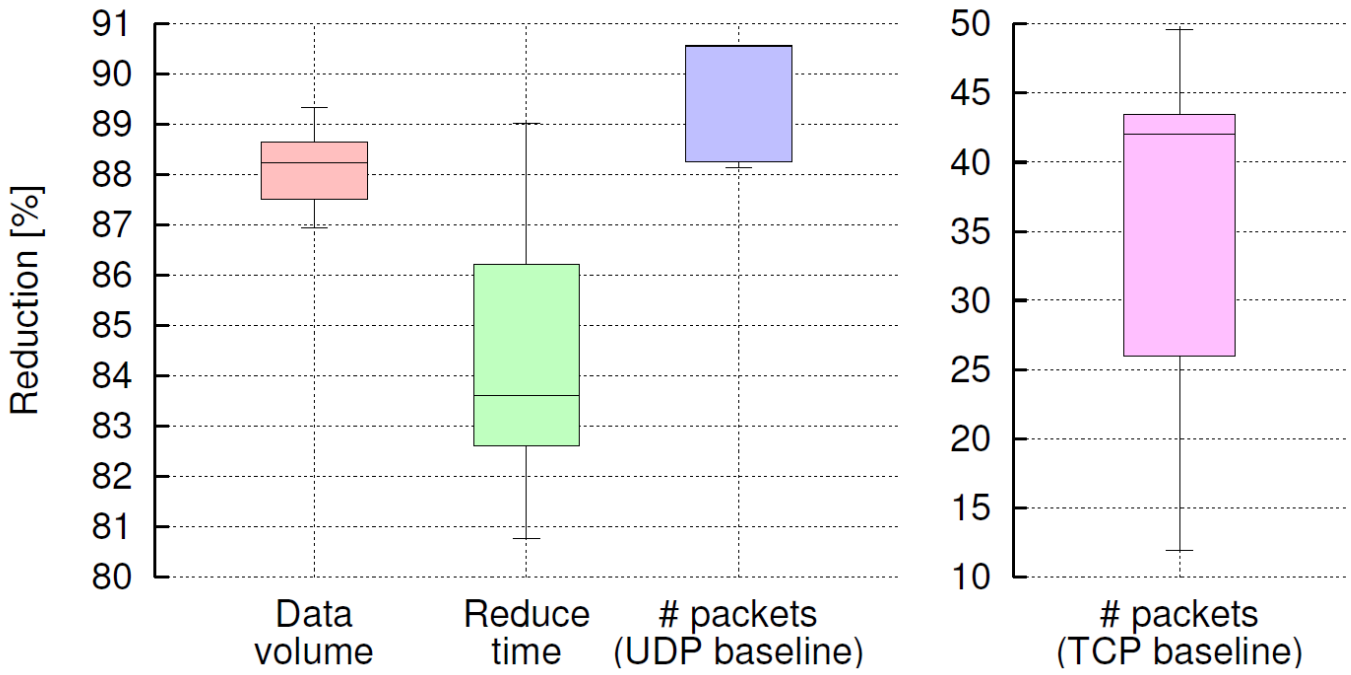
- How much is the traffic reduced?
- How much computation can we offload?
- Compare with TCP and UDP baselines (no in-network aggregation)

Settings:

- Emulated environment
- Single software switch
- 12 containers as workers
- 500 MB dataset

Data Aggregation Results

- 86.9%-89.3% reduction of the amount of data received by the reducers
- 83.6% median decrease in the execution time at the reducer
- 88.1% - 90.5% reduction of number of packets received by the reducers (UDP baseline)
- 42% median reduction of number of packets received by the reducers (TCP baseline)



Concluding remarks

- Computation can be offloaded to data plane hardware
  - with some limitations
- Spare CPU cycles and reduce network traffic
- Opportunistic in-network aggregation
- Applications:
  - Machine Learning
  - Batch, stream processing and graph analytics

Conclusions

Conclusions

- With modern networks, many opportunities arise for deploying services throughout the network infrastructure
- This dissertation shows how the different components of a modern NSP infrastructure can be used to provide several services designed factoring in their different characteristics and constraints.
- A judicious design of the service architecture is required to match the specific limitations
- NFV and programmable dataplane are two of the key enablers to provide additional distributed services that:
  - can simplify network management
  - reduce network overhead
  - be a new source of revenues for service providers

Publications

2015

- A. Sapio, M. Baldi and G. Pongrácz. **Cross-Platform Estimation of Network Function Performance.** IEEE EWSDN, 2015
- M. Baldi and A. Sapio. **A Network Function Modeling Approach for Performance Estimation.** IEEE RTSI, 2015

2016

- R. Bonafiglia, S. Miano, S. Nuccio, F. Risso and A. Sapio. **Enabling NFV Services on Resource-Constrained CPEs.** IEEE CloudNet, 2016
- M. Baldi, R. Bonafiglia, F. Risso and A. Sapio. **Modeling Native Software Components as Virtual Network Functions.** ACM SIGCOMM, 2016

2017

- R. Bonafiglia, A. Sapio, M. Baldi, F. Risso and P. C. Pomi. **Enforcement of dynamic HTTP policies on resource-constrained residential gateways.** Elsevier Computer Networks, 2017
- A. Sapio, I. Abdelaziz, M. Canini and P. Kalnis. **DAIET: a system for data aggregation inside the network.** ACM SoCC, 2017
- A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini and P. Kalnis. **In-network computation is a dumb idea whose time has come.** ACM HotNets, 2017
- A. Sapio, M. Baldi, F. Risso, N. Anand and A. Nucci. **Packet Capture and Analysis on MEDINA, a Massively Distributed Network Data caching platform.** Parallel Processing Letters, 2017

2018

- M. Baldi and A. Sapio. **Network Function Modeling and Performance Estimation.** International Journal of Electrical and Computer Engineering, 2018