Multitenant In-Network Acceleration with SwitchVM

Sajy Khashab, Alon Rashelbach, Mark Silberstein

NSDI '24

In-Network Computing

Offloading parts of applications from hosts to the network switches

Performance

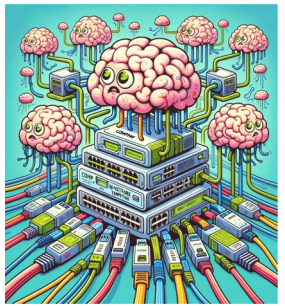
NetChain NetCache NetRPC

SwitchML Eris

NetPaxos HPCC

UnivMon P4xos

Beamer NetLock



Why aren’t

offering INCaaS to tenants?

Benefits of INCaaS to Tenants

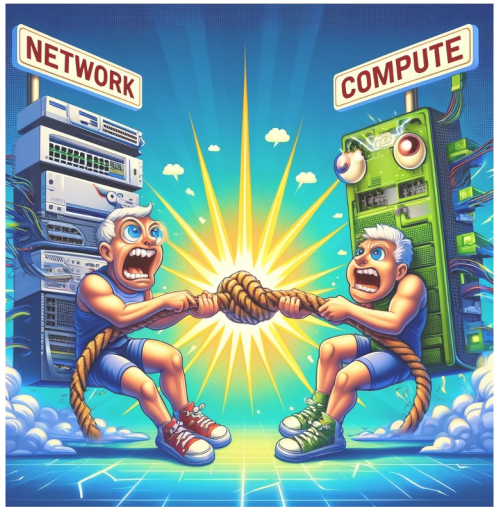
- Precise Network Monitoring and Performance Debugging
- Low Latency In-Network Services:
  - Consensus
  - Barriers
  - Key-Value Cache
- Application-Specific Load Balancing

and many more

INCaaS is NOT Possible Today

Existing hardware does not provide:

- ☐ Resource Isolation
- ☐ Fault Isolation
- ☐ Performance Isolation
- ☐ Runtime Programmability
- ☐ Virtual Network Isolation



Agenda

- Motivation
- **SwitchVM**
- Challenges
- Design
- Implementation
- Evaluation

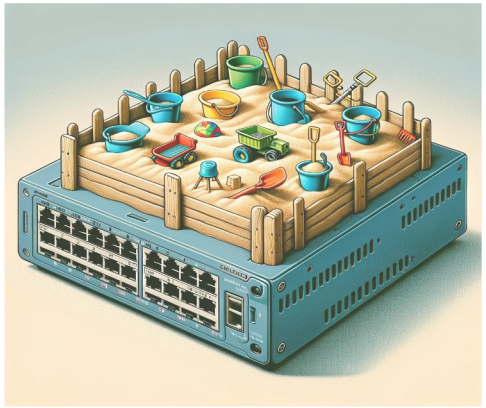
SwitchVM: In-Network Computing Secure Sandbox

- **Tenants**

  Write Data-Plane Filters (DPFs)

  For tenant's packets:

  - Access Packet/Switch Memory
  - Perform Computation
  - Change Routing within Virtual Network

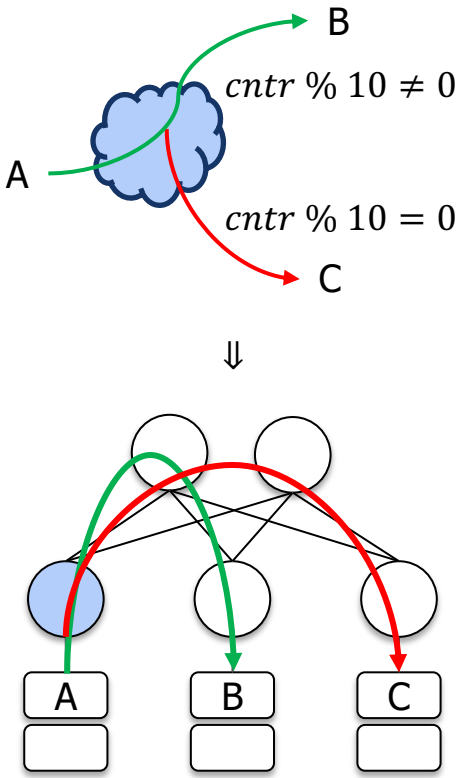


- **Operators**

  Control DPF permissions with per-tenant policies

SwitchVM Deployment

- **DPF**
  - Count packets from A to B
  - Send every 10-th packet to C
- **Setup**
  - Placement (Virtual Network)
  - Allocate Memory for Counter
  - Install VIP-to-PIP Mapping for C
  - Load DPF into Switch
    - or authorize in-packet DPF execution



Language-Level Virtualization

- DPFs are written in the SwitchVM ISA
  - RISC-like, Turing Complete
- SwitchVM = Interpreter
  - Load and execute DPF per-packet
- Sandbox enforced at runtime
- Inspired by Linux eBPF

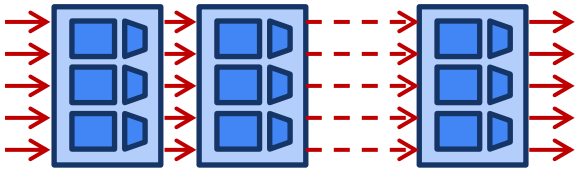
Instructions	Description	Instructions	Description
Data Movement		ALU Operations	
MOV	R1=R2	ADD SUB	A = A op B
LOAD_IMM	B=imm	AND OR XOR	
LOAD_CONST	B=const_tbl[imm]	LSH RSH NEG	A = op(A)
HASH	A=hash(A)	MIN MAX	A = op(A, B)
Memory Operations		Control Flow	
LOAD	B = mem[A]	HALT	Goto END
STORE	mem[A] = B	JMP	Goto pc
FAADD FAOR FAAND FAMAX	t=Mem[A] Mem[A]=op(t, B) B=t	BEQ BSET BLT BGT BLTS BGTS	Goto A==B ? pc_taken : pc_not_taken
Prolog		Epilog	
POP	R=stack.pop()	PUSH	stack.push(R)
PEEK	R=stack.peek()	FWD	fwd according to R
LOAD_MD	R=MD[md_idx]	NEXT_PC	pkt.next_pc = R
RAND	R=rng.get()	HDR_MOD	drop DPF from pkt

Agenda

- Motivation
- SwitchVM
- **Challenges**
- Design
- Implementation
- Evaluation

Challenge: SwitchVM on RMT Pipeline

- RMT Pipelines
  - Match-Action Table stages
  - Multiple tables per stage
- Stateful Memory
- Feed Forward
- Line-Rate Processing
- Program in P4
- Can’t add/remove Tables at Runtime
  - But can modify Entries.



SwitchVM: Design Principles

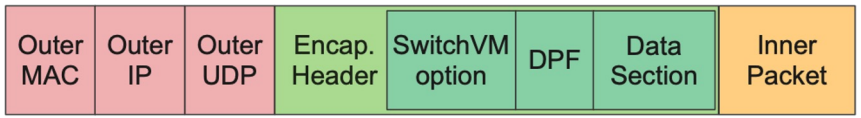
- ISA with Match-Action Tables
  - Key = Opcode, Action = Operation
- Tables shared by all DPFs
- DPFs on Switch (as table entries) or in Packet
- VLIW Instructions: Multiple ISA instructions per stage
- Sequential VLIW Instruction Execution

Agenda

- Motivation
- SwitchVM
- Challenges
- **Design**
- Implementation
- Evaluation

SwitchVM Packet

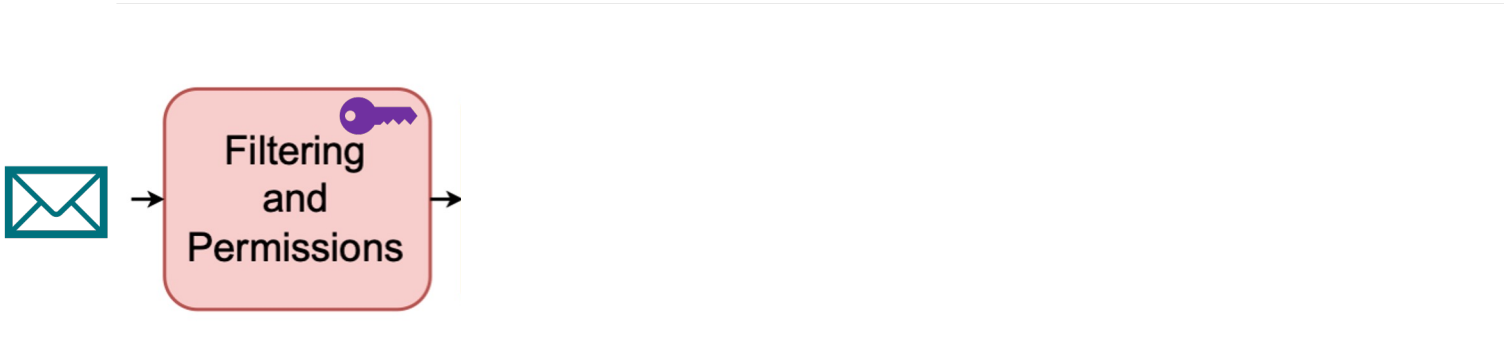
- Virtual Network Identifier = Tenant ID
- SwitchVM encapsulation header option
  - Program Pointer (points to on-switch program memory or in-packet)
  - Program Code (optional)
  - Data Stack (inputs/outputs)
- The same for all applications
  - Cannot define new headers.



SwitchVM Data-Plane Pipeline

① **Filtering**

- Authorized → execute
- Otherwise → forward

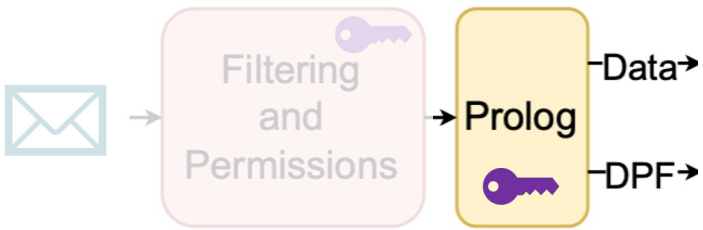


SwitchVM Data-Plane Pipeline

② **Prolog**

- Packet data stack
- Switch metadata

Prolog	
POP	R=stack.pop()
PEEK	R=stack.peek()
LOAD_MD	R=MD[md_idx]
RAND	R=rng.get()

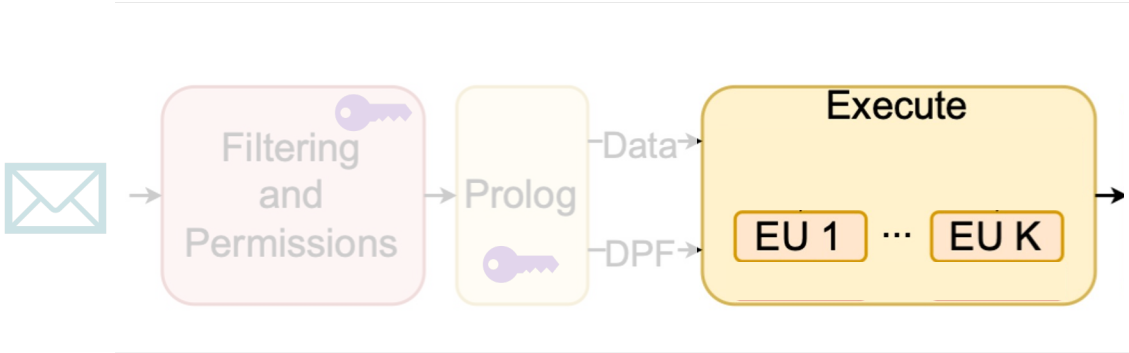


SwitchVM Data-Plane Pipeline

③ **Execution Pipeline**

- Execution Units – VLIW
- #EUs = #VLIW Instructions
- #Tables in Stage = VLIW Width

Instructions	Description	Instructions	Description
Data Movement		ALU Operations	
MOV LOAD_IMM LOAD_CONST HASH	R1=R2 B=imm B=const_tbl[imm] A=hash(A)	ADD SUB AND OR XOR LSH RSH NEG MIN MAX	A = A op B A = op(A) A = op(A, B)
Memory Operations		Control Flow	
LOAD STORE	B = mem[A] mem[A] = B	HALT JMP	Goto END Goto pc
FAADD FAOR FAAND FAMAX	t=Mem[A] Mem[A]=op(t, B) B=t	BEQ BSET BLT BGT BLTS BGTS	Goto A==B ? pc_taken : pc_not_taken

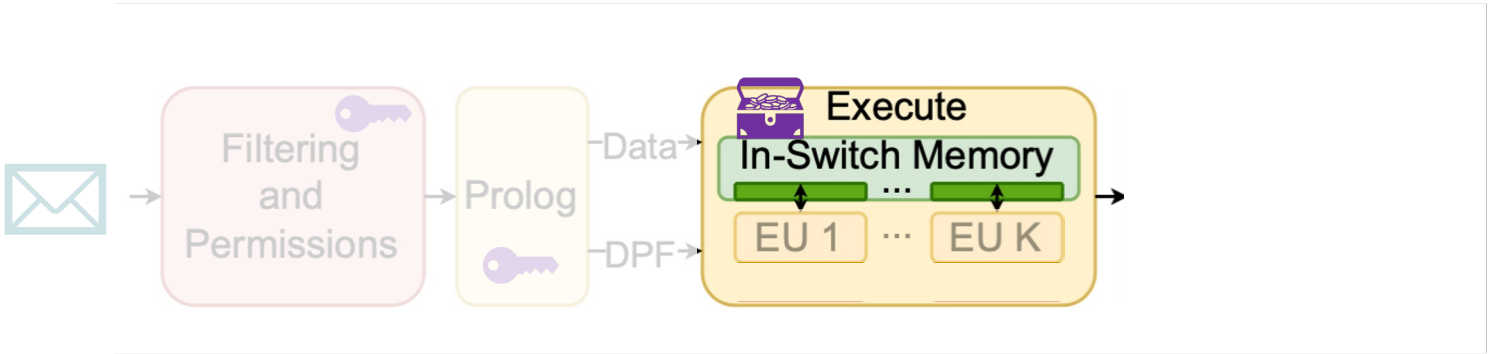


SwitchVM Data-Plane Pipeline

④ **Virtual Memory**

- Allocated by operator
- Ensure resource isolation

Memory Operations	
LOAD STORE	B = mem[A] mem[A] = B
FAADD FAOR FAAND FAMAX	t=Mem[A] Mem[A]=op(t, B) B=t

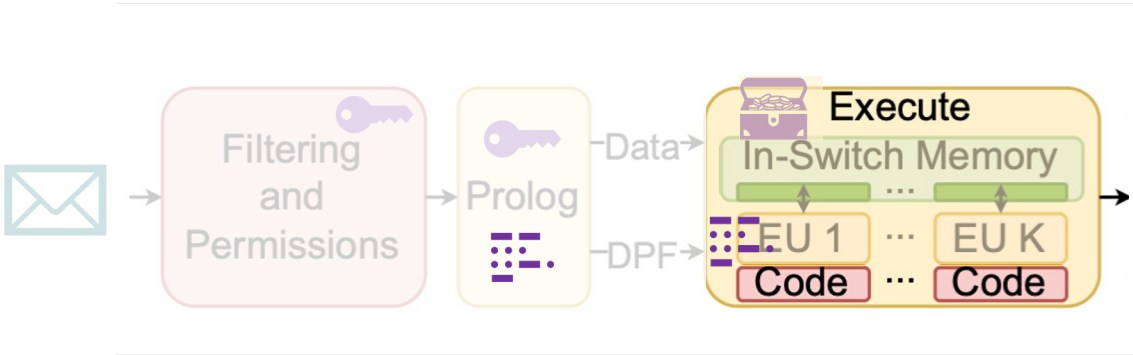


SwitchVM Data-Plane Pipeline

⑤ **Control Flow**

- Conditional branches and Jumps
- Code from switch/packet

Control Flow	
HALT	Goto END
JMP	Goto pc
BEQ BSET BLT BGT BLTS BGTS	Goto A==B ? pc_taken : pc_not_taken

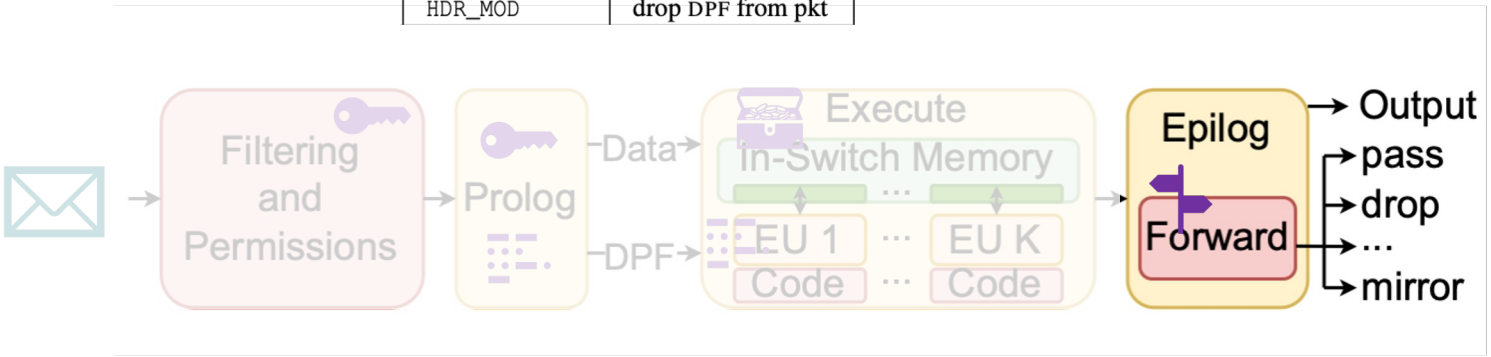


SwitchVM Data-Plane Pipeline

⑥ **Epilog**
- Push data to packet

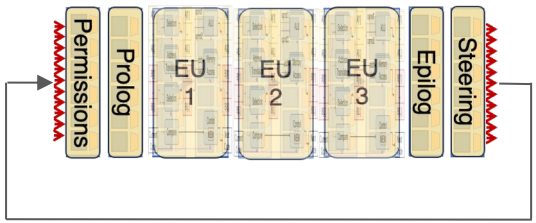
⑦ **Forwarding**
- Mapping installed at setup

Epilog	
PUSH	stack.push(R)
FWD	fwd according to R
NEXT_PC	pkt.next_pc = R
HDR_MOD	drop DPF from pkt



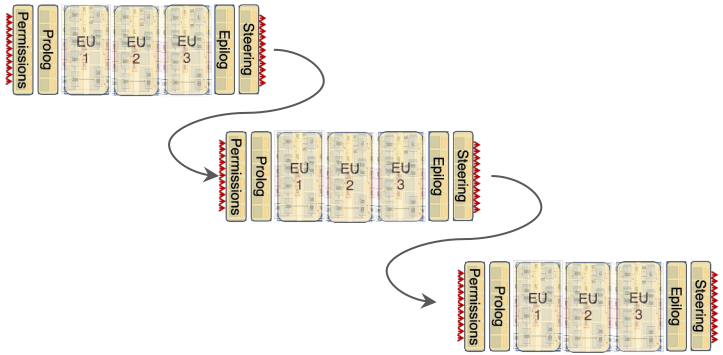
How to Support Longer Programs ?

Modify program identifier to DPFs in same switch and recirculate



DPFs are Turing Complete

Modify program identifier to point to DPFs in different switches



DPF Chaining

Agenda

- Motivation
- SwitchVM
- Challenges
- Design
- **Implementation**
- Evaluation

Execution Unit Structure

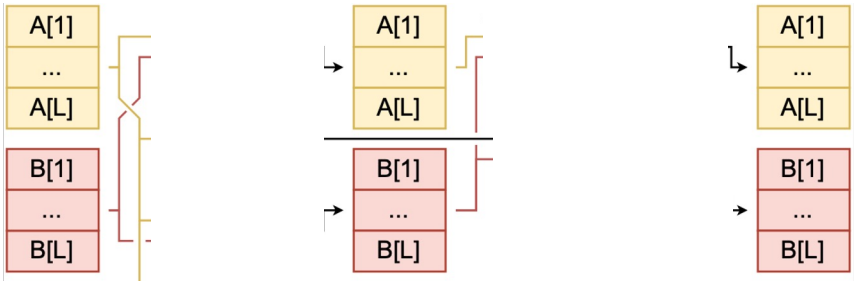
- VLIW Units, with MA tables (P4)
  - Limitation: Resource overhead
- Unit spans two stages
- Registers = A's and B's

- Data Movement
- ALU
- Memory
- Control Flow

➢ More details in paper

Data Movement	
MOV	R1=R2
LOAD_IMM	B=imm
LOAD_CONST	B=const_tbl[imm]
HASH	A=hash(A)

ALU Operations	
ADD SUB	A = A op B
AND OR XOR	
LSH RSH NEG	A = op(A)
MIN MAX	A = op(A, B)



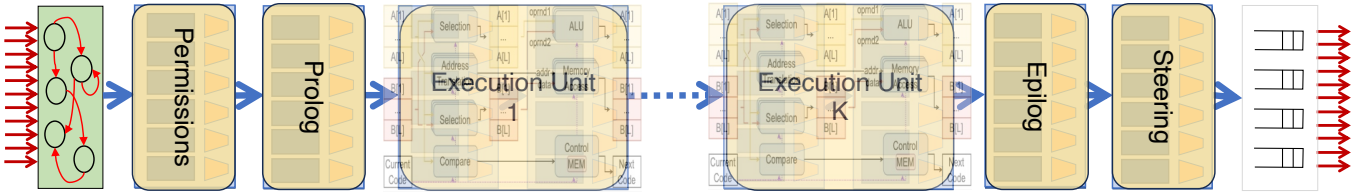
Control Flow	
HALT	Goto END
JMP	Goto pc
BEQ BSET BLT BGT BLTS BGTS	Goto A==B ? pc_taken : pc_not_taken

Memory Operations	
LOAD	B = mem[A]
STORE	mem[A] = B
FAADD FAOR FAAND FAMAX	t=Mem[A] Mem[A]=op(t, B) B=t

Mapping to Match-Action Pipeline

- Filtering – 1 stage
- Prolog – 1 stage
- Execution Unit – 2 stages
- Epilog – 1 stage
- Forwarding – 1 stage

Main Challenge
Making it all fit!



Limitations

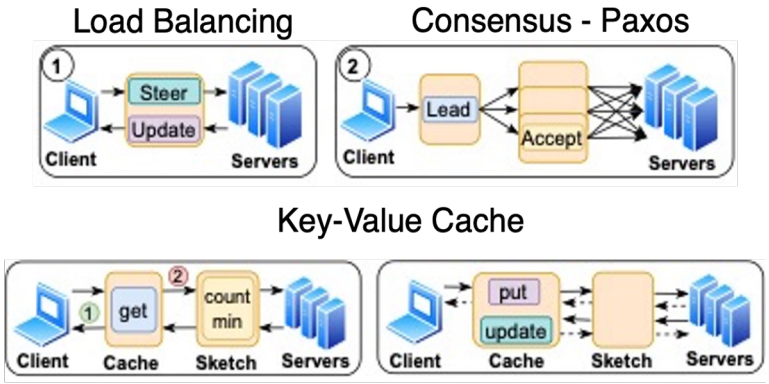
- Number of VLIW Instructions Per-DPF:
  - 4 in Tofino
  - 8 in Tofino2
- VLIW Structure:
  - 8x Movement
  - 4x ALU
  - 4x MEM
  - 1x Control
- High resource utilization
- Not compatible with P4
- Non-programmable parser

Agenda

- Motivation
- SwitchVM
- Challenges
- Design
- Implementation
- **Evaluation**

Evaluation

- Prototype using P4 on Intel Tofino
- Applications:



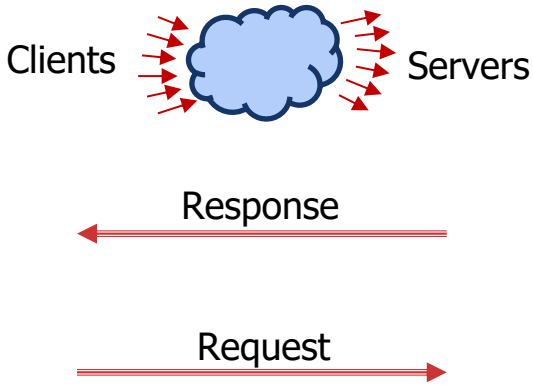
➢ More details in paper

Evaluation Results:

- End-to-End Applications
  - Load-Aware Load Balancer
    - In-Place Policy Replacement
  - In-Switch Cache for Key-Value Stores
- Microbenchmarks
  - Throughput vs. Latency
  - Performance Isolation
  - Scalability
  - DPF Bandwidth Overheads
- Resource Usage

Load-Aware Load Balancing

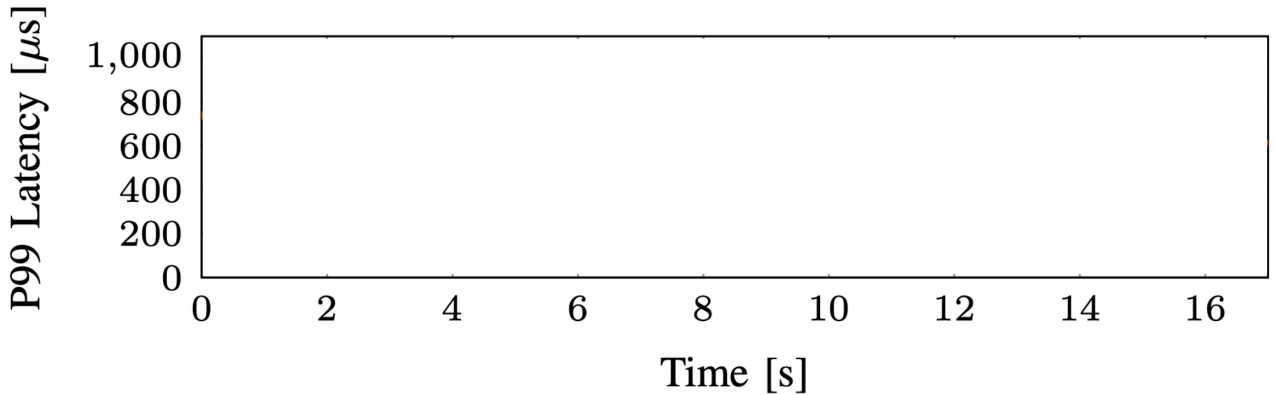
- Load balance requests across the different servers
- Switch stores current load on each server



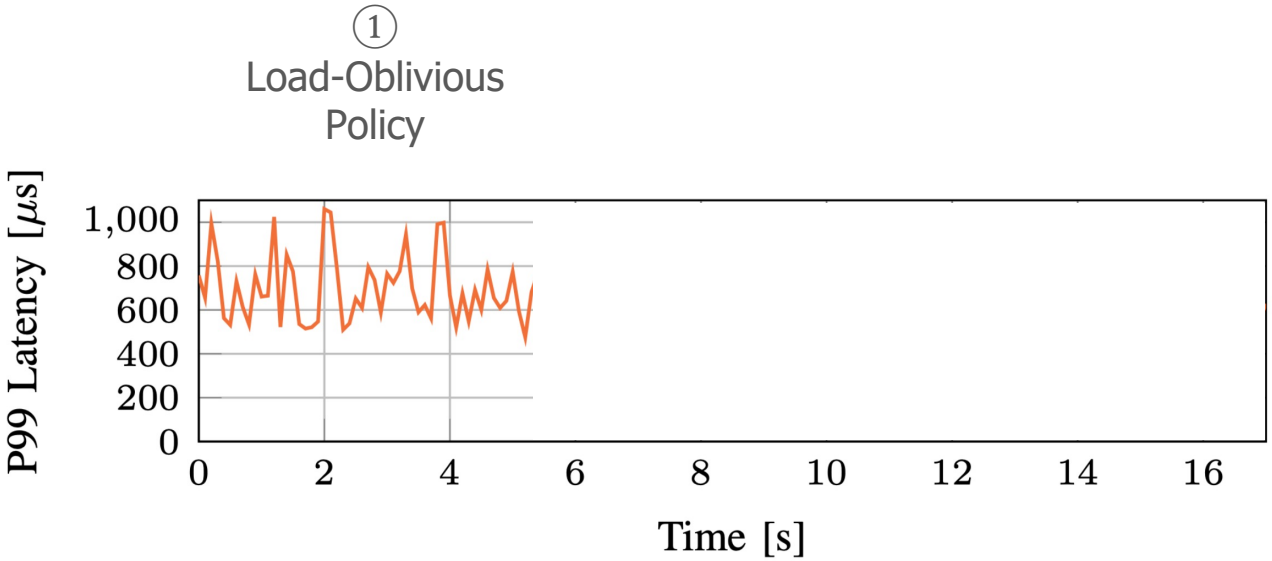
- **Response DPF:** Update server load
- **Request DPF:** Choose server
- **Policy**: Power of Two Choices
  Sample two servers, use less loaded one

Runtime Policy Change

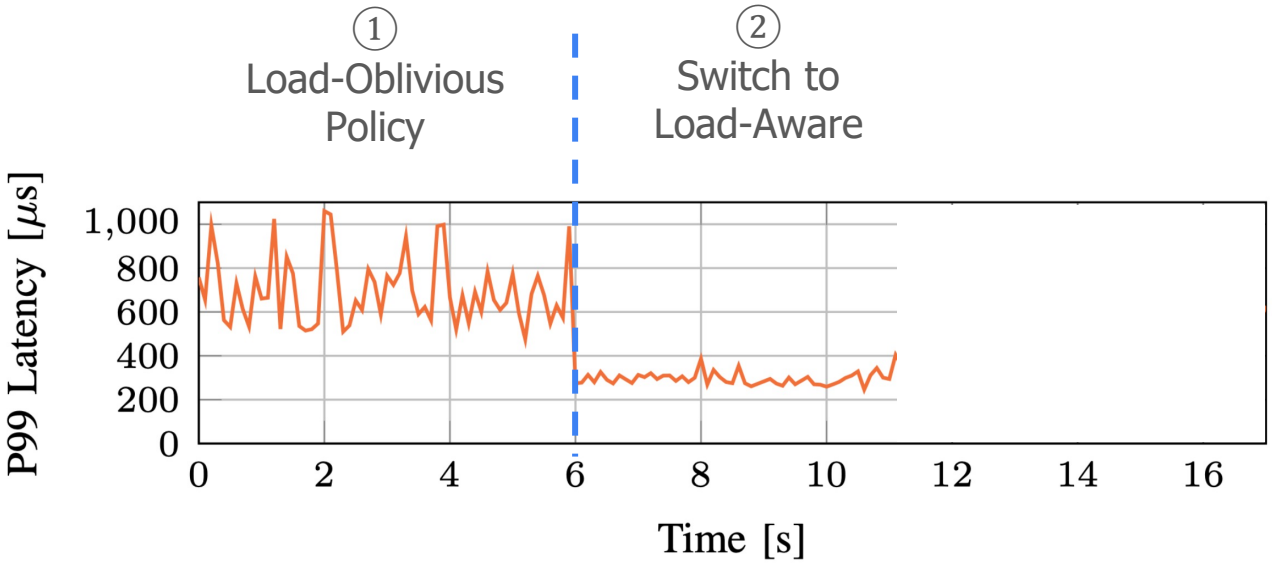
- **Response DPF**: Update load
- **Request DPF**: Two Policies (different DPFs)
  1. Load-oblivious
  2. Load-aware



Runtime Policy Change

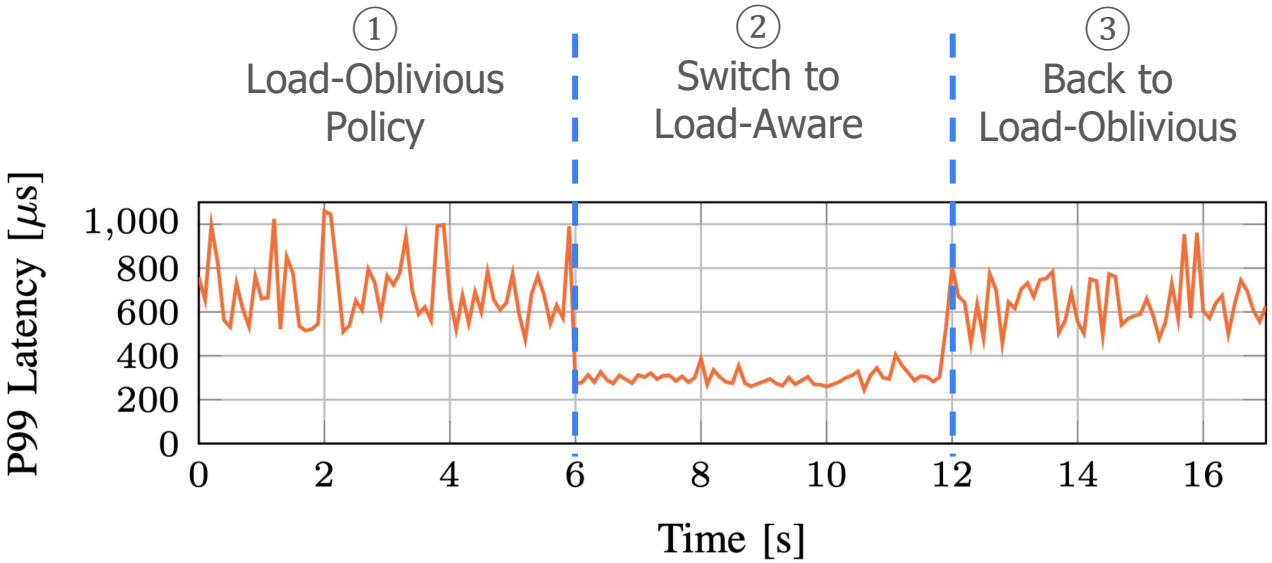


Runtime Policy Change



We record **NO** packet loss!

Runtime Policy Change



We record **NO** packet loss again!

Details in paper...

More Applications in Paper

- Key-Value Store Cache
- Count-Min Sketch
- Load Balancing
- Paxos Acceleration (Leader/Acceptor)
- Tiny Packet Programs
- Distributed Coordination
- Replication

Related Work

- P4 Hypervisors
  - Use a single P4 program, able to execute other P4 programs.
  - High performance/resource overheads
- Source Code Merge
  - Merge a few programs at compile-time into a single super program, enforcing isolation.
  - Requires recompilation upon changes, space-partitioning of resources.
- New Hardware

Conclusion

- SwitchVM provides an In-Network Computing Secure Sandbox
  - ✓ Fault Isolation
  - ✓ Resource Isolation
  - ✓ Performance Isolation
  - ✓ Runtime Programmability
  - ✓ Resource Sharing
  - ✓ Virtual Network Isolation
- SwitchVM: Gives access to In-Network Computing in multitenant environments, with existing hardware switches.

Thank You!

Sajy Khashab
sajykhashab@gmail.com

Alon Rashelbach
alonrs@campus.technion.ac.il

Mark Silberstein
mark@ee.technion.ac.il