SoftRing: Taming the Reactive Model for Software Defined Networks

Chengchen Hu, **Kaiyu Hou**, Hao Li, Ruilong Wang
Peng Zheng, Peng Zhang, Huanzhao Wang

MOE KLINNS Lab
Xi'an Jiaotong University

西安交通大學
XI'AN JIAOTONG UNIVERSITY

Match-Action Abstraction

- Software Defined Networking (SDN) employs a *Match-Action* model as processing abstraction

- A packet: ① match a rule in switches

  ② execute the related actions

  - Forward, Drop, Modify

- Flow entry stored in Flow Table

Proactive vs. Reactive

Proactive

- Pre-install flow entry in switches
- Anticipate the network issues in advance

Reactive

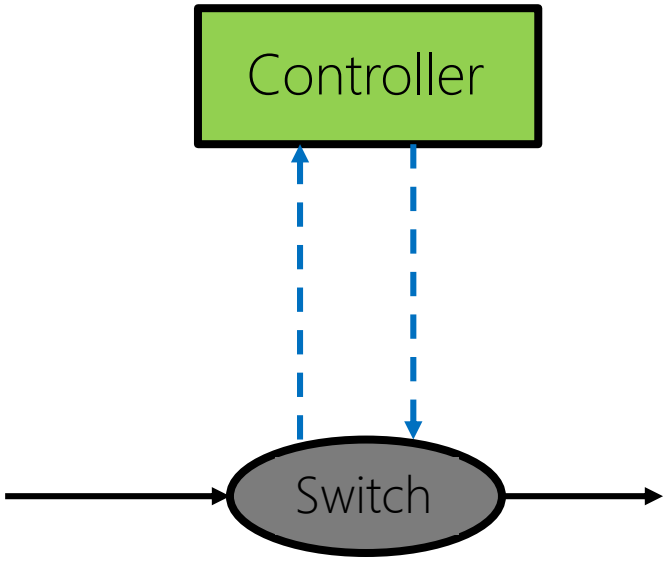
- Switches invoke the controller on network event
- Controller pushes corresponding flow entry

Proactive Model Only?

- Limited flow entry memory in switch
  - TCAM, up to 5000~ flow entries
  - Limited flexibility, finer-grained control
- Unawareness of data plane
  - Controller cannot be aware of failures
  - Flow entry removed: bugs, attacks
    - Continuously drop packets

Overhead in Reactive

- Reactive introduces communicate overhead between controller and switches (South-Bound Interface)



- Table-miss event
  - One common trigger of reactive
  - Example: OpenFlow

Table-miss

- New incoming packet doesn't match any flow entry
  - Raise a **Table-miss** event

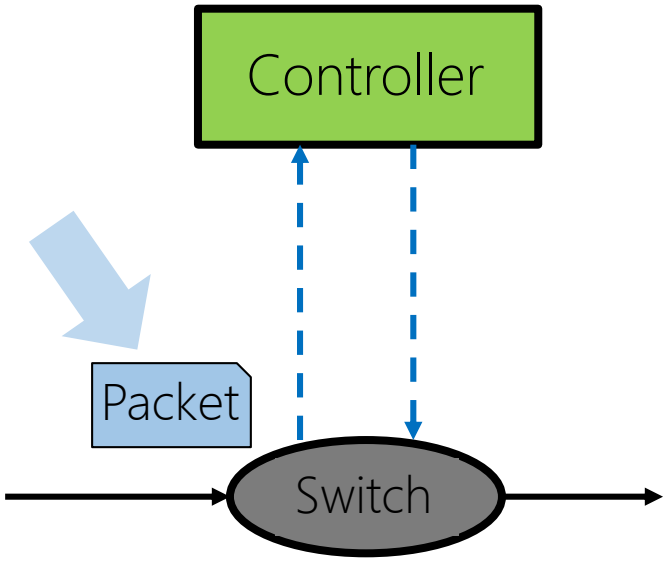


Table-miss

- Switch buffers the packet
- Sends a ***Packet-in*** message to controller
  - first 128 bytes of the packet

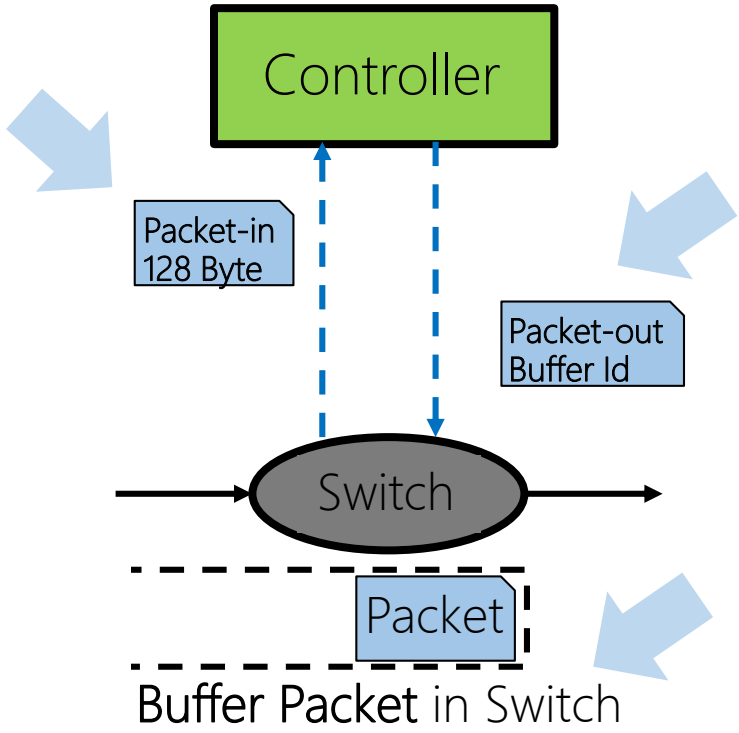


Table-miss

- The buffer overflows
- Switch has to send the entire packet through the ***Packet-in*** message

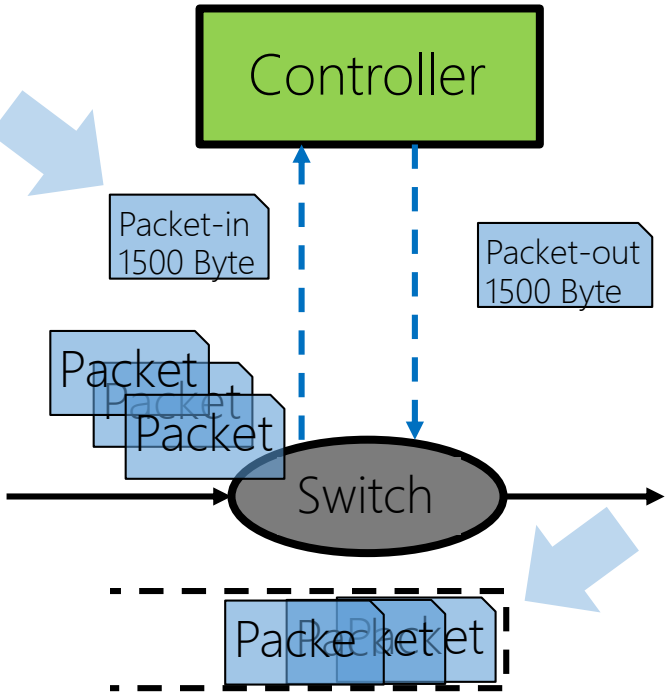
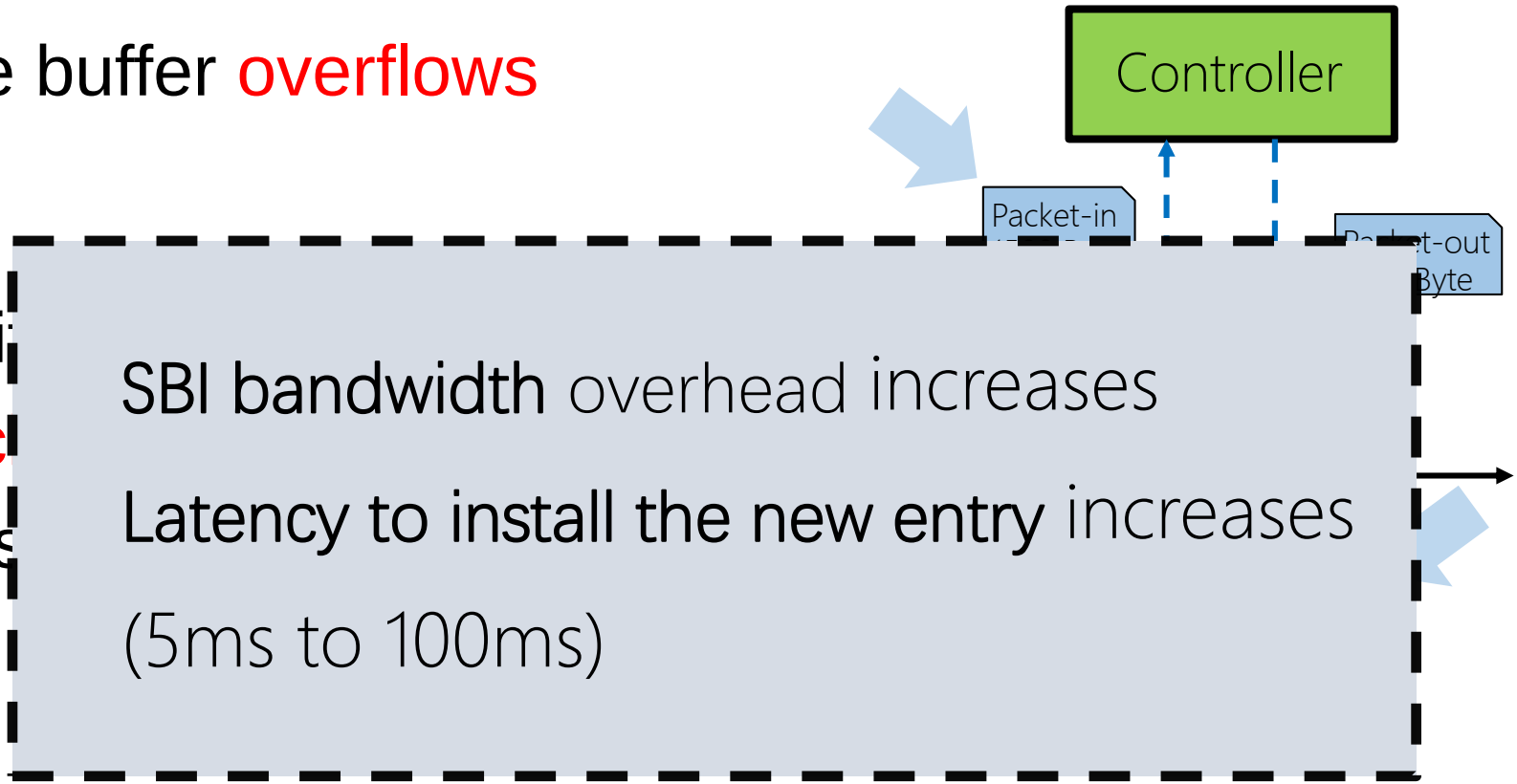


Table-miss

- The buffer overflows
- Swi
  pac
  mes



When Does Reactive Happen?

1. Small flows come successively.
   - 20% traffic occupies 80% of total flows.

2. Limited flow entry memory, replacement
   - E.g., LRU Policy
   - Subsequent packets of TCP can trigger Table-miss

3. Subsequent packets also trigger Table-miss
   - Before related new rule installed

Where to buffer the Table-miss packet?

- Keep in Switch

Brand	Model	Port	Buffer	Control BW
Pica8	AS7712-32X	100GbE*32	16MB	1000Mbps
Brocade	ICX7750-26Q	40GbE*26	12.2MB	1000Mbps
Dell	Z9100-ON	100GbE*32	16MB	1000Mbps
Huawei	CE8860	100GbE*32	16MB	N/A
Netgear	M5300-52G	48+10GbE*4	4MB	10Gbps

  - Buffer overflow: new flow exceeds 0.26% of switch capacity

- Send to Controller

  - Only 0.06% of the traffic consume all the control bandwidth

Goal of SoftRing

1. Reduce Buffer
2. Reduce Bandwidth
3. Less Packet Drop
4. Compatibility

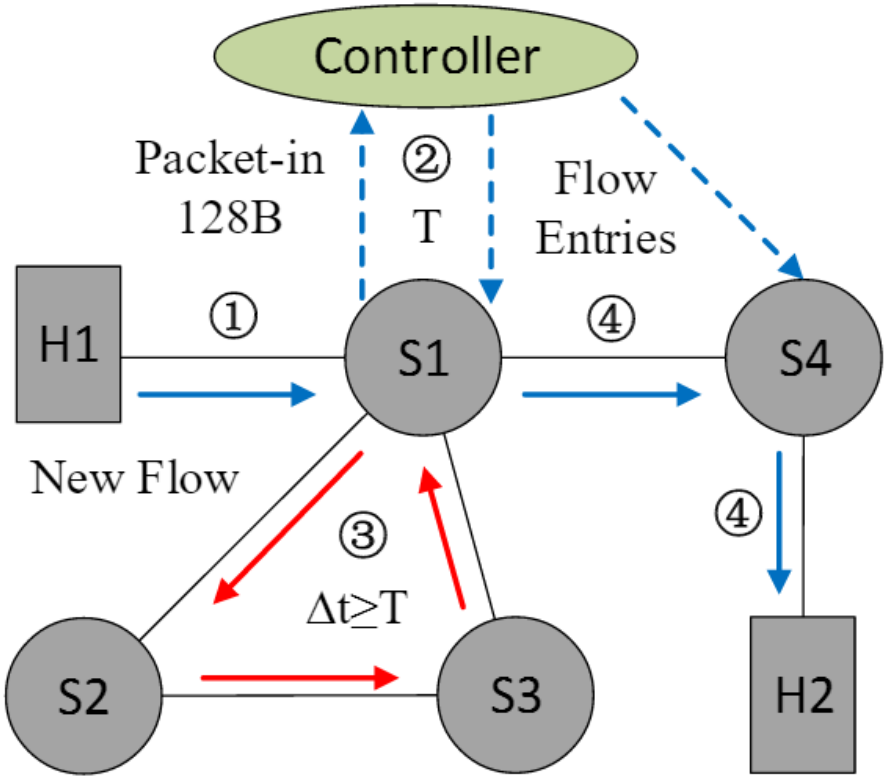
Popular Restaurant Policy

Popular restaurant cannot have enough tables to serve everyone (“table-miss”)

Waiters: record the customer, arrange a waiting number

Customers: walk away and back later when the seat availability.

Basic Idea



- Extend loops in network
- Remaining bandwidth and memory in dataplane
  - Internet: > 50%
  - Data Center: > 75%
- Accurate control of packet forwarding

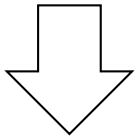
Three Challenges

1. Seek and Select waiting-loops
2. Enforce waiting-loop policy in switches
3. Dynamically adjust waiting-loops

1 Seek and Select

Loop Seeking Algorithm

(collect enough loops as waiting-loop candidate)



Loop Selection Algorithm

(select a loop subset to cover all the switches)

Loop Seeking Algorithm

- Based on Johnson's loop searching algorithm
  - Directed graph
- Accelerate
  - Graph partition
  - Loop length and scale control
  - Random shuffle
- Virtual Loop
  - $v1 \rightarrow v2 \rightarrow \ldots \rightarrow v2 \rightarrow v1$

Waiting-loop Candidate Set

Loop Selection Algorithm

- Select a loop subset to meet the requirements:
  - Cover all the switches $\bigcup_{j=1}^{|S|} V_j = \mathrm{V}$
  - Minimize the extra delay $0 \le a_j L_j - T \le \alpha$
  - Bandwidth in Data Plane $C_i = \sum_j a_j I_{ij}$
  - Minimize the extra flow entries $N_i = \sum_j I_{ij}$
- Map to weighted set cover problem

2 Switch Enforcement

	Priority	Match	Action	Timeout
OpenFlow	0	*	① Buffer **Table-miss** packet in switch ② Send **Packet-in** with 128 Bytes	0

2 Switch Enforcement

	Priority	Match	Action	Timeout
OpenFlow	0	*	① Buffer **Table-miss** packet in switch ② Send **Packet-in** with 128 Bytes	0
ES (Entry Switch)	0	*	① Send **Packet-in** with 128 Bytes ② Push VLAN = 1 ④ Send to next switch by queue	0

✓ VLAN tag: distinguish waiting-loop packet

2 Switch Enforcement

	Priority	Match	Action	Timeout
OpenFlow	0	*	① Buffer **Table-miss** packet in switch ② Send **Packet-in** with 128 Bytes	0
ES (Entry Switch)	0	*	① Send **Packet-in** with 128 Bytes ② Push VLAN = 1 ④ Send to next switch by queue	0
LS (Loop Switch)	1	VLAN = 1 in-port = pre-switch	② Send to next Switch by queue	0

- ✓ VLAN tag: distinguish waiting-loop packet
- ✓ Priority: never affect normal traffic

2 Switch Enforcement

	Priority	Match	Action	Timeout
OpenFlow	0	*	① Buffer **Table-miss** packet in switch ② Send **Packet-in** with 128 Bytes	0
ES (Entry Switch)	0	*	① Send **Packet-in** with 128 Bytes ② Push VLAN = 1 ③ Set TTL = β ④ Send to next switch by queue	0
LS (Loop Switch)	1	VLAN = 1 in-port = pre-switch	① Set TTL = TTL -1 ② Send to next Switch by queue	0

- ✓ VLAN tag: distinguish waiting-loop packet
- ✓ Priority: never affect normal traffic
- ✓ TTL: avoid endless loop

2 Switch Enforcement

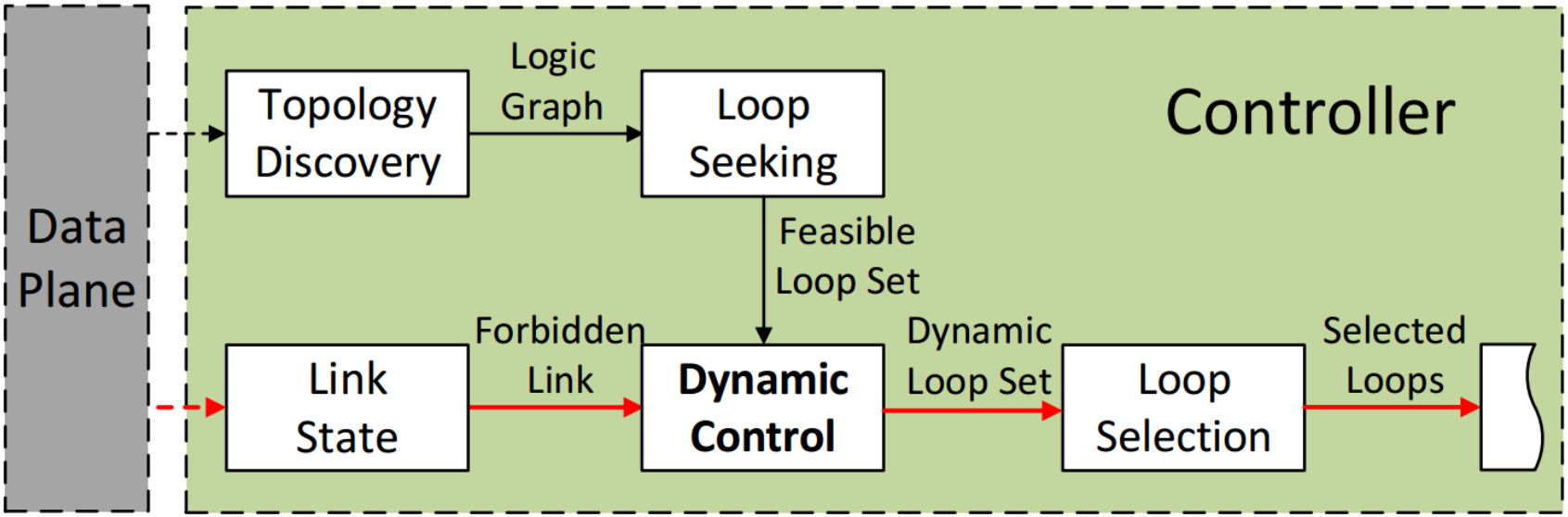
Controller-to-Switch Message: OpenFlow

Type	Priority	Match/Data Field	Action	Timeout
Packet-out	/	Buffer ID or Entire Packet	Forwarding	/
Flow-add	>2	*ip_src ip_dst* etc.	Forwarding	normal

Controller-to-Switch Message: SoftRing

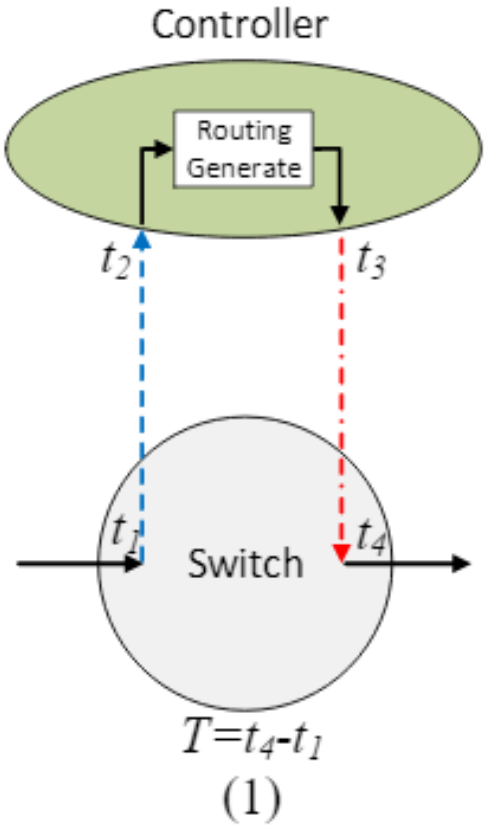
Type	Priority	Match/Data Field	Action	Timeout
Flow-add	>2	VLAN = None *ip_src ip_dst* etc.	Forwarding	normal
Flow-add	>2	VLAN = 1 *ip_src ip_dst* etc.	① Pop VLAN ② Reset TTL ③ Forwarding	short

3 Handling the Dynamic

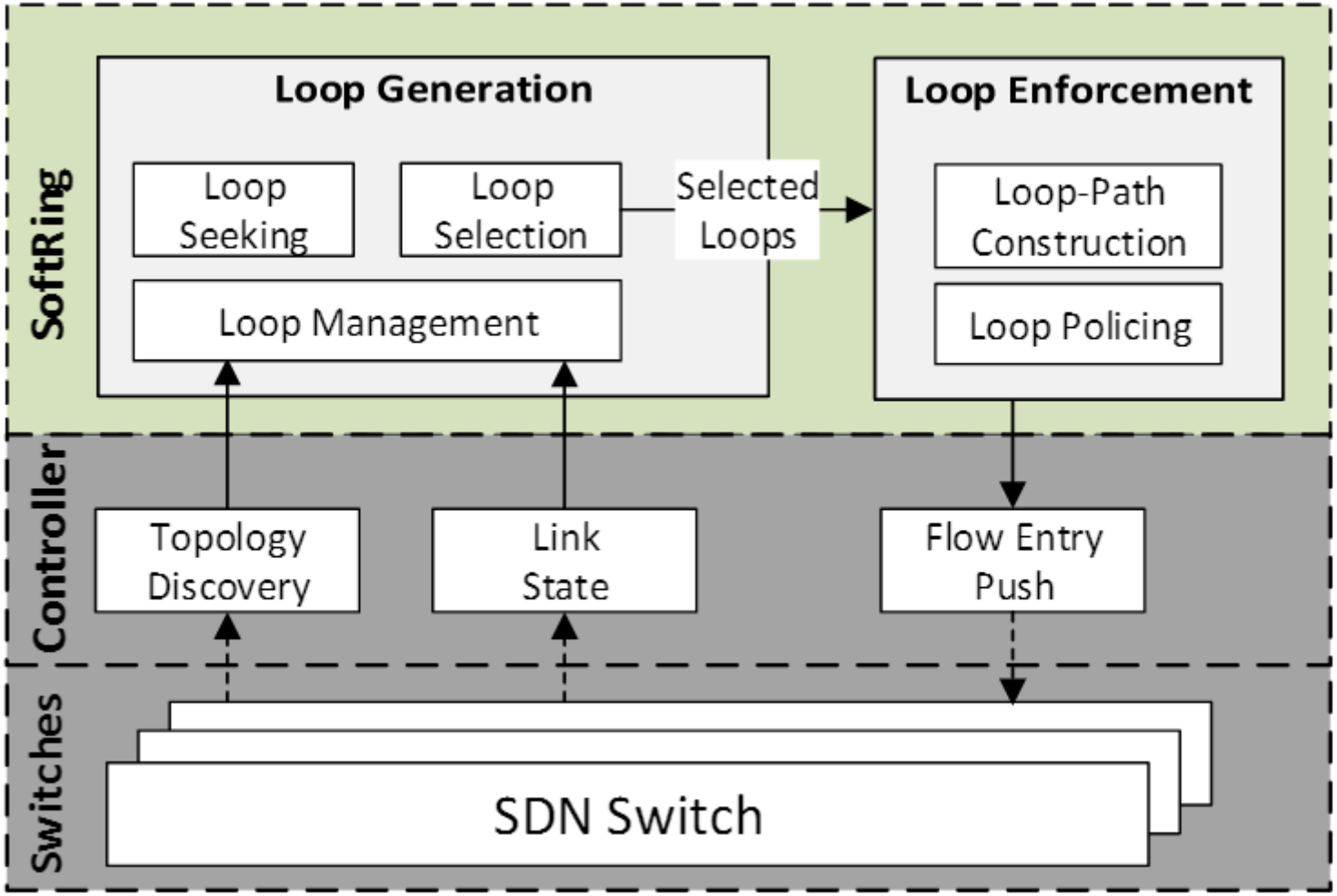


- Forbidden Link: overloaded or failed links
- Re-execute: Only Loop Selection Algorithm (< 1s)

3 Handling the Dynamic



SoftRing: Implementation

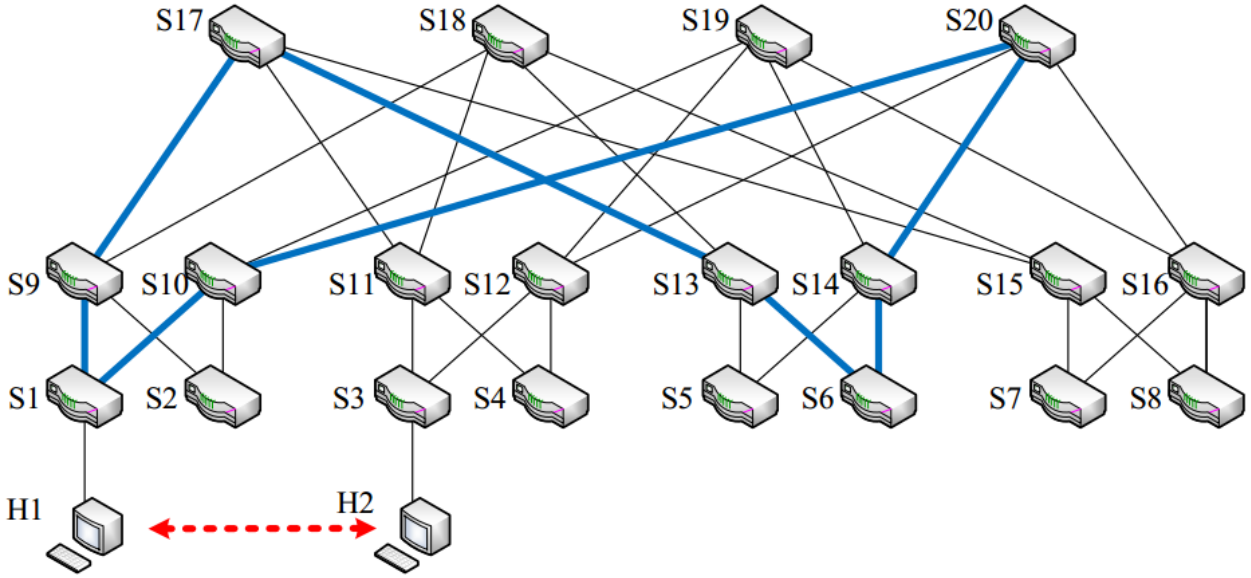


SoftRing: Implementation

Fattree(4) with 20 ONetSwitch

- Controller:
  - Floodlight
- Switch:
  - Open vSwitch: Software Switch
  - ONetSwitch 30: Hardware Switch

SoftRing: Implementation



Loop ID	Switch ID								Loop Length
1	1	10	20	14	6	13	17	9	8
2	2	9	17	13	5	14	19	10	8
3	3	12	19	16	8	15	18	11	8
4	4	11	18	15	7	16	20	12	8

Evaluation

	Topology	#Switch	#Link
Data Center Topology	BCube (1 , 4)	24	32
	BCube (2 , 6)	324	648
	BCube (3 , 8)	6,144	16,384
	DCell (1 , 4)	25	30
	DCell (2 , 6)	2170	3,612
	DCell (2 , 8)	5,913	10,512
	Fattree (4)	20	32
	Fattree (8)	80	256
	Fattree (32)	1,280	16,384
Internet Topology	Stanford	26	46
	CERNET	41	39
	KDL	754	899
	CAIDA	10,827	37,734

Evaluation

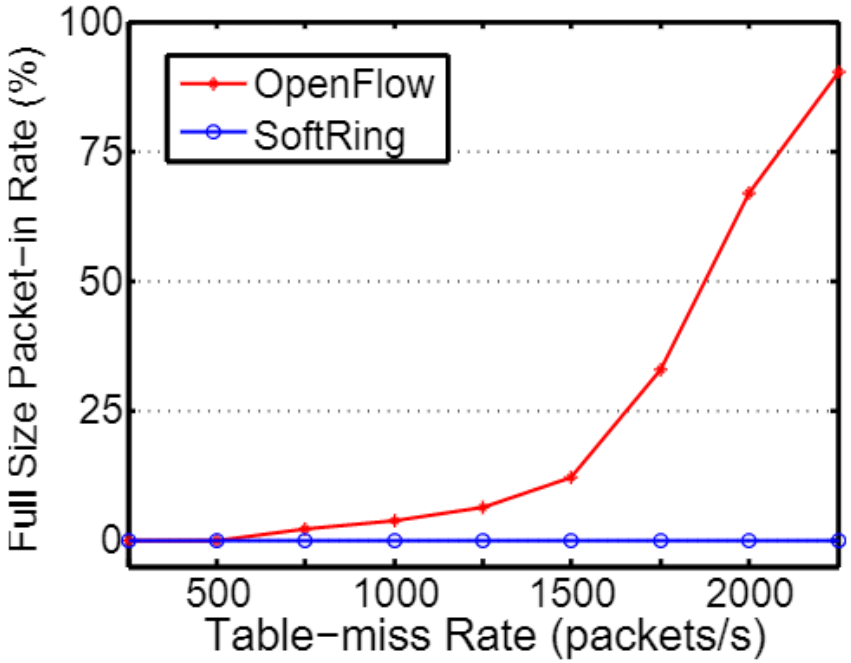
Topology	Time (s)		#Loop	#Flow Entry
	Seeking	Selection		
BCube (1 , 4)	0	0	4	2.3
BCube (2 , 6)	0.81	0.43	58	2.4
BCube (3 , 8)	122	81.3	1077	2.4
DCell (1 , 4)	0	0	4	2.4
DCell (2 , 6)	2.02	3.11	304	2.3
DCell (2 , 8)	12	32.3	816	2.3
Fattree (4)	0	0.01	4	2.6
Fattree (8)	0.03	0.01	14	2.3
Fattree (32)	6.49	2.89	208	2.5
Stanford	0.01	0	5	2.3
CERNET*	0	0	3	2.7
KDL*	0.12	0.02	62	3.3
CAIDA*	2,160	84.8	4,097	4.6

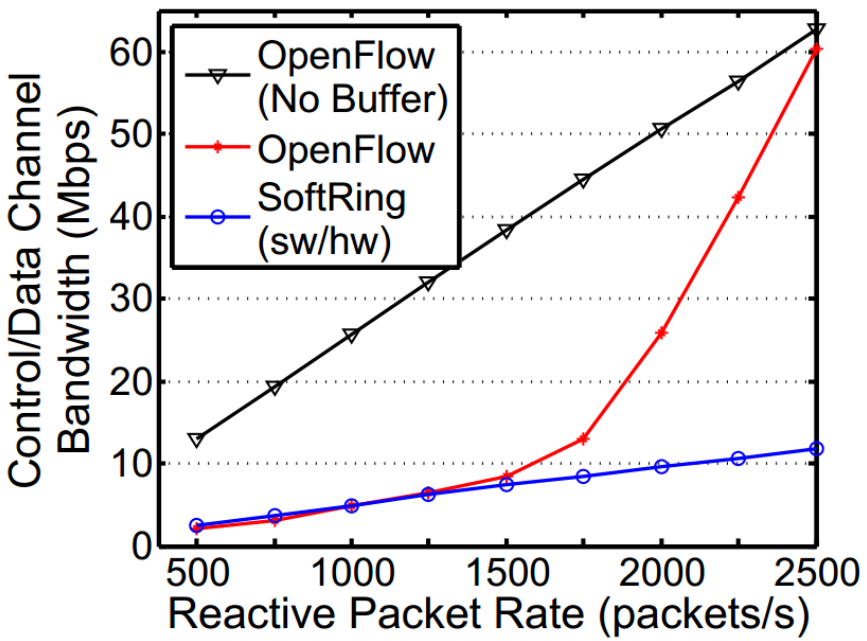
- 100% Coverage,
  - Virtual Loop used in last three topology

Evaluation

Topology	Time (s)		#Loop	#Flow Entry
	Seeking	Selection		
BCube (1 , 4)	0	0	4	2.3
BCube (2 , 6)	0.81	0.43	58	2.4
BCube (3 , 8)	122	81.3	1077	2.4
DCell (1 , 4)	0	0	4	2.4
DCell (2 , 6)	2.02	3.11	304	2.3
DCell (2 , 8)	12	32.3	816	2.3
Fattree (4)	0	0.01	4	2.6
Fattree (8)	0.03	0.01	14	2.3
Fattree (32)	6.49	2.89	208	2.5
Stanford	0.01	0	5	2.3
CERNET*	0	0	3	2.7
KDL*	0.12	0.02	62	3.3
CAIDA*	2,160	84.8	4,097	4.6

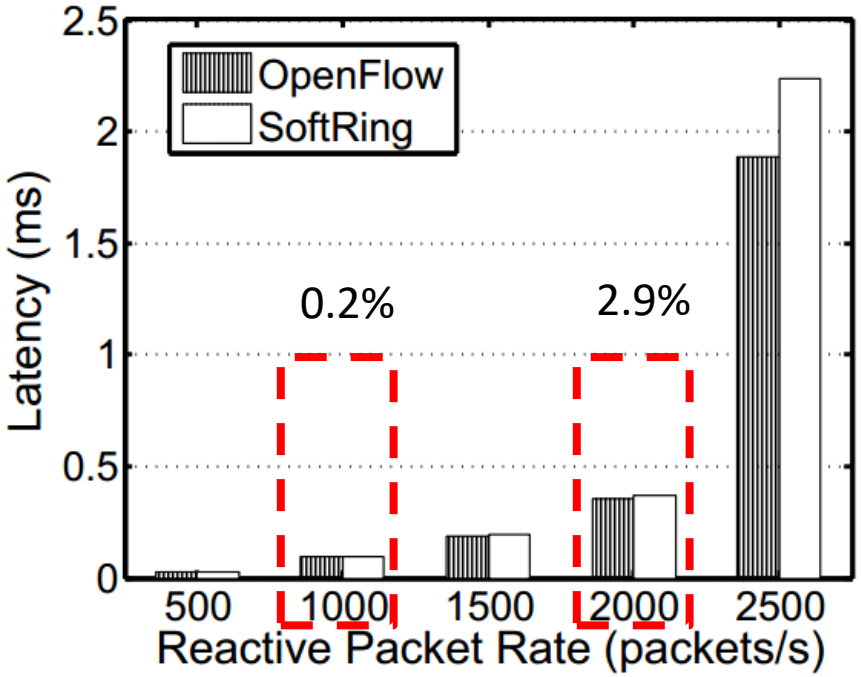
Benefit of SoftRing



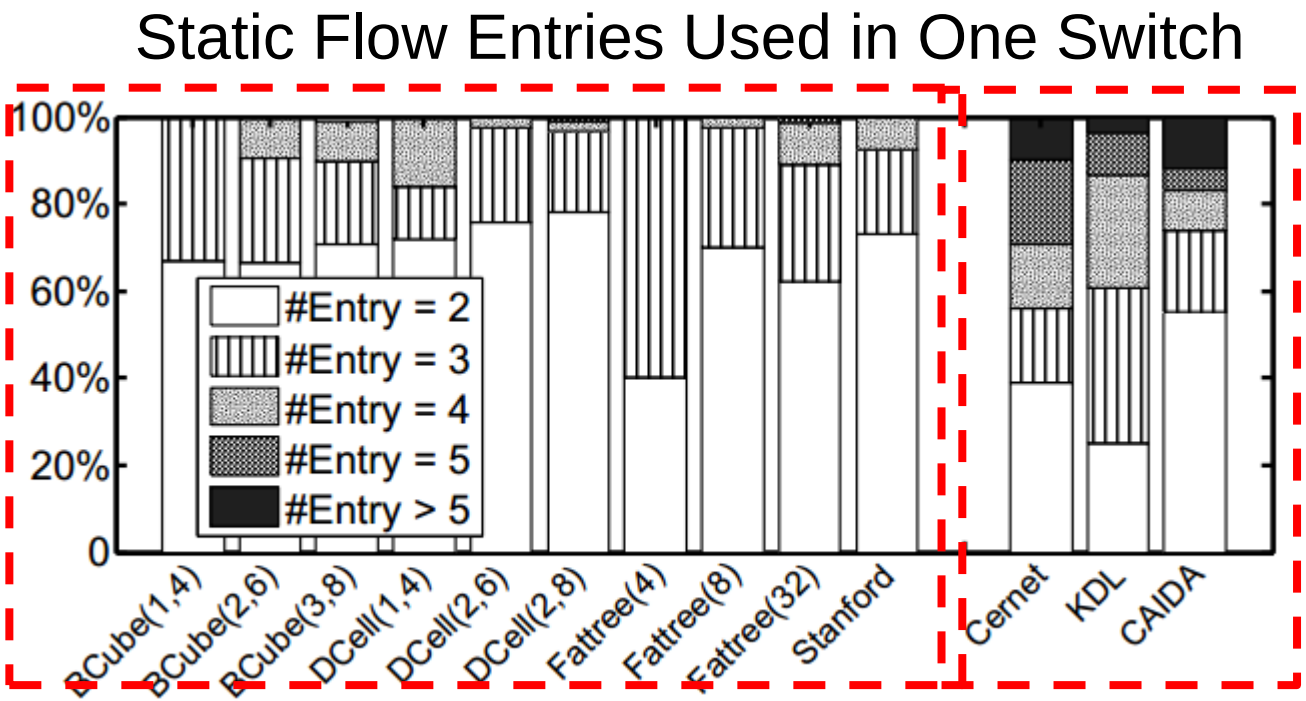


Cost of SoftRing

Average Packet Delay in Different Reactive Packet Rate



Cost of SoftRing



Conclusion

Claim	We advocate the coexistence of proactive model and reactive model.
SoftRing	A reactive packet goes through a pre-computed waiting-loop before get related rules from controller.
Evaluation	Reduce the control channel bandwidth up to 80%. With the cost of 3 flow entries, minor latency.

Thank You

Questions ?