

Universität Stuttgart

Institute of Parallel and
Distributed Systems (IPVS)

Universitätsstraße 38
D-70569 Stuttgart

Tutorial: Software-defined Networking

Part 2: Introduction to OpenFlow

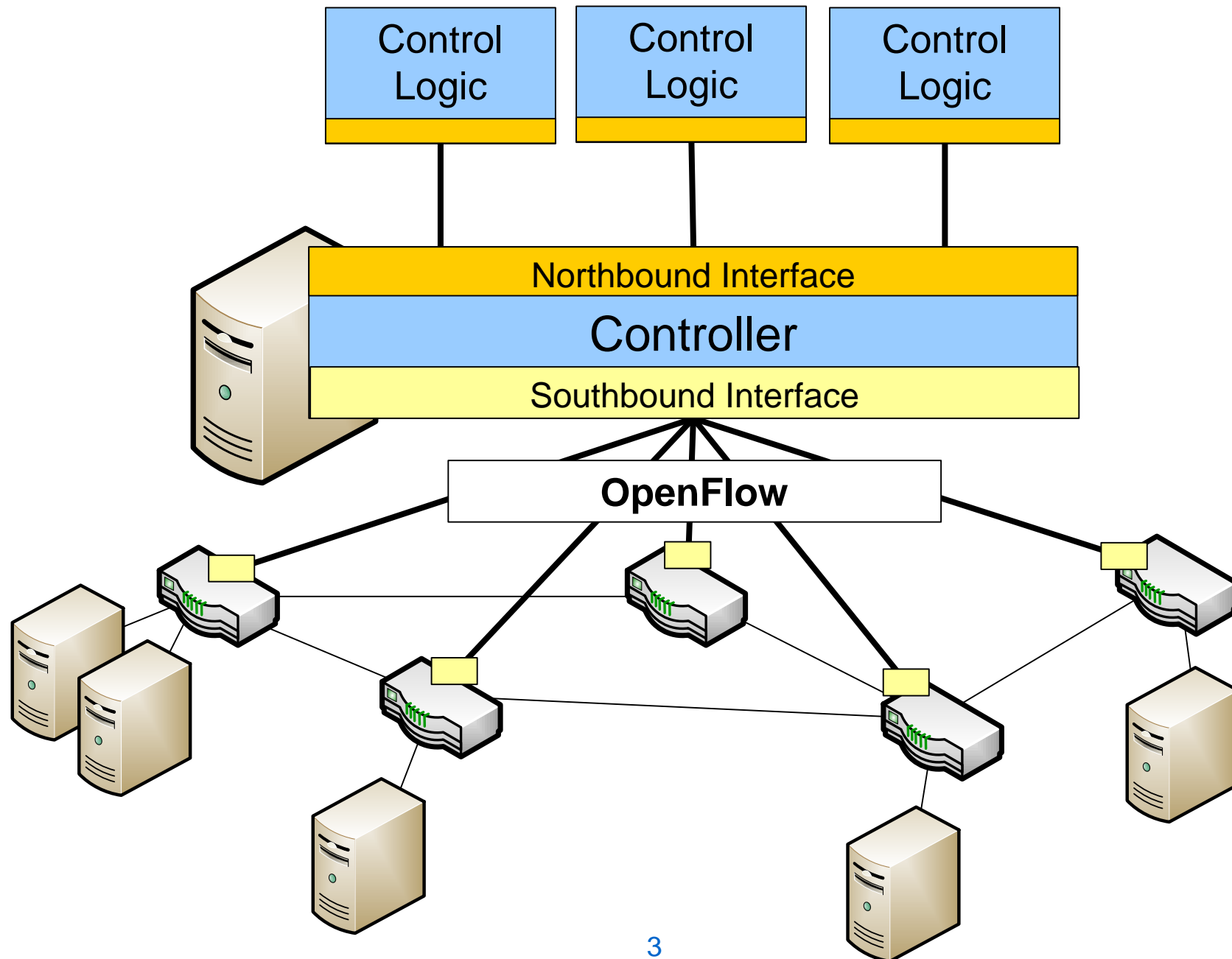
Frank Dürr

Overview

- The OpenFlow Protocol
- Proactive and Reactive Routing



Southbound Interface: The OpenFlow Protocol

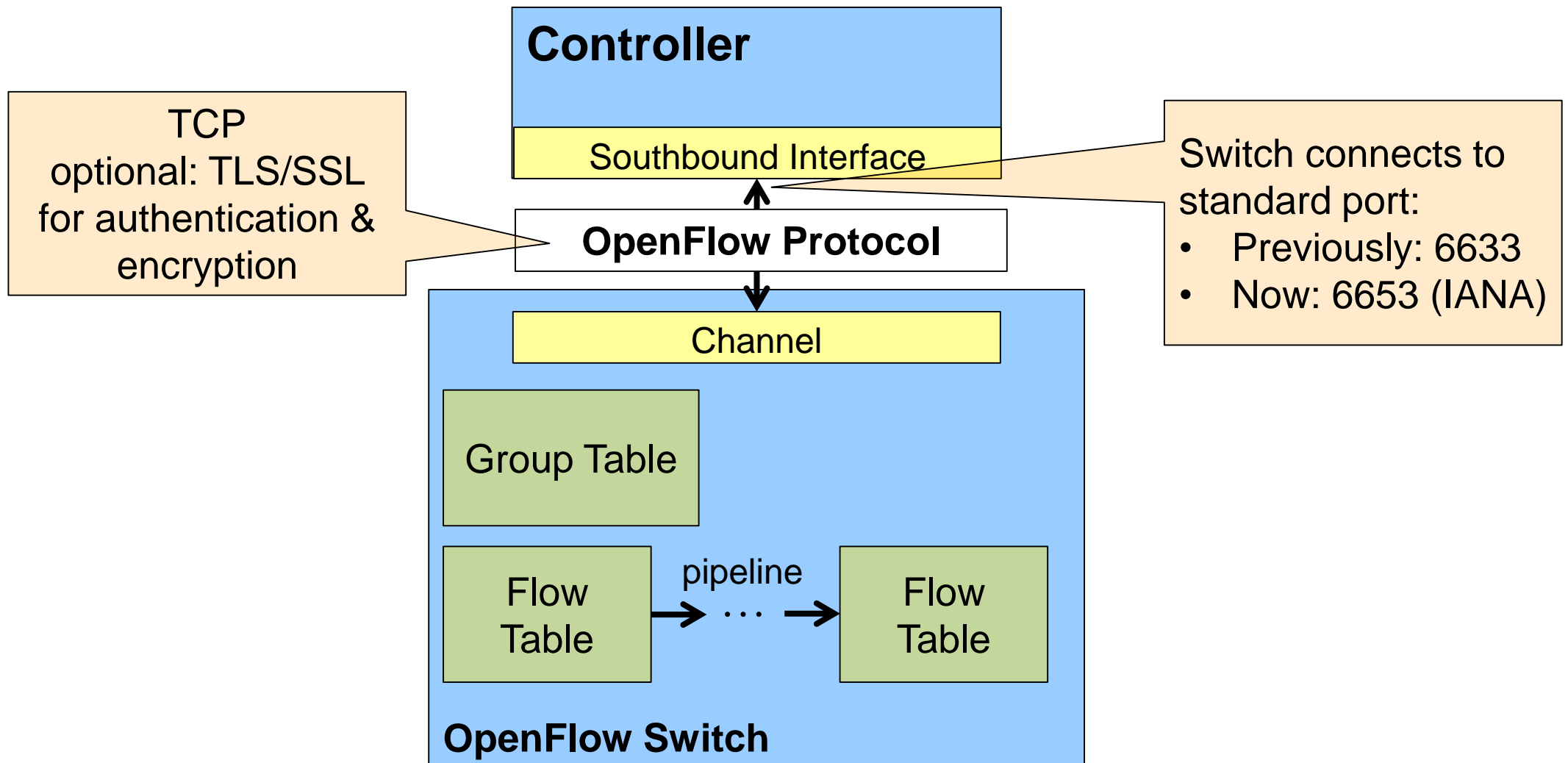


The OpenFlow Protocol: Overview

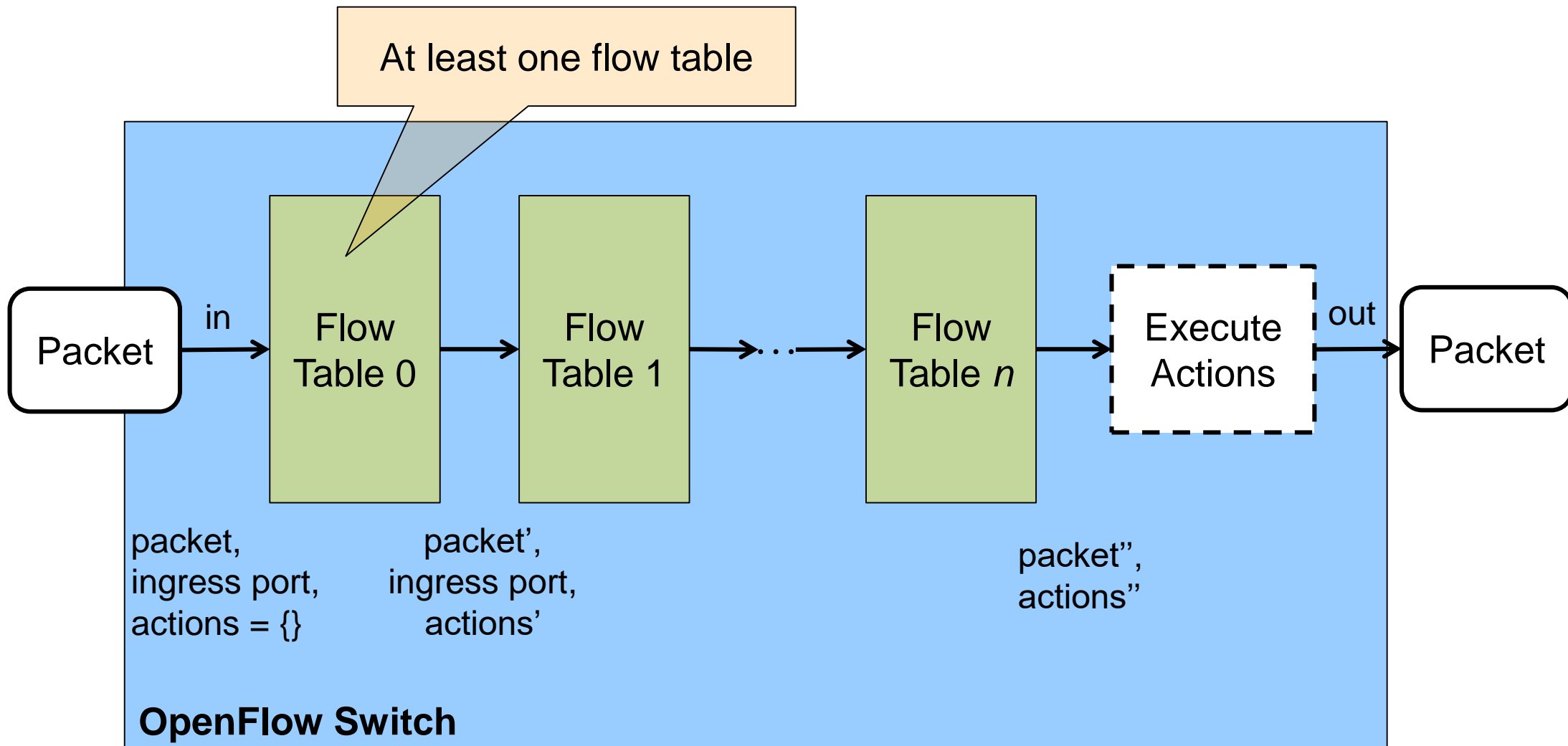
- OpenFlow de facto standard for southbound interface
- Defined by Open Networking Foundation
 - Major vendors (Cisco, IBM, NEC, HP, Alcatel-Lucent, VMware, ...)
- Interface to a single switch
 - No aspects of control plane distribution defined
- Basic functionality
 - Modification of flow tables (adding, removing, modifying entries)
 - Injecting packets
 - Events for receiving packets (reactive routing)
 - Querying traffic statistics (counters)



Architecture of an OpenFlow System



The Way of a Packet through a Switch



Flow Tables & Flow Entries

- **Flow tables** consist of a list of flow entries
- **Flow entry (slightly simplified):**
 - *Match field*: Defines matching packets
 - *Priority*: Precedence if multiple entries match
 - *Counters*: Counts matches
 - *Instructions*:
 - Modification and forwarding of packet
 - *Timeouts*: Removes entry after a certain (idle) time

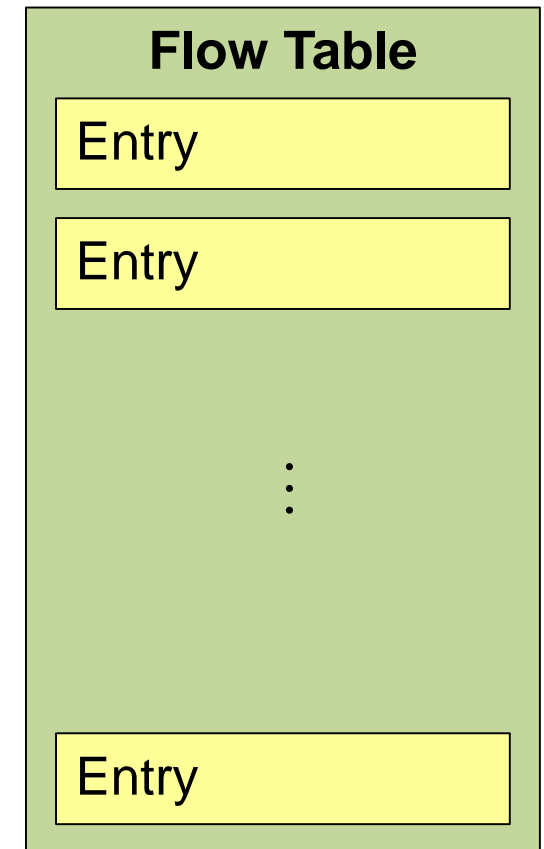
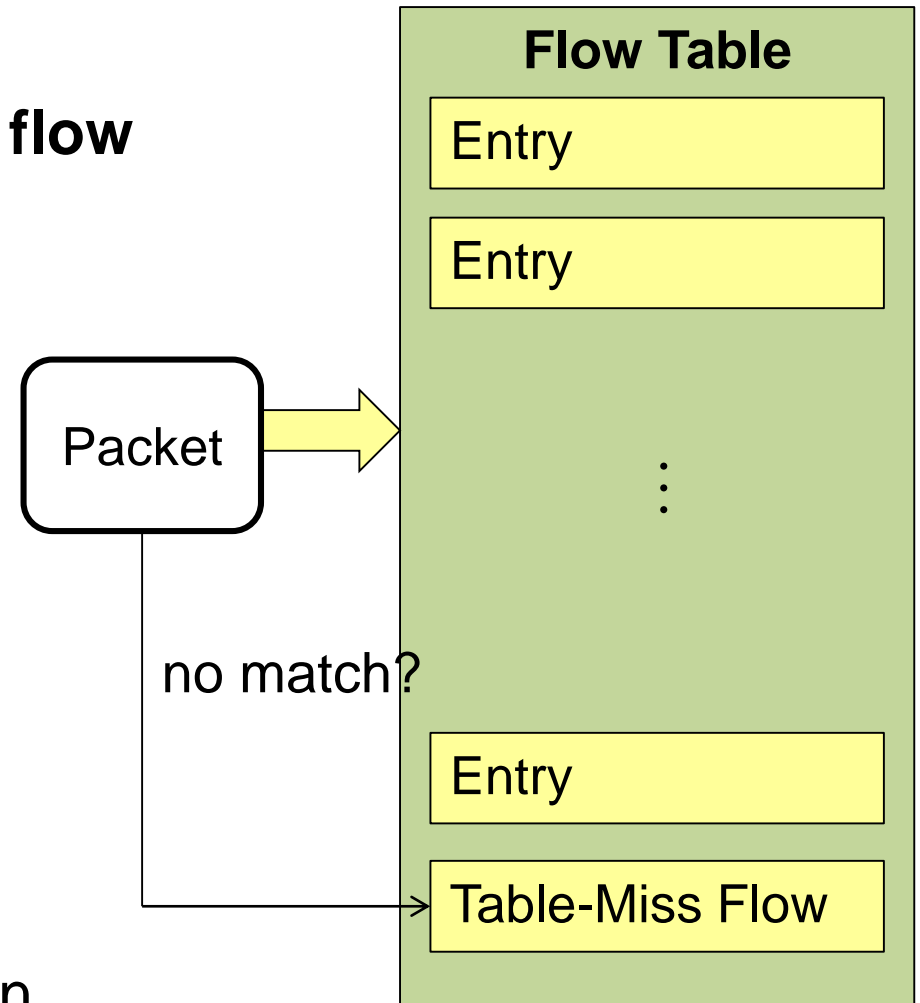


Table Misses

Important for dynamic routing

- Each table supports a **table-miss flow entry**
 - Lowest priority
 - Matches all packets
- Possible actions (at least):
 - Drop
 - **Send to controller**
- If no table-miss entry is defined
 - Drop packet (default in OF1.3)
 - Or define another default action



Match Fields

Subset of L2-4 header fields

- 10 tuple (must be supported)
 - For more (optional) fields please see OF standard
- Note: hardware switches might only support hardware-accelerated matching on some combinations!
 - Rest goes the “slow path”

Ingress Port			
Ether Src	Ether Dst	VLAN ID	Ether Type
IP SRC	IP DST	IP PROTO	
TCP/UDP SRC PORT	TCP/UDP SRC PORT		

- Specifies layer 3 protocol
- E.g, 0x0800 = IPv4

- Specifies layer 4 protocol
- E.g, 6 = TCP, 17 = UDP



Wildcard Matching (1)

- Not all fields need to be specified: Wildcard *
 - Matches any value
- For IP addresses, bitmasks can be specified (CIDR)
 - Example: Subnet mask of IPv4 address 192.168.1.1/24 (netmask 255.255.255.0)

In Port	Eth Src	Eth Dst	Eth Type	VLAN ID	IP SRC	IP DST	IP Proto	Src Port	Dst Port
*	*	*	0x0800 (IPv4)	*	*	10.2.3.4	6 (TCP)	*	80
2	*	*	*	42	*	*	*	*	*
*	*	*	0x0800 (IPv4)	*	*	10.1.2.3	*	*	*

IPv4 traffic to a certain machine (e.g., result of a routing algorithm)

Traffic of a certain VLAN from a certain port

All traffic to a certain web server (port 80)

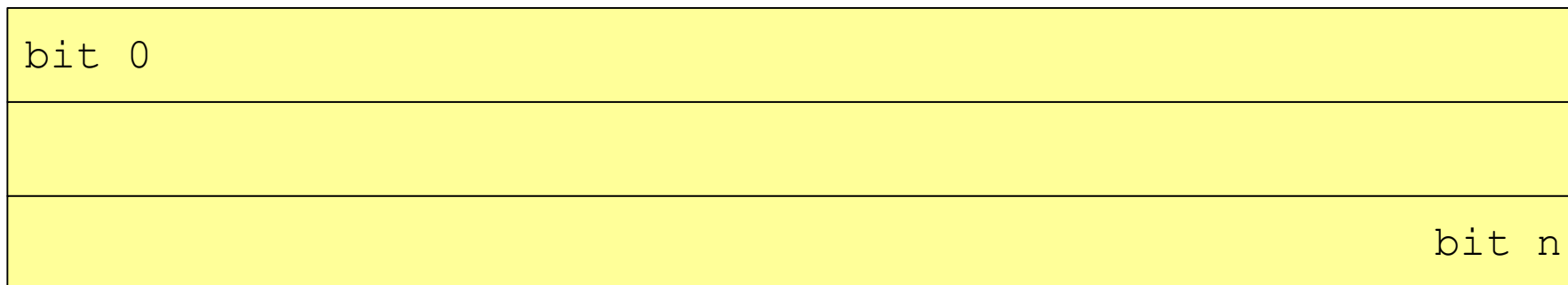
Wildcard Matching (2)

- Hardware switches can perform very fast matching using Content Addressable Memory (CAM)
 - Parallel matching of all entries in one clock cycle (micro-seconds)
- Two types of CAM
 - Binary CAM (BCAM): ordinary bits 0, 1
 - Good for exact match
 - Ternary CAM (TCAM): ordinary bits + wildcard (don't care) 0, 1, *
 - Implementation of longest prefix match on IP addresses
- Drawbacks: Consumes significant energy, silicon space
 - Limited memory size in switches (hundreds to hundred thousand entries)
 - Remember the day of the 512k problem!



A Thought about Flexibility

- OpenFlow specification tailored to specific protocols
 - IP, TCP, UDP, ICMP, ARP, ...
 - Did not support IPv6 in version 1.0!
- Again: Stuck with standard protocols
 - Practical trade-off: existing hardware switches can be easily modified to support OpenFlow
- Theoretically, we could just assume a big bit field and bit mask
 - Fields could be freely defined



Actions

- **Output:** output packet on the specified port
- **TTL modifications**
 - Decrement TTL, copy TTL outwards/inwards
- **Push and pop tags**
 - Add or remove VLAN/MPLS/PBB (MAC-in-MAC) tags to/from the packet
- **Set header fields**
 - Example: IP- or MAC-address re-writing
- **Group actions**
 - Example: Multicast

Order of execution of actions is well-defined by action type



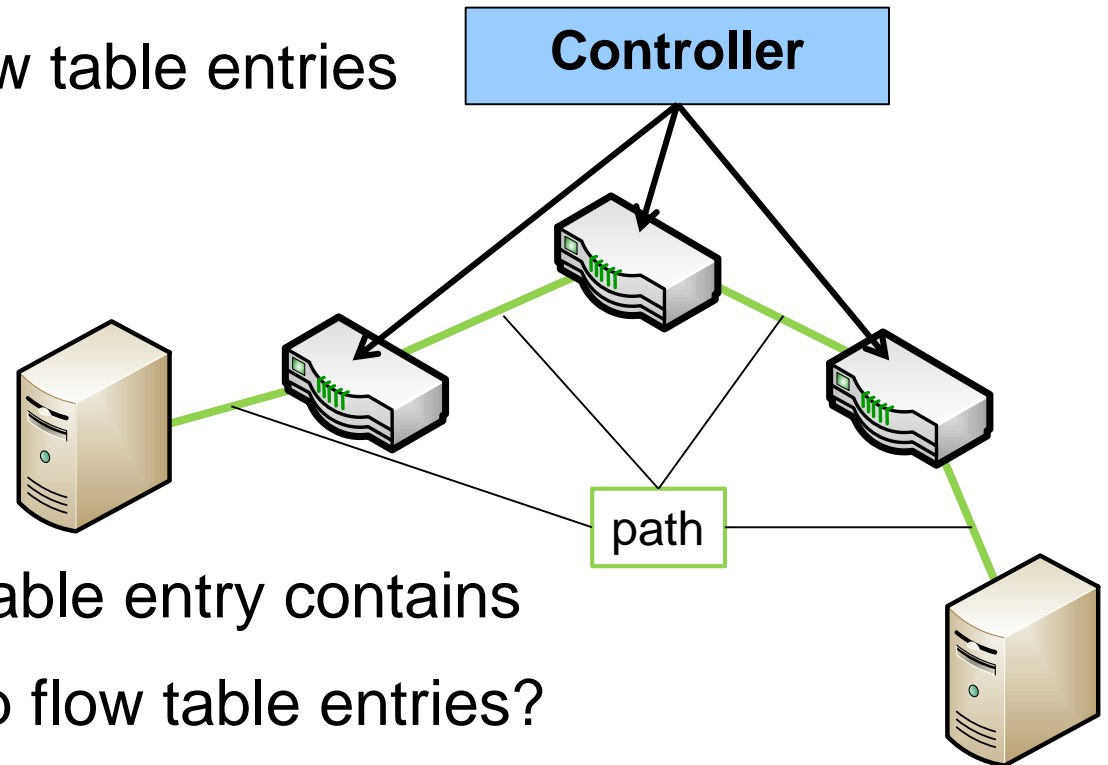
Instructions

- Write actions
 - Adds given actions to the action set
 - Overwrites actions of the same type
- Go to table with given id
 - Hybrid switches can also go to the “normal” table
- Apply specific actions immediately
 - Modify packet before going to next table
- Clear action set
- Meter id: send packet to a given meter (e.g. rate limitation)



Proactive vs Reactive Routing

- Routes defined by a set of flow table entries
 - Along the path of packets

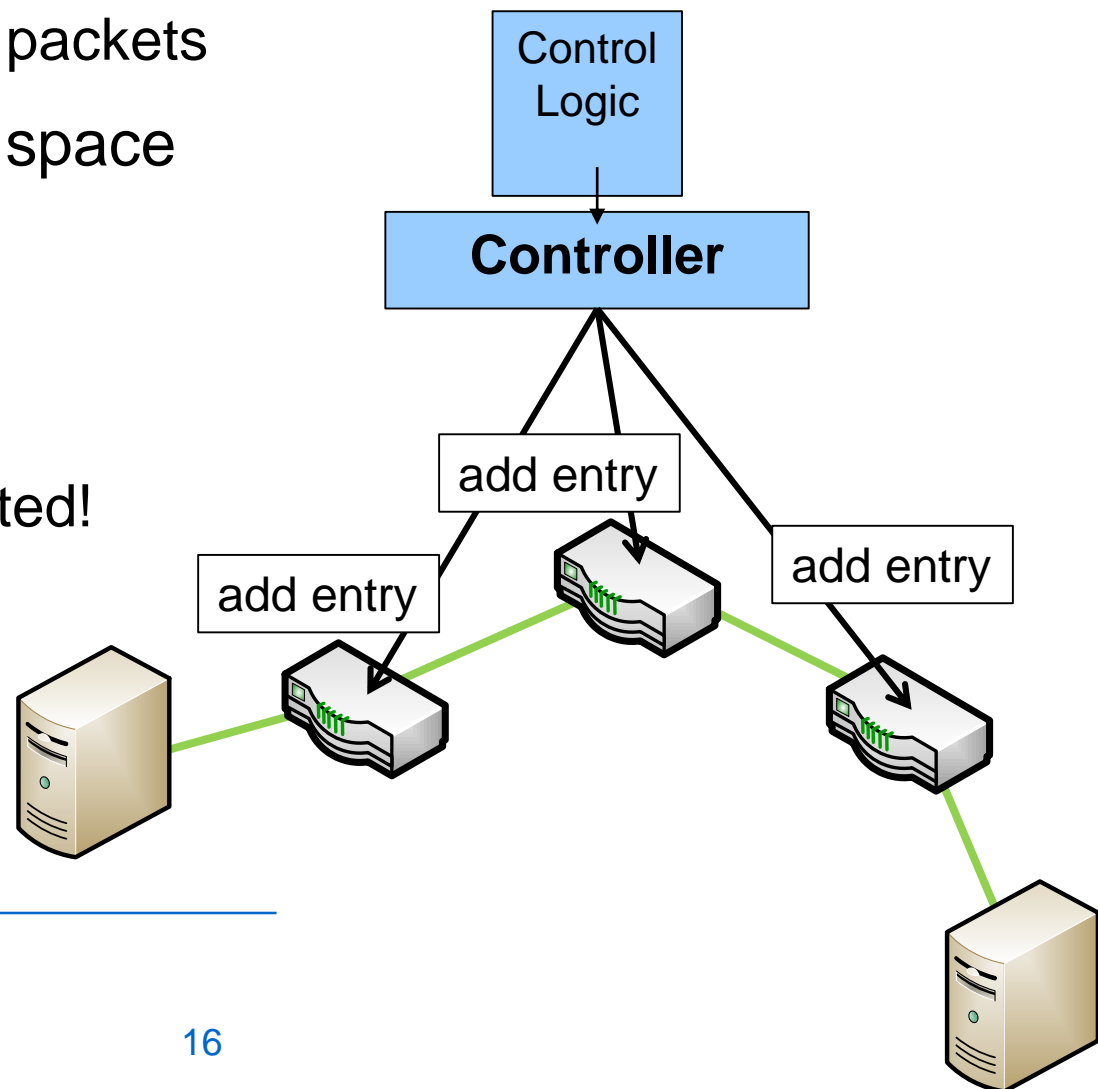


- So far, we know *what* a flow table entry contains
- Question now: *When* to set up flow table entries?
- Two options:
 - Proactively: before the flow starts
 - Reactively: as soon as the flow starts



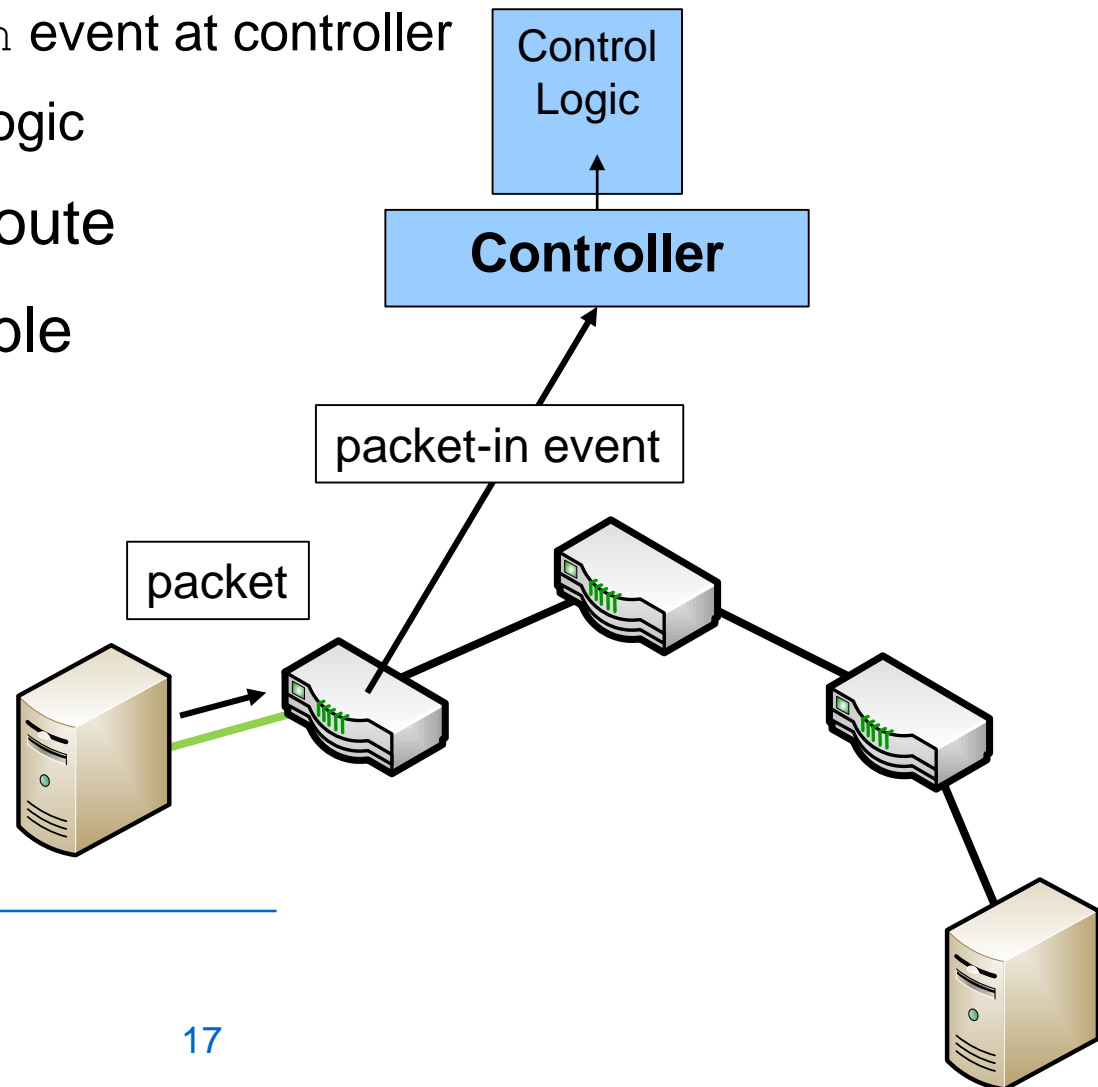
Proactive Routing

- Controller proactively “pushes” flow table entries onto switches
- Advantage: Reduces controller load
 - No reactive handling of packets
- Disadvantage: Occupies space in flow table of switch
 - Even without traffic
 - Remember:
Size of flow table is limited!



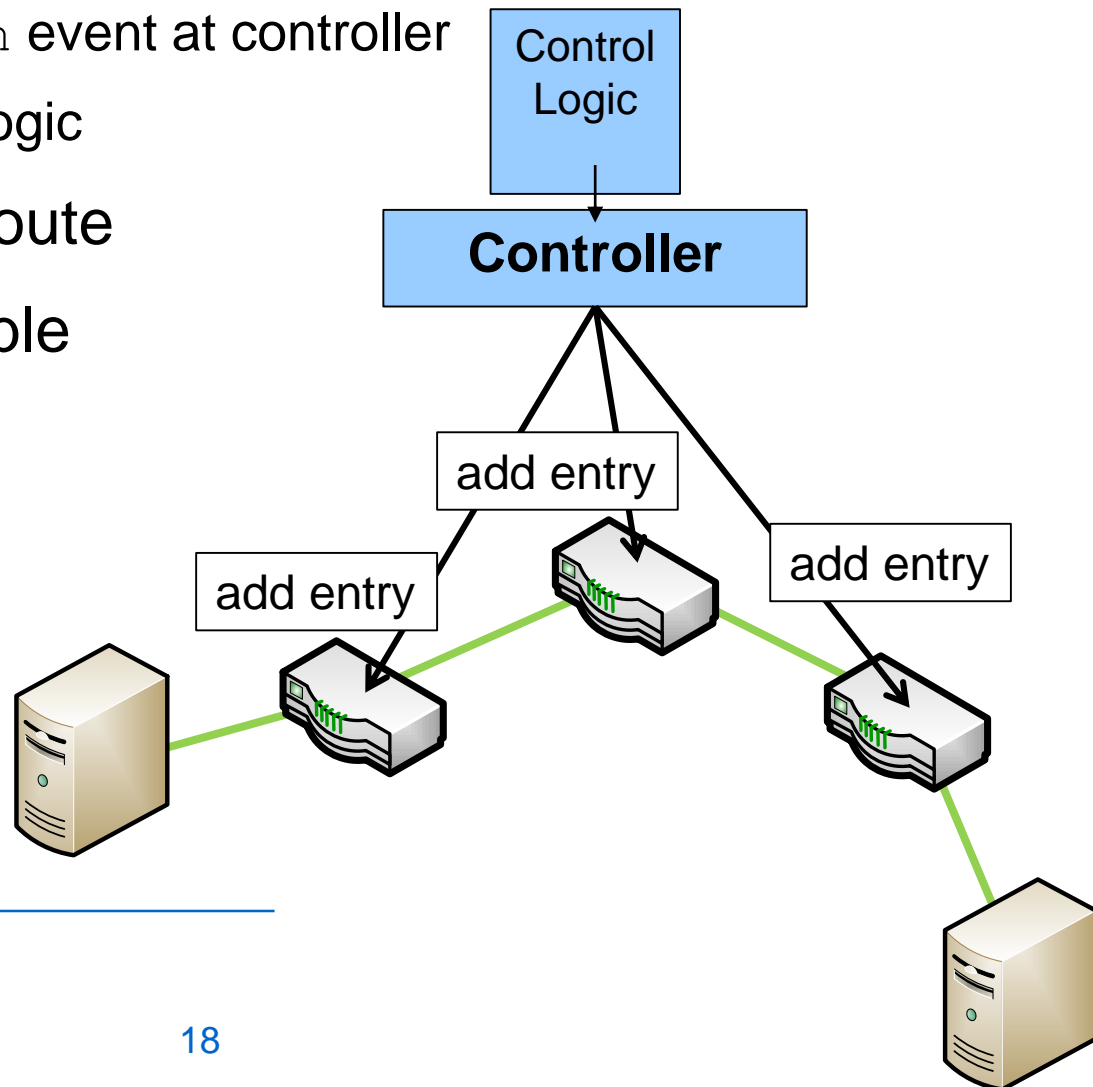
Reactive Routing (1)

- Switch receives a packet without matching flow table entry
 - Switch redirects packet to controller
 - Open flow `packet_in` event at controller
 - Forwarded to control logic
- Control logic calculates route
- Controller installs flow table entries along path
 - Further packets of flow don't involve controller again



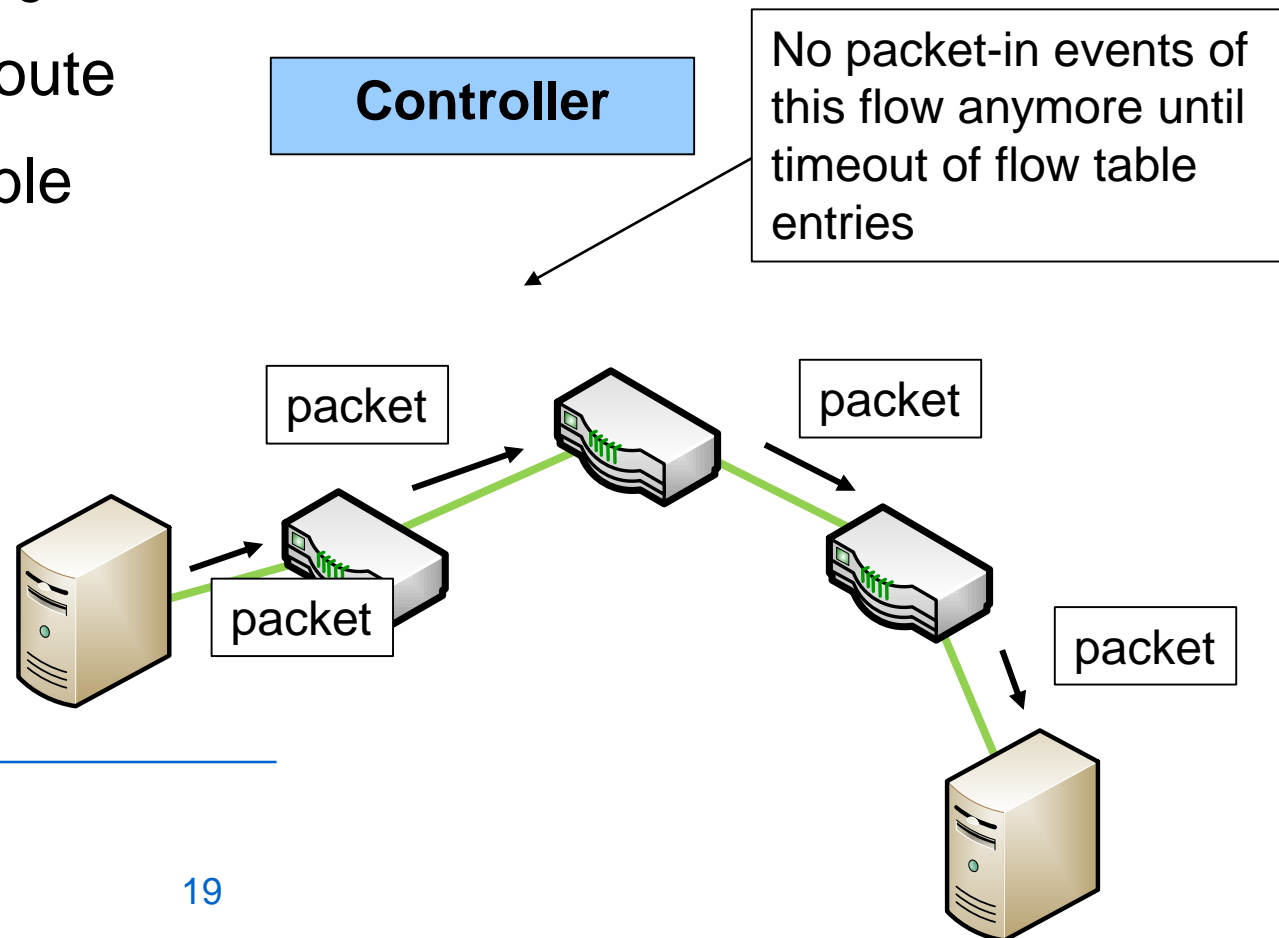
Reactive Routing (1)

- Switch receives a packet without matching flow table entry
 - Switch redirects packet to controller
 - Open flow `packet_in` event at controller
 - Forwarded to control logic
- Control logic calculates route
- Controller installs flow table entries along path
 - Further packets of flow don't involve controller again



Reactive Routing (1)

- Switch receives a packet without matching flow table entry
 - Switch redirects packet to controller
 - Open flow `packet_in` event at controller
 - Forwarded to control logic
- Control logic calculates route
- Controller installs flow table entries along path
 - Further packets of flow don't involve controller again



Reactive Routing (2)

- Advantage: Saves flow table space
- Disadvantage: Puts load onto controller and control network
 - Not such a big problem for TCP
 - Sender blocked until connection setup is done
 - Beware: Connection-less UDP can send at full rate immediately (without warning)!



Required Information for Routing

Dynamic routing requires knowledge of network status

- **Network topology** (nodes and links):
 - Switches and hosts
 - Links between switches (from switch port to switch port)
 - Links between hosts and switches (connection host to switch port)
 - Bandwidth of links
- **Traffic statistics**
 - Number of packets or bytes
 - Number of dropped packets, receive/transmit errors, etc.
 - Per flow (entry), link/port, group, etc.



Network Topology

- Discovery of network topology not covered by OpenFlow
- Controller implements standard protocols
 - Logical Link Layer Discovery Protocol (LLDP)
 - ARP handling for host discovery
- Controller exposes discovered information to control logic
 - See northbound interface discussion



Traffic Statistics

- OpenFlow switches implement **counters**
- Controller can **query counters** using OpenFlow messages:
 - Flow Statistics (OFPMP_FLOW, OFPMP_AGGREGATE)
 - Port Statistics (OFPMP_PORT_STATS)
 - Table Statistics (OFPMP_TABLE)
 - Queue Statistics (OFPMP_QUEUE)
 - Group Statistics (OFPMP_GROUP)
 - Meter Statistics (OFPMT_METER)



Counters

Counter	Bits	
Per Flow Table		
Reference Count (active entries)	32	<i>Required</i>
Packet Lookups	64	<i>Optional</i>
Packet Matches	64	<i>Optional</i>
Per Flow Entry		
Received Packets	64	<i>Optional</i>
Received Bytes	64	<i>Optional</i>
Duration (seconds)	32	<i>Required</i>
Duration (nanoseconds)	32	<i>Optional</i>
Per Port		
Received Packets	64	<i>Required</i>
Transmitted Packets	64	<i>Required</i>
Received Bytes	64	<i>Optional</i>
Transmitted Bytes	64	<i>Optional</i>
Receive Drops	64	<i>Optional</i>
Transmit Drops	64	<i>Optional</i>
Receive Errors	64	<i>Optional</i>
Transmit Errors	64	<i>Optional</i>
Receive Frame Alignment Errors	64	<i>Optional</i>
Receive Overrun Errors	64	<i>Optional</i>
Receive CRC Errors	64	<i>Optional</i>
Collisions	64	<i>Optional</i>
Duration (seconds)	32	<i>Required</i>
Duration (nanoseconds)	32	<i>Optional</i>

Per Queue		
Transmit Packets	64	<i>Required</i>
Transmit Bytes	64	<i>Optional</i>
Transmit Overrun Errors	64	<i>Optional</i>
Duration (seconds)	32	<i>Required</i>
Duration (nanoseconds)	32	<i>Optional</i>
Per Group		
Reference Count (flow entries)	32	<i>Optional</i>
Packet Count	64	<i>Optional</i>
Byte Count	64	<i>Optional</i>
Duration (seconds)	32	<i>Required</i>
Duration (nanoseconds)	32	<i>Optional</i>
Per Group Bucket		
Packet Count	64	<i>Optional</i>
Byte Count	64	<i>Optional</i>
Per Meter		
Flow Count	32	<i>Optional</i>
Input Packet Count	64	<i>Optional</i>
Input Byte Count	64	<i>Optional</i>
Duration (seconds)	32	<i>Required</i>
Duration (nanoseconds)	32	<i>Optional</i>
Per Meter Band		
In Band Packet Count	64	<i>Optional</i>
In Band Byte Count	64	<i>Optional</i>

Summary

- OpenFlow basics
 - Standard protocol between switch and controller (southbound)
 - Flow tables: pipeline processing
 - Flow table entries: match fields, instructions, actions
- Reactive and proactive flow table setup



Questions?



IPVS

Research Group
“Distributed Systems”

26

Universität Stuttgart
IPVS