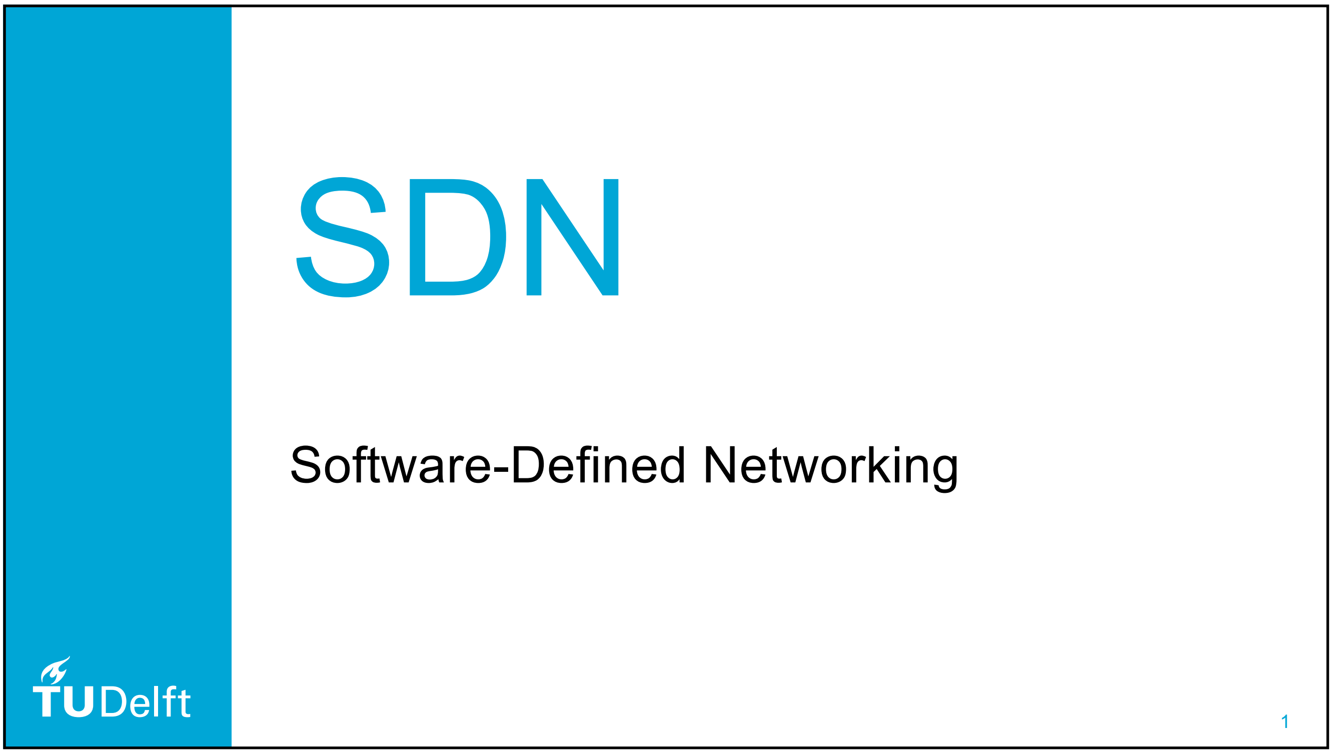


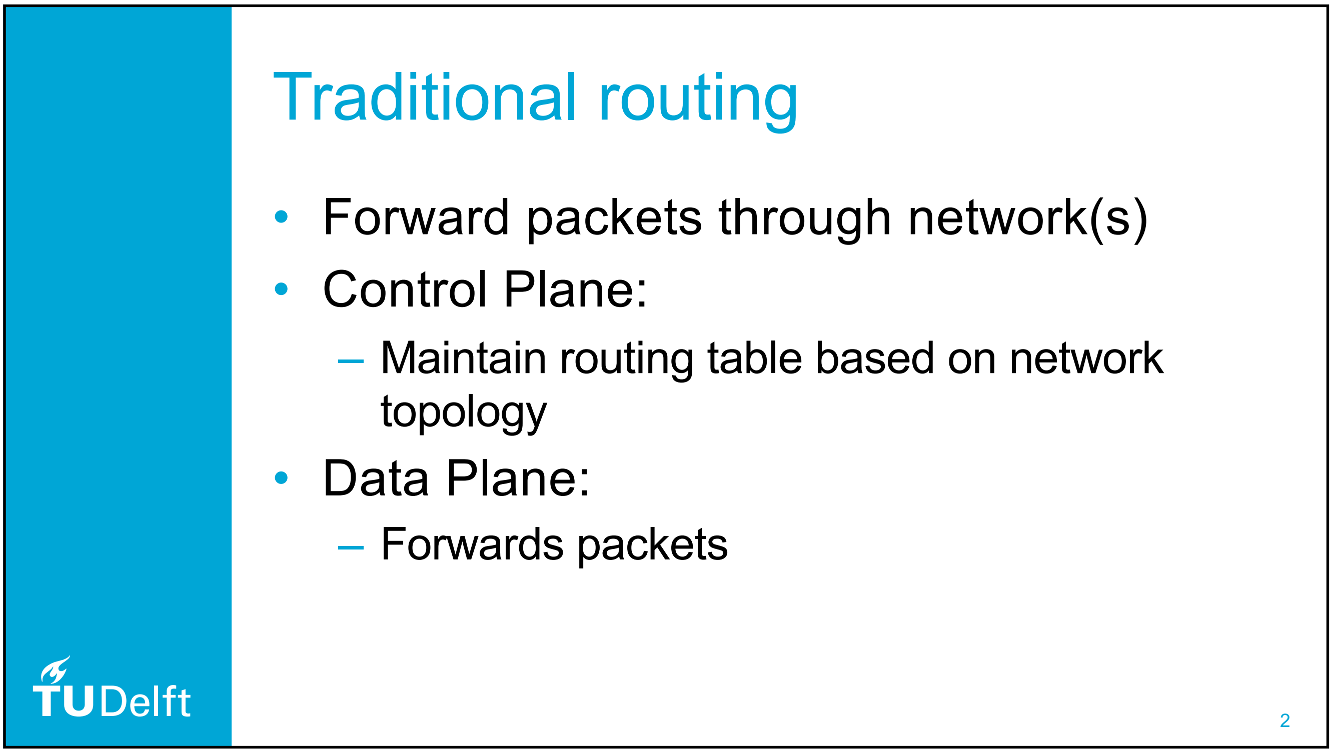
SDN

Software-Defined Networking

TU Delft

1

1



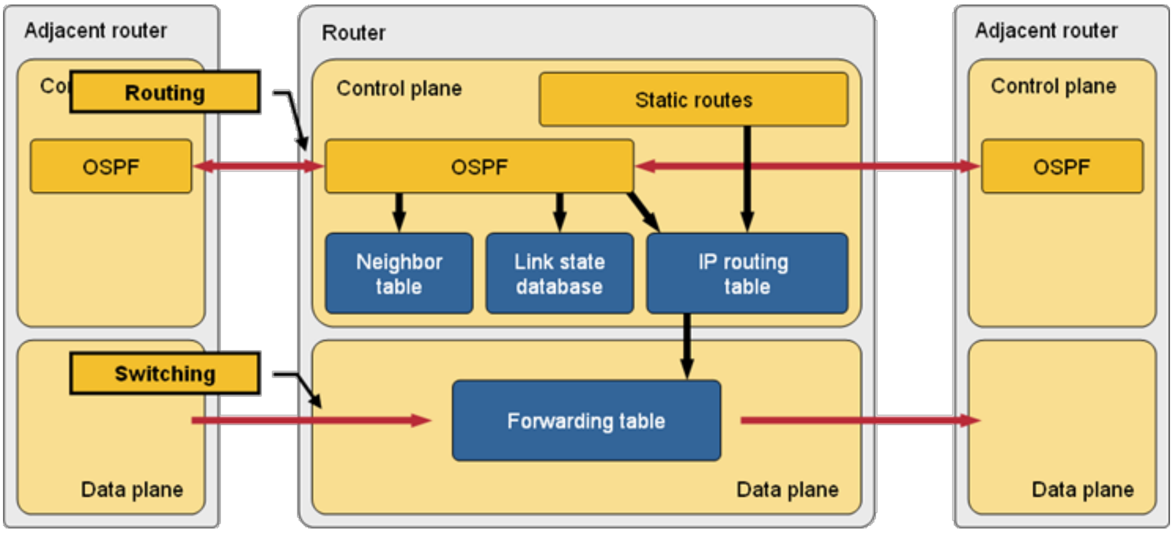
Traditional routing

- Forward packets through network(s)
- Control Plane:
  - Maintain routing table based on network topology
- Data Plane:
  - Forwards packets

TU Delft

2

2



Adjacent router

Co

Routing

OSPF

Switching

Data plane

Router

Control plane

Static routes

OSPF

Neighbor table

Link state database

IP routing table

Forwarding table

Data plane

Adjacent router

Control plane

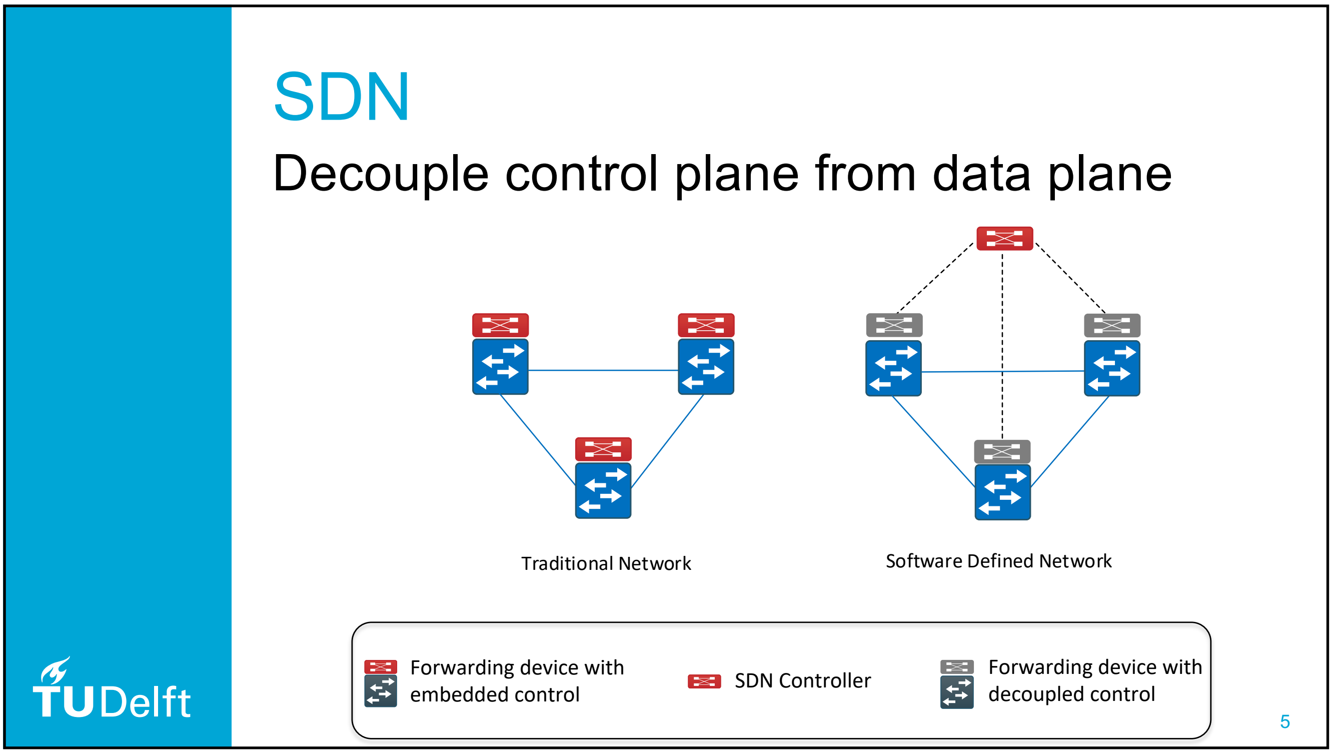
OSPF

Data plane

Source: http://wiki.nil.com

Traditional routing: Disadvantages

- Difficult to make changes
- Constant communication between routers
- Fairly static (long convergence time)
- Dependent on hardware (vendors)

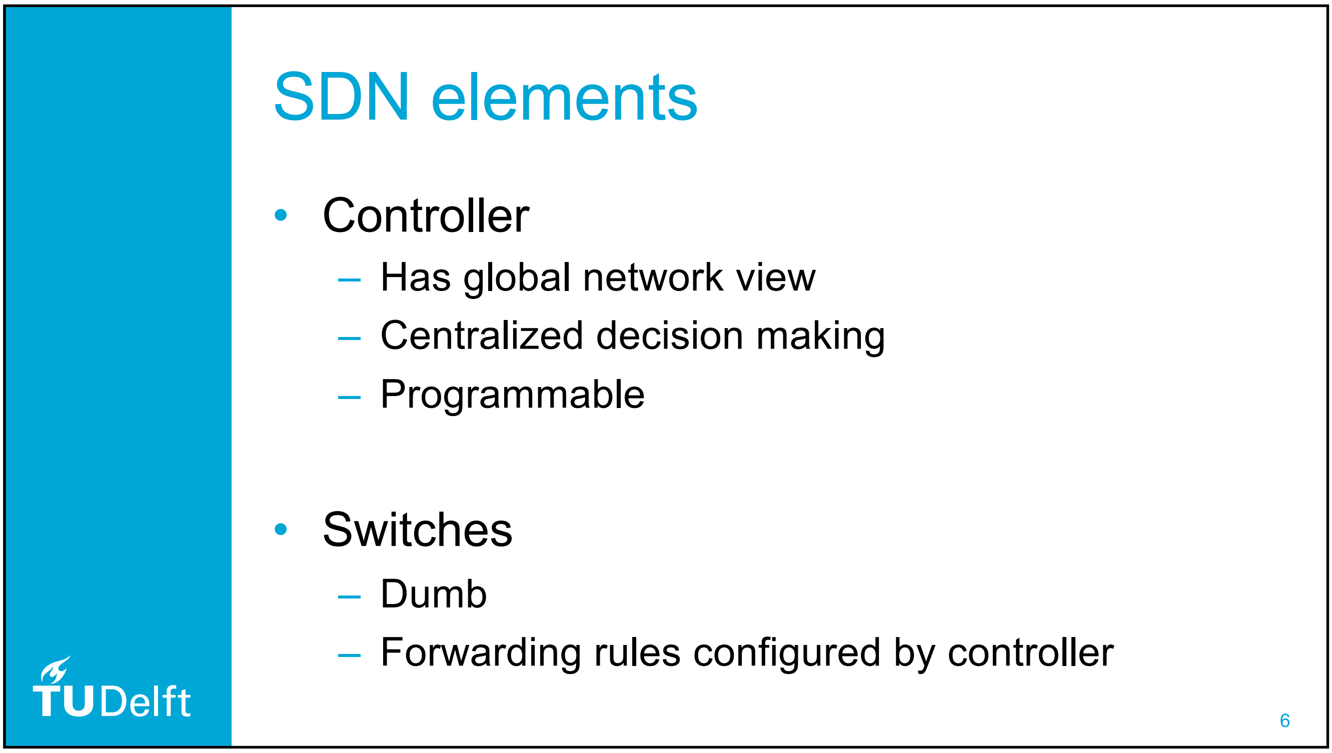


SDN

Decouple control plane from data plane

Traditional Network
Software Defined Network
Forwarding device with embedded control
SDN Controller
Forwarding device with decoupled control

5



SDN elements

- Controller
  - Has global network view
  - Centralized decision making
  - Programmable

- Switches
  - Dumb
  - Forwarding rules configured by controller

6

Advantages

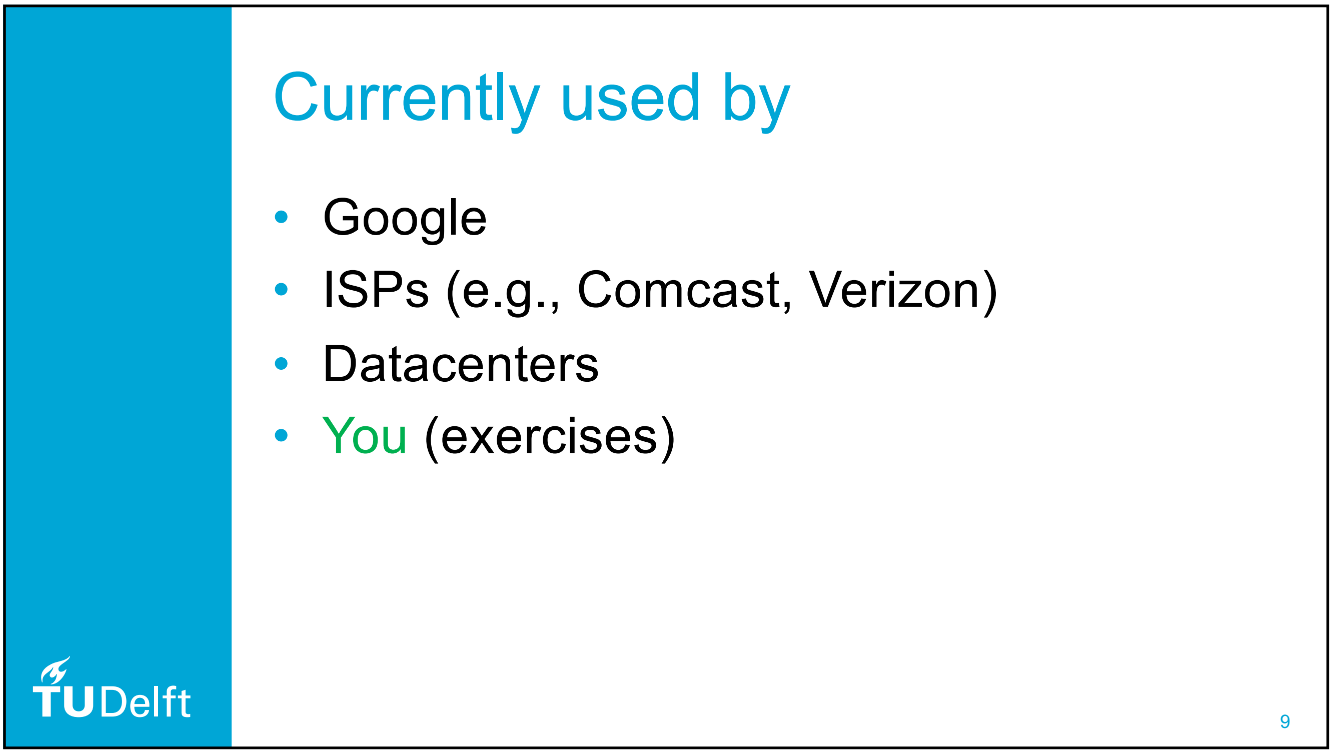
- Programmable
  - Flexible
  - Fine-grained traffic management
- Centralized view of network, so easier to:
  - Compute paths/trees
  - Add security
  - Provide fault tolerance
  - Etc.

TU Delft

Disadvantages

- Centralized
  - Single point of failure (multiple controllers can be used)
- Scalability:
  - Processing power bottleneck (at the controller)
- Initial delay when installing flows reactively

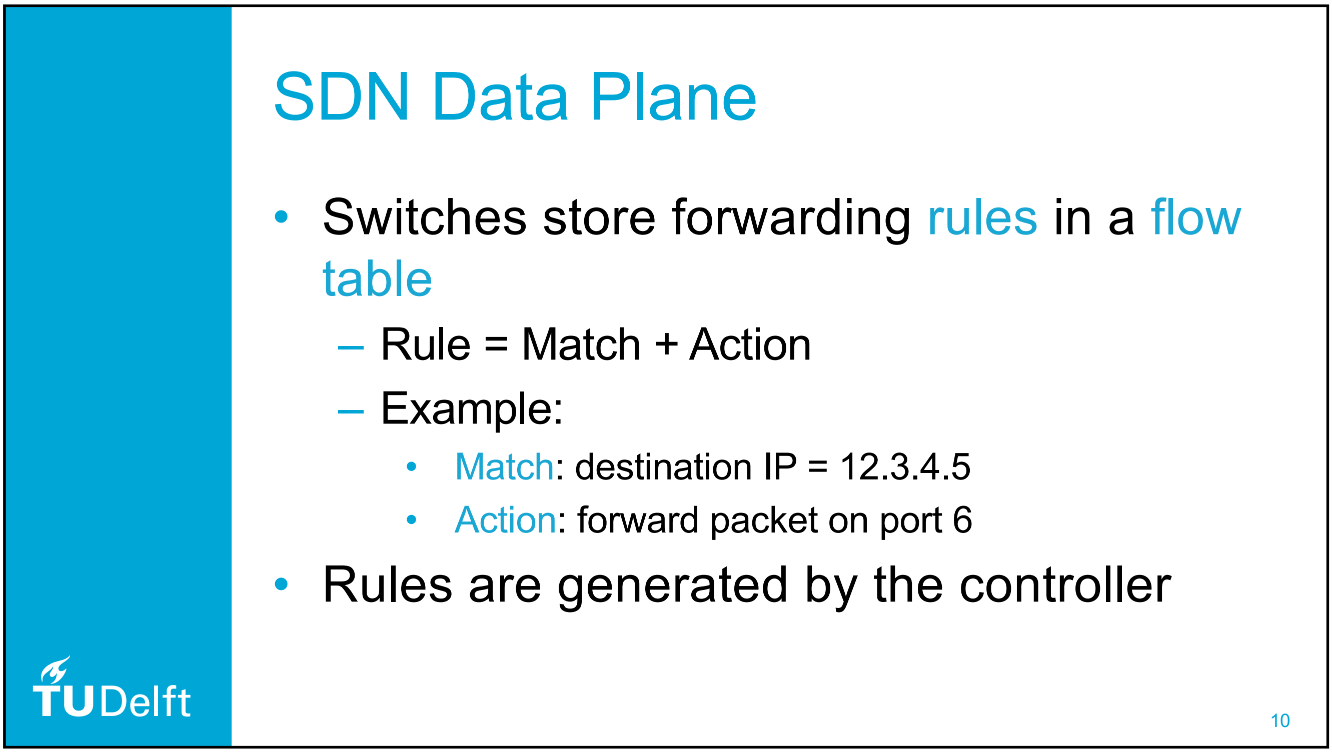
TU Delft



Currently used by

- Google
- ISPs (e.g., Comcast, Verizon)
- Datacenters
- You (exercises)

9



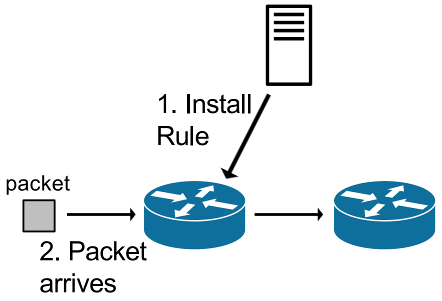
SDN Data Plane

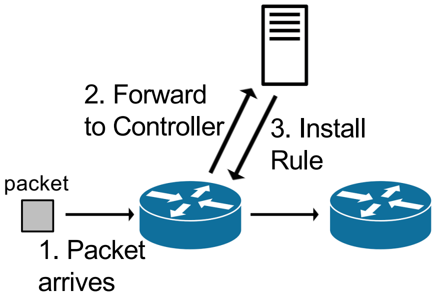
- Switches store forwarding rules in a flow table
  - Rule = Match + Action
  - Example:
    - Match: destination IP = 12.3.4.5
    - Action: forward packet on port 6
- Rules are generated by the controller

10

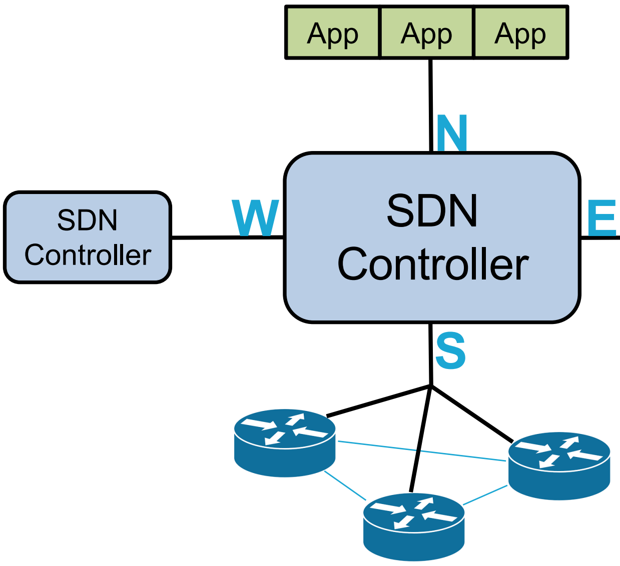
SDN Modes of Operation

- Proactive: controller installs rules on switches before packets arrive
- Reactive: controller installs rules on switches as soon as packets arrive





SDN Control Plane

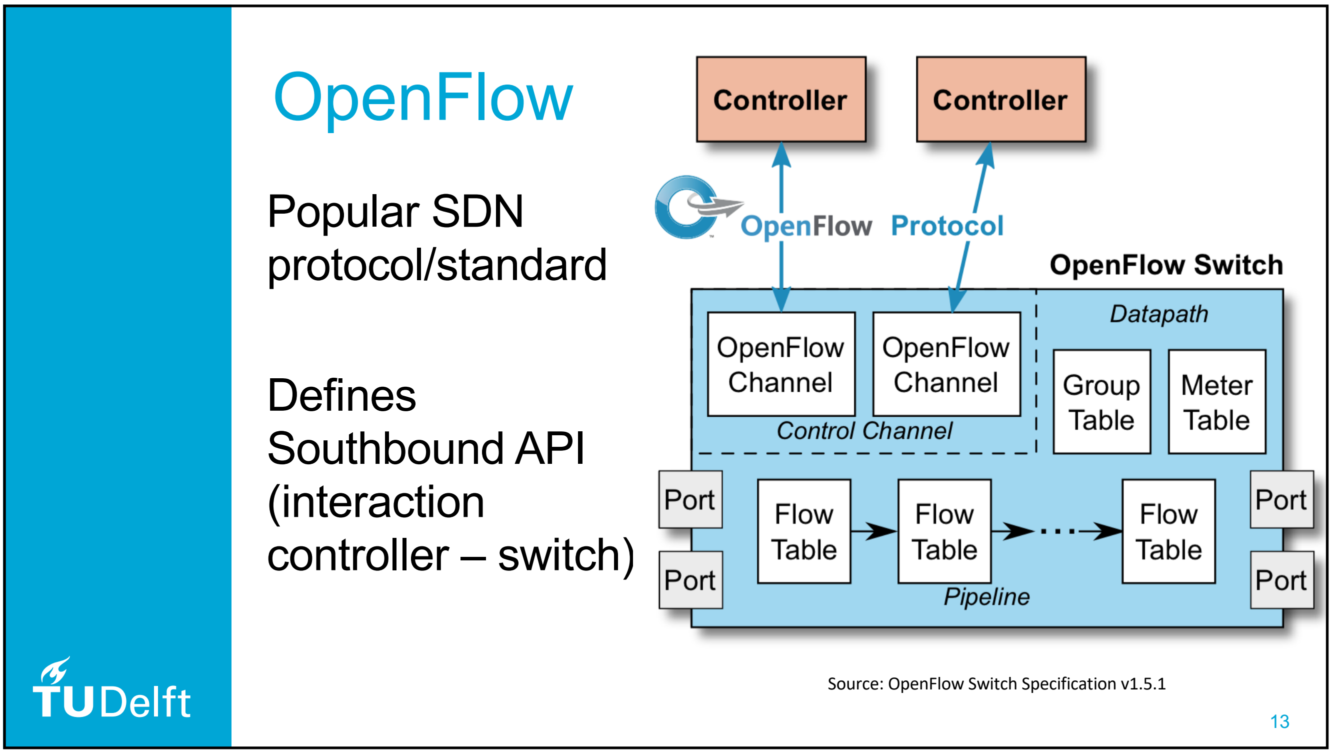


N Northbound API: Interact with network applications

S Southbound API: Interact with switches

W Westbound API: Interface to other controllers

E Eastbound API: Interface to legacy infrastructure



OpenFlow

Popular SDN protocol/standard

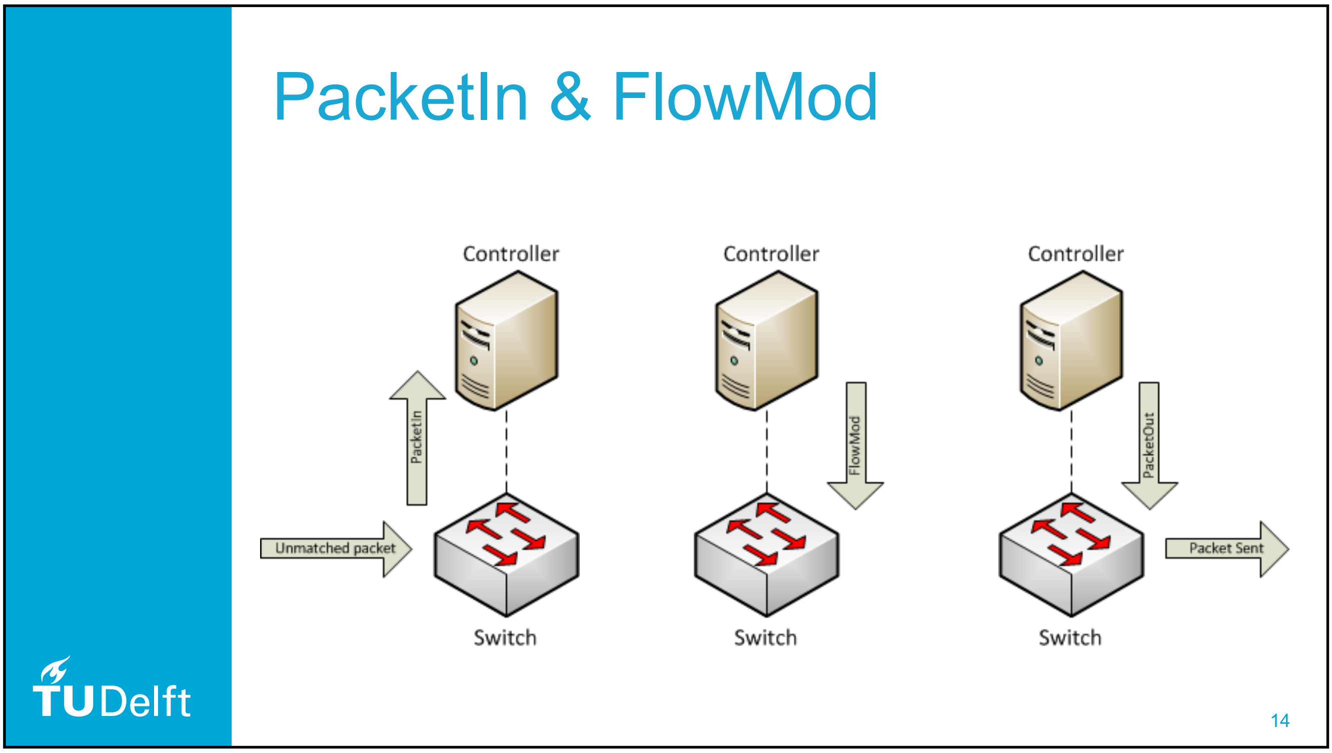
Defines Southbound API (interaction controller – switch)

Source: OpenFlow Switch Specification v1.5.1

TUDelft

13

13

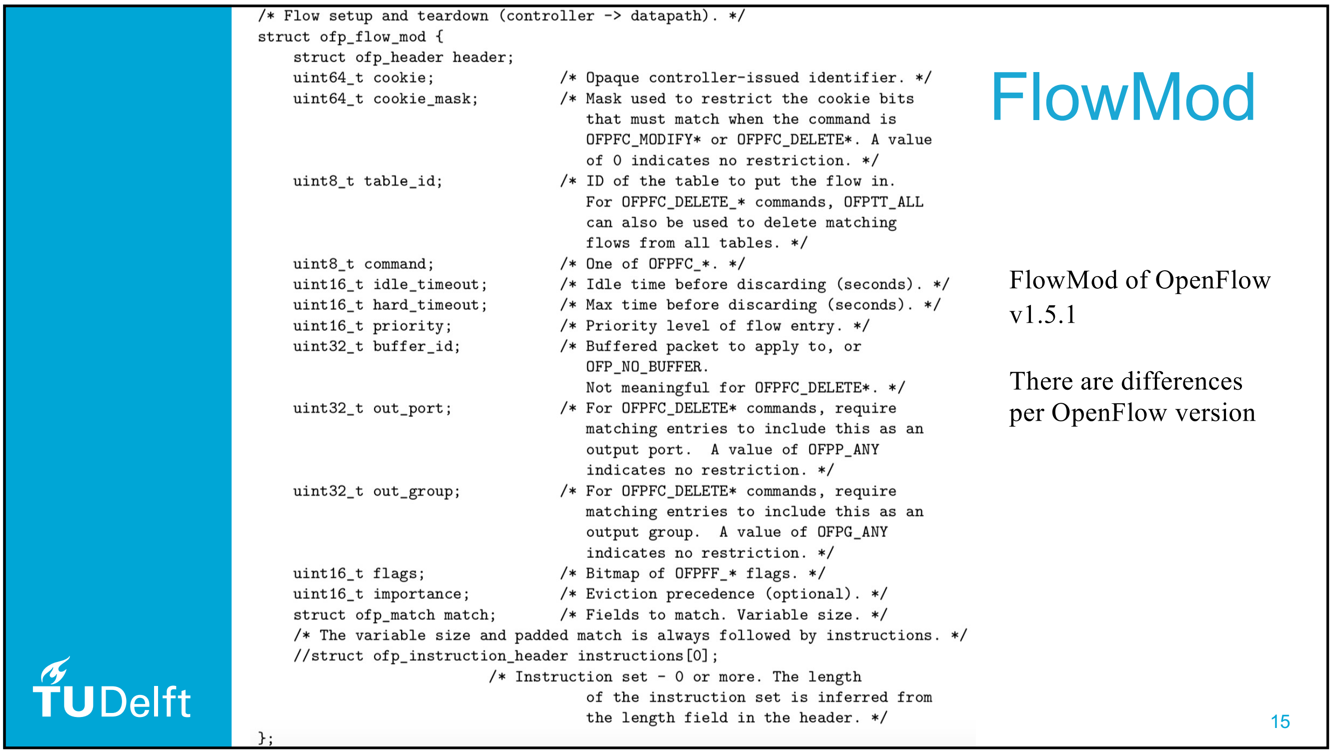


PacketIn & FlowMod

TUDelft

14

14



```
/* Flow setup and teardown (controller -> datapath). */
struct ofp_flow_mod {
    struct ofp_header header;
    uint64_t cookie;             /* Opaque controller-issued identifier. */
    uint64_t cookie_mask;        /* Mask used to restrict the cookie bits
                                    that must match when the command is
                                    OFPFC_MODIFY* or OFPFC_DELETE*. A value
                                    of 0 indicates no restriction. */
    uint8_t table_id;            /* ID of the table to put the flow in.
                                    For OFPFC_DELETE_* commands, OFPTT_ALL
                                    can also be used to delete matching
                                    flows from all tables. */
    uint8_t command;             /* One of OFPFC_*. */
    uint16_t idle_timeout;       /* Idle time before discarding (seconds). */
    uint16_t hard_timeout;       /* Max time before discarding (seconds). */
    uint16_t priority;           /* Priority level of flow entry. */
    uint32_t buffer_id;          /* Buffered packet to apply to, or
                                    OFP_NO_BUFFER.
                                    Not meaningful for OFPFC_DELETE*. */
    uint32_t out_port;           /* For OFPFC_DELETE* commands, require
                                    matching entries to include this as an
                                    output port.  A value of OFPP_ANY
                                    indicates no restriction. */
    uint32_t out_group;          /* For OFPFC_DELETE* commands, require
                                    matching entries to include this as an
                                    output group.  A value of OFPG_ANY
                                    indicates no restriction. */
    uint16_t flags;              /* Bitmap of OFPFF_* flags. */
    uint16_t importance;         /* Eviction precedence (optional). */
    struct ofp_match match;      /* Fields to match. Variable size. */
    /* The variable size and padded match is always followed by instructions. */
    //struct ofp_instruction_header instructions[0];
                             /* Instruction set - 0 or more. The length
                                    of the instruction set is inferred from
                                    the length field in the header. */
};
```

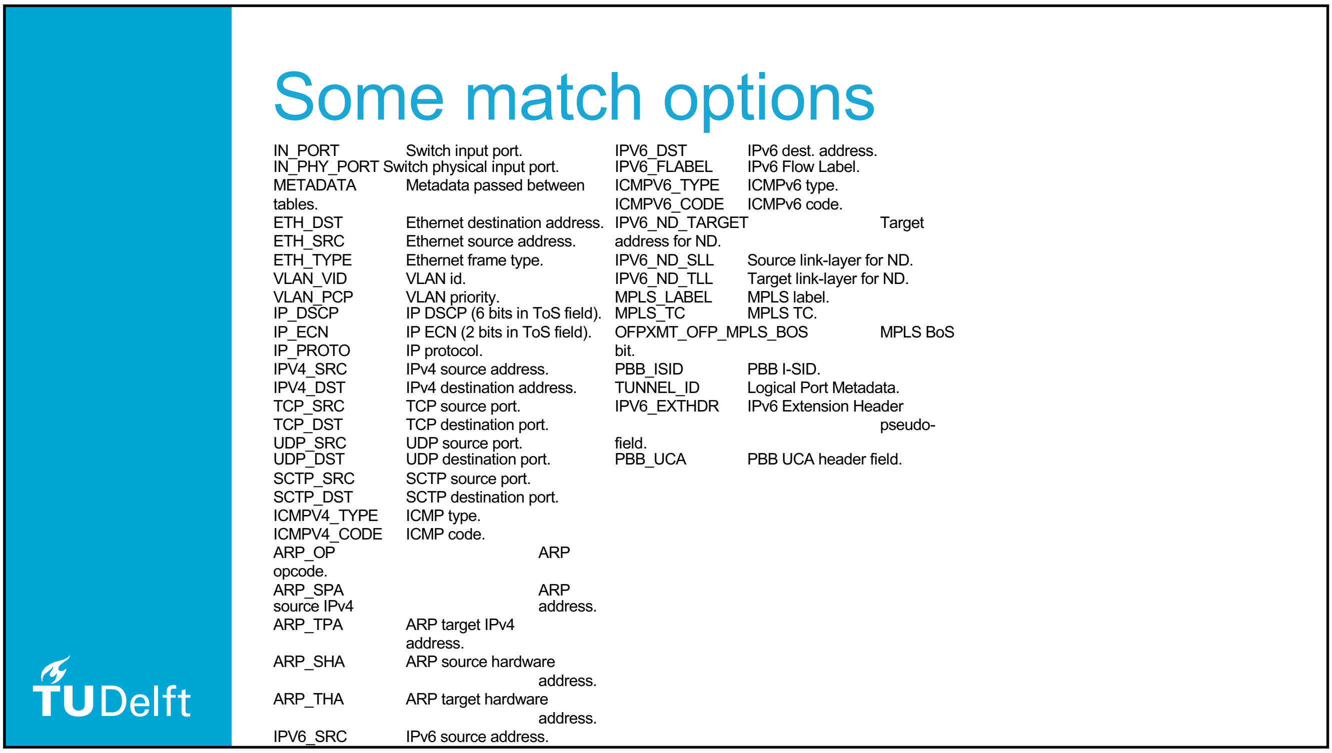
FlowMod

FlowMod of OpenFlow v1.5.1

There are differences per OpenFlow version

15

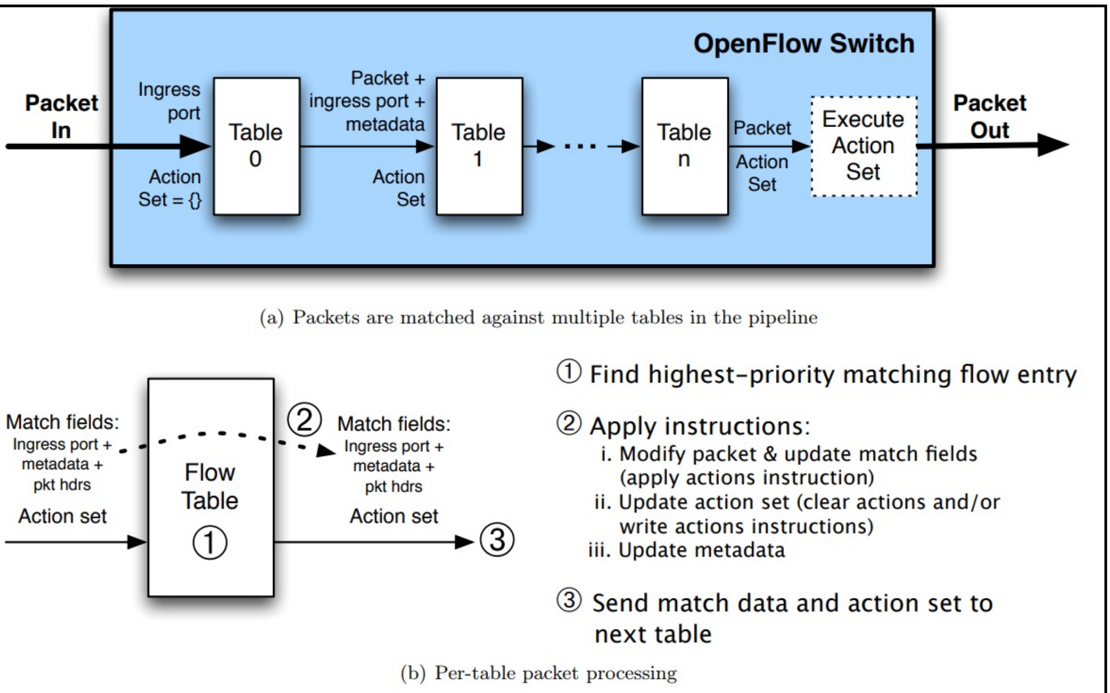
15



Some match options

IN_PORT	Switch input port.
IN_PHY_PORT	Switch physical input port.
METADATA	Metadata passed between tables.
ETH_DST	Ethernet destination address.
ETH_SRC	Ethernet source address.
ETH_TYPE	Ethernet frame type.
VLAN_VID	VLAN id.
VLAN_PCP	VLAN priority.
IP_DSCP	IP DSCP (6 bits in ToS field).
IP_ECN	IP ECN (2 bits in ToS field).
IP_PROTO	IP protocol.
IPV4_SRC	IPv4 source address.
IPV4_DST	IPv4 destination address.
TCP_SRC	TCP source port.
TCP_DST	TCP destination port.
UDP_SRC	UDP source port.
UDP_DST	UDP destination port.
SCTP_SRC	SCTP source port.
SCTP_DST	SCTP destination port.
ICMPV4_TYPE	ICMP type.
ICMPV4_CODE	ICMP code.
ARP_OP	ARP opcode.
ARP_SPA	ARP source IPv4 address.
ARP_TPA	ARP target IPv4 address.
ARP_SHA	ARP source hardware address.
ARP_THA	ARP target hardware address.
IPV6_SRC	IPv6 source address.
IPV6_DST	IPv6 dest. address.
IPV6_FLABEL	IPv6 Flow Label.
ICMPV6_TYPE	ICMPv6 type.
ICMPV6_CODE	ICMPv6 code.
IPV6_ND_TARGET	Target address for ND.
IPV6_ND_SLL	Source link-layer for ND.
IPV6_ND_TLL	Target link-layer for ND.
MPLS_LABEL	MPLS label.
MPLS_TC	MPLS TC.
OFPXMT_OFP_MPLS_BOS	MPLS BoS bit.
PBB_ISID	PBB I-SID.
TUNNEL_ID	Logical Port Metadata.
IPV6_EXTHDR	IPv6 Extension Header pseudo-field.
PBB_UCA	PBB UCA header field.

16



(a) Packets are matched against multiple tables in the pipeline

① Find highest-priority matching flow entry

② Apply instructions:
i. Modify packet & update match fields (apply actions instruction)
ii. Update action set (clear actions and/or write actions instructions)
iii. Update metadata

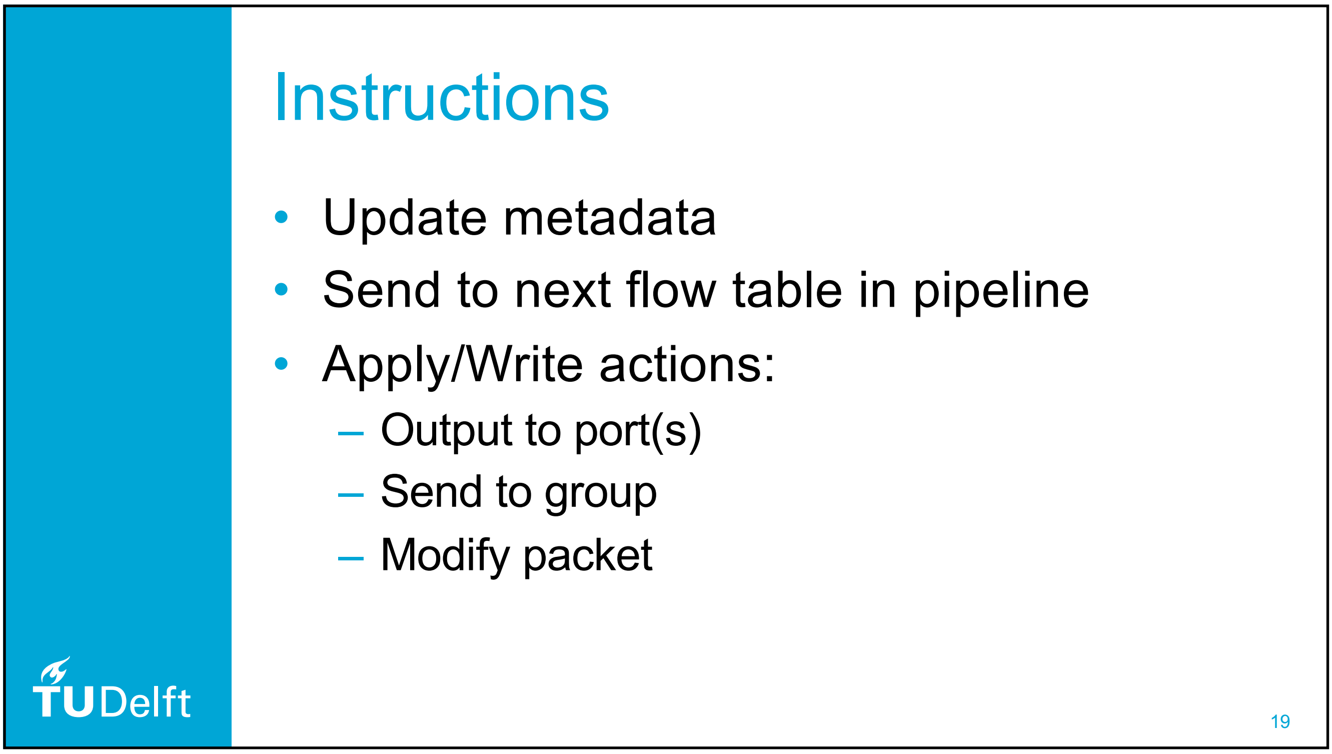
③ Send match data and action set to next table

(b) Per-table packet processing

Source: OpenFlow Switch Specification v1.3

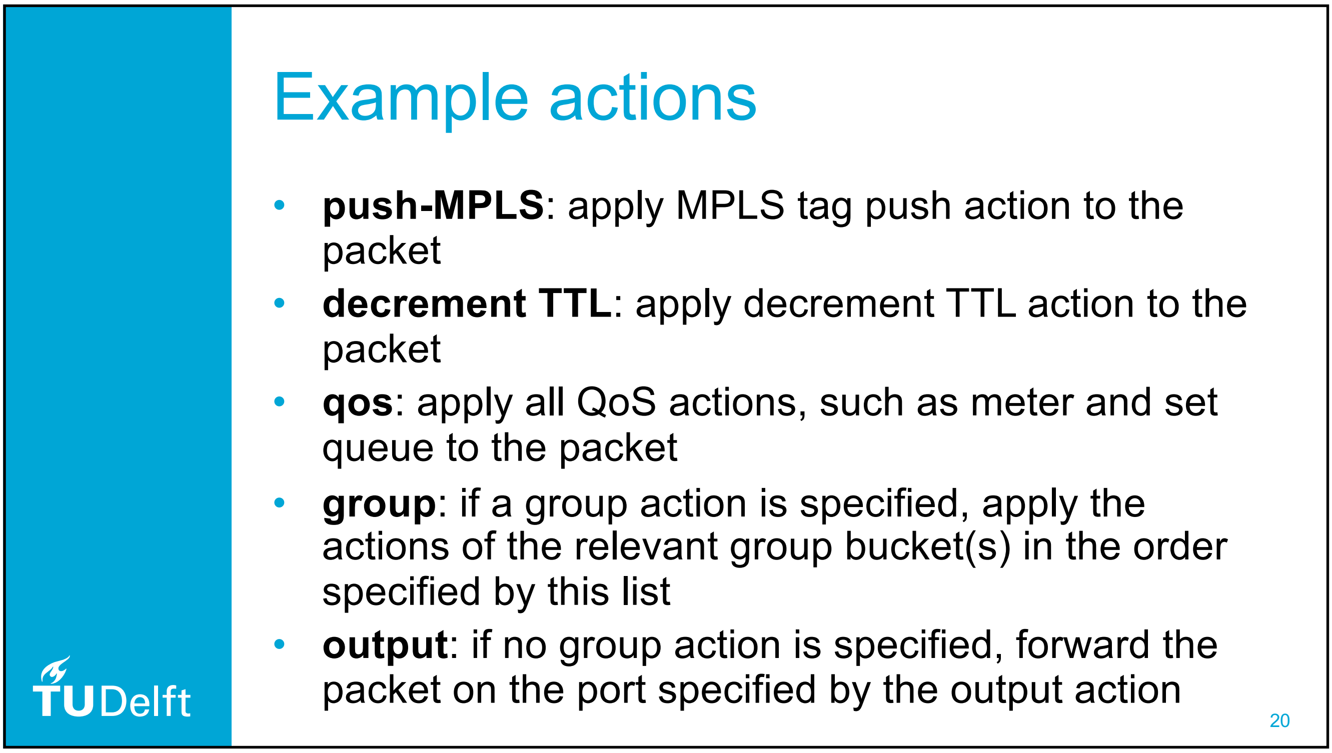
Flow Table entries

- Match Fields:
  - Ingress port
  - Packet headers (e.g., protocol, dst)
  - Metadata
- Priority
- Instructions



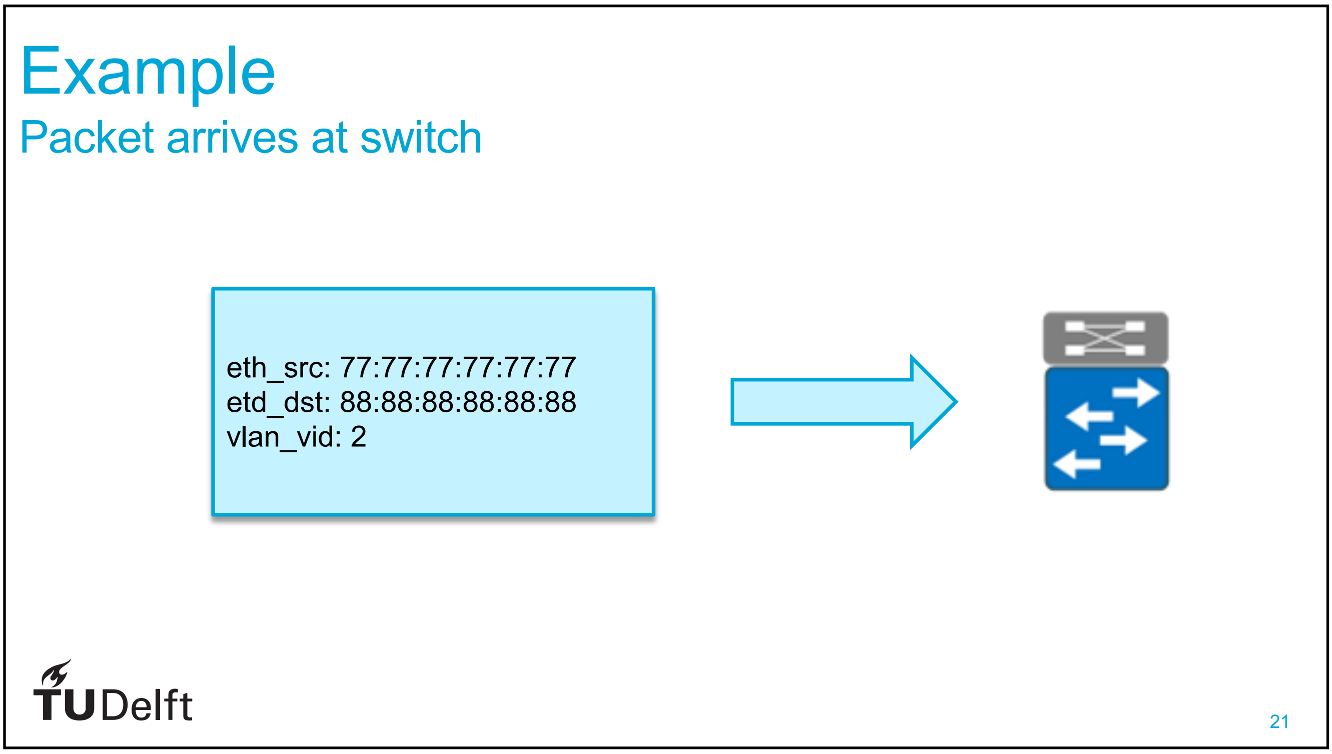
Instructions

- Update metadata
- Send to next flow table in pipeline
- Apply/Write actions:
  - Output to port(s)
  - Send to group
  - Modify packet



Example actions

- **push-MPLS**: apply MPLS tag push action to the packet
- **decrement TTL**: apply decrement TTL action to the packet
- **qos**: apply all QoS actions, such as meter and set queue to the packet
- **group**: if a group action is specified, apply the actions of the relevant group bucket(s) in the order specified by this list
- **output**: if no group action is specified, forward the packet on the port specified by the output action

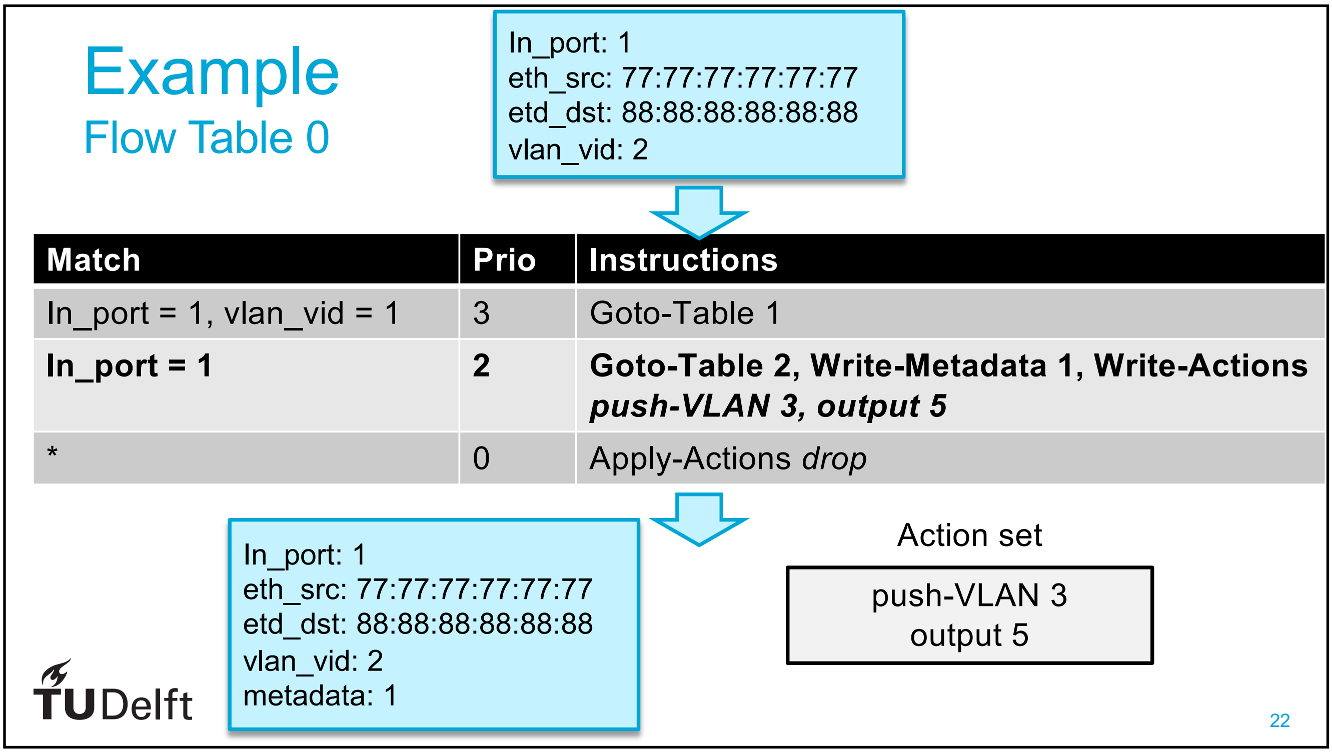


Example

Packet arrives at switch

eth_src: 77:77:77:77:77:77
etd_dst: 88:88:88:88:88:88
vlan_vid: 2

TUDelft



Example

Flow Table 0

In_port: 1
eth_src: 77:77:77:77:77:77
etd_dst: 88:88:88:88:88:88
vlan_vid: 2

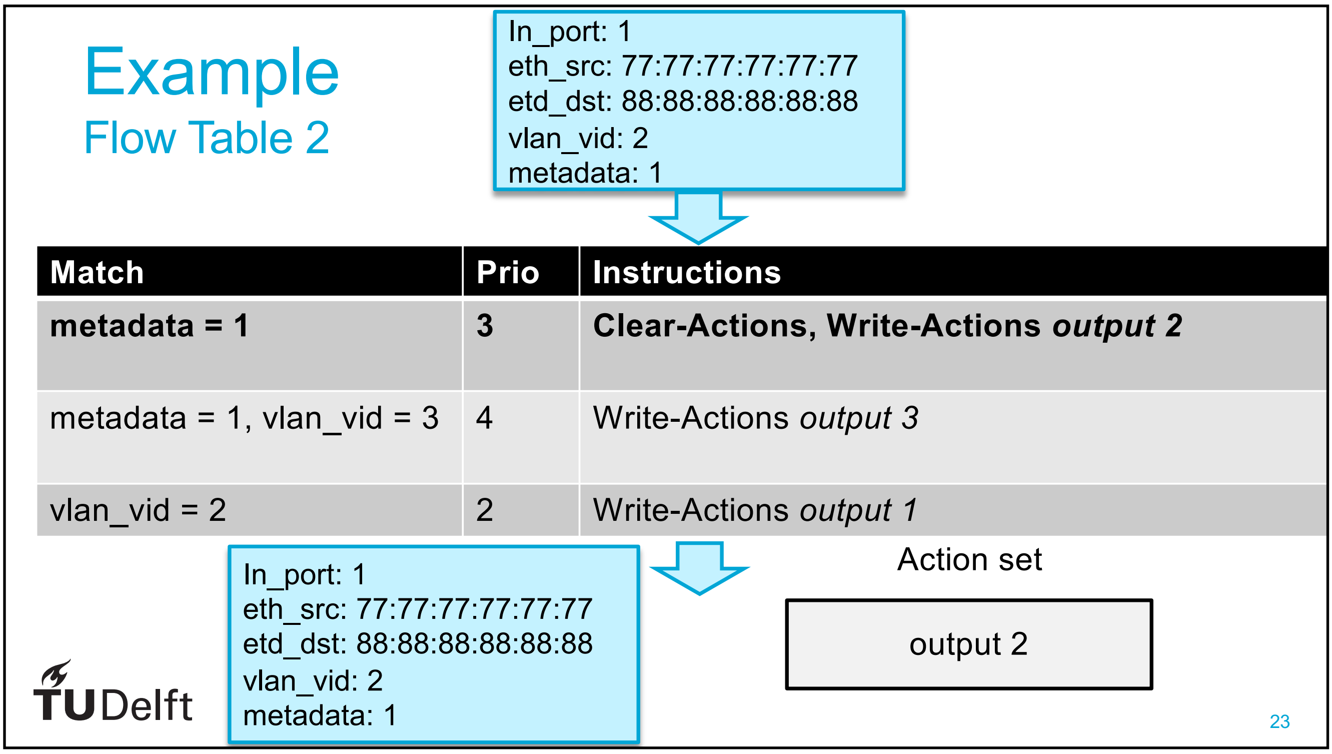
Match	Prio	Instructions
In_port = 1, vlan_vid = 1	3	Goto-Table 1
In_port = 1	**2**	**Goto-Table 2, Write-Metadata 1, Write-Actions *push-VLAN 3, output 5***
*	0	Apply-Actions *drop*

In_port: 1
eth_src: 77:77:77:77:77:77
etd_dst: 88:88:88:88:88:88
vlan_vid: 2
metadata: 1

Action set

push-VLAN 3
output 5

TUDelft



Example

Flow Table 2

In_port: 1
eth_src: 77:77:77:77:77:77
etd_dst: 88:88:88:88:88:88
vlan_vid: 2
metadata: 1

Match	Prio	Instructions
metadata = 1	**3**	**Clear-Actions, Write-Actions *output 2***
metadata = 1, vlan_vid = 3	4	Write-Actions *output 3*
vlan_vid = 2	2	Write-Actions *output 1*

In_port: 1
eth_src: 77:77:77:77:77:77
etd_dst: 88:88:88:88:88:88
vlan_vid: 2
metadata: 1

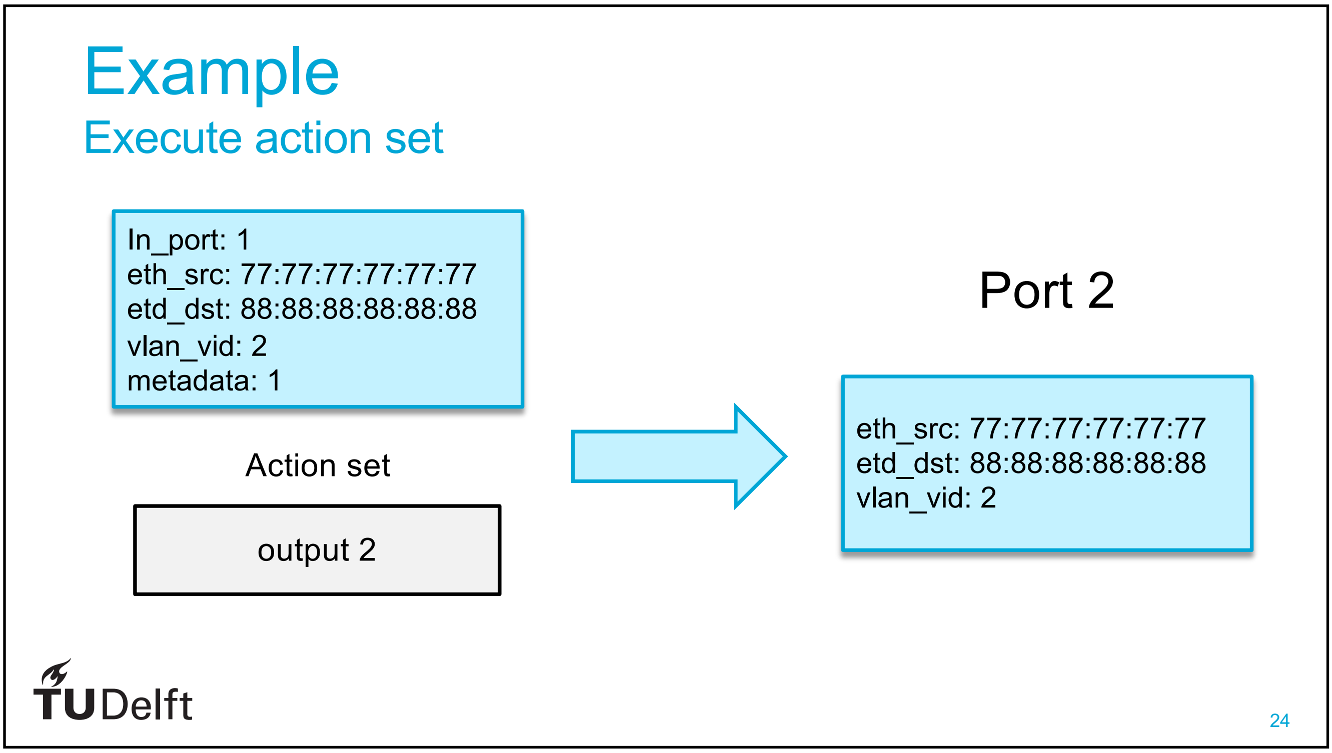
Action set

output 2

TUDelft

23

23



Example

Execute action set

In_port: 1
eth_src: 77:77:77:77:77:77
etd_dst: 88:88:88:88:88:88
vlan_vid: 2
metadata: 1

Action set

output 2

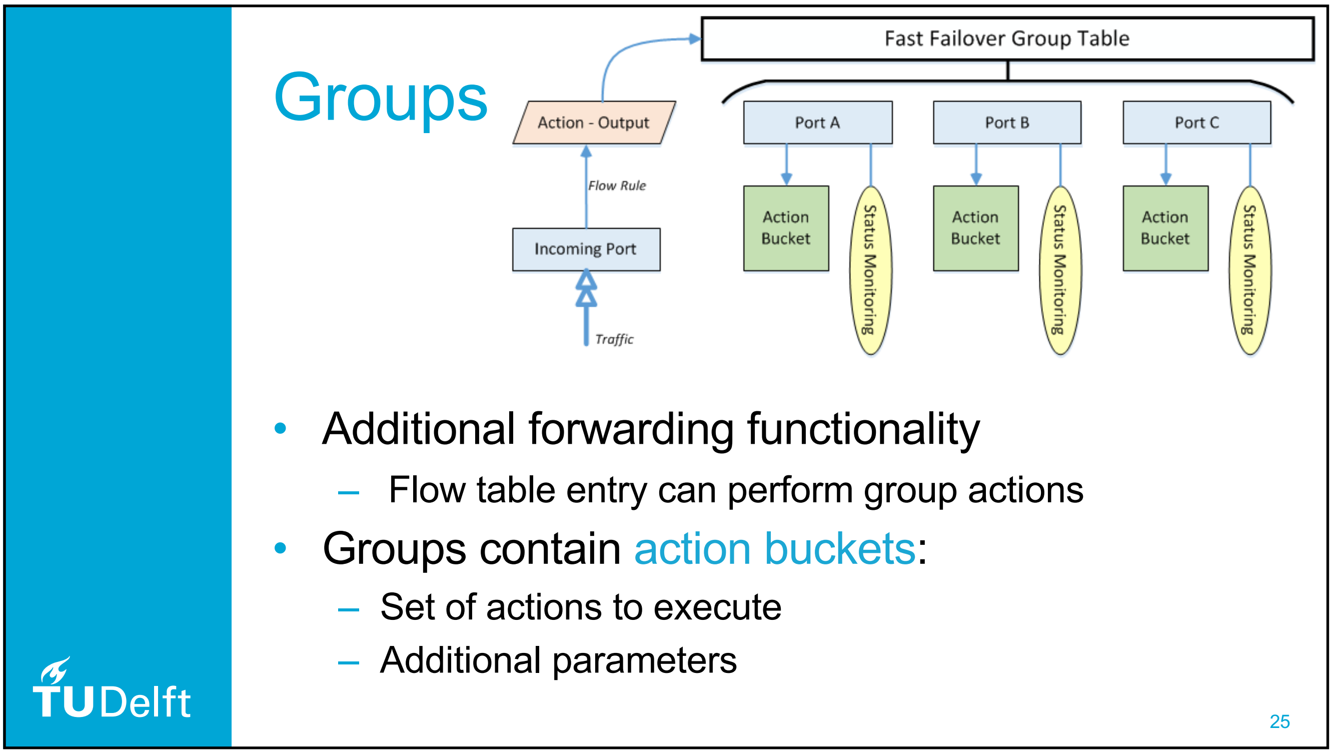
Port 2

eth_src: 77:77:77:77:77:77
etd_dst: 88:88:88:88:88:88
vlan_vid: 2

TUDelft

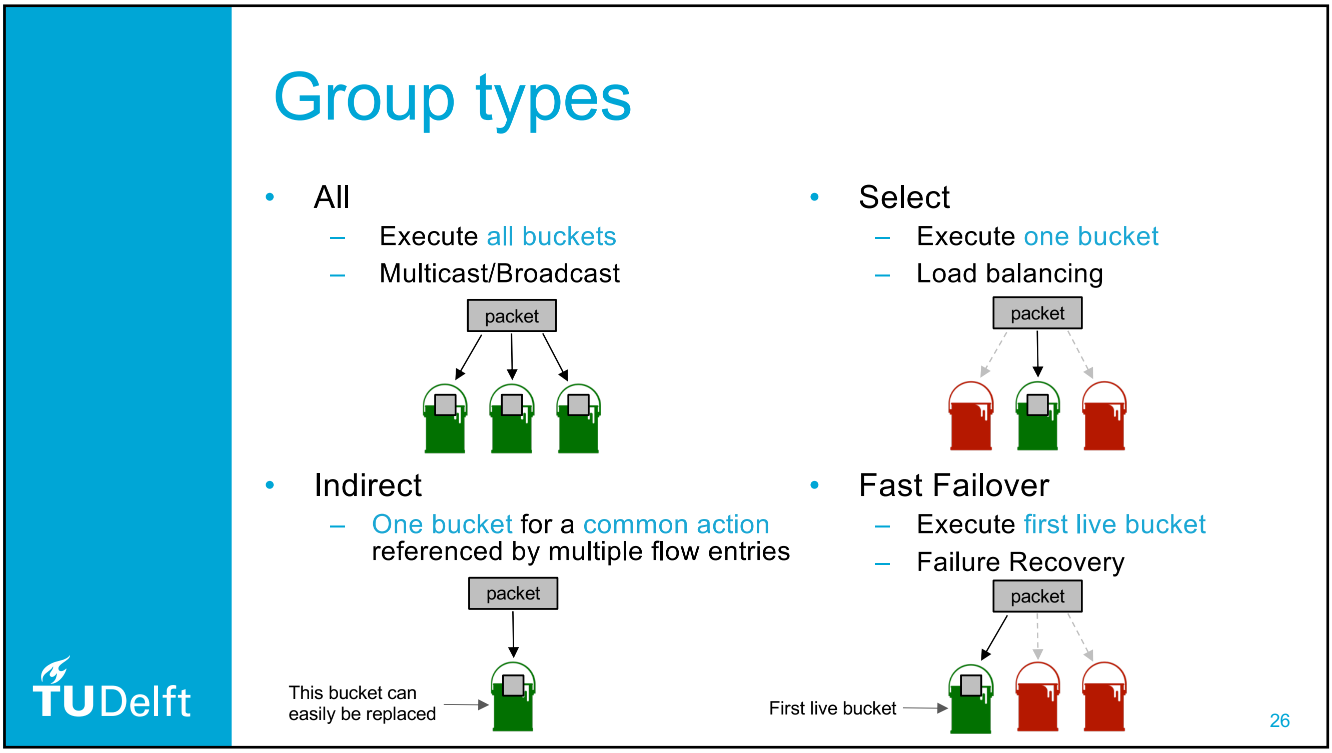
24

24



Groups

- Additional forwarding functionality
  - Flow table entry can perform group actions
- Groups contain action buckets:
  - Set of actions to execute
  - Additional parameters



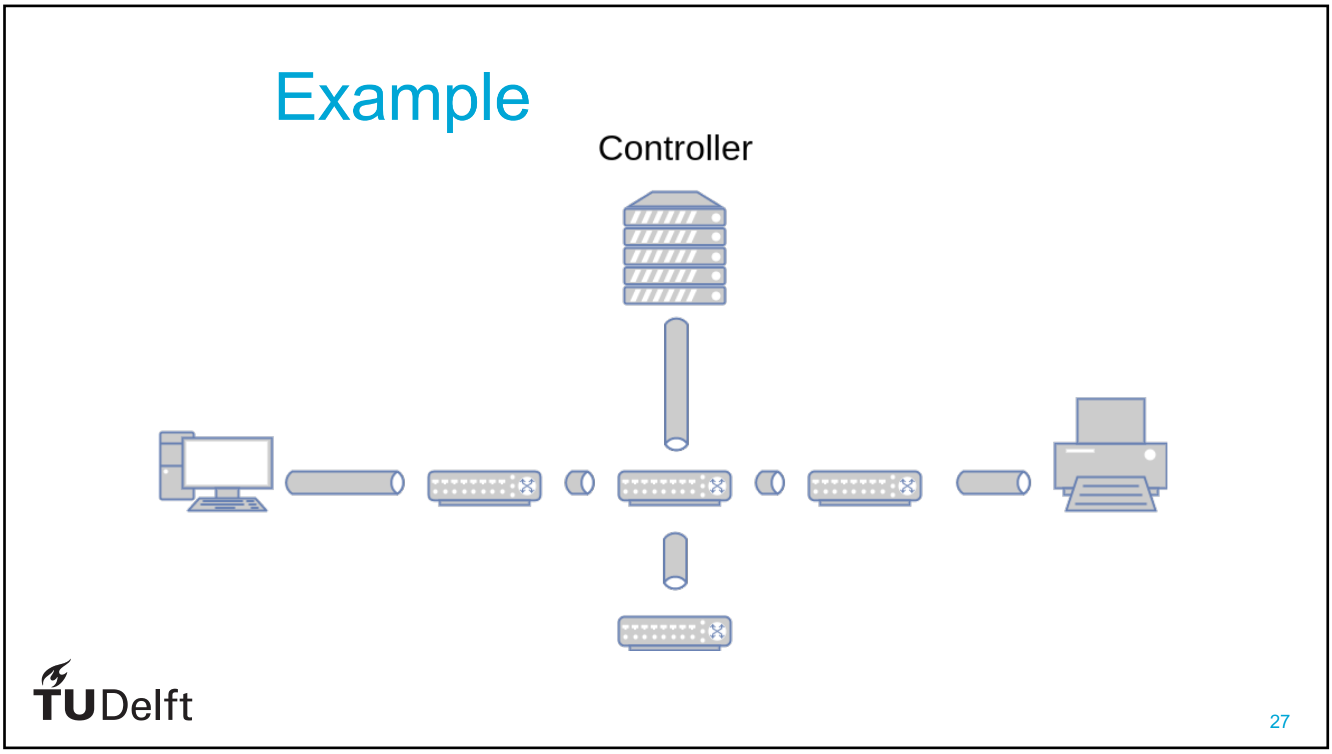
Group types

- All
  - Execute all buckets
  - Multicast/Broadcast

- Select
  - Execute one bucket
  - Load balancing

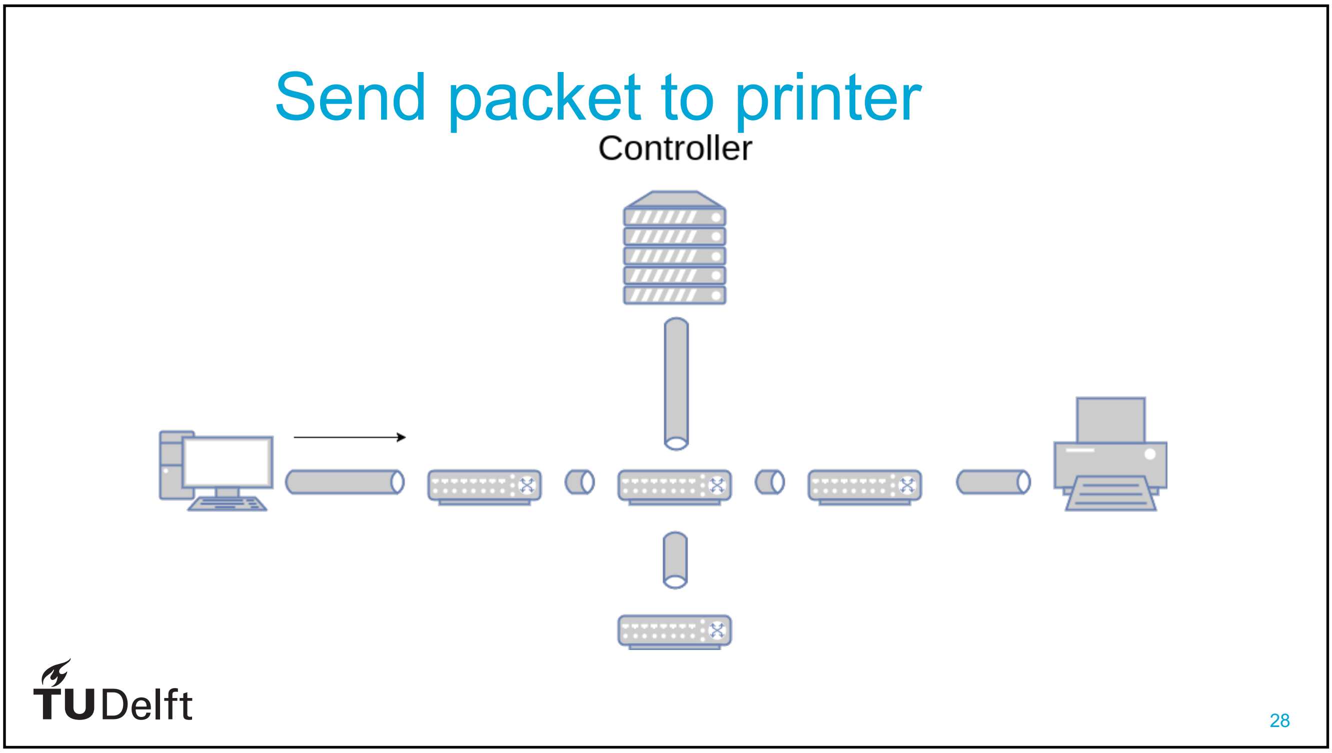
- Indirect
  - One bucket for a common action referenced by multiple flow entries

- Fast Failover
  - Execute first live bucket
  - Failure Recovery



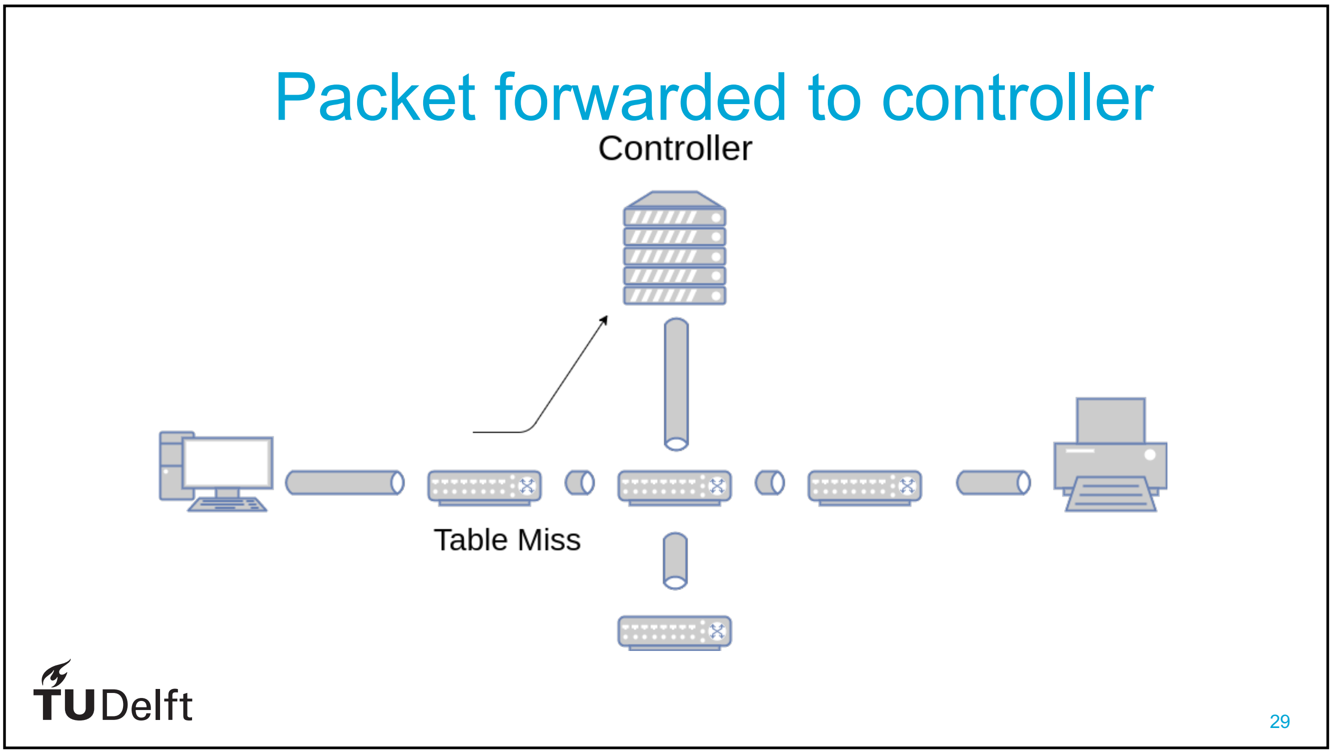
Example

Controller



Send packet to printer

Controller

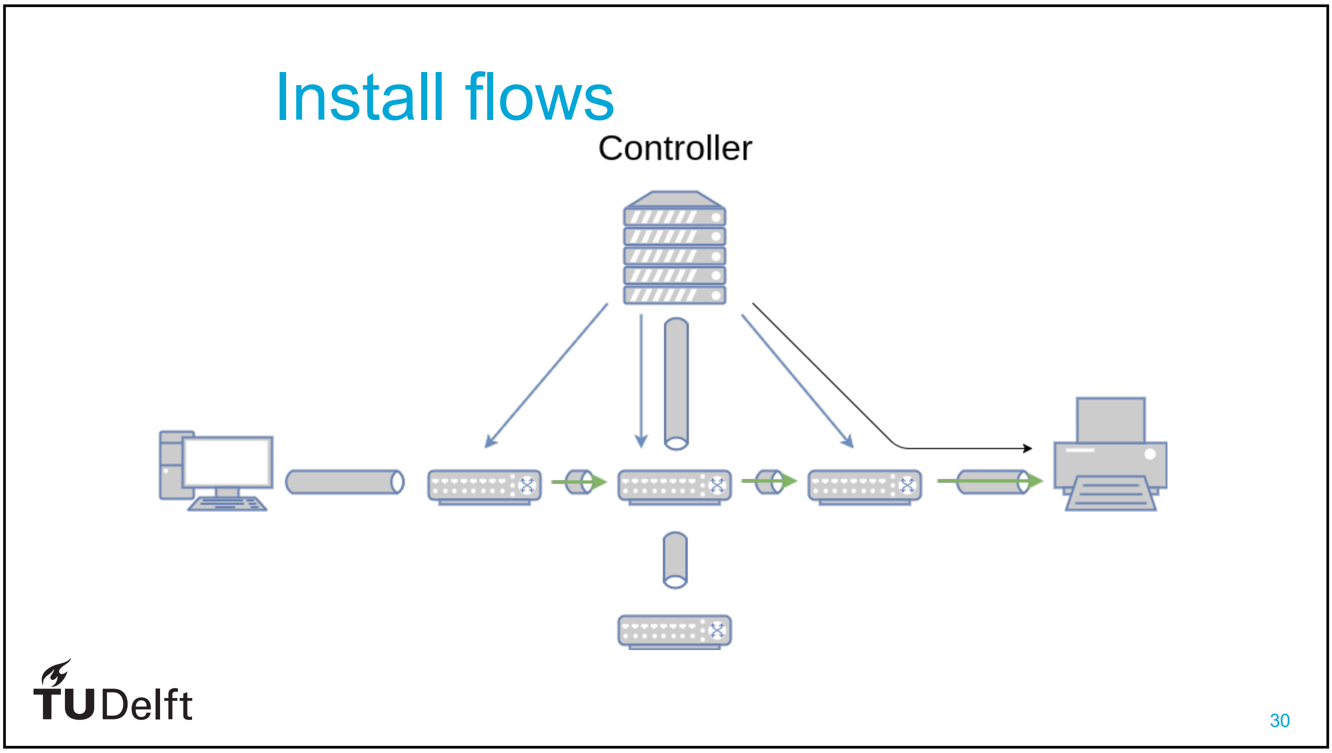


Packet forwarded to controller

TUDelft

29

29

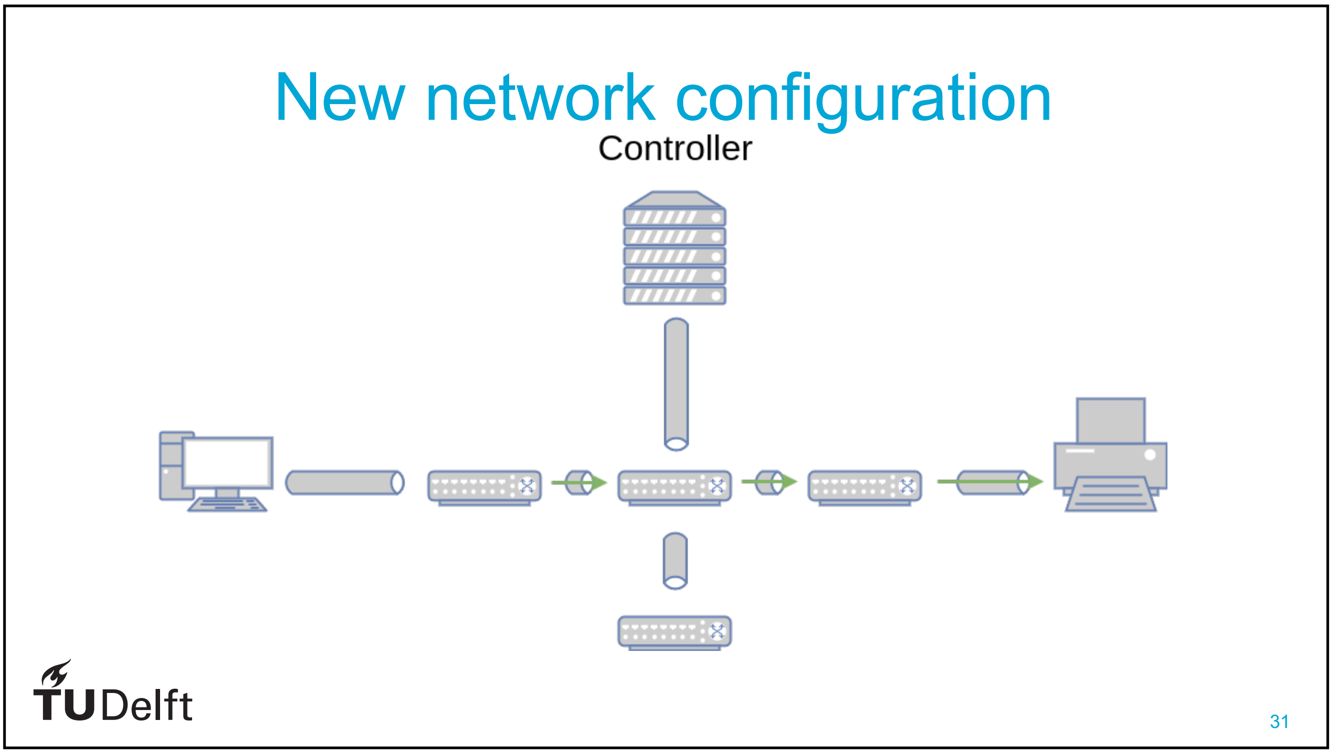


Install flows

TUDelft

30

30

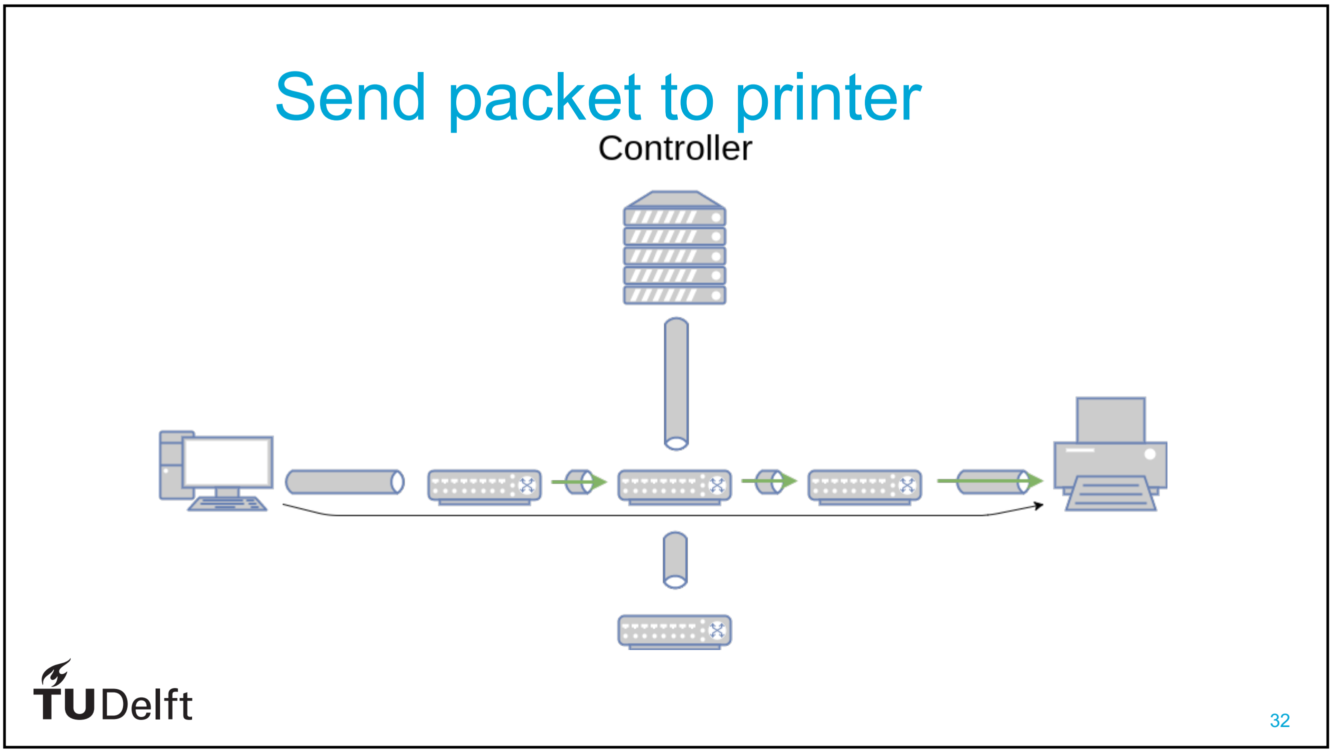


New network configuration

TUDelft

31

31

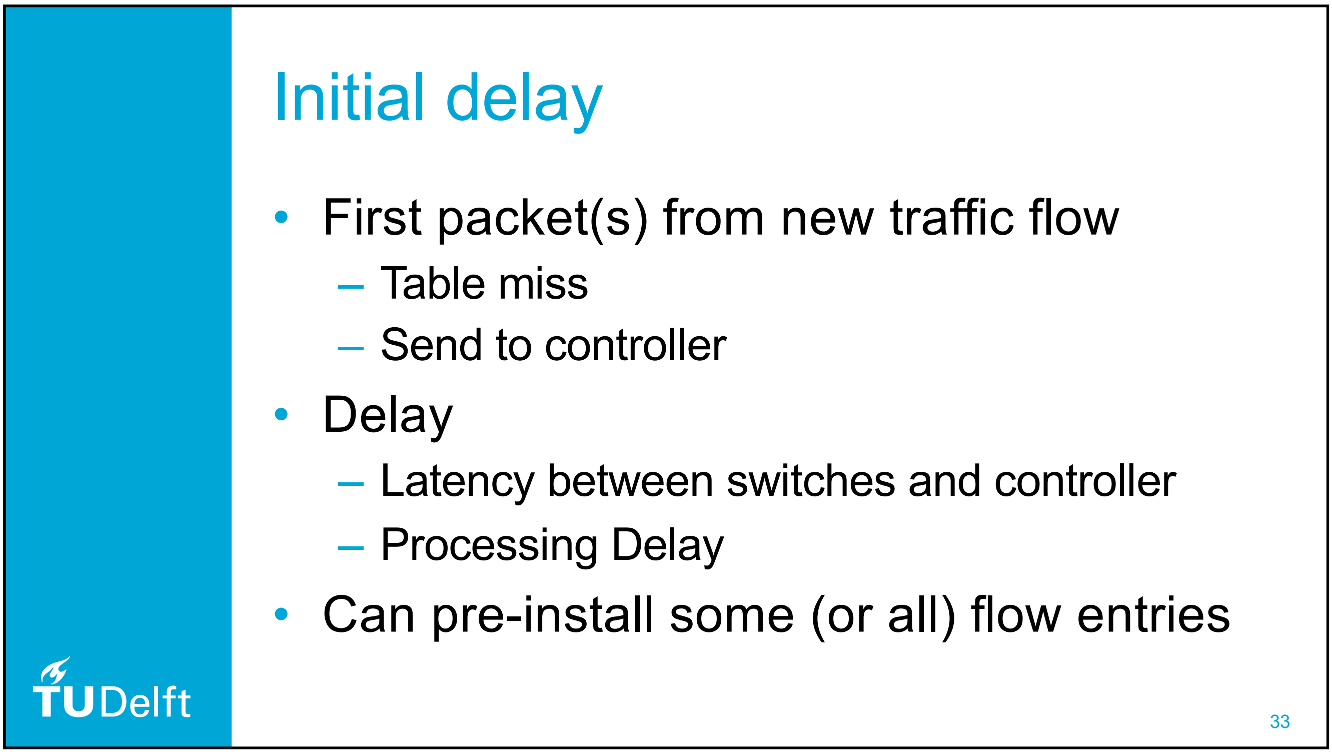


Send packet to printer

TUDelft

32

32



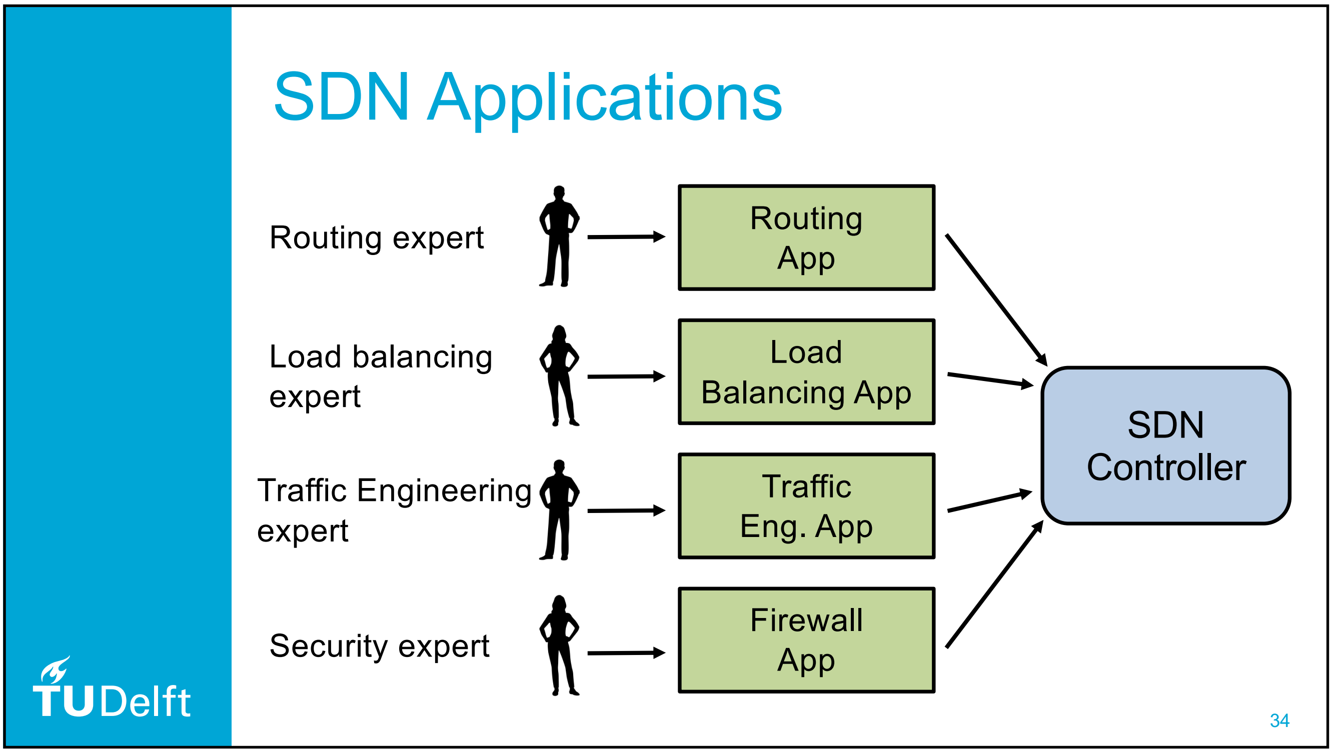
Initial delay

- First packet(s) from new traffic flow
  - Table miss
  - Send to controller
- Delay
  - Latency between switches and controller
  - Processing Delay
- Can pre-install some (or all) flow entries

TUDelft

33

33

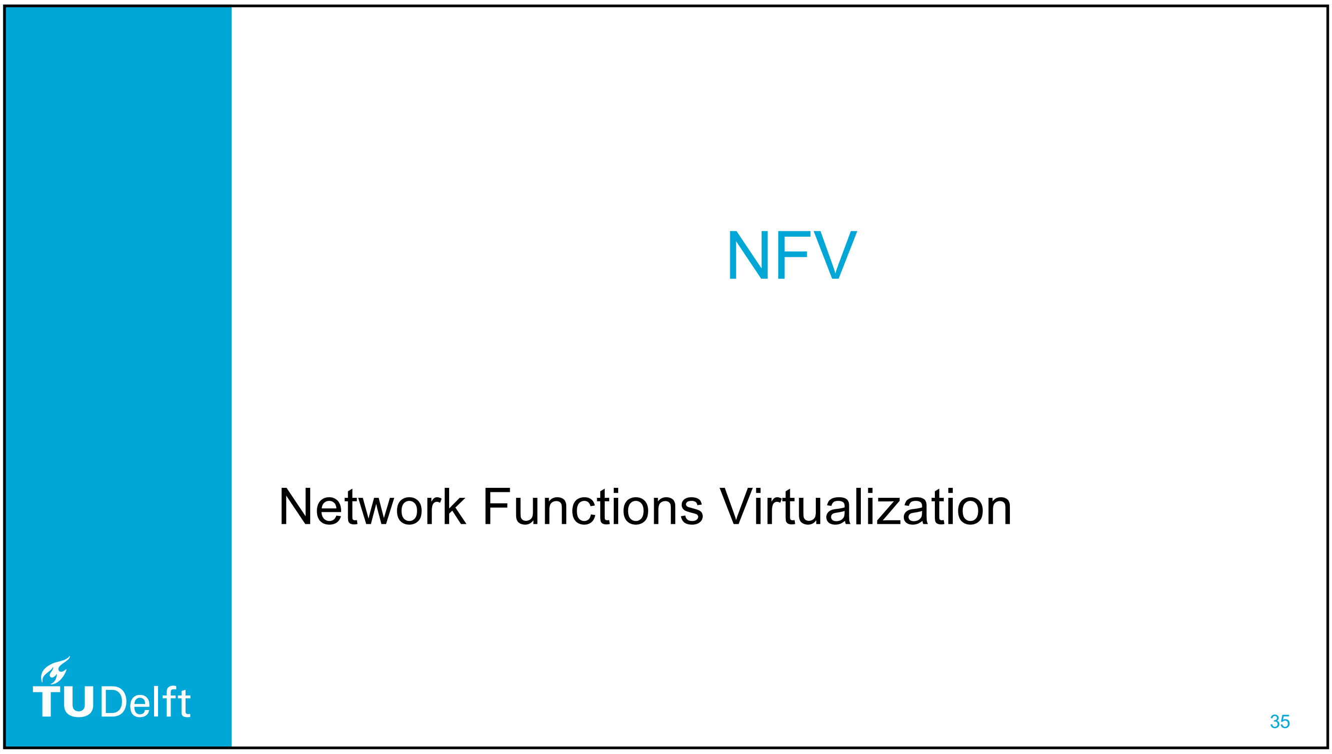


SDN Applications

TUDelft

34

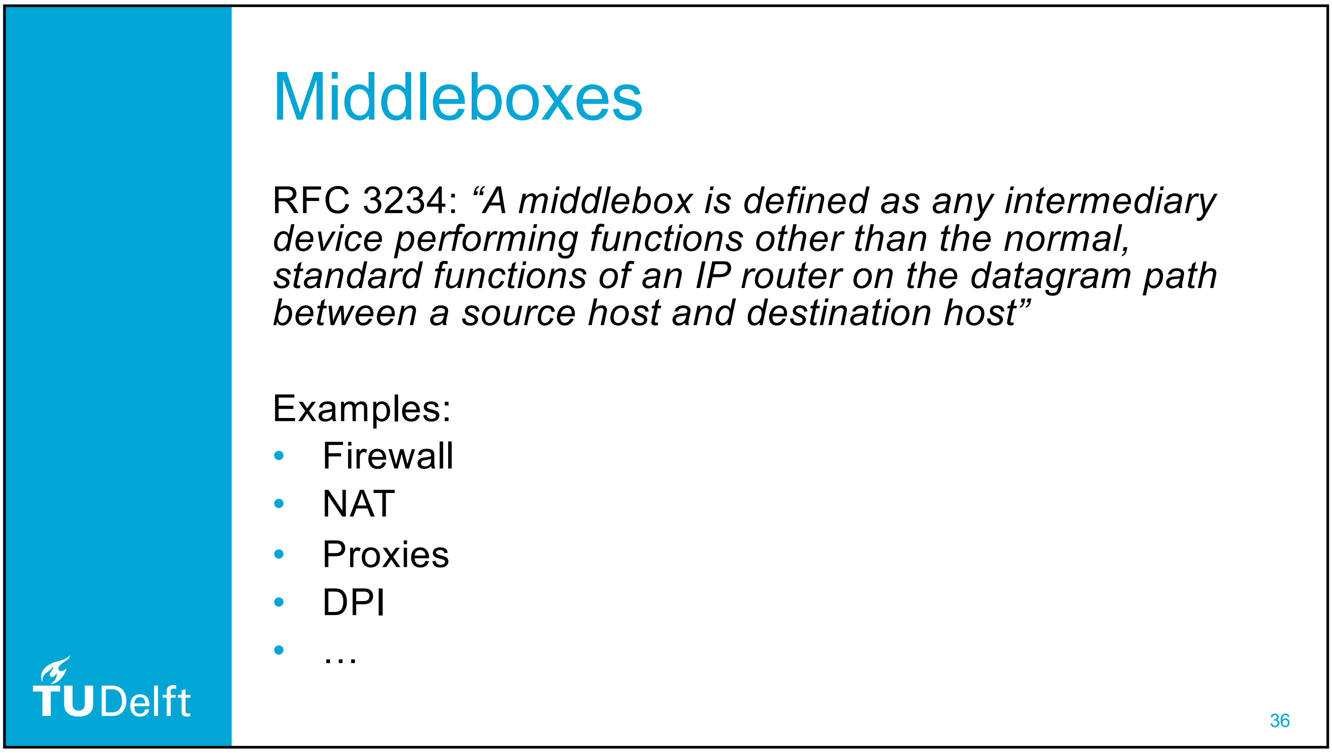
34



NFV

Network Functions Virtualization

TUDelft



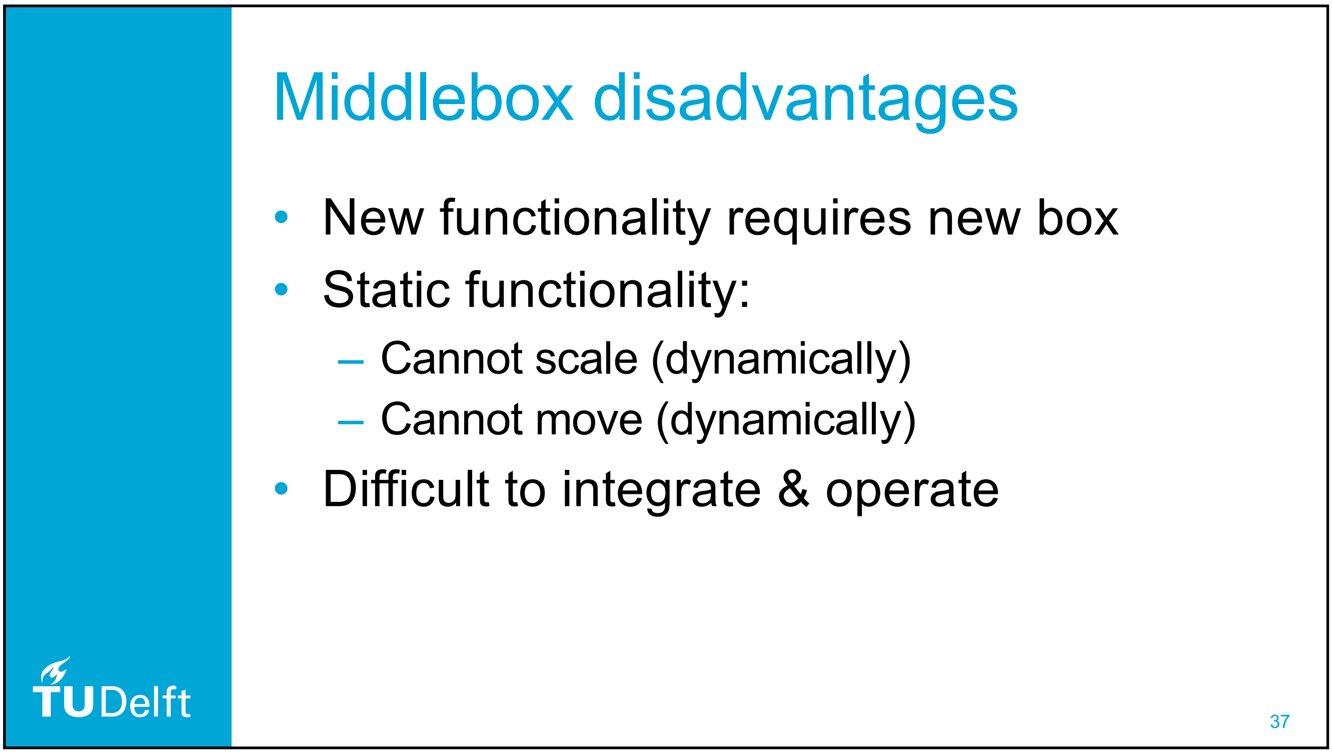
Middleboxes

RFC 3234: *"A middlebox is defined as any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host"*

Examples:

- Firewall
- NAT
- Proxies
- DPI
- …

TUDelft



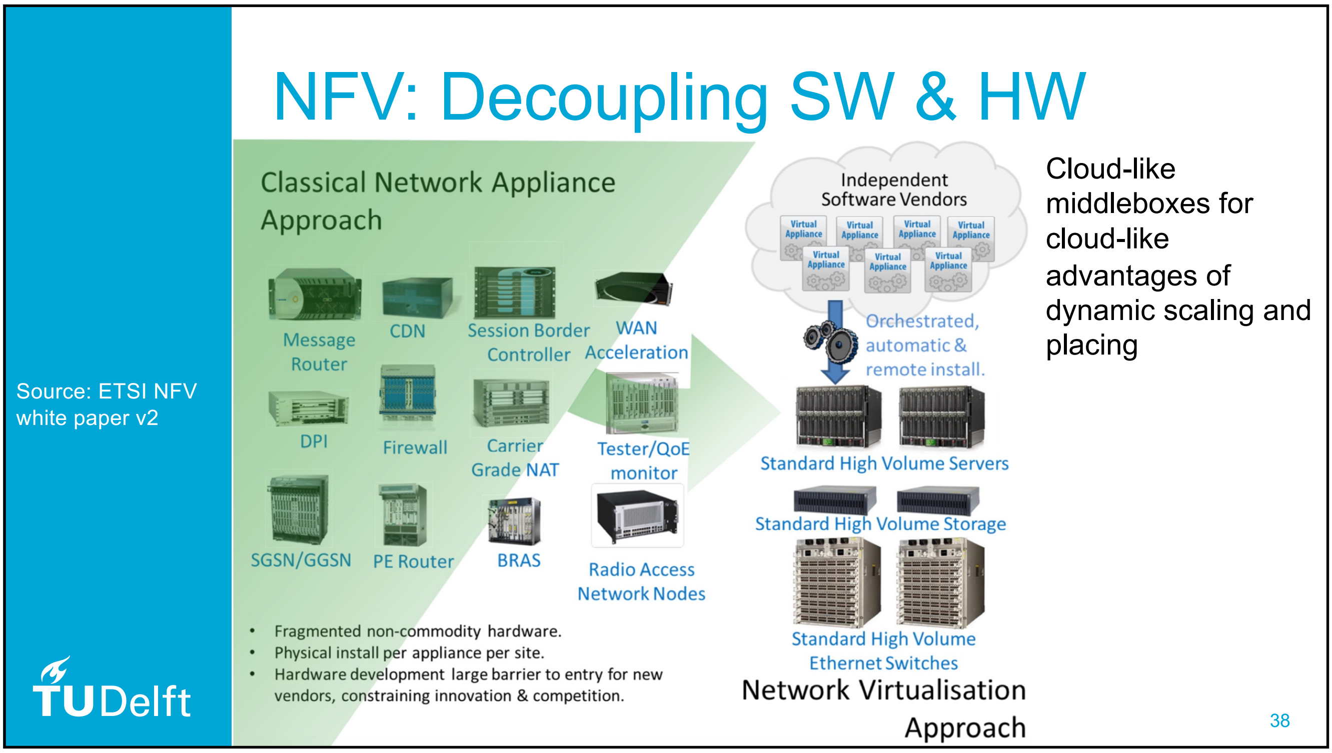
Middlebox disadvantages

- New functionality requires new box
- Static functionality:
  - Cannot scale (dynamically)
  - Cannot move (dynamically)
- Difficult to integrate & operate

TUDelft

37

37



NFV: Decoupling SW & HW

Source: ETSI NFV white paper v2

Classical Network Appliance Approach

Message Router
CDN
Session Border Controller
WAN Acceleration
DPI
Firewall
Carrier Grade NAT
Tester/QoE monitor
SGSN/GGSN
PE Router
BRAS
Radio Access Network Nodes

- Fragmented non-commodity hardware.
- Physical install per appliance per site.
- Hardware development large barrier to entry for new vendors, constraining innovation & competition.

Independent Software Vendors
Virtual Appliance
Orchestrated, automatic & remote install.
Standard High Volume Servers
Standard High Volume Storage
Standard High Volume Ethernet Switches

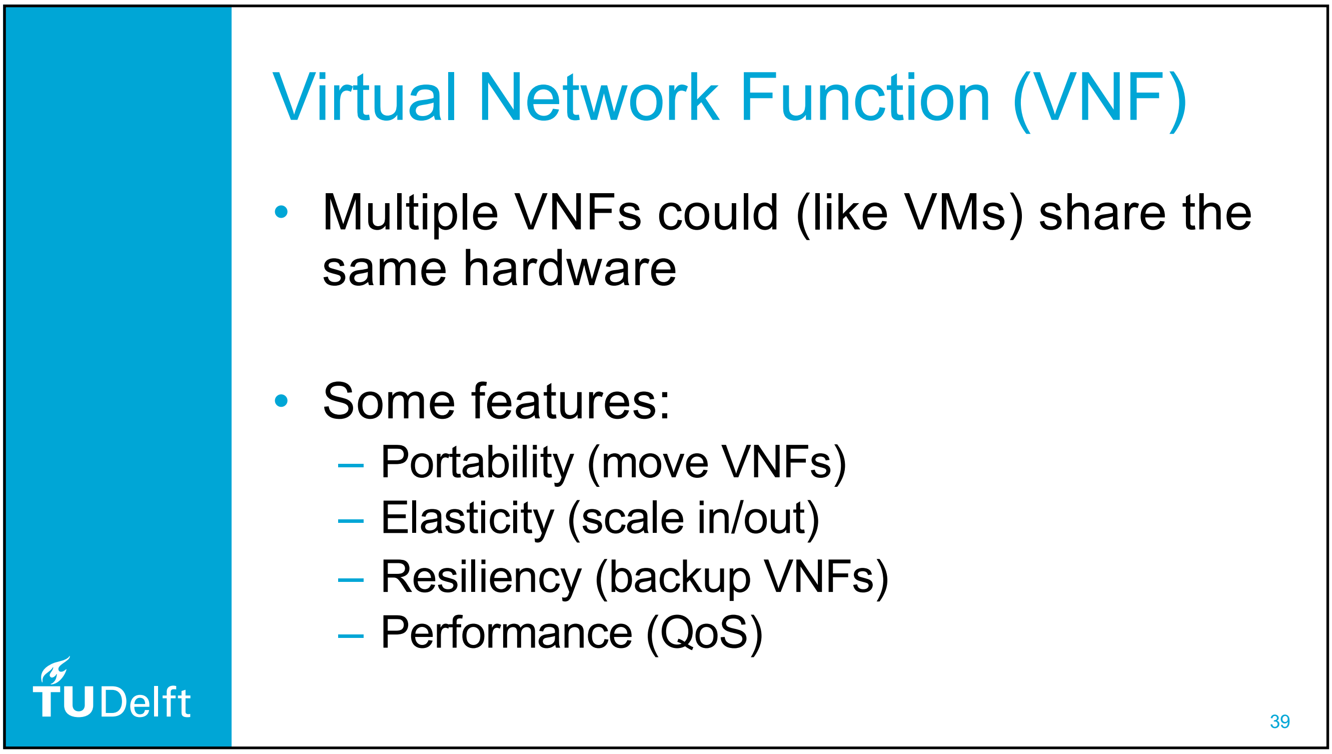
Network Virtualisation Approach

Cloud-like middleboxes for cloud-like advantages of dynamic scaling and placing

TUDelft

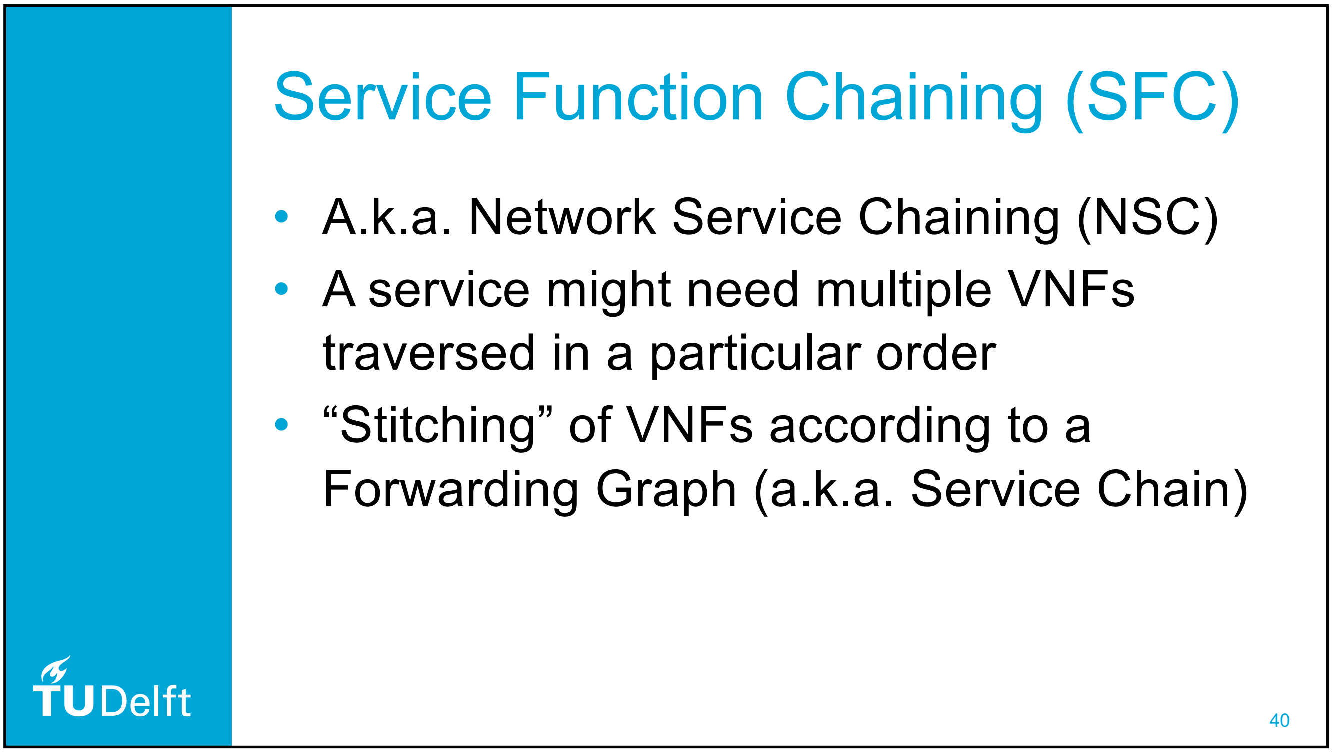
38

38



Virtual Network Function (VNF)

- Multiple VNFs could (like VMs) share the same hardware
- Some features:
  - Portability (move VNFs)
  - Elasticity (scale in/out)
  - Resiliency (backup VNFs)
  - Performance (QoS)



Service Function Chaining (SFC)

- A.k.a. Network Service Chaining (NSC)
- A service might need multiple VNFs traversed in a particular order
- “Stitching” of VNFs according to a Forwarding Graph (a.k.a. Service Chain)