DVAD41 - Introduction to Data Plane Programming

The Data Plane and SDN

What is Data Plane Programming?

- What is the Data Plane?
- What is Software Defined Networking (SDN)?

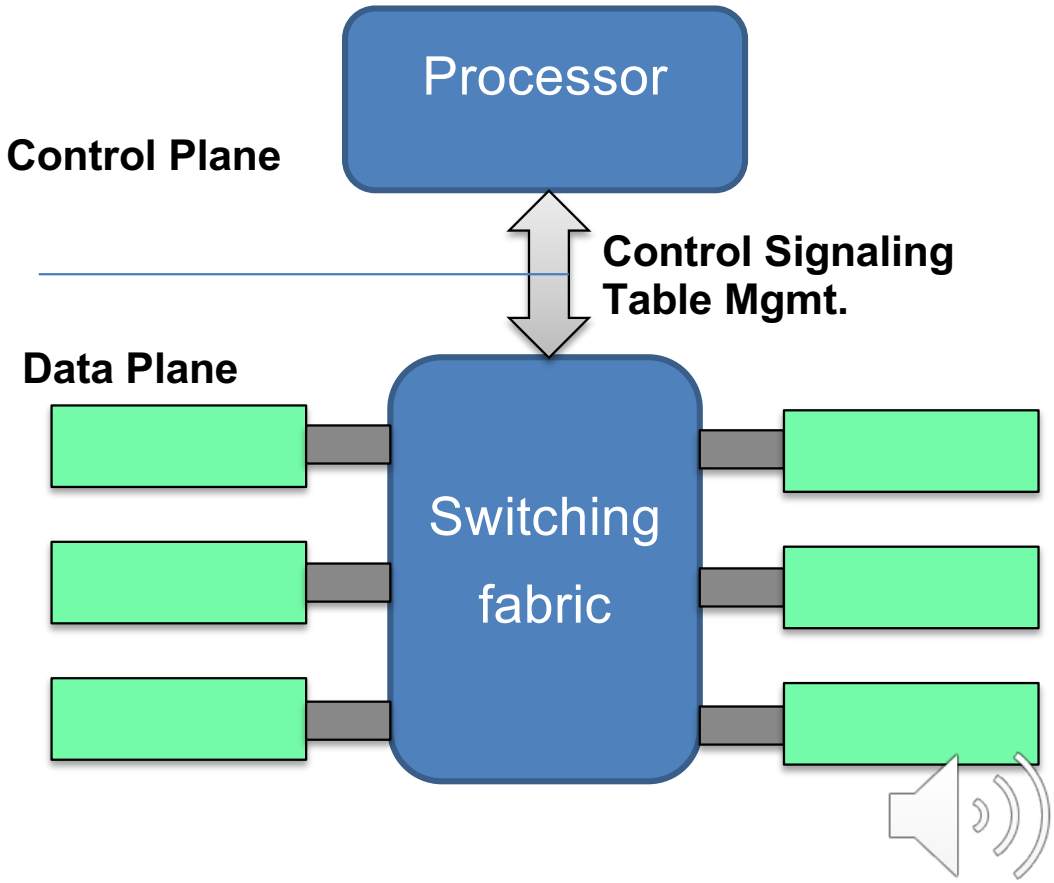
What is the data plane?

- **Processing packet streams**
  - Large volume, packets come in streams, algorithms process them
    - super fast → small time to process single packet
    - matching bitfields, simple actions
    - at end hosts → NIC
    - inside the network → router, switch, firewall

- **Bunch of different functionality**
  - packet forwarding (switch)
  - access control (firewall)
  - tunneling
  - traffic monitoring
  - buffering and marking
  - shaping and scheduling
  - Deep packet inspection (DPI box)

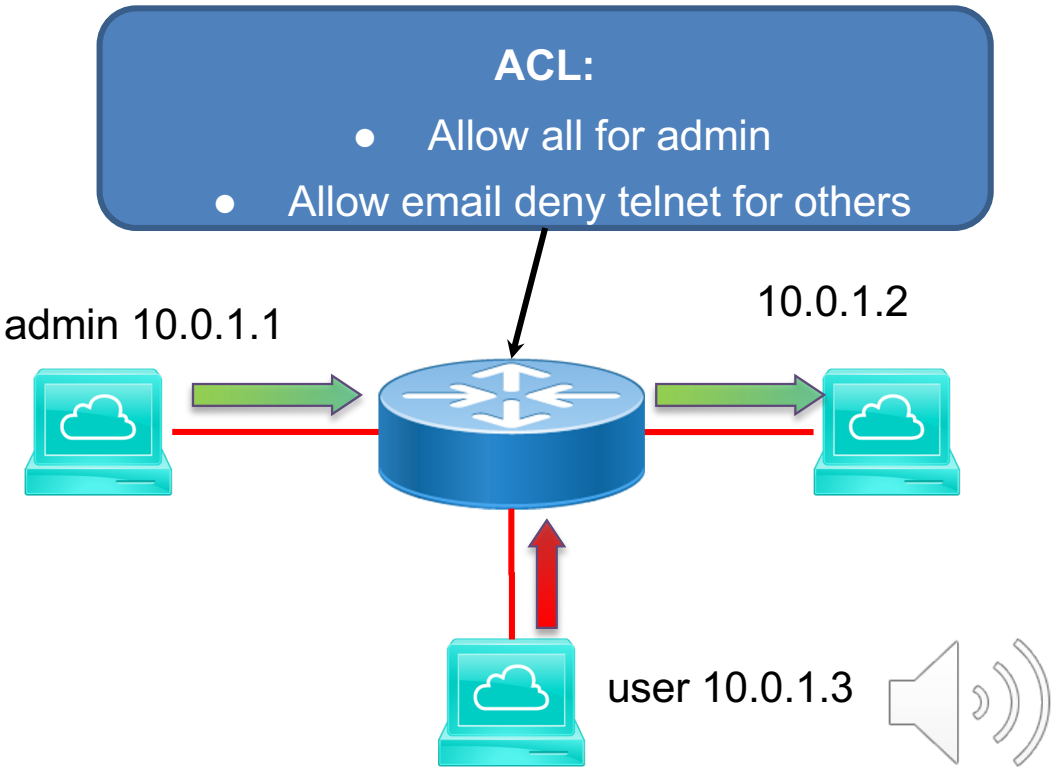
Packet Forwarding

- **Control Plane**
  - calculates the forwarding table
  - determines output port based on destination address
- **Data Plane**
  - manages individual incoming packets
  - matches destination address
    - switch: Dst MAC addr
    - routers: longest IP Prefix
  - lookup the output port
  - action: the packet is sent to output port
  - switching fabric: directs packets from input to output



Access Control

- **Packet Filtering → Access Control Lists (ACL)**
  - Src, Dst IP address
  - Src, Dst ports
  - Protocol ID
- **Stateful operations**
  - also for security, e.g. attacks
  - e.g. block all TCP syn packets from outside
  - requires to parse TCP headers and maintain flow state



Access Control List

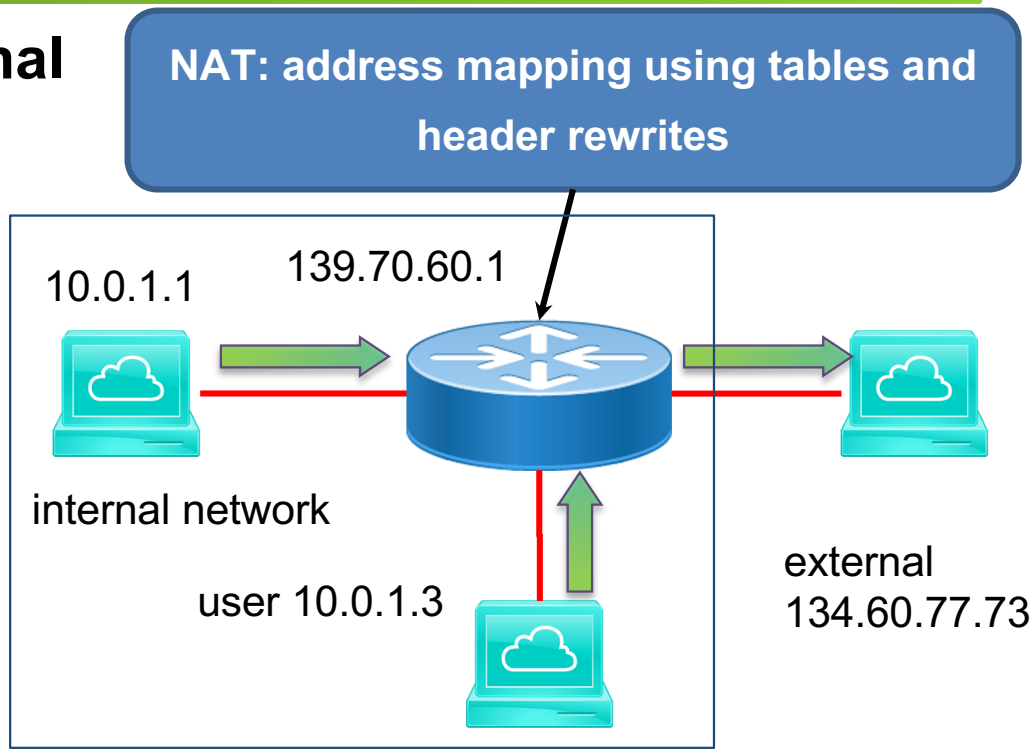
- **Accept/Drop actions**
  - ordered and list
  - Wildcard rules possible
  - list entries can overlap → priority
- **Packet classification**
  - match header fields
  - identify match with highest priority
- **Different approaches**
  - multi-dimensional classification algorithms
  - Use TCAMs: ternary content addressable memory

Src=1.2.3.4, Dest=5.6.7.8	accept
Dest=1.2.3.*	drop
Dest=1.2.3.8, Dport!=53	accept
Src=1.2.3.7, Dport=100	accept
Dport=100	drop

Network Address Translation - NAT

- **Mapping between internal and external addresses**
  - IP-addresses: between end-hosts and NAT
  - ports: each connection needs to be unique

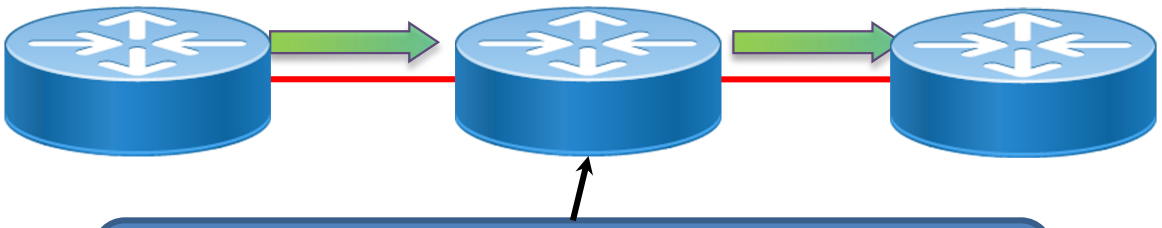
- **NAT Table**
  - entries are dynamically created
  - when to remove entries?
  - what if both ends are behind NAT?



  - Example:
    - Src 10.0.1.3, Sport 1024, Dest 134.6077.73, Dport 80
    - NAT Map to Src 139.70.60.1, Sport 1024, Dest 134.60.77.73, Dport 80
    - Src 10.0.1.1, Sport 1024, Dest 134.6077.73, Dport 80
    - NAT Map to Src 139.70.60.1, Sport 1025, Dest 134.60.77.73, Dport 80

Traffic Monitoring

- **Why Traffic Monitoring?**
  - volume based charging, traffic engineering, anomaly detection, ...
- **How?**
  - matching header fields
  - updating counter of packets/bytes
- **Challenges**
  - identify correct aggregates: proactive vs. reactive
  - more information, e.g. time in queue, congestion states,..
  - some packets of a flow might pass through other nodes, e.g. MPTCP



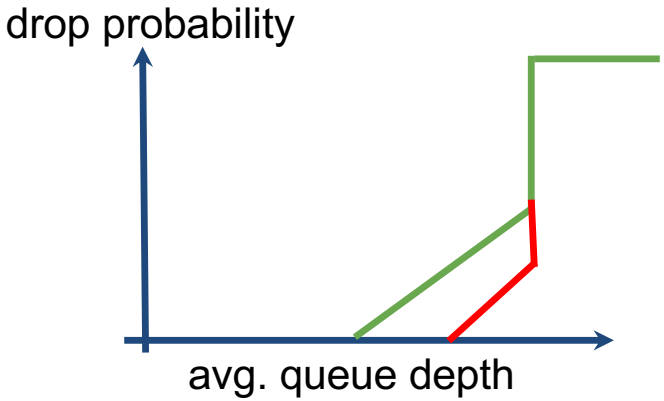
for each input and output port

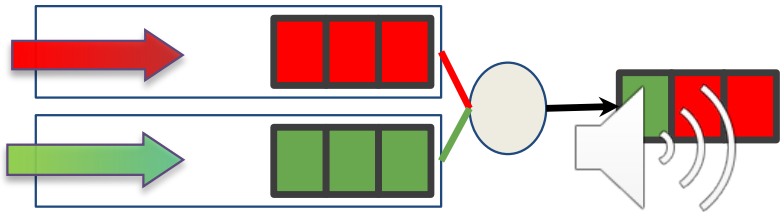
e.g. for each source and destination prefix

Match	packets	bytes
Dest=1.2.3.*	2	3000
Dest=1.2.3.8, Dport=100	10	14000
Dest=1.2.3.7, Dport=80	1000	141200

Buffering and Queue Management

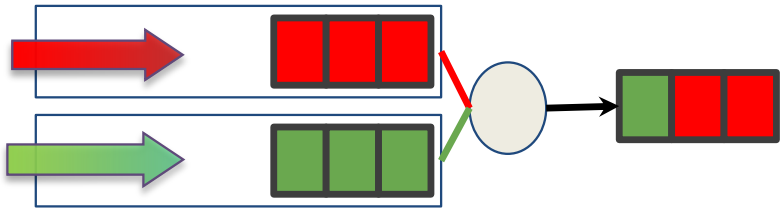
- **First In First Out (FIFO) → Drop Tail**
  - packets served in arriving order
  - if queue is full, arriving packet is dropped
- **Random Early Detection (RED)**
  - drop earlier (function of buffer size)
  - or mark to signal congestion to end hosts
  - different traffic classes can be handled differently
- **Multiple Traffic Classes**
  - separate FIFO queue
    - for each flow or traffic class (e.g. voice, video, web)
    - need scheduler to decide serving order
- **Active Queue Management (AQM)**
  - queue autotunes itself to e.g. latency target
    - CoDel, PIE, FqCoDel,...
    - Packet Value based dropping

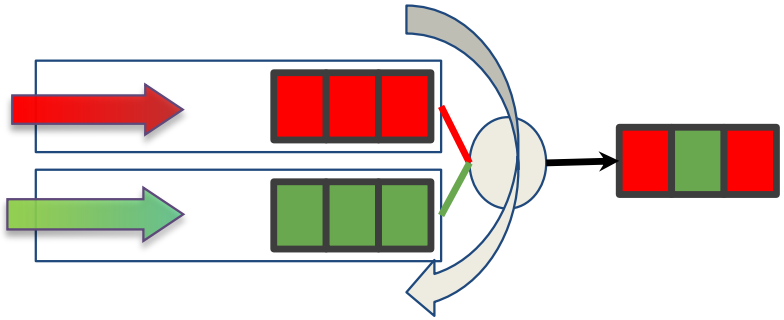




Packet Scheduling

- **Determines the serving order of packets**
  - when there are multiple queues to serve
  - multiple algorithms, different complexity
- **Strict priority**
  - assign to each queue a priority number
  - serve always the queue with highest priority first if it has packets
- **Round Robin**
  - go through queues in round robin way
  - if packet in the queue, serve, otherwise check next one
- **Weighted Fair Scheduling**
  - assign weights to queues
  - serve proportionally many packets

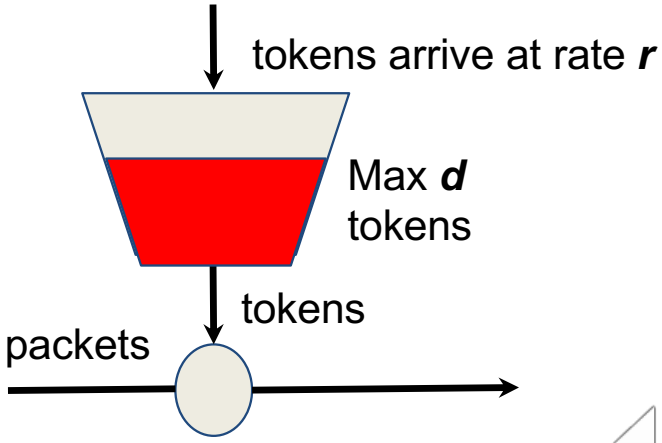




Rate Shaping of Traffic

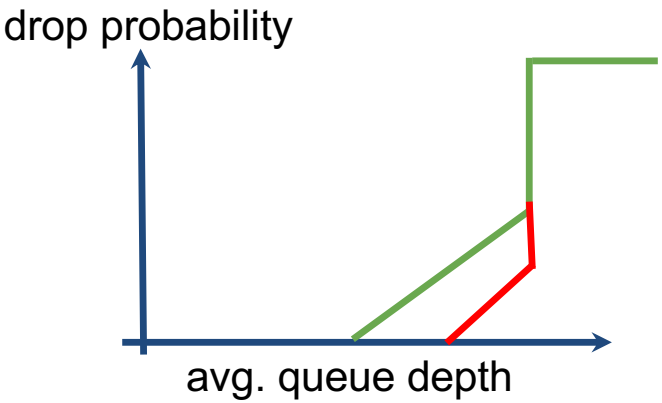
- **Modify packet rate to conform to a profile**
  - e.g. customer contract on max rate
  - e.g. avoiding congestion of downstream nodes

- **Several algorithms**
  - e.g. Leaky bucket
    - control maximum rate and burst duration
    - for each flow or traffic aggregate
    - tokens are created at rate ***r***, bucket with depth ***d***



Packet Marking and Traffic Classification

- **Mark a packet to signal downstream or end-hosts**
  - ECN - Early Congestion notification
    - Reuse some of the IP header fields, e.g. Type of Service (ToS) bits
    - can use bufferstate → RED, CoDel,...
- **How to mark?**
  - End hosts based on applications
    - how can the network trust the endhosts?
  - Network nodes based on traffic classes
    - how can the network infer application requirements?
- **How to identify traffic classes?**
  - using flow specification based on five tuple
  - rate limitations, what conforms to profile, what is out of profile



Data Plane and Software Defined Networking - SDN

- **What is Software Defined Networking?**

Data Plane - Similar Functionality

- **Router**
  - Forwarding on Destination Address
  - Access Control on Five Tuple
  - Packet Scheduling, marking queuing
  - Traffic Monitoring
- **Firewall**
  - Access Control on Five Tuple
  - Stateful vs. Stateless
- **NAT**
  - mapping port numbers and addresses
- **Switch**
  - Forwarding on Destination MAC address
- **Packet Shaper**
  - classify packets and shape traffic

What are the common abstractions?

OpenFlow

- **Match**
  - Header fields
    - Subset
    - IP, MAC, MPLS
  - Ports
  - supported by most TCAM implementations

Prio	Match	Action
1	Src=1.2.3.4, Dest=5.6.7.8	forward (1)
2	Dest=1.2.3.*	drop
3	Dest=1.2.3.8, Dport!=53	forward (2)
4	Src=1.2.3.7, Dport=100	forward (3)
5	Dport=100	forward controller

- **Actions**
  - If match found, perform action on packet
    - drop
    - rewrite header fields
    - forward on port X
    - count packets
    - flood

Software Defined Networking (SDN)

- **Main contributions**
  - OpenFlow = standardized *protocol* to interact with switch
    - download flow table entries, query statistics, etc.
  - OpenFlow = standardized *model*
    - match/action abstraction
  - *Concept* of *logically* centralized control via a single entity (“SDN controller”)
    - Simplifies control plane
- **Issues**
  - Data-plane protocol evolution requires changes to standards (12 → 40 OpenFlow match fields)
  - Limited interoperability between vendors (OpenFlow / netconf / JSON / XML variants)
  - Limited programmability