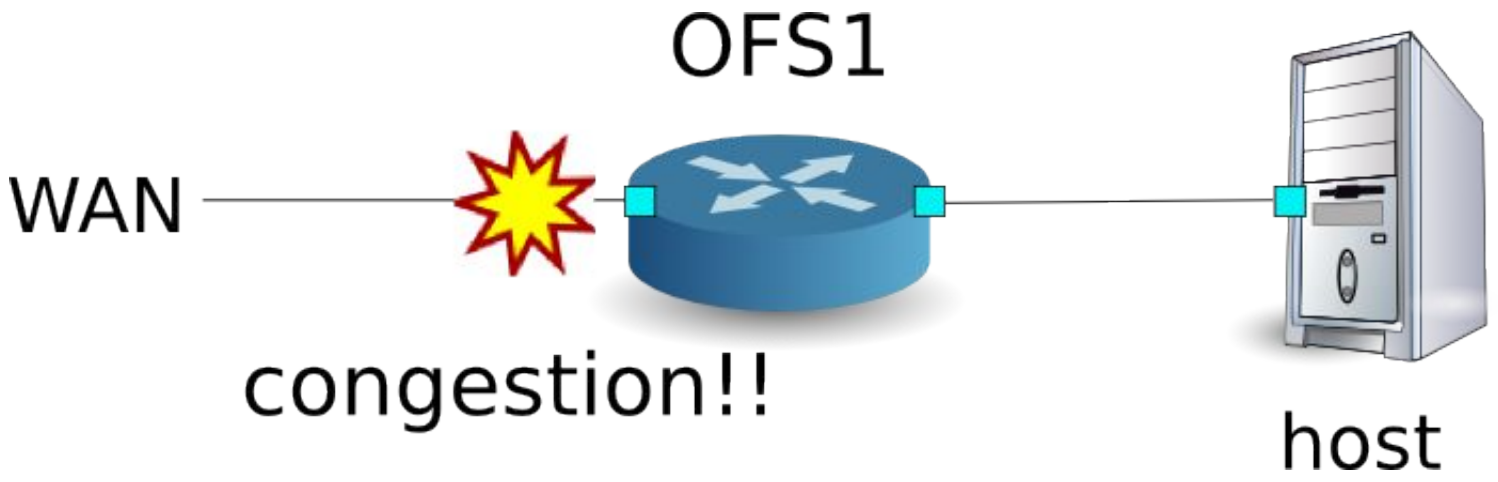
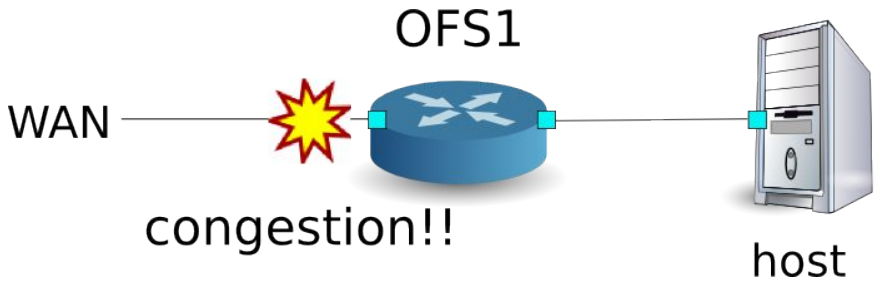
QoS

Exemplo de operação da QoS por fluxo



Criando Topologia

- *sudo mn --mac --switch ovsk --controller remote -x*



Configurando Switch

- É necessário definir a versão do OpenFlow para ser usada em cada switch, para a versão 1.3 também é fundamental definir para ouvir na porta 6632 para acessar OVSDB.
- Em s1:
  - *ovs-vsctl set Bridge s1 protocols=OpenFlow13*
  - *ovs-vsctl set-manager ptcp: 6632*

Modificação no simple_switch_13

- É necessário modificar o código do simple_switch_13 para registrar a entrada de flows na tabela 1.
  - *sed '/OFPFlowMod(/,/)/s/)/, table_id=1)/' ryu/ryu/app/simple_switch_13.py > ryu/ryu/app/qos_simple_switch_13.py*

Start controlador RYU

- No xterm c0:
  - *# ~/ryu/bin/ryu-manager ryu.app.rest_qos ryu.app.qos_simple_switch_13 ryu.app.rest_conf_switch*
    - *[QoS][INFO] dpid=0000000000000001: Join qos switch.*

Configuração da Queue

Queue ID	Taxa máxima	Taxa mínima
0	500Kpbs	-
1	1Mbps	800Kbps

Configurar o acesso do OVSDB

- Em c0.
  - *curl -X PUT -d ‘“tcp:127.0.0.1:6632”’ http://localhost:8080/v1.0/conf/switches/0000000000000001/ovsdb_addr*

  - **Dessa forma é possível configurar à Queue**
    - *curl -X POST -d '{"port_name": "s1-eth1", "type": "linux-htb", "max_rate": "1000000", "queues": [{"max_rate": "500000"}, {"min_rate": "800000"}]}' http://localhost:8080/qos/queue/0000000000000001*

Definindo a Queue


```
[
  {
    "switch_id": "0000000000000001",
    "command_result": {
      "result": "success",
      "details": {
        "0": {
          "config": {
            "max-rate": "500000"
          }
        },
        "1": {
          "config": {
            "min-rate": "800000"
          }
        }
      }
    }
  }
]
```

Inserindo Flow no Switch s1

Prioridade	Endereço Destino	Porta Destino	Protocolo	ID Queue	ID QoS
1	10.0.0.1	5002	UDP	1	1

Inserindo Flow no Switch s1

- Em c0:
  - *curl -X POST -d '{"match": {"nw_dst": "10.0.0.1", "nw_proto": "UDP", "tp_dst": "5002"}, "actions":{"queue": "1"}}'* http://localhost:8080/qos/rules/0000000000000001


```
[
  {
    "switch_id": "0000000000000001",
    "command_result": [
      {
        "result": "success",
        "details": "QoS added. : qos_id=1"
      }
    ]
  }
]
```

Analisando as configurações

- Em c0:
  - *curl -X GET http://localhost:8080/qos/rules/0000000000000001*


```
{
  "switch_id": "0000000000000001",
  "command_result": [
    {
      "qos": [
        {
          "priority": 1,
          "dl_type": "IPv4",
          "nw_proto": "UDP",
          "tp_dst": 5002,
          "qos_id": 1,
          "nw_dst": "10.0.0.1",
          "actions": [
            {
              "queue": "1"
```

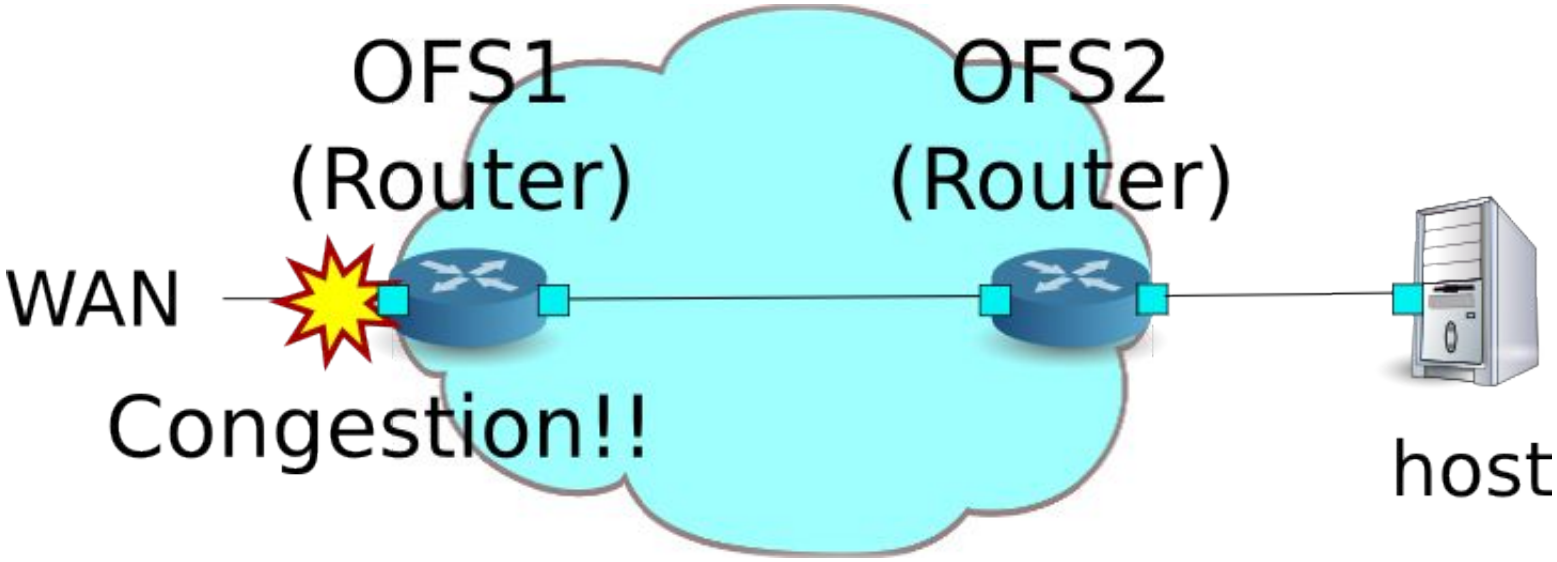
Medindo a largura de banda

- *mininet > xterm h1*
- *mininet > xterm h2*
- *Em h1 (1):*
  - *iperf -s -u -i 1 -p 5001*
- *Em h1 (2):*
  - *iperf -s -u -i 1 -p 5002*
- *Em h2 (1):*
  - *iperf -c 10.0.0.1 -p 5001 -u -b 1M*
- *Em h2 (2):*
  - *iperf -c 10.0.0.1 -p 5001 -u -b 1M*

Conclusão

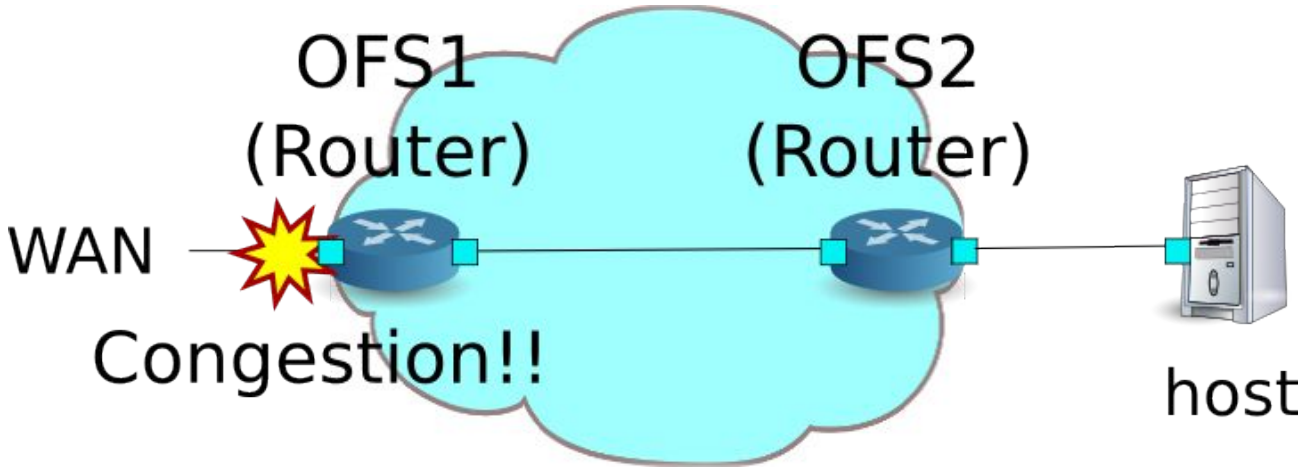
Queue ID	Taxa máxima	Taxa mínima
0	500Kpbs	-
1	1Mbps	800Kbps

Exemplo de QoS usando DiffServ



Criando Topologia

- *sudo mn --topo linear, 2 --mac --switch ovsk --controlador remoto -x*



Configurando Switch s1 e s2

- É necessário definir a versão do OpenFlow para ser usada em cada switch, para a versão 1.3 também é fundamental definir para ouvir na porta 6632 para acessar OVSDB.
- Em s1:
  - *ovs-vsctl set Protocolos Bridge s1 = OpenFlow13*
  - *ovs-vsctl set-manager ptcp: 6632*
  - Em s2:
    - *ovs-vsctl set Protocolos Bridge s2 = OpenFlow13*

Mudando endereços dos Hosts

- Em h1:
  - *ip addr del 10.0.0.1/8 dev h1-eth0*
  - *ip addr add 172.16.20.10/24 dev h1-eth0*
- Em h2:
  - *ip addr del 10.0.0.2/8 dev h2-eth0*
  - *ip addr add 172.16.10.10/24 dev h2-eth0*

Modificação no rest_router

- É necessário modificar o código do rest_router.py para registrar a entrada de flows na tabela 1.
- Em c0:
  - *# sed '/OFPFlowMod(/,/)/s/0, cmd/1, cmd/' ryu/ryu/app/rest_router.py > ryu/ryu/app/qos_rest_router.py*

Iniciando controlador

- Em c0:
  - *ryu-manager ryu.app.rest_qos ryu.app.qos_rest_router ryu.app.rest_conf_switch*
    - *[INFO] switch_id=0000000000000002: Join as router.*
    - *[INFO] dpid=0000000000000001: Join qos switch.*

Configuração da Queue

ID Queue	Taxa máxima	Taxa Mínima	Classe
0	1Mbps	-	Padrão
1	1Mbps	200 Kbps	AF3
2	1Mbps	500 Kbps	AF4

Configurar o acesso do OVSDB

- Em c0.
  - *curl -X PUT -d "'tcp:127.0.0.1:6632'" http://localhost:8080/v1.0/conf/switches/0000000000000001/ovsdb_addr*

Configurando Queue

- **Dessa forma é possível configurar à Queue**
  - *# curl -X POST -d '{"port_name": "s1-eth1", "type": "linux-htb", "max_rate": "1000000", "queues":[{"max_rate": "1000000"}, {"min_rate": "200000"}, {"min_rate": "500000"}]}' http://localhost:8080/qos/queue/0000000000000001*


```
  "switch_id": "0000000000000001",
  "command_result": {
    "result": "success",
    "details": {
      "0": {
        "config": {
          "max-rate": "1000000"
        }
      },
      "1": {
        "config": {
          "min-rate": "200000"
        }
      },
      "2": {
        "config": {
          "min-rate": "500000"
        }
```

Configurando router em s1

- curl -X POST -d '{"address": "172.16.20.1/24"}' http://localhost:8080/router/0000000000000001
- curl -X POST -d '{"address": "172.16.30.10/24"}' http://localhost:8080/router/0000000000000001
- curl -X POST -d '{"gateway": "172.16.30.1"}' http://localhost:8080/router/0000000000000001

Configurando router em s2

- curl -X POST -d '{"address": "172.16.10.1/24"}' http://localhost:8080/router/0000000000000002
- curl -X POST -d '{"address": "172.16.30.1/24"}' http://localhost:8080/router/0000000000000002
- curl -X POST -d '{"gateway": "172.16.30.10"}' http://localhost:8080/router/0000000000000002

Configuração gatway padrão para h1 e h2

- Em h1:
  - *# ip route add default via 172.16.20.1*
- Em h2:
  - *# ip route add default via 172.16.10.1*

Definindo flows com valor DSCP em s1

Prioridade	DSCP	ID da Queue	ID QoS
1	26 (AF31)	1	1
1	34 (AF41)	2	2

Definindo flows com valor DSCP em s1

- Em c0:
  - *curl -X POST -d '{"match": {"ip_dscp": "26"}, "actions": {"queue": "1"}}' 'http: // localhost: 8080 / qos / rules / 0000000000000001*
  - *curl -X POST -d ' {"match": {"ip_dscp": "34"}, "actions": {"queue": "2"}} " http: // localhost:8080 / qos / rules / 0000000000000001*

Definindo flows com valor DSCP em s2

Prioridade	Endereço Destino	Porta Destino	Protocolo	DSCP	ID QoS
1	172.16.20.10	5002	UDP	26 (AF31)	1
1	172.16.20.10	5003	UDP	34 (AF41)	2

Definindo flows com valor DSCP em s2

- Em c0:
  - *# curl -X POST -d '{"match": {"nw_dst": "172.16.20.10", "nw_proto": "UDP", "tp_dst": "5002"}, "actions":{"mark": "26"}}' http://localhost:8080/qos/rules/0000000000000002*
  - *# curl -X POST -d '{"match": {"nw_dst": "172.16.20.10", "nw_proto": "UDP", "tp_dst": "5003"}, "actions":{"mark": "34"}}' http://localhost:8080/qos/rules/0000000000000002*

Verificando a Configuração

- Em c0:
  - # curl -X GET http://localhost:8080/qos/rules/0000000000000001


```
"switch_id": "0000000000000001",
"command_result": [
  {
    "qos": [
      {
        "priority": 1,
        "dl_type": "IPv4",
        "ip_dscp": 34,
        "actions": [
          {
            "queue": "2"
          }
        ],
        "qos_id": 2
      },
      {
        "priority": 1,
        "dl_type": "IPv4",
        "ip_dscp": 26,
        "actions": [
          {
            "queue": "1"
          }
        ],
        "qos_id": 1
```

Verificando a Configuração

- Em c0:
  - # curl -X GET http://localhost:8080/qos/rules/0000000000000002


```
"switch_id": "0000000000000002",
"command_result": [
  {
    "qos": [
      {
        "priority": 1,
        "dl_type": "IPv4",
        "nw_proto": "UDP",
        "tp_dst": 5002,
        "qos_id": 1,
        "nw_dst": "172.16.20.10",
        "actions": [
          {
            "mark": "26"
          }
        ]
      },
      {
        "priority": 1,
        "dl_type": "IPv4",
        "nw_proto": "UDP",
        "tp_dst": 5003,
        "qos_id": 2,
        "nw_dst": "172.16.20.10",
        "actions": [
          {
            "mark": "34"
```

Medindo a largura de banda

- mininet > xterm h2
- mininet > xterm h2
- Em h1 (1):
  - *# iperf -s -u -p 5002 & # iperf -s -u -p 5003 & # iperf -s -u -i 1 5001*
- *Em h2 (1):*
  - *# iperf -c 172.16.20.10 -p 5001 -u -b 1M*
- *Em h2 (2):*
  - *# iperf -c 172.16.20.10 -p 5002 -u -b 300K*
- *Em h2 (3):*
  - *# iperf -c 172.16.20.10 -p 5003 -u -b 600K*