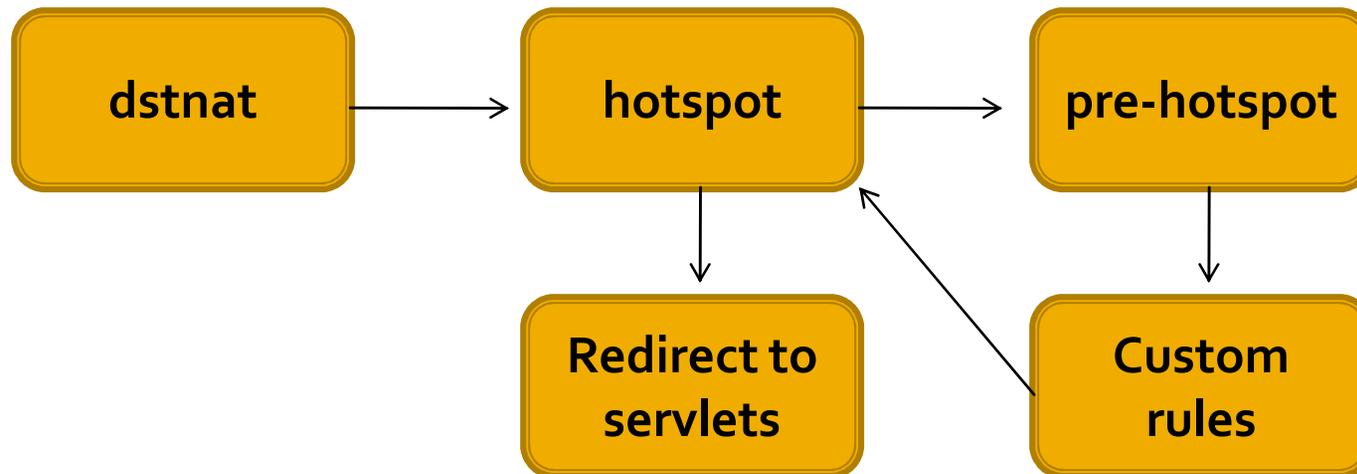


# Customizing Hotspot Deployments

# NAT hooks

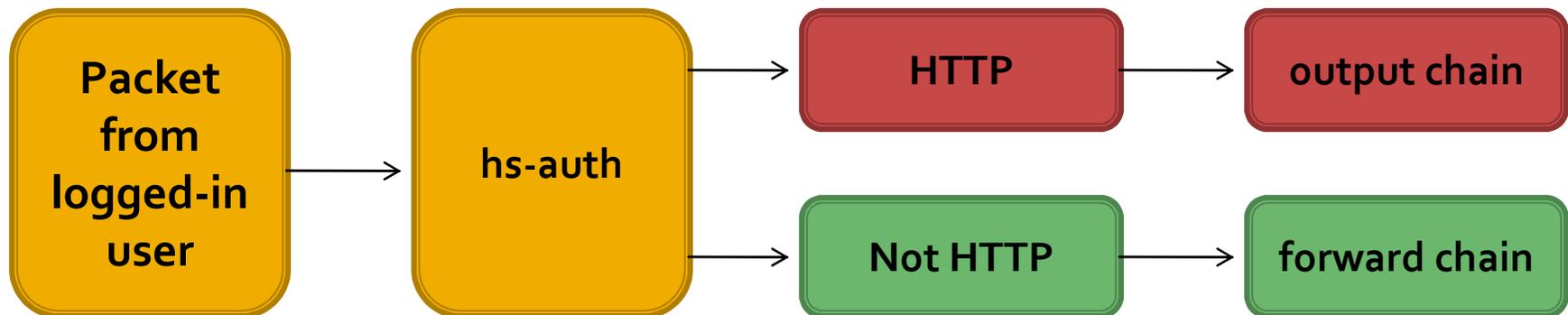
```
0 D chain=dstnat action=jump  
  jump-target=hotspot hotspot=from-client  
1 I chain=hotspot action=jump  
  jump-target=pre-hotspot
```



- Use pre-hotspot chain to prevent Hotspot from redirecting traffic to servlets

# Authenticated traffic

```
7 D chain=hotspot action=jump jump-target=  
  hs-auth protocol=tcp hotspot=auth  
13 D chain=hs-auth action=redirect  
  to-ports=64874 protocol=tcp hotspot=http
```



- Even authenticated users have HTTP redirected through servlet

# What is affected?

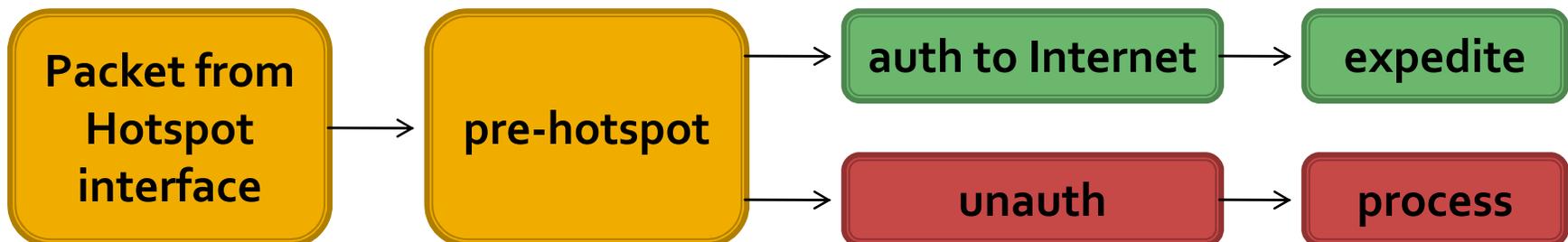
- PCC + Hotspot
- Performance
  - Authenticated traffic traverses 8 rules
  - Servlet proxying is slower than just routing packets
- Vast majority of traffic seen on Hotspots is HTTP and DNS

# PCC + Hotspot

- Textbook examples expect traffic to flow through router, not from it
- Usually router generated traffic is very specific and shouldn't be balanced
- On routers with normal and Hotspot networks the ruleset would double in size

# Short circuit authenticated traffic

```
/ip firewall nat
add chain=pre-hotspot action=accept
    dst-address-type=!local hotspot=auth
```



- Authenticated traffic now only takes three rules to process
- Authenticated traffic is always in forward chain when traversing the router

# Filter hooks

```
2 D chain=input action=jump  
   jump-target=hs-input  
   hotspot=from-client
```

```
3 I chain=hs-input action=jump  
   jump-target=pre-hs-input
```

- Only input chain can be customized manually
- Forward chain is customized via dynamic entries inserted via Walled Garden IP rules for unauthenticated traffic

# Servlet load



- Lots of applications use HTTP but are not prepared to handle Hotspots
- We see an average of 14 redirects to the login page before the user interacts with it
- Malware can spawn HTTP requests at a very high rate
- Servlet operation is rather expensive as it listens to each request and issues a response

# Protecting the HTTP servlets

```
/ip firewall filter
add chain=pre-hs-input action=drop
    connection-limit=5,32 protocol=tcp
    dst-port=64872-64875
add chain=pre-hs-input action=drop
    connection-limit=100,24 protocol=tcp
    dst-port=64872-64875
```

- Implement limits for hosts and networks
- More hosts mean more legitimate requests
- Can block legitimate clients – only use when necessary

# Rate limits: simple queues

Queue for client 1



Queue for client 2



Queue for client n



Queue on interface, shared by all bypassed and unauthenticated users

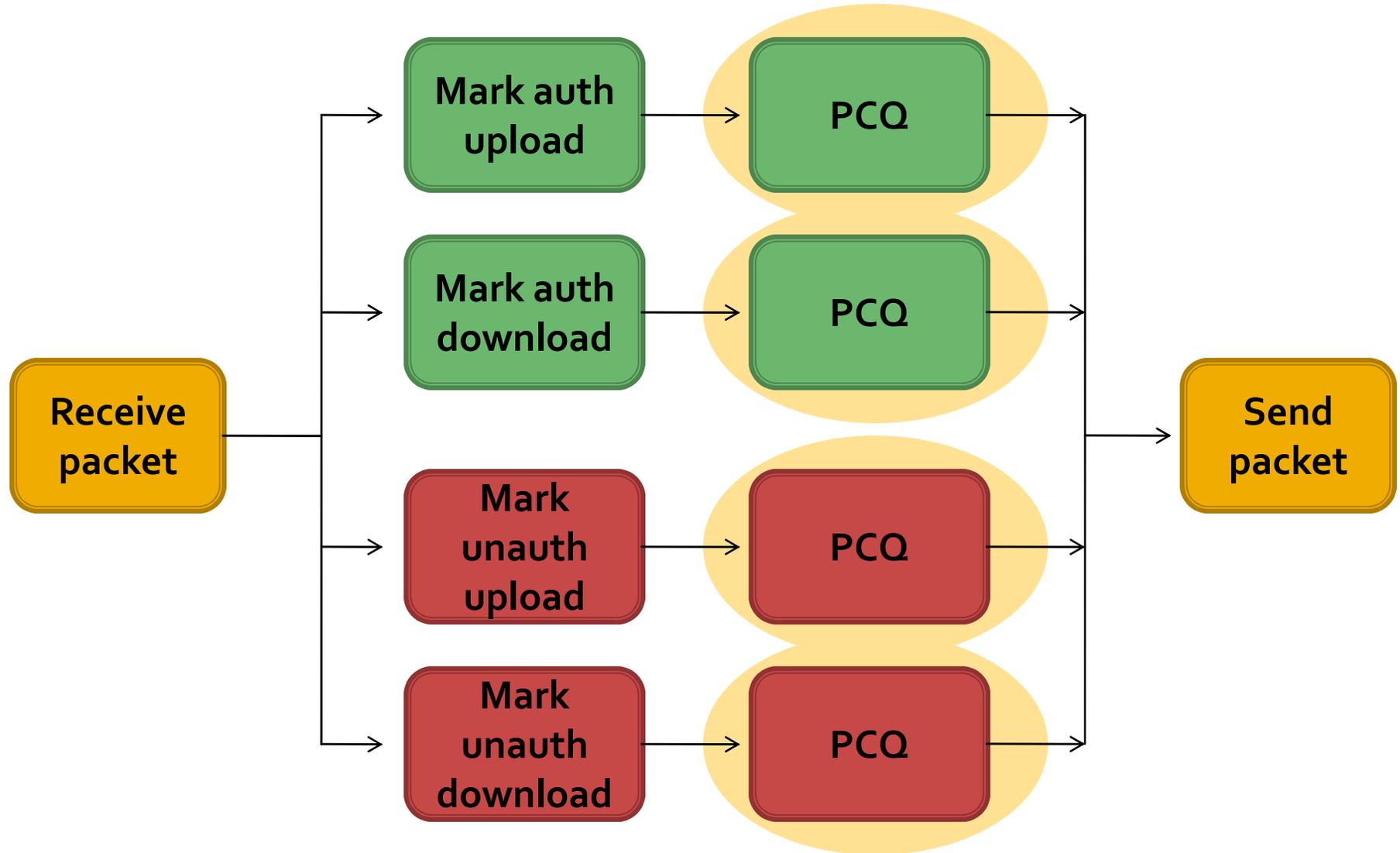
# Simple queue problems

- Simple queues don't scale well
- Bypassed users:
  - Shortcut to troubleshooting users
  - Share bandwidth pool with unauth
  - Require manually shifted simple queues and static DHCP leases
- It's not possible to rate limit the Hotspot network as a whole

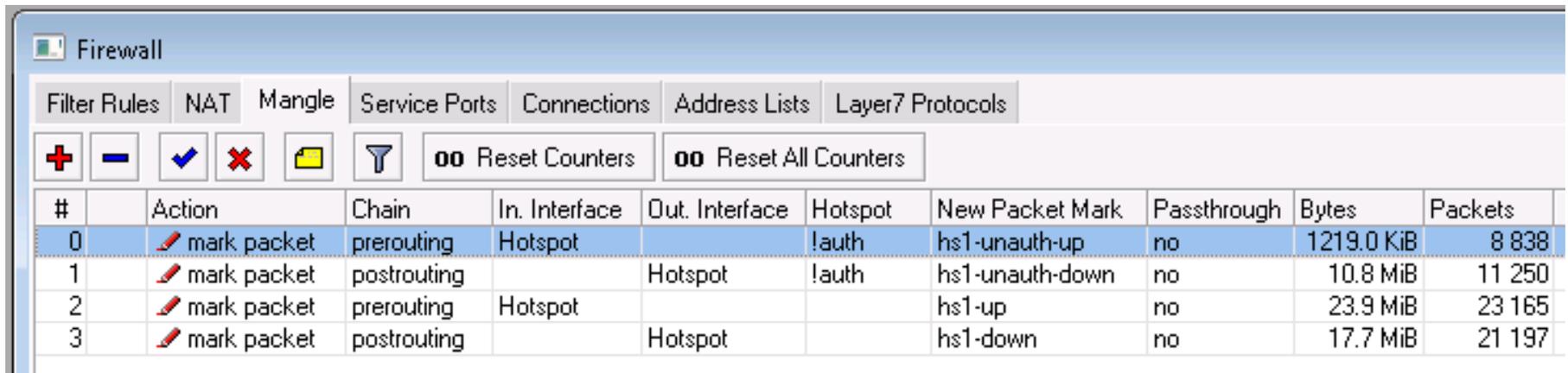
# Solution: PCQ

- Scales well as sub-queues get picked fast, is processed in parallel
- Unauthenticated users can be identified and rate limited per user
- Leaf can have max-limit aggregated over sub-queues

# Scenario 1: one profile/network



# Scenario 1: marking traffic



The screenshot shows the Mikrotik WinBox Firewall Filter Rules configuration window. The 'Filter Rules' tab is active. The table below lists the configured rules:

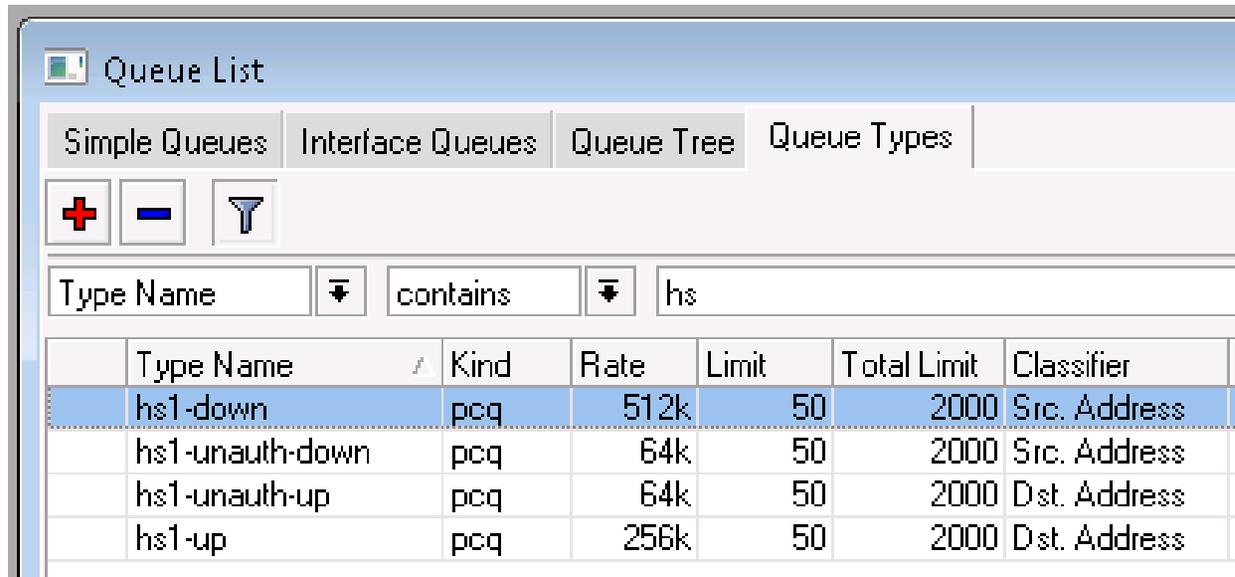
#	Action	Chain	In. Interface	Out. Interface	Hotspot	New Packet Mark	Passthrough	Bytes	Packets
0	mark packet	prerouting	Hotspot		!auth	hs1-unauth-up	no	1219.0 KiB	8 838
1	mark packet	postrouting		Hotspot	!auth	hs1-unauth-down	no	10.8 MiB	11 250
2	mark packet	prerouting	Hotspot			hs1-up	no	23.9 MiB	23 165
3	mark packet	postrouting		Hotspot		hs1-down	no	17.7 MiB	21 197

- Prerouting for upload, postrouting for download
- Unauthenticated traffic first, then fall through to authenticated

# Scenario 1: mangle export

```
/ip firewall mangle
add action=mark-packet chain=prerouting
    hotspot=!auth in-interface=Hotspot new-
    packet-mark=hs1-unauth-up passthrough=no
add action=mark-packet chain=postrouting
    hotspot=!auth new-packet-mark=hs1-unauth-down
    out-interface=Hotspot passthrough=no
add action=mark-packet chain=prerouting in-
    interface=Hotspot new-packet-mark=hs1-up
    passthrough=no
add action=mark-packet chain=postrouting new-
    packet-mark=hs1-down out-interface=Hotspot
    passthrough=no
```

# Scenario 1: queue types

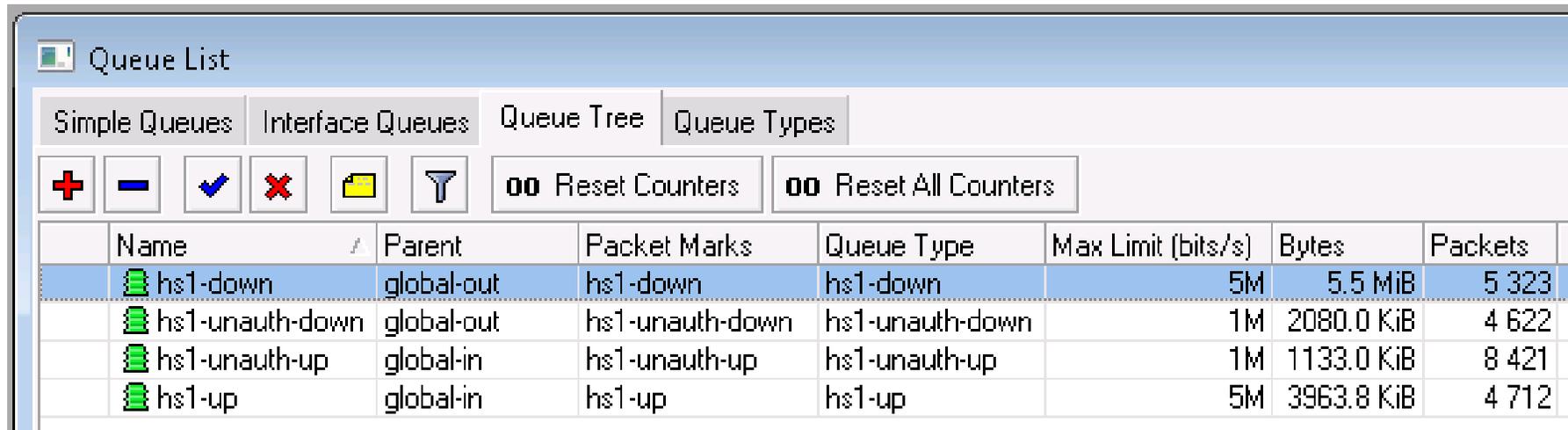


The screenshot shows a window titled "Queue List" with four tabs: "Simple Queues", "Interface Queues", "Queue Tree", and "Queue Types". The "Queue Types" tab is selected. Below the tabs are three icons: a red plus sign, a blue minus sign, and a funnel. A search bar contains the text "Type Name" followed by a dropdown arrow, the word "contains", another dropdown arrow, and the text "hs". Below the search bar is a table with the following data:

Type Name	Kind	Rate	Limit	Total Limit	Classifier
hs1-down	pcq	512k	50	2000	Src. Address
hs1-unauth-down	pcq	64k	50	2000	Src. Address
hs1-unauth-up	pcq	64k	50	2000	Dst. Address
hs1-up	pcq	256k	50	2000	Dst. Address

- Adjust limit and total-limit as required for number of users, make sure not to exceed available memory

# Scenario 1: queue tree



The screenshot shows a 'Queue List' window with four tabs: 'Simple Queues', 'Interface Queues', 'Queue Tree', and 'Queue Types'. The 'Queue Tree' tab is active. Below the tabs is a toolbar with icons for adding, deleting, and filtering, along with buttons for 'Reset Counters' and 'Reset All Counters'. The main area contains a table with the following data:

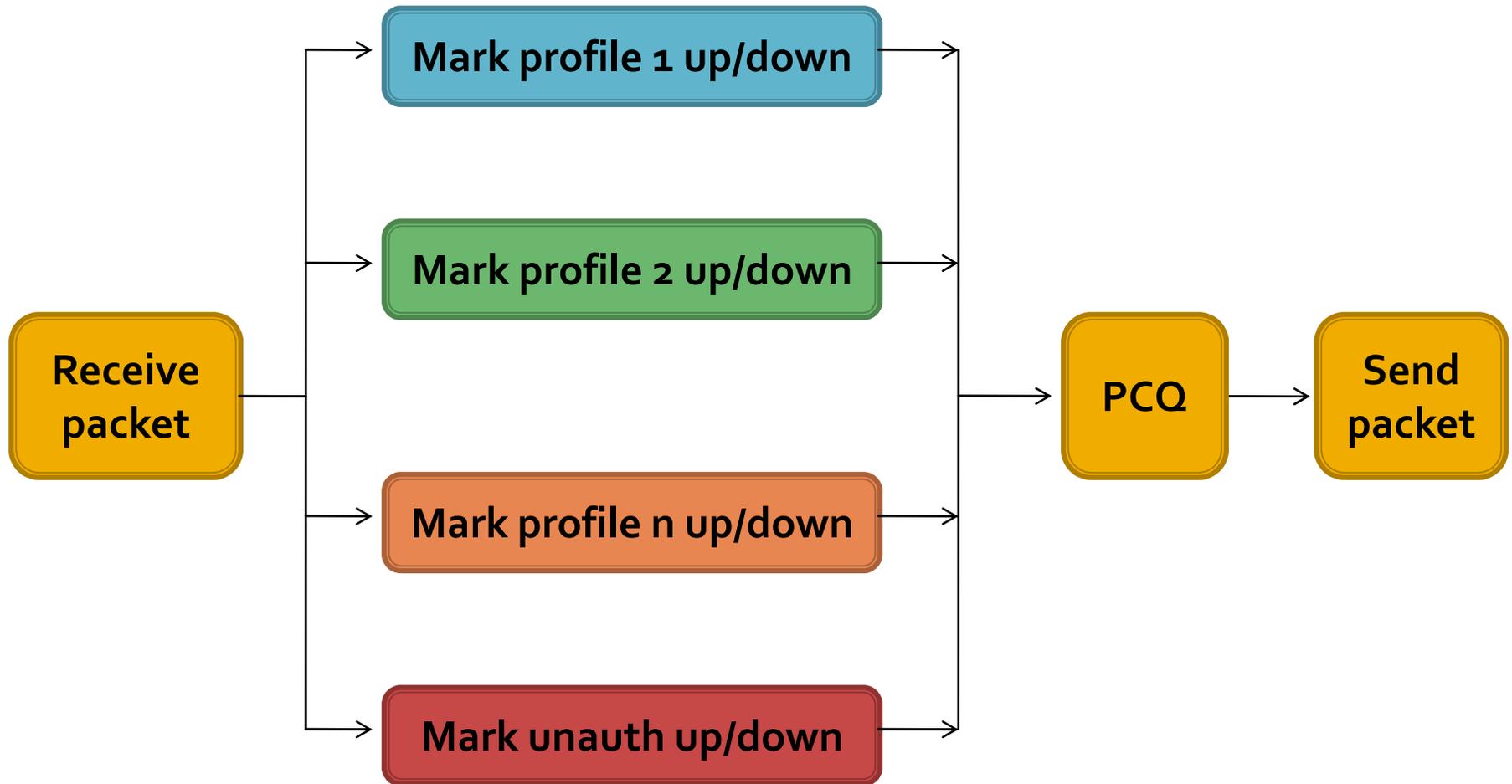
Name	Parent	Packet Marks	Queue Type	Max Limit (bits/s)	Bytes	Packets
hs1-down	global-out	hs1-down	hs1-down	5M	5.5 MiB	5 323
hs1-unauth-down	global-out	hs1-unauth-down	hs1-unauth-down	1M	2080.0 KiB	4 622
hs1-unauth-up	global-in	hs1-unauth-up	hs1-unauth-up	1M	1133.0 KiB	8 421
hs1-up	global-in	hs1-up	hs1-up	5M	3963.8 KiB	4 712

- Download goes in global-out, upload goes in global-in
- max-limit sets total network bandwidth
- Prioritizing is possible

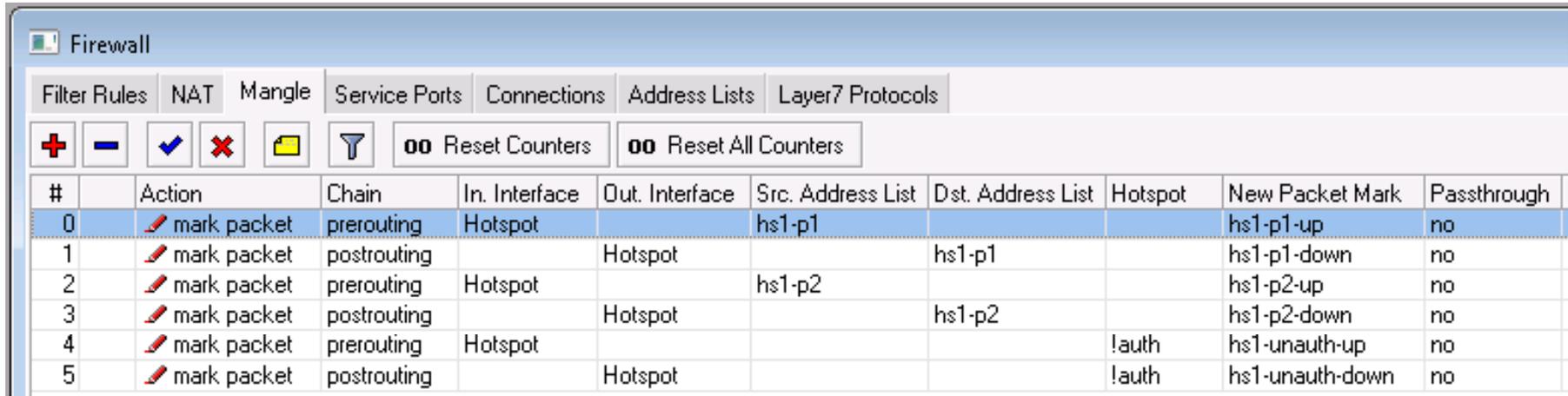
# Scenario 1: queue export

```
/queue type
add kind=pcq name=hs1-unauth-up pcq-classifier=dst-address
    pcq-limit=50 pcq-rate=64000 pcq-total-limit=2000
add kind=pcq name=hs1-unauth-down pcq-classifier=src-address
    pcq-limit=50 pcq-rate=64000 pcq-total-limit=2000
add kind=pcq name=hs1-up pcq-classifier=dst-address pcq-limit=50
    pcq-rate=256000 pcq-total-limit=2000
add kind=pcq name=hs1-down pcq-classifier=src-address
    pcq-limit=50 pcq-rate=512000 pcq-total-limit=2000
/queue tree
add max-limit=1M name=hs1-unauth-up packet-mark=hs1-unauth-up
    parent=global-in queue=hs1-unauth-up
add max-limit=1M name=hs1-unauth-down packet-mark=hs1-unauth-down
    parent=global-out queue=hs1-unauth-down
add max-limit=5M name=hs1-down packet-mark=hs1-down
    parent=global-out queue=hs1-down
add max-limit=5M name=hs1-up packet-mark=hs1-up parent=global-in
    queue=hs1-up
```

# Scenario 2: many profiles



# Scenario 2: marking traffic

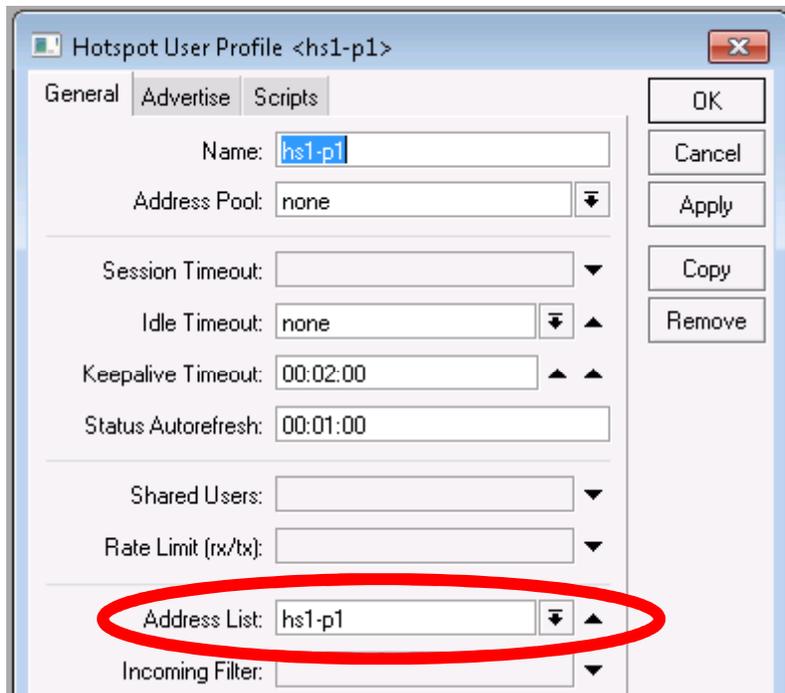


The screenshot shows the Mikrotik WinBox Firewall configuration window. The 'Filter Rules' tab is active. The table below lists the configured rules:

#	Action	Chain	In. Interface	Out. Interface	Src. Address List	Dst. Address List	Hotspot	New Packet Mark	Passthrough
0	mark packet	prerouting	Hotspot		hs1-p1			hs1-p1-up	no
1	mark packet	postrouting		Hotspot		hs1-p1		hs1-p1-down	no
2	mark packet	prerouting	Hotspot		hs1-p2			hs1-p2-up	no
3	mark packet	postrouting		Hotspot		hs1-p2		hs1-p2-down	no
4	mark packet	prerouting	Hotspot				!auth	hs1-unauth-up	no
5	mark packet	postrouting		Hotspot			!auth	hs1-unauth-down	no

- Determine user profile based on firewall address lists
- One rule for upload/download each
- Traffic not on any of the address lists checked falls through to simple queues

# Scenario 2: populating lists



- Address lists are set via User Profiles. AAA can inherit via Mikrotik-Group attribute

- AAA can also set address list directly via Mikrotik-Address-List (vendor 14988, id 19, type string)

# Adjusting PCQ limits

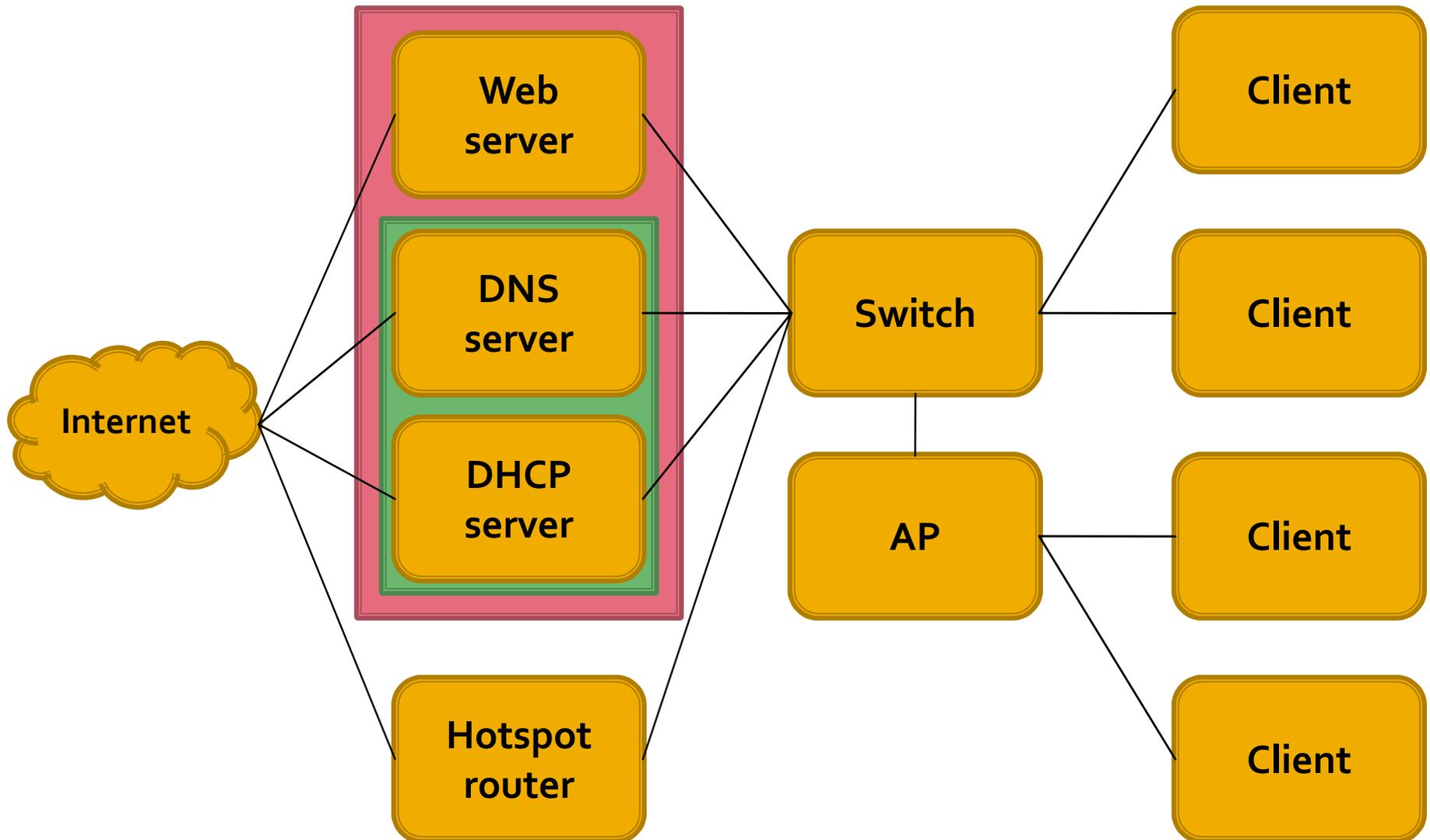
```
/queue type {  
    :local total([:len [/ip hot act find server=hs1]] * 50);  
    :local total ([:len [/ip fire addr find name=hs1-p1]] * 50);  
    set [find name=hs1-down] pcq-total-limit=$total;  
    set [find name=hs1-up] pcq-total-limit=$total;  
};
```

- Defaults can only serve 40 users
- Scale packets/user down to save RAM
- Should probably not be run scheduled to prevent RAM exhaustion if you don't have much memory

# Scaling to thousands of users

- Turn off all unnecessary services
- Offload the necessary services:
  - DHCP
  - DNS
  - User Authentication
- Minimize what can't be offloaded:
  - Servlets / login pages
- Don't skimp on hardware

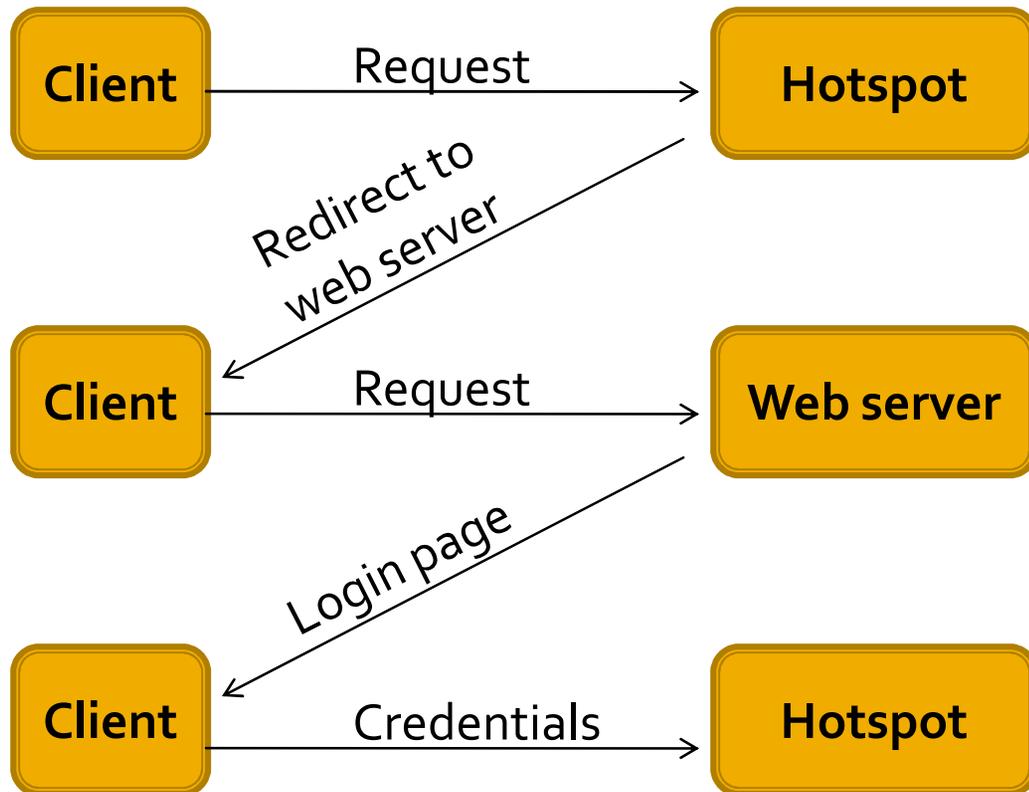
# Offloading network layout



# DHCP and DNS

- Use long DHCP leases to minimize traffic, RFC1918 space is free
- Can be offloaded to a second RouterOS device with interfaces on the network
- By default a Hotspot servlet intercepts DNS, using a parallel DNS server could affect functionality (IP to Hotspot DNS name mapping must be perfect)

# Web server



- Login page can be comprised of many large elements
- All variables the servlet can set can be passed on via GET

# Hotspot redirect to external

```
<html><head>
<meta http-equiv="refresh" content="0;
  url=https://login.example.org/?mac=$(mac)">
<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="expires" content="-1">
</head></html>
```

- Whitelist in walled garden IP for non-local servers
- meta refreshes are implemented by more clients than JavaScript

# Switch and AP hardware

- Use switches that know spanning tree to prevent loops
- Isolate clients on the edge to reduce broadcast related traffic
- In dense coverage areas add APs and lower TX power rather than increase it
- Offer 5GHz SSIDs