Netfilter e o Subsistema de Rede do Linux

Juliano F. Ravasi

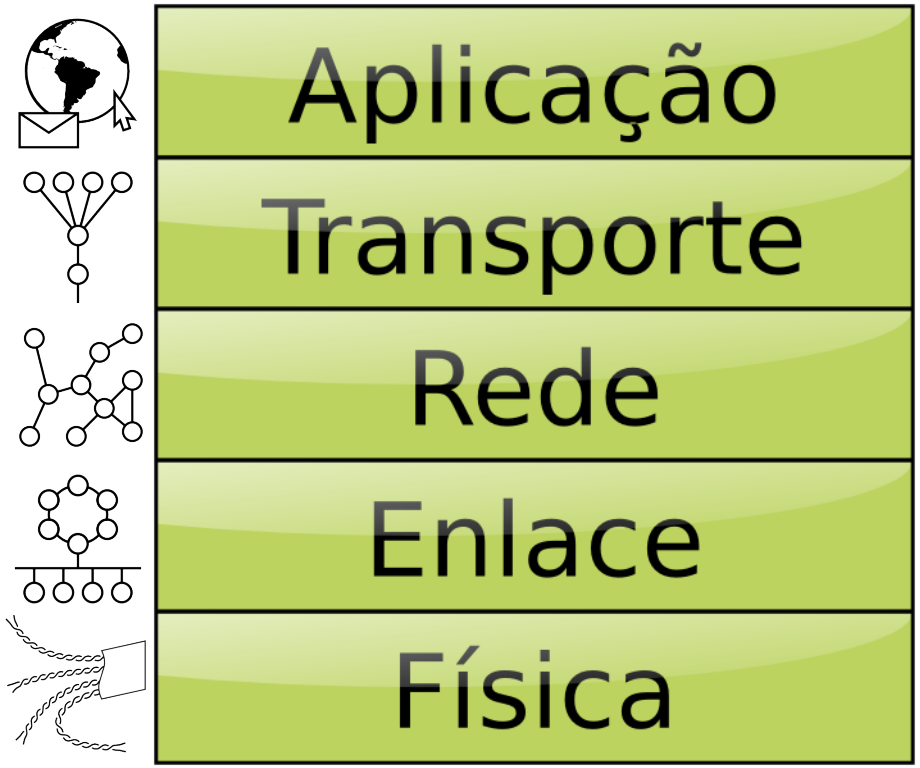
Novembro 2010
http://juliano.info/

Introdução a
Redes de Computadores

Redes de Computadores

- Internet → Rede baseada em datagramas
  - Transmissão de pequenas unidades de informação (chamadas de “pacotes”).
  - Cada pacote é “roteado” individualmente.
  - Conexões, ou sessões:
    - Conjunto de pacotes relacionados que definem uma única transferência de dados, como por exemplo, a transferência de um arquivo.
    - Ex: Conexão TCP.

Redes de Computadores



Modelo de referência geralmente utilizado com o conjunto de protocolos da Internet.
É uma simplificação do modelo OSI, adaptado ao modelo TCP/IP.

O Sistema Operacional Linux

O Sistema Operacional Linux

- Projeto GNU + *kernel* Linux
- O projeto GNU:
  - Criado em 1983 por Richard Stallman
  - Objetivo: um sistema operacional livre
- O *kernel* Linux:
  - Criado em 1991 por Linus Torvalds
  - Objetivo: suprir a falta de um bom *kernel* livre

O Sistema Operacional Linux

- Código-fonte aberto, é um software livre
- Segue a filosofia de SOs Unix (*Unix-like*)
- Versátil:
  - Dispositivos embarcados
    - Android, WebOS, MeeGo, WRT, etc.
  - Computação pessoal
    - Ubuntu, Fedora, openSUSE, etc.
  - Servidores
    - SLES, RHEL, CentOS, etc.
  - Supercomputadores
    - Os dez supercomputadores mais rápidos do mundo.

O Sistema Operacional Linux

- É o SO mais utilizado em servidores
- Possui um vasto ecossistema de aplicações:
  - Apache HTTP Server, nginx, lighttpd, etc.
  - MySQL, PostgreSQL, Firebird, etc.
  - Perl, PHP, Python, Ruby, OpenJDK, etc.
  - Sendmail, Postfix, Exim, etc.
  - vsftpd, ProFTPd, Pure-FTPd, etc.
  - Xen, KVM, VirtualBox, etc.
  - OpenSSH, OpenVNC, X.org,
    BIND, INN, NTP, SAMBA, IRCD, etc...
    (e tudo isso é software de código-fonte aberto)

O Subsistema de Rede do Linux

O Subsistema de Rede do Linux

- Versátil:
  - Carregamento dinâmico de módulos do *kernel* para diversas finalidades (*netfilter* é um deles).
  - Componentes em espaço de usuário (interfaces tun/tap, fileiras de pacotes, roteamento, etc.).
- Poderoso:
  - Possui componentes para filtragem, roteamento, comutação, criptografia, tunelamento, etc.

Múltiplas Finalidades

- *Firewall*:
  - Filtro de pacotes L1-L4
  - Filtro de pacotes L7 (camada de aplicação)
  - Rastreamento de conexões (*stateful firewall*)
  - Memória de tráfego recente
  - Registro de tráfego (*logging*)
  - NAT e NAPT (tradução de endereços)
  - Redireção de tráfego (*proxy* transparente)

Múltiplas Finalidades

- Roteador:
  - Múltiplas regras e tabelas de rotas
  - Roteamento *multicast*
  - Roteamento utilizando *Proxy ARP*
  - Suporte completo a IPv4 e IPv6
  - Suporte a IPSec e tunelamento
  - Balanceamento e equalização de tráfego

Múltiplas Finalidades

- Comutador (*switch*, *bridge*):
  - Múltiplas interfaces de rede podem ser unidas e transformadas em um comutador de camada de enlace (*bridge*).
  - Permite monitoramento passivo
  - Permite filtragem de pacotes/quadros
  - Suporte ao protocolo *spanning-tree*

Múltiplas Finalidades

- Controle de tráfego (*shaper*):
  - Regula e limita o tráfego de dados
  - Escalona o uso da largura de banda disponível
  - Oferece priorização e garantia de tráfego (*QoS*)

Ferramentas

- Configuração:
  - ip, tc, arp, brctl, iptables, ebtables, ethtool
- Testes:
  - ping, traceroute, nmap
- Monitoramento:
  - netstat, conntrack, tcpdump, wireshark
- *Daemons*:
  - quagga, ulogd
- Utilitários:
  - ipset

Netfilter

Histórico

- Linux 2.0: *ipfwadm*
  - Baseado no *ipfw* do BSD.
  - Quatro tabelas:
    - *Input* (entrada)
    - *Output* (saída)
    - *Forward* (encaminhamento)
    - *Accounting* (contabilidade)
  - Cada tabela contém uma lista de regras que são aplicadas a cada pacote que entra, sai ou passa pelo sistema.
  - Cada regra testa os endereços e portas do pacote e determina se este é aceito ou rejeitado.

Histórico

- Linux 2.2: *ipchains*
  - Reimplementação do *ipfwadm*
  - Conceito de “correntes” (*chains*) de regras
    - Cada corrente contém uma lista de regras, cada pacote é testado contra cada regra da corrente.
    - O usuário pode criar novas correntes.
    - Correntes podem ser encadeadas, criando dependências hierárquicas de filtragem de pacotes.
  - Correntes padrões:
    - INPUT (entrada)
    - OUTPUT (saída)
    - FORWARD (encaminhamento)

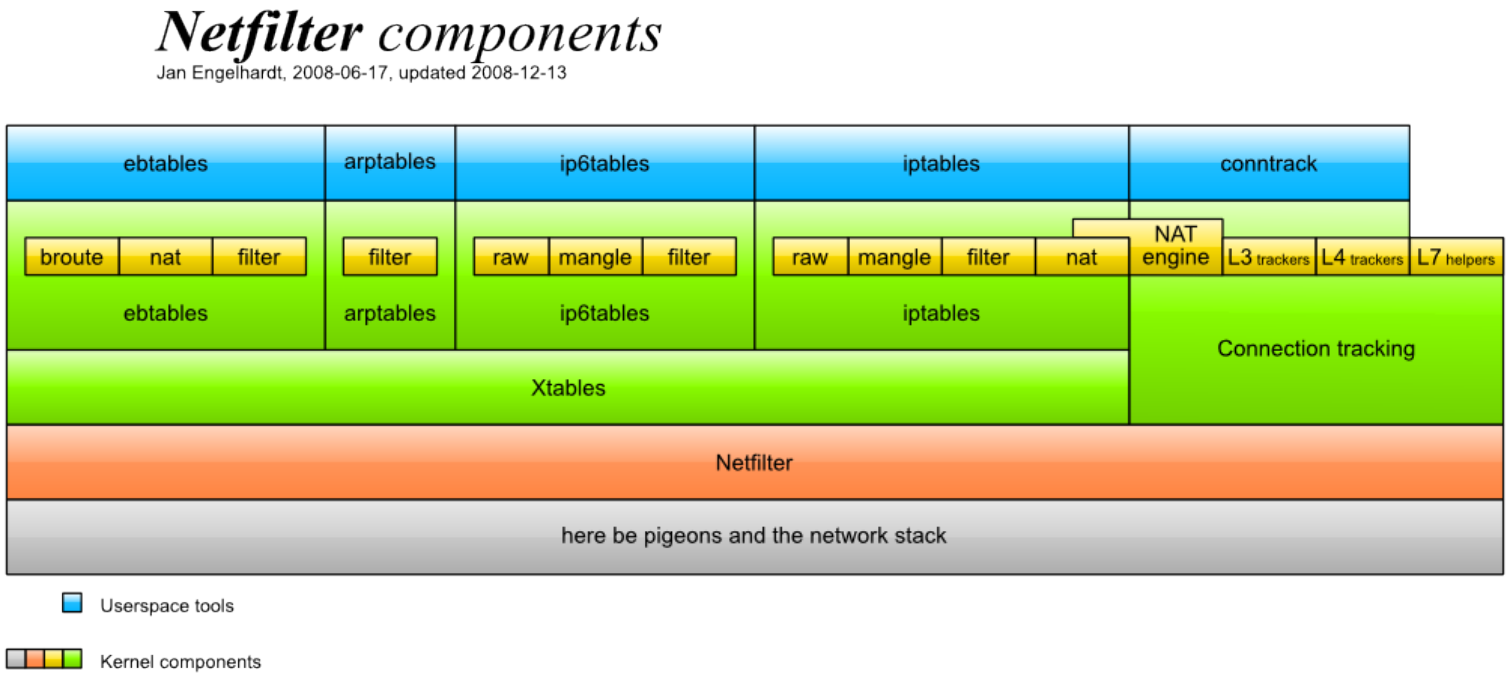
Histórico

- Linux 2.4 – 2.6: *netfilter* + *iptables*
  - Novo *framework* para manipulação de pacotes pelo *kernel*, baseado em “ganchos” onde módulos carregados dinamicamente podem processar pacotes individualmente.
  - Separação entre:
    - Filtragem de pacotes
    - Rastreamento de conexões
    - NAT e tradução de endereços em geral

Arquitetura

- Netfilter:
  - Uma série de “ganchos” dentro do *kernel*, nos procedimentos que tratam da comunicação em rede.
  - Módulos registram procedimentos nestes ganchos, que são chamados quando um respectivo pacote passa por aquele gancho.
- *iptables*:
  - Um dos módulos do Netfilter, fornece uma estrutura de tabelas para a definição de regras de filtragem de pacotes.

Arquitetura



Os componentes do Netfilter.

iptables
(filtro de pacotes)

iptables: filtro de pacotes

- Filtro de pacotes:
  - Para cada pacote de dados que passa pelo sistema, analisa determinadas características do pacote (definidos pelo usuário e/ou pelo *firewall*) e determina se o pacote pode ser aceito ou deve ser descartado, ou ainda, se alguma ação deve ser executada.

iptables: filtro de pacotes

- Composto por:
  - Tabelas
  - Correntes
  - Regras
  - Testes
  - Destinos

iptables: filtro de pacotes

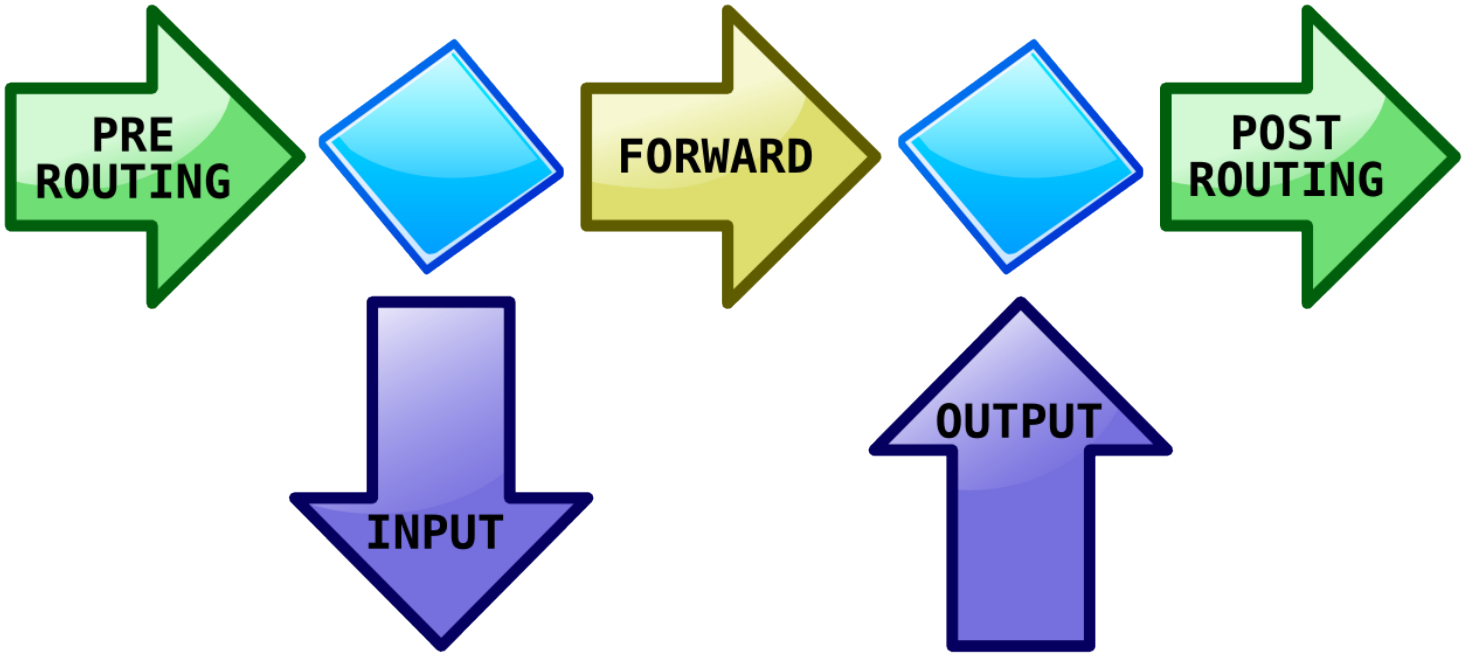
- Tabelas:
  - O *iptables* fornece 4 tabelas pré-definidas, para finalidades específicas:
    - *filter* (filtragem de pacotes)
    - *nat* (tradução de endereços)
    - *mangle* (alteração de pacotes)
    - *raw* (configuração de rastreamento de pacotes)
  - Cada tabela contém correntes de regras.
  - Algumas correntes são padrão, e o usuário pode criar suas próprias correntes, e encadeá-las às correntes-padrão.

iptables: filtro de pacotes

- Correntes:
  - Cada corrente contém uma lista de regras que são testadas e aplicadas a cada pacote que percorre aquela corrente.
  - O usuário pode criar novas correntes e encadeá-las umas às outras e às correntes-padrão.
  - As seguintes correntes-padrão são definidas:
    - PREROUTING (todos os pacotes que chegam)
    - INPUT (pacotes que entram para o próprio sistema)
    - FORWARD (pacotes roteados através do sistema)
    - OUTPUT (pacotes gerados pelo próprio sistema)
    - POSTROUTING (todos os pacotes que saem)

iptables: filtro de pacotes

- Correntes-padrão:



iptables: filtro de pacotes

- Regras:
  - Cada regra é composta por um número de testes e um destino.
  - Os testes verificam características do pacote (endereço, protocolo, porta, etc.). Cada teste pode ser positivo ou negativo.
  - O destino determina o que acontece com os pacotes que combinar com todos os testes.
    - Pode ser uma outra corrente na mesma tabela, um dos destinos-padrão (ACCEPT, DROP, QUEUE, REJECT), ou ainda, uma das várias extensões do *iptables* (SNAT, DNAT, LOG, REDIRECT, REJECT, etc.).

Encadeamento de Correntes

```
Chain: INPUT
target  prot  source          destination
ACCEPT  tcp   anywhere        anywhere     tcp dpt:80
ACCEPT  tcp   10.0.0.0/8      anywhere     tcp dpt:22
DROP    tcp   anywhere        anywhere     tcp dpt:22
foobar  udp   anywhere        anywhere     udp dpt:53
foobar  udp   anywhere        anywhere     udp dpt:123
REJECT  all   192.0.2.0/24    anywhere

Chain: foobar
target  prot  source          destination
LOG     all   anywhere        anywhere
REJECT  all   10.5.1.0/24     anywhere
ACCEPT  all   10.0.0.0/8      anywhere
```

Encadeamento de Correntes

Pacote:
IP origem: 192.0.2.15
IP destino: 192.168.2.5
Protocolo: UDP
Porta origem: 49212
Porta destino: 53

Chain: INPUT

	target	prot	source	destination	
–	ACCEPT	tcp	anywhere	anywhere	tcp dpt:80
–	ACCEPT	tcp	10.0.0.0/8	anywhere	tcp dpt:22
–	DROP	tcp	anywhere	anywhere	tcp dpt:22
+	foobar	udp	anywhere	anywhere	udp dpt:53
–	foobar	udp	anywhere	anywhere	udp dpt:123
+	REJECT	all	192.0.2.0/24	anywhere	

Chain: foobar

	target	prot	source	destination
+	LOG	all	anywhere	anywhere
–	REJECT	all	10.5.1.0/24	anywhere
–	ACCEPT	all	10.0.0.0/8	anywhere

Encadeamento de Correntes

Pacote:
IP origem: 10.1.5.6
IP destino: 192.168.2.5
Protocolo: UDP
Porta origem: 49212
Porta destino: 53

Chain: INPUT

	target	prot	source	destination	
–	ACCEPT	tcp	anywhere	anywhere	tcp dpt:80
–	ACCEPT	tcp	10.0.0.0/8	anywhere	tcp dpt:22
–	DROP	tcp	anywhere	anywhere	tcp dpt:22
+	foobar	udp	anywhere	anywhere	udp dpt:53
–	foobar	udp	anywhere	anywhere	udp dpt:123
–	REJECT	all	192.0.2.0/24	anywhere	

Chain: foobar

	target	prot	source	destination
+	LOG	all	anywhere	anywhere
–	REJECT	all	10.5.1.0/24	anywhere
+	ACCEPT	all	10.0.0.0/8	anywhere

iptables: filtro de pacotes

- Política (destino padrão):
  - Define o destino de um pacote que chegar ao final de uma das correntes-padrão se nenhuma regra for ativada ou fornecer um veredito:
    - ACCEPT (o pacote é aceito)
    - DROP (o pacote é descartado)

Testes

- Testes-padrão:
  - Endereços de origem e destino:
    - `-s «endereço»/«pref»`
    - `-d «endereço»/«pref»`
  - Interfaces de rede de entrada e saída:
    - `-i «eth»`
    - `-o «eth»`
  - Protocolo (camada de transporte):
    - `-p «proto»` (“tcp”, “udp”, “icmp”, etc.)

Testes

- Testes modulares:
  - Portas (TCP e UDP) de origem e destino:

    `-p tcp --sport «porta»`

    `-p tcp --dport «porta»`
  - Mensagens ICMP:

    `-p icmp --icmp-type «type»/«code»`
  - Limite de taxa de pacotes:

    `-m limit --limit «taxa»`
  - Tráfego recente:

    `-m recent --name «name» --set/--update`
  - Sequência de bytes:

    `-m string --algo «algo» --string «string»`

Testes

- Muitos outros testes disponíveis:
  - addrtype, ah, cluster, connbytes, connmark, conntrack, dccp, dscp, ecn, esp, hashlimit, helper, icmp, iprange, length, mac, mark, owner, physdev, pkttype, policy, quota, rateest, realm, sctp, set, socket, state, statistic, tcpmss, time, tos, ttl, u32, unclean.

Destinos

- Destinos-padrão:
  - Aceita o pacote:

    ```
    -j ACCEPT
    ```
  - Descarta o pacote:

    ```
    -j DROP
    ```
  - Enfileira o pacote para o espaço de usuário:

    ```
    -j QUEUE
    ```
  - Retorna para a corrente anterior:

    ```
    -j RETURN
    ```

Destinos

- Destinos modulares:
  - Ajusta a classe de tráfego:
    `-j CLASSIFY --set-class «major»:«minor»`
  - Rejeita o pacote (retorna um pacote de erro):
    `-j REJECT --reject-with «tipo»`
  - Registro de pacotes:
    `-j LOG`
  - Tradução de endereços (tabela "*nat*"):
    `-j SNAT --to-source «endereço»:«porta»`
    `-j DNAT --to-destination «endereço»:«porta»`
  - Redireção de tráfego (tabela "*nat*"):
    `-j REDIRECT --to-ports «porta»`

Destinos

- Vários outros destinos disponíveis:
  - CLUSTERIP, CONNMARK, CONNSECMARK, DSCP, ECN, MARK, MASQUERADE, MIRROR, NETMAP, NFLOG, NFQUEUE, NOTRACK, RATEEST, SAME, SECMARK, SET, TCPMSS, TCPOPTSTRIP, TOS, TPROXY, TRACE, TTL, ULOG.

Rastreamento de Conexões

- Módulo "*conntrack*":
  - Com este módulo, iptables se torna um *stateful firewall*, ou seja, é capaz de tratar conexões ao invés de apenas pacotes individuais.
- Testes para o estado da conexão:

```
-m state --state INVALID
-m state --state NEW
-m state --state ESTABLISHED
-m state --state RELATED
```

Exemplos

- Política (destino padrão) das correntes:
  - `iptables -P INPUT ACCEPT`
  - `iptables -P OUTPUT ACCEPT`
  - `iptables -P FORWARD DROP`
- Permitir pacotes de conexões conhecidas:
  - `iptables -A INPUT -j ACCEPT -m state --state ESTABLISHED,RELATED`
- Permitir acesso ao serviço SMB apenas da rede local:
  - `iptables -A INPUT -j ACCEPT -s 10.0.0.0/24 -p tcp --dport 135:139`
  - `iptables -A INPUT -j REJECT -p tcp –dport 135:139`

Exemplos

- Rotear pacotes da rede local para a Internet
  - `iptables -A FORWARD -j ACCEPT -i eth0 -o eth1 -s 10.0.0.0/24`
  - `iptables -A FORWARD -j ACCEPT -i eth1 -o eth0 -d 10.0.0.0/24`
- Fazer tradução de endereços (NAT):
  - `iptables -t nat -A POSTROUTING -o eth1 -s 10.0.0.0/24 -j SNAT --to 192.0.2.5`

tc
(controle de tráfego)

tc: controle de tráfego

- Composto por:
  - Disciplinas de enfileiramento (*qdisc*)
  - Classes de tráfego
  - Filtros
- Importante:
  - Apenas o tráfego que sai do sistema pode ser regulado e limitado. O tráfego que entra no sistema só pode ser policiado ou descartado.

tc: controle de tráfego

Comportamento típico de um enlace ADSL (500 kbit/s downstream, 128 kbit/s upstream).

tc: controle de tráfego

- Disciplinas de enfileiramento (*qdisc*)
  - Elemento fundamental do controle de tráfego.
  - Sempre que o *kernel* precisa enviar um pacote para a rede, o pacote é enfileirado em uma *qdisc* configurada para a interface de saída.
  - A *qdisc* é responsável por dizer ao kernel quando e como os pacotes enfileirados irão sair para a rede.
  - Dois tipos:
    - *qdiscs* básicas (livres de classes)
    - *qdiscs* baseadas em classes

tc: controle de tráfego

- Classes de tráfego:
  - Estão contidas nas *qdiscs* baseadas em classes.
  - Cada *qdisc* pode conter múltiplas classes.
  - São utilizadas para classificar o tráfego que deve passar por aquela *qdisc*.
  - Cada classe pode conter outras classes, ou outra *qdisc*, formando uma hierarquia de *qdiscs* e classes.

tc: controle de tráfego

- *Qdiscs* básicas (sem classes):
  - *PFIFO*, *BFIFO* (*First-In*, *First-Out*)
  - *PFIFO_FAST*: FIFO estendida com três bandas, permitindo a classificação pelas *flags* de Tipo de Serviço do pacote.
  - *RED* (*Random Early Detection*): descarta pacotes aleatórios quando próximo do limite de banda.
  - *SFQ* (*Stochastic Fair Queueing*): enfileiramento justo para todas as sessões.
  - *TBF* (*Token Bucket Filter*): reluga o tráfego para uma taxa precisamente definida.

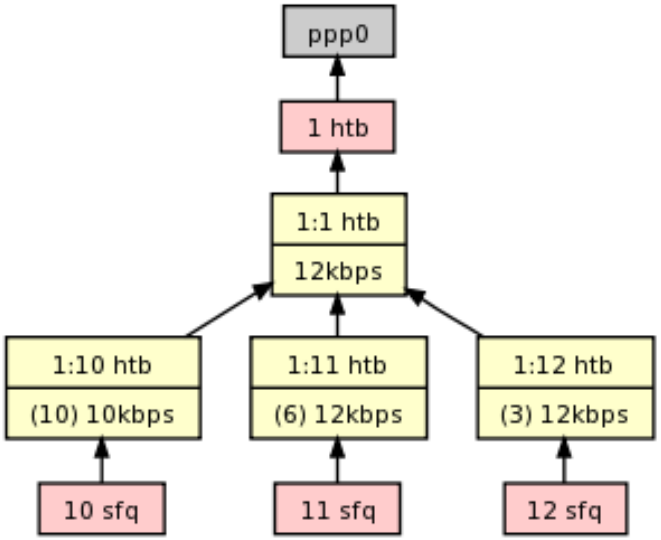
tc: controle de tráfego

- *Qdiscs* baseadas em classes:
  - *CBQ* (*Class Based Queueing*): permite uma rica hierarquia de classes, suportando limitação e priorização de tráfego.
  - *HTB* (*Hierarchy Token Bucket*): como o CBQ, permite uma rica hierarquia de classes, e permite garantir largura de banda para certas classes e compartilhamento de banda entre classes.
  - *PRIO*: permite definir a ordem (prioridade) de desenfileiramento entre as classes, mas não possui a capacidade de regular o tráfego.

tc: controle de tráfego

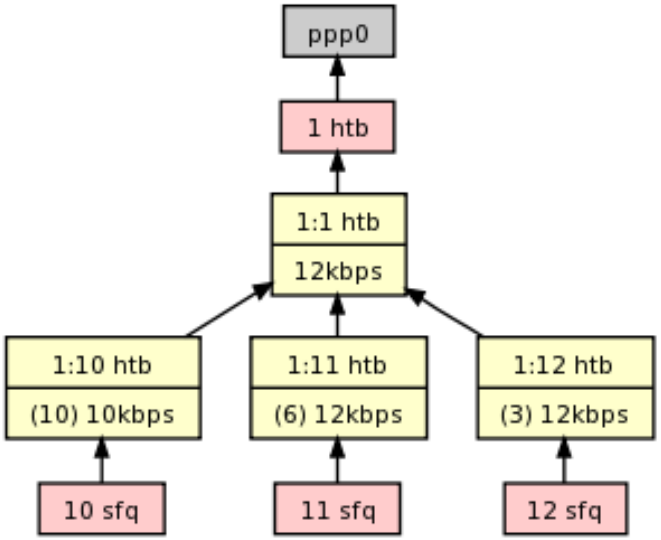
- Filtros:
  - Estão contidos nas *qdiscs* baseadas em classes, e são utilizados para determinar em qual classe determinado pacote deve ser enfileirado.
  - A classificação também pode ser feita pelo *iptables* (destino CLASSIFY), e é geralmente preferido, uma vez que há mais recursos para classificação de pacotes, como o rastreamento de conexões.

Exemplo




```
PPP="ppp0"
ifconfig $PPP txqueuelen 1000
tc qdisc add dev $PPP root handle 1:0 htb default 12
tc class add dev $PPP parent 1:0 classid 1:1 htb rate 12kbps
tc class add dev $PPP parent 1:1 classid 1:10 htb rate 10kbps prio 0
tc class add dev $PPP parent 1:1 classid 1:11 htb rate 6kbps ceil 12kbps prio 1
tc class add dev $PPP parent 1:1 classid 1:12 htb rate 3kbps ceil 12kbps prio 1
tc qdisc add dev $PPP parent 1:10 handle 10:0 sfq perturb 10
tc qdisc add dev $PPP parent 1:11 handle 11:0 sfq perturb 10
tc qdisc add dev $PPP parent 1:12 handle 12:0 sfq perturb 10
```

Exemplo

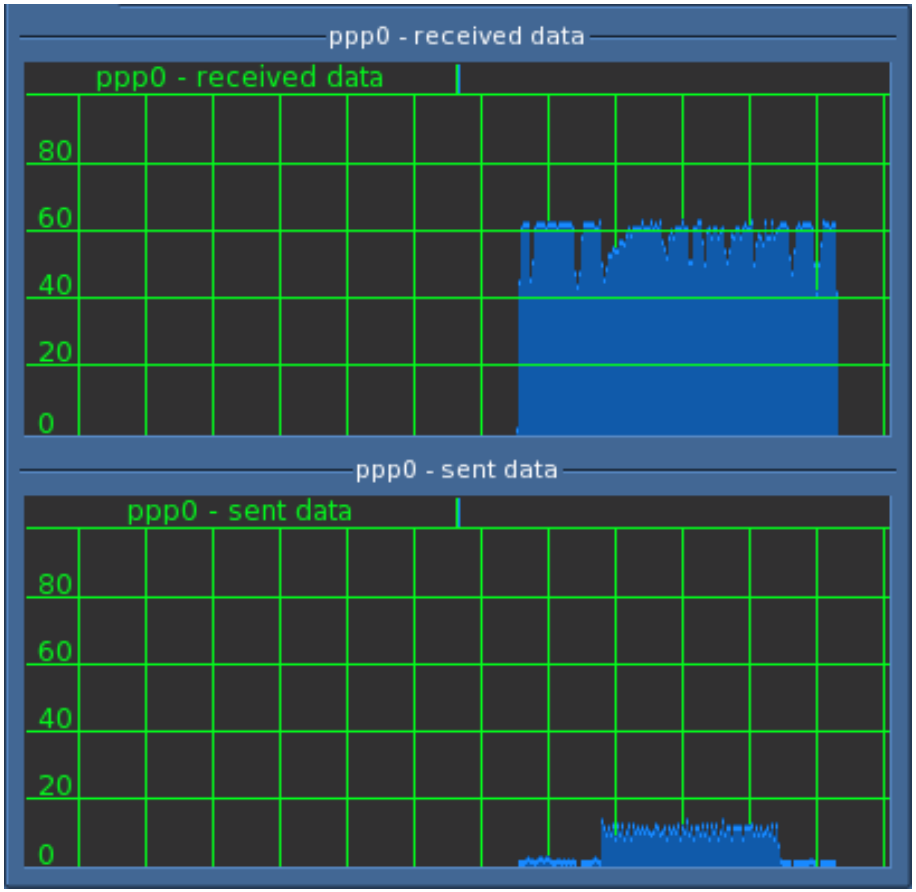



```
PPP="ppp0"

iptables -t mangle -A OUTPUT -o $PPP -m length --length 0:128 \
          -j CLASSIFY --set-class 1:10

iptables -t mangle -A OUTPUT -o $PPP -p tcp --dport 25 \
          -j CLASSIFY --set-class 1:11
```

Exemplo



Comportamento do enlace ADSL após aplicar o controle de tráfego.

O quê mais?

- quagga: Roteamento dinâmico.
- radvd: Anúncio de roteador IPv6.
- brctl: Criação de *bridges*.
- ip tunnel: Criação de túneis.
- setkey, racoon: Configuração de IPSec.