K-SHAPER

K-Shaper è un'applicazione che permette di creare un sistema di code di una certa complessità, nascondendo i dettagli implementativi (quali, ad esempio, la sintassi non proprio semplice di tc e iptables, i comandi che, effettivamente, sono necessari per configurare un sistema di traffic shaping), e permettendo che ci si concentri, piuttosto, sulla progettazione.

In particolare, è uno script in linguaggio PERL che, analizzato un file di configurazione, crea un sistema di code suddiviso in classi, aggiunge i relativi filtri, eventuali algoritmi di scheduling, e assegna la quantità di banda decisa ad ogni classe.

In dettaglio, lo script esegue un parsing del file di configurazione, scritto in XML, costruisce stringhe con i dati recuperati dal file, scrive il tutto in uno script bash, e alfine esegue quest'ultimo.

La scelta del linguaggio Perl è dettata dal fatto che è un linguaggio molto adatto a manipolare testi, stringhe ed espressioni regolari, e comunque ha una buona (intesa come velocità di esecuzione) interazione con il sistema ospite.

Per quanto riguarda l'uso di XML per il file di configurazione, la motivazione è da trovarsi nella natura stessa di questo linguaggio: un documento XML presenta i dati secondo una struttura ad albero, e un sistema (più o meno) complesso di code ha, anch'esso, una struttura ad albero. Chi, quindi, redigerà il suddetto file di configurazione non avrà poi molti problema nel rendere in “testo”, usando gli appositi tag, l'albero delle code più appropriato per quel particolare contesto.

Inoltre, un documento XML è sempre accompagnato da altri linguaggi (Schema e derivati) che controllano la validità del documento di istanza, in particolare per quanto riguarda la sintassi e la semantica dei dati, e le dipendenze tra questi.

In tal modo lo script non si deve preoccupare (per la maggior parte dei casi, in realtà...) di controllare la validità del file di configurazione man mano che lo analizza, perche il “check” viene fatto una tantum nella fase iniziale del parsing.

Infine, XML è un linguaggio di interscambio dati molto diffuso, ed il suo utilizzo potrebbe tornare molto utile in caso di sviluppi molto massivi di questo software (come ad esempio, una totale riscrittura in Java o Python).

Segue il menu delle scelte (più significative) disponibili invocando l'--help del programma:

`-a|--add-rules`

Aggiunge le regole di accodamento/filtraggio al kernel analizzando il file di configurazione

`-c|--cache-file=<cache.sh>`

Usa un file di cache.sh (lo script bash creato) particolare invece del predefinito /var/lib/k-shaper/cache.sh

`-f|--flush-rules`

Elimina le regole esistenti

`-l|--list-rules`

Mostra le regole esistenti

`-t|--view-rules`

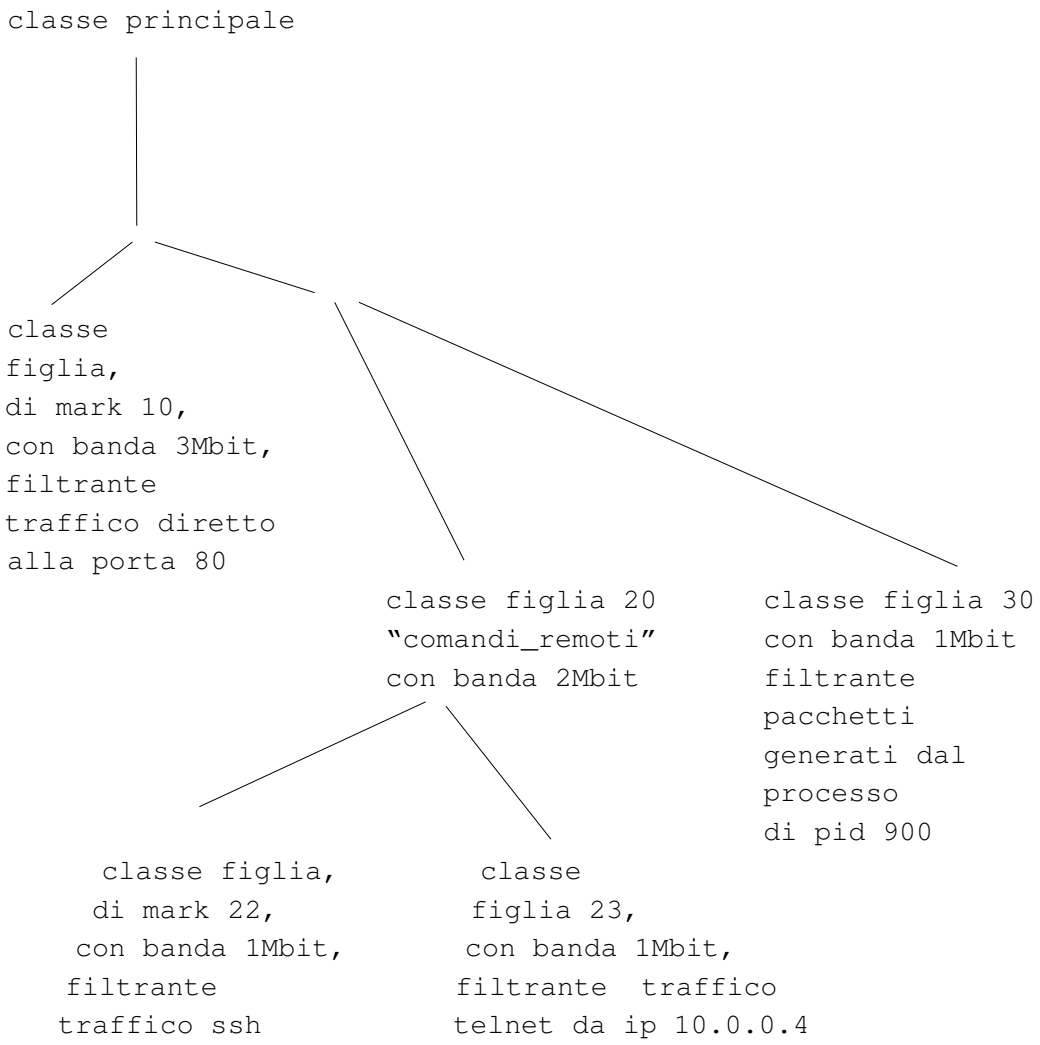
Testa il file di configurazione mostrando a video le righe di comando che saranno eseguite a partire dal file cache.sh

`-x|--config-file=<file.xml>`

Usa un file di configurazione particolare invece del predefinito /etc/k-shaper/config.xml

Infine una breve spiegazione del file di configurazione, illustrato con un esempio.

Si ipotizzi una lan con 10 Mbit di banda, in uscita e in ingresso; si vuole avere la seguente struttura:



Ed ecco il corrispondente file xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="config.xsd">
  <queue direction="both">
    <root default="100">
      <class>
        <id name="first">2</id>
        <rate unit="mbit">10</rate>
        <class>
            <id name="http">10</id>
            <rate unit="mbit">3</rate>
            <filter>
                  <port target="destination" protocol="tcp">80</port>
            </filter>
        </class>
        <class>
            <id name="remote_commands">20</id>
            <rate unit="mbit">2</rate>
            <class>
                  <id name="ssh">22</id>
                  <rate unit="mbit">1</rate>
                  <filter>
                        <application>ssh</application>
                  </filter>
            </class>
            <class>
                  <id name="telnet">23</id>
                  <rate unit="mbit">1</rate>
                  <filter>
                        <ip target="source">10.0.0.4</ip>
                        <application>telnet</application>
                  </filter>
            </class>
        </class>
        <class>
            <id name="local_p2p">30</id>
            <rate unit="mbit">1</rate>
            <filter>
                  <pid>900</pid>
            </filter>
        </class>
        <class>
            <id name="default">100</id>
            <rate unit="kbit">1</rate>
        </class>
      </class>
    </root>
  </queue>
</config>
```

La maggior parte dei tag sono esplicativi, come “class”, “id”, “rate”, “filter”: per altri è bene spiegarne il significato e la struttura:

- ✔ il documento deve sempre iniziare con il tag <config>, che accetta come figlio il tag <queue> (coda), al minimo 1, al massimo 2 (rispettivamente 1 coda per traffico uscente, 1 per traffico entrante, o 1 sola coda con attributo *direction*=”both” se c'è banda simmetrica);
- ✔ una coda <queue>, come tale, deve avere al minimo una classe (<classe>), di cui una di default per il traffico non specificato (default=”numero_intero”, che corrisponde all'id della coda di default);
- ✔ la classe <class> deve sempre avere un identificativo numerico <id> (che, a sua volta, ha un *name)*, una certa banda <rate>, e può avere un certo numero di code figlie, o essere una coda figlia: in quest'ultimo caso deve avere l'attributo *leaf*=”yes” e 1 filtro <filter> associato, ed eventualmente uno scheduler <sched> diverso da fifo (può essere pfifo, bfifo, sfq) [nel caso l'attributo leaf non fosse presente, il parser non leggerebbe in nessun caso i sub-tag <filter> e <sched>];
- ✔ infine <filter>: deve avere almeno un sub-tag, che specifica un qualche filtro, tra i seguenti <ip> <port> <application> (inteso come strato applicativo, o protocollo di livello 7: vedasi prima “l7-filter”) <pid> <group> <user> <command> <session> <string>. Per altri particolari tralasciati si rimanda direttamente al file di Schema config.xsd, dove per ogni tag è presente un commento che ne spiega l'utilizzo.

E' infine da aggiungere che, nonostante i molti tag e attributi presenti, la creazione del file config.xml è alquanto banale, soprattutto se vengono utilizzati editor grafici xml (uno per tutti, Eclipse) che, a partire dallo schema, forniscono automaticamente le scelte disponibili per quel particolare tag o attributo.