snmpPlugin

Fusco Francesco
Giuseppe Giardina

snmpPlugin:

di Fusco Francesco e Giuseppe Giardina
Copyright © 2004 Fusco FrancescoGiuseppe Giardina

Sommario

Capitolo 1. Introduzione

Questo documento descrive la realizzazione di un plugin per Ntop [ntop] per aggiungere al programma il supporto a snmp.

Il fine del plugin è quello di permettere la lettura dei dati raccolti da Ntop attraverso il protocollo snmp.

Ntop infatti permette di leggere i dati attraverso un'interfaccia web con i vantaggi e gli svantaggi che comporta. Se da un lato l'interfaccia web semplifica la consultazione dei dati agli utenti, dall'altro rende più difficile la realizzazione di programmi per l'analisi dei dati di traffico. Questo tipo di programmi devono fondamentalmente eseguire delle *HTTP GET* sul server web di ntop e fare il parsing dei dati ottenuti.

L'aggiunta del supporto SNMP semplifica la realizzazione di programmi di analisi sui dati di Ntop perchè permette di utilizzare i tools già disponibili per SNMP.

Per la realizzazione del plugin abbiamo definito un MIB ed abbiamo realizzato un agent la cui esecuzione è controllata da ntop. Per la realizzazione dell'agent abbiamo deciso di utilizzare Net-snmp [netsnmp] perchè è un prodotto opensource disponibile per le maggiori piattaforme e molto diffuso.

Capitolo 2. I dati da gestire

Prima di iniziare qualsiasi attività di progettazione abbiamo cercato di capire quale fosse l'architettura di massima si Ntop e quali fossero le strutture dati ed i tipi di dato più importanti. Nel file *globals-struc-types.h* sono definiti i tipi di dato del programma, mentre il file *globals-core.h* contiene i prototipi di funzioni.

Un tipo di dato fondamentale per Ntop è la struct *HostTraffic* che memorizza dati di traffico relativi ad un host. Nel seguito riporto la struct solo per avere un idea di quali sono i campi che contiene.

```
typedef struct hostTraffic {
  u_short           magic;
  u_short           l2Family;
  u_int             hostTrafficBucket;
  u_int             originalHostTrafficBucket;
  u_short           refCount;
  HostSerial        hostSerial;
  HostAddr          hostIpAddress;
  short             vlanId;
  u_int16_t         hostAS;
  time_t            firstSeen, lastSeen;
  u_char            ethAddress[LEN_ETHERNET_ADDRESS];
  u_char            lastEthAddress[LEN_ETHERNET_ADDRESS];
  char              ethAddressString[LEN_ETHERNET_ADDRESS_DISPLAY];
  char              hostNumIpAddress[20] /* xxx.xxx.xxx.xxx */,
                    *dnsDomainValue,
                    *dnsTLDValue;
  char              *ip2ccValue,
                    hostResolvedName[MAX_LEN_SYM_HOST_NAME],
                   *fingerprint;
  short             hostResolvedNameType;
  u_short           minTTL, maxTTL; /* IP TTL (Time-To-Live) */
  struct timeval    minLatency, maxLatency;

  NonIPTraffic     *nonIPTraffic;
  /* Info about further non IP protos */
  NonIpProtoTrafficInfo *nonIpProtoTrafficInfos;

  fd_set            flags;
  TrafficCounter    pktSent, pktRcvd, pktSentSession, pktRcvdSession,
    pktDuplicatedAckSent, pktDuplicatedAckRcvd;
  TrafficCounter    lastPktSent, lastPktRcvd;
  TrafficCounter    pktBroadcastSent, bytesBroadcastSent;
  TrafficCounter    pktMulticastSent, bytesMulticastSent,
    pktMulticastRcvd, bytesMulticastRcvd;
  TrafficCounter    lastBytesSent, lastHourBytesSent,
    bytesSent, bytesSentLoc, bytesSentRem, bytesSentSession;
  TrafficCounter    lastBytesRcvd, lastHourBytesRcvd, bytesRcvd,
    bytesRcvdLoc, bytesRcvdFromRem, bytesRcvdSession;
  float             actualRcvdThpt, lastHourRcvdThpt,
                    averageRcvdThpt, peakRcvdThpt,
                    actualSentThpt, lastHourSentThpt,
                    averageSentThpt, peakSentThpt,
                    actualTThpt, averageTThpt, peakTThpt;
  float             actualRcvdPktThpt, averageRcvdPktThpt,
                    peakRcvdPktThpt,actualSentPktThpt,
                    averageSentPktThpt, peakSentPktThpt,
                    actualTPktThpt, averageTPktThpt, peakTPktThpt;
  unsigned short    actBandwidthUsage, actBandwidthUsageS,
                    actBandwidthUsageR;
```

```
  TrafficDistribution *trafficDistribution;
  u_int32_t        numHostSessions;

  /* Routing */
  RoutingCounter   *routedTraffic;

  /* IP */
  PortUsage        **portsUsage; /* 0...MAX_ASSIGNED_IP_PORTS */
  /* Don't change the recentl... to unsigned ! */
  int              recentlyUsedClientPorts[MAX_NUM_RECENT_PORTS],
                   recentlyUsedServerPorts[MAX_NUM_RECENT_PORTS];
  int              otherIpPortsRcvd[MAX_NUM_RECENT_PORTS],
                   otherIpPortsSent[MAX_NUM_RECENT_PORTS];
  TrafficCounter   ipBytesSent, ipBytesRcvd, ipv6Sent, ipv6Rcvd;
  TrafficCounter   tcpSentLoc, tcpSentRem, udpSentLoc,
                   udpSentRem, icmpSent,icmp6Sent;
  TrafficCounter   tcpRcvdLoc, tcpRcvdFromRem, udpRcvdLoc,
                   udpRcvdFromRem, icmpRcvd, icmp6Rcvd;

  TrafficCounter   tcpFragmentsSent,  tcpFragmentsRcvd,
    udpFragmentsSent, udpFragmentsRcvd,
    icmpFragmentsSent, icmpFragmentsRcvd,
    icmp6FragmentsSent, icmp6FragmentsRcvd;

  /* Protocol decoders */
  ProtocolInfo     *protocolInfo;

  /* Interesting Packets */
  SecurityHostProbes *secHostPkts;
  IcmpHostInfo       *icmpInfo;
  /* List of myGlobals.numIpProtosList entries */
  ShortProtoTrafficInfo **ipProtosList;
  /* Info about IP traffic generated/rcvd by this host */
  ProtoTrafficInfo      **protoIPTrafficInfos;

  /* Fiber Channel/SCSI */
  FcScsiCounters   *fcCounters;
  /* # of different contacted peers */
  Counter          totContactedSentPeers, totContactedRcvdPeers;
  /* peers that talked with this host */
  UsageCounter     contactedSentPeers;
  /* peers that talked with this host */
  UsageCounter     contactedRcvdPeers;
  /* routers contacted by this host */
  UsageCounter     contactedRouters;
  /* pointer to the next element */
  struct hostTraffic *next;
} HostTraffic;
```

Come si può bene vedere la maggioranza dei campi sono dei contatori di tipo *TrafficCounter* e di tipo *Counter*. Tali contatori sono aggiornati da ntop mano a mano che legge pacchetti dalla rete, quindi sono dati molto variabili nel tempo. Lo scopo di questo progetto è fondamentalmente quello di permettere la lettura di questi campi attraverso snmp.

Una struct importante contenuta nella *HostTraffic* è la *HostSerial*.

```
typedef struct hostSerial {
  u_char serialType;     /* 0 == empty */
  union {
    EthSerial ethSerial; /* hostSerial == SERIAL_MAC */
```

```
    IpSerial  ipSerial;  /* hostSerial == SERIAL_IPV4/SERIAL_IPV6 */
    FcSerial  fcSerial;
  } value;
} HostSerial;
```

Questa struct è una chiave utilizzata per ottenere dati di traffico di uno specifico host dalla hash di ntop.

Capitolo 3. Progettazione del MIB

Prima di effettuare alcun tipo di scelta progettuale sul MIB che doveva essere realizzato abbiamo deciso di metterci nell'ottica di un potenziale utente del nostro plugin. Era infatti di primaria importanza capire quale poteva essere il tipico utilizzo di snmp nel contesto di ntop per creare uno strumento che fosse di una qualche utilità pratica. Le idee che ne sono uscite sono le seguenti:

un utente deve poter richiedere in modo semplice dati relativi ad un host
: Quindi l'indice di qualsiasi tabella concettuale definita doveva rappresentare in un qualche modo un host

un utente deve poter richiedere il valore di un contatore di traffico con una sola operazione snmp
: Quindi usare troppe tabelle avrebbe reso meno usabile per l'utente il plugin e complicato la realizzazione

Alla luce di queste riflessioni abbiamo deciso di utilizzare una singola tabella che include tutti i parametri di interesse della *struct HostTraffic*. L'indice di questa tabella è un indice multiplo costituito da:

- tipo di HostSerial
- device dal quale arriva la HostTraffic
- vlan o vsan
- Indirizzo

La scelta di utilizzare una sola grande tabella è stata fatta per rendere il mib il più semplice possibile. Avevamo valutato l'ipotesi di suddividere i tipi di host creando una tabella per ogni tipo. Questa soluzione è stata subito scartata perchè le tabelle avrebbero avuto la maggioranza delle colonne in comune e si sarebbero differenziate solo per una minoranza.

Per comodità abbiamo definito delle *TEXTUAL-CONVENTION* per i campi che costituiscono l'indice. Tali tc sono:

NtopSerialType
: Specifica quale è il tipo di Serial che viene utilizzato nell'indice. Il tipo di serial può essere uno tra NtopSerialEth, NtopSerialIpv4, NtopSerialIpv6, NtopSerialFc.

NtopSerial
: Rappresenta un Serial generico ed è utilizzato per specificare un indice di lunghezza variabile nel mib.

NtopSerialEth
: Rappresenta un Serial di un indirizzo ethernet

NtopSerialIpv4
: Rappresenta un Serial di un indirizzo ipv4

NtopSerialIpv6
: Rappresenta un Serial di un indirizzo ipv6

NtopSerialFc
: Rappresenta un Serial dell'indirizzo di un dispositivo Fibre Channel

NtopActualDevice
Rappresenta il device sul quale si vuol fare la richiesta di traffico per uno specifico host.

Il mib

```
NTOP-MIB DEFINITIONS ::= BEGIN

IMPORTS
MODULE-IDENTITY, OBJECT-TYPE, enterprises, Counter32,
Integer32,Counter64                          FROM SNMPv2-SMI
TEXTUAL-CONVENTION  ,DisplayString           FROM SNMPv2-TC
OBJECT-GROUP,MODULE-COMPLIANCE               FROM SNMPv2-CONF;

ntop MODULE-IDENTITY
LAST-UPDATED "9902100000Z"
ORGANIZATION "Universita' di Pisa"
CONTACT-INFO "

Fusco Francesco (Editor)
E-mail: fuscof@cli.di.unipi.it"

DESCRIPTION
"The MIB module for ntop "
::= { enterprises 30000 }

--TODO: registrare presso lo iana


NtopActualDevice ::= TEXTUAL-CONVENTION
STATUS      current
DESCRIPTION
"A value that represents the actual sniffing device."
SYNTAX    Integer32 (0..8191)


NtopSerialType ::= TEXTUAL-CONVENTION
STATUS      current
DESCRIPTION
"A value that represents a type of serial used by ntop.

unknown(0)  An unknown serial type.

ethSerial   A serial for MAC serial

ipSerial    A serial for ipv4 or ipv6 serial

fcSerial    A serial for fibre channel serial


Each definition of a concrete serialType value must be
accompanied by a definition of a textual convention for use
with that SerialType."


SYNTAX      INTEGER {
unknown(0),
ethSerial(1),
ipv4Serial(2),
```

```
ipv6Serial(3),
fcSerial(4)
}

NtopSerial ::= TEXTUAL-CONVENTION
STATUS       current
DESCRIPTION
"Denotes a generic serial used by ntop.

A NtopSerial value is always interpreted within the
context of an NtopSerialType value."

SYNTAX      OCTET STRING (SIZE (4|6|16))

NtopSerialEth ::= TEXTUAL-CONVENTION
DISPLAY-HINT "1x:1x:1x:1x"
STATUS       current
DESCRIPTION
"Represents a mac  address:

The corresponding NtopSerialType value is ethSerial(1)."

SYNTAX       OCTET STRING (SIZE (6))

NtopSerialIPv4 ::= TEXTUAL-CONVENTION
DISPLAY-HINT "1d.1d.1d.1d"
STATUS       current
DESCRIPTION
"Represents an IPv4 network address.

The corresponding NtopSerialType value is ipv4Serial(2)."

SYNTAX       OCTET STRING (SIZE (4))

NtopSerialIPv6 ::= TEXTUAL-CONVENTION
DISPLAY-HINT "2x:2x:2x:2x:2x:2x:2x:2x"
STATUS       current
DESCRIPTION

"Represents an IPv6 network address.

The corresponding NtopSerialType value is ipv6(2)."

SYNTAX       OCTET STRING (SIZE (16))

NtopSerialFc::= TEXTUAL-CONVENTION
DISPLAY-HINT "255a"  --TODO: how to print?
STATUS       current
DESCRIPTION
"Represents a fibre channel serial.

The corresponding NtopSerialType value is fcSerial(4)."

SYNTAX       OCTET STRING (SIZE (6))



-- Some fun with tables:)

ntopTable OBJECT-TYPE
SYNTAX      SEQUENCE OF NtopEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
```

```
"A list of communication peers."
::= { ntop 1 }

ntopEntry OBJECT-TYPE
SYNTAX      NtopEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"An entry ."
INDEX       { ntopSerialType, ntopActualDevice , ntopVanId, ntopSerial }
::= { ntopTable 1 }

NtopEntry ::= SEQUENCE {

--indexes
ntopSerialType              NtopSerialType,
ntopActualDevice            NtopActualDevice,
ntopVanId                   Integer32,--(vsan e vlan)
ntopSerial                  NtopSerial,
-- end indexes

hostResolvedName            DisplayString,
fingerprint                 DisplayString,

pktSent                 Counter64,
pktRcvd                 Counter64,
pktSentSession         Counter64,
pktRcvdSession         Counter64,
pktDuplicatedAckSent Counter64,
pktDuplicatedAckRcvd Counter64,
pktBroadcastSent         Counter64,
bytesBroadcastSent         Counter64,
pktMulticastSent         Counter64,
bytesMulticastSent         Counter64,
pktMulticastRcvd         Counter64,
bytesMulticastRcvd         Counter64,
bytesSent                Counter64,
bytesSentLoc         Counter64,
bytesSentRem         Counter64,
bytesSentSession         Counter64,
bytesRcvd                Counter64,
bytesRcvdLoc         Counter64,
bytesRcvdFromRem         Counter64,
bytesRcvdSession         Counter64,
numHostSessions         Counter64,
ipBytesSent                Counter64,
ipBytesRcvd                Counter64,
ipv6Sent                 Counter64,
ipv6Rcvd                 Counter64,
tcpSentLoc                 Counter64,
tcpSentRem                 Counter64,
udpSentLoc                 Counter64,
udpSentRem                 Counter64,
icmpSent                 Counter64,
icmp6Sent                Counter64,
tcpRcvdLoc                 Counter64,
tcpRcvdFromRem         Counter64,
udpRcvdLoc                 Counter64,
udpRcvdFromRem         Counter64,
icmpRcvd                 Counter64,
icmp6Rcvd                Counter64,
tcpFragmentsSent         Counter64,
tcpFragmentsRcvd         Counter64,
udpFragmentsSent         Counter64,
```

```
udpFragmentsRcvd         Counter64,
icmpFragmentsSent         Counter64,
icmpFragmentsRcvd         Counter64,
icmp6FragmentsSent         Counter64,
icmp6FragmentsRcvd         Counter64,
totContactedSentPeers Counter64,
totContactedRcvdPeers Counter64,
contactedSentPeers         Counter64,
contactedRcvdPeers         Counter64,
contactedRouters         Counter64
}


ntopSerialType OBJECT-TYPE
SYNTAX      NtopSerialType
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
"The type of ntop serial. "
::= { ntopEntry 1 }

ntopActualDevice OBJECT-TYPE
SYNTAX      NtopActualDevice
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
"the actual sniffing device for ntop. "
::= { ntopEntry 2 }

ntopVanId OBJECT-TYPE
SYNTAX      Integer32 (0..65535)
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
"VLan o Vsan, O if not set "
::= { ntopEntry 3 }

ntopSerial OBJECT-TYPE
SYNTAX      NtopSerial
MAX-ACCESS  read-only
STATUS      current

DESCRIPTION
"The serial wich we care about."

::= { ntopEntry 4 }

hostResolvedName OBJECT-TYPE
SYNTAX                 DisplayString
MAX-ACCESS         read-only
STATUS                 current
DESCRIPTION
""
::= {ntopEntry 5 }

fingerprint         OBJECT-TYPE
SYNTAX                 DisplayString
MAX-ACCESS         read-only
STATUS                 current
DESCRIPTION
""
::= {ntopEntry 6 }
```

```
pktSent OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 7 }

pktRcvd OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 8 }

pktSentSession OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 9 }

pktRcvdSession OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 10 }

pktDuplicatedAckSent OBJECT-TYPE
SYNTAX                 Counter64
MAX-ACCESS         read-only
STATUS                 current
DESCRIPTION
""
::= {ntopEntry 11 }

pktDuplicatedAckRcvd OBJECT-TYPE
SYNTAX                 Counter64
MAX-ACCESS         read-only
STATUS                 current
DESCRIPTION
""
::= {ntopEntry 12 }

pktBroadcastSent OBJECT-TYPE
SYNTAX                 Counter64
MAX-ACCESS         read-only
STATUS                 current
DESCRIPTION
""
::= {ntopEntry 13 }

bytesBroadcastSent OBJECT-TYPE
SYNTAX                 Counter64
MAX-ACCESS         read-only
STATUS                 current
DESCRIPTION
""
::= {ntopEntry 14}
```

```
pktMulticastSent OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 15}

bytesMulticastSent OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 16}

pktMulticastRcvd OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 17}

bytesMulticastRcvd OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 18}


bytesSent OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 19}

bytesSentLoc OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 20}

bytesSentRem OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 21}

bytesSentSession OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 22}
```

```
bytesRcvd OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 23}

bytesRcvdLoc OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 24}

bytesRcvdFromRem OBJECT-TYPE
SYNTAX                 Counter64
MAX-ACCESS         read-only
STATUS                 current
DESCRIPTION
""
::= {ntopEntry 25}

bytesRcvdSession OBJECT-TYPE
SYNTAX                 Counter64
MAX-ACCESS         read-only
STATUS                 current
DESCRIPTION
""
::= {ntopEntry 26}

numHostSessions OBJECT-TYPE
SYNTAX                 Counter64
MAX-ACCESS         read-only
STATUS                 current
DESCRIPTION
""
::= {ntopEntry 27}

ipBytesSent OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 28}

ipBytesRcvd OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 29}

ipv6Sent OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 30}
```

```
ipv6Rcvd OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 31}

tcpSentLoc OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 32}

tcpSentRem OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 33}

udpSentLoc OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 34}

udpSentRem OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 35}

icmpSent OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 36}

icmp6Sent OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 37}

tcpRcvdLoc OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 38}
```

```
tcpRcvdFromRem OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 39}

udpRcvdLoc OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 40}

udpRcvdFromRem OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 41}

icmpRcvd OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 42}

icmp6Rcvd OBJECT-TYPE
SYNTAX         Counter64
MAX-ACCESS read-only
STATUS         current
DESCRIPTION
""
::= {ntopEntry 43}

tcpFragmentsSent OBJECT-TYPE
SYNTAX                 Counter64
MAX-ACCESS         read-only
STATUS                 current
DESCRIPTION
""
::= {ntopEntry 44}

tcpFragmentsRcvd OBJECT-TYPE
SYNTAX                 Counter64
MAX-ACCESS         read-only
STATUS                 current
DESCRIPTION
""
::= {ntopEntry 45}

udpFragmentsSent OBJECT-TYPE
SYNTAX                 Counter64
MAX-ACCESS         read-only
STATUS                 current
DESCRIPTION
""
::= {ntopEntry 46}
```

```
udpFragmentsRcvd OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 47}

icmpFragmentsSent OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 48}

icmpFragmentsRcvd OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 49}

icmp6FragmentsSent OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 50}

icmp6FragmentsRcvd OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 51}

totContactedSentPeers OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 52}

totContactedRcvdPeers OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 53}

contactedSentPeers OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 54}
```

```
contactedRcvdPeers OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 55}

contactedRouters OBJECT-TYPE
SYNTAX                Counter64
MAX-ACCESS         read-only
STATUS                current
DESCRIPTION
""
::= {ntopEntry 56}


ntopMIBConformance  OBJECT IDENTIFIER ::= { ntop 2 }

ntopMIBGroups     OBJECT IDENTIFIER ::= { ntopMIBConformance 1 }

ntopMIBCompliances  OBJECT IDENTIFIER ::= { ntopMIBConformance 2 }

---compliance statement

ntopMiBCompliance MODULE-COMPLIANCE
STATUS      current
DESCRIPTION
"The compliance statement for entities which
implement the nTOP MIB."

MODULE      -- this module
MANDATORY-GROUPS    { ntopGroup }

::={ ntopMIBCompliances 1}

-- units of conformance

ntopGroup OBJECT-GROUP
OBJECTS   {
ntopSerialType,
ntopSerial,
ntopActualDevice,
pktSent,
pktRcvd,
pktSentSession,
pktRcvdSession,
pktDuplicatedAckSent,
pktDuplicatedAckRcvd,
pktBroadcastSent,
bytesBroadcastSent,
pktMulticastSent,
bytesMulticastSent,
pktMulticastRcvd,
bytesMulticastRcvd,
bytesSent,
bytesSentLoc,
bytesSentRem,
bytesSentSession,
bytesRcvd,
bytesRcvdLoc,
bytesRcvdFromRem,
```

```
bytesRcvdSession,
numHostSessions,
ipBytesSent,
ipBytesRcvd,
ipv6Sent,
ipv6Rcvd,
tcpSentLoc,
tcpSentRem,
udpSentLoc,
udpSentRem,
icmpSent,
icmp6Sent,
tcpRcvdLoc,
tcpRcvdFromRem,
udpRcvdLoc,
udpRcvdFromRem,
icmpRcvd,
icmp6Rcvd,
tcpFragmentsSent,
tcpFragmentsRcvd,
udpFragmentsSent,
udpFragmentsRcvd,
icmpFragmentsSent,
icmpFragmentsRcvd,
icmp6FragmentsSent,
icmp6FragmentsRcvd,
totContactedSentPeers,
totContactedRcvdPeers,
contactedSentPeers,
contactedRcvdPeers,
contactedRouters
}
STATUS    current
DESCRIPTION
"HostTraffic values for a particular serial"
::= { ntopMIBGroups 2 }


END
```

Capitolo 4. Implementazione plugin

Il plugin è un software che permette il collegamento tra un agent SNMP ed Ntop. In particolare il plugin permette l'avvio e l'interruzione di un agent che prende i dati dalle strutture dati interne di Ntop. La realizzazione del plugin è stata fatta per gradi. Prima abbiamo implementato un plugin minimale e poi abbiamo esteso il plugin minimale con il codice per l'agent.

L'implementazione dell'agent è stata la parte più impegnativa del progetto. La tipologia e la quantità di dati da gestire sono stati dei fattori critici che hanno influenzato pesantemente le scelte implementative. L'agent infatti doveva essere pensato per lavorare in situazioni di carico notevoli. Non sono rare le situazioni in cui Ntop riesce ad avere nella sua hash più di 50000 host. Inoltre la tipologia dei dati estremamente variabili nel tempo ha impedito l'utilizzo di molte forme di caching. Gli strumenti messi a disposizione Net-Snmp si sono dimostrati inadeguati per il nostro contesto di utilizzo, pertanto abbiamo dovuto scrivere più codice di quello necessario per implementare agent che lavorano su tabelle più piccole. Nelle sezioni successive saranno discussi i passi fondamentali per la realizzazione del plugin.

Realizzazione di un plugin generico per ntop

Per realizzare un plugin di Ntop abbiamo preso in esame il file *pluginSkeleton.c*. Questo file è uno skeleton di un plugin minimale per ntop e permette di capire come creare nuovi plugin.

Un plugin minimale di ntop deve contenere un array che serve per definire le funzioni che vengono chiamate da Ntop in determinate circostanze. Vediamo come esempio l'array definito in *snmpPlugin.c:*

```
static PluginInfo snmpPluginInfo[] = {
 {
    VERSION,                 /* current ntop version */
   "snmpPlugin",
   "This plugin is used to monitor host traffic using snmp protocol .",
   "0.1",                  /* version */
  "<a href=mailto:fuscof@cli.di.unipi.it>F.Fusco</a>,"
  "  <a href=mailto:giardina@cli.di.unipi.it>G.Giardina</a>",
   "snmpPlugin",
   1,                        /* Active by default */
   1,                        /* Inactive setup */
   initSnmpFunct,          /* InitFunc */
   termSnmpFunct,          /* TermFunc */
   NULL,                   /* PluginFunc */
   handleSnmpHTTPrequest,
   simplehandlePluginHostCreationDeletion,/* host creation/deletion handle */
   NULL,                  /* no capture */
   NULL                        /* no status */
 }
};
```

Le funzioni *initSnmpFunct*, *termsnmpFunct* sono chiamate rispettivamente all'attivazione e alla disattivazione del plugin. La funzione *handleSnmpHTTPRequest* viene chiamata da Ntop ogni volta che si accede alla pagina web del plugin. *simplehandlePluginHostCreationDeletition* viene invece chiamata da Ntop quando viene aggiunto o rimosso dall'hash di Ntop un host. Questa funzione ha come prototipo

```
   simplehandlePluginHostCreationDeletion (HostTraffic * el,
                                           u_short deviceId,
                                           u_char hostCreation);
```

dove hostCreation indica se si tratta di una rimozione o di un inserimento, deviceId indica il device dal quale è stato sniffato l'host e HostTraffic è struttura del traffico per quell'host.

Un plugin deve anche definire la funzione

```
#ifdef MAKE_STATIC_PLUGIN
PluginInfo *
snmpPluginEntryFctn (void)
{
#else
PluginInfo *
PluginEntryFctn (void)
{
#endif
  traceEvent (CONST_TRACE_ALWAYSDISPLAY,
              "SNMP: Welcome to %s. (C) 1999-2004 by "
              "Fusco Francesco , Giardina Giuseppe",
              snmpPluginInfo->pluginName);
  return (snmpPluginInfo);
}
```

che viene chiamata la prima volta che il plugin viene attivato.

Utilizzo di mib2c per la generazione dello skeleton dell'agent

Net-snmp fornisce il tool *mib2c* che semplifica la scrittura del codice di un agent che usi le librerie net-snmp. Con mib2c vengono distribuiti anche dei files di configurazione che permettono di controllare la generazione del codice. L'utilizzo del tool è estremamente semplice. Basta copiare un MIB, in questo caso NTOP-MIB.txt in *~/.snmp/mib* e dare il seguente comando:

```
fuscof@black:~/sgr/ntopAgent$ env MIBS="+NTOP-MIB" mib2c -c \
>/etc/snmp/mib2c.iterate.conf ntop
writing to ntop.h
writing to ntop_columns.h
writing to ntop_enums.h
writing to ntop.c
running indent on ntop_enums.h
running indent on ntop.c
running indent on ntop_columns.h
running indent on ntop.h
fuscof@black:~/sgr/ntopAgent$
```

Il comando *env* è necessario per permettere al tool di riconoscere il MIB. In questo caso è stato usato il files di configurazione *mib2c.iterate.conf*. Questo file permette la creazione di uno skel che è particolarmente utile nel caso si abbiano MIB con tabelle che devono utilizzare dati esterni. In questo caso infatti è sufficiente riempire il codice delle due funzioni:

```
netsnmp_variable_list *
ntopTable_get_first_data_point(void **my_loop_context,
                               void **my_data_context,
                               netsnmp_variable_list * put_index_data,
                               netsnmp_iterator_info *mydata);
netsnmp_variable_list *
ntopTable_get_next_data_point(void **my_loop_context,
```

```
                                void **my_data_context,
                                netsnmp_variable_list * put_index_data,
                                netsnmp_iterator_info *mydata);
```

e modificare poche righe della funzione

```
int
ntopTable_handler(netsnmp_mib_handler *handler,
                  netsnmp_handler_registration *reginfo,
                  netsnmp_agent_request_info *reqinfo,
                  netsnmp_request_info *requests);
```

per avere un agent funzionante che è in grado di gestisce tabelle . Il codice generato in modo automatico si occupa di ottenere un puntatore al dato corrispondente alla richiesta (snmpget o snmpgetnext) inoltrata.

Nell'implementare il nostro agent come plugin di ntop abbiamo però notato che mib2c era si utile, ma non nel nostro caso o almeno non quanto speravamo. Il codice generato in automatico con mib2c è infatti poco efficiente e quindi inutilizzabile per creare un agent che lavora su molti dati. Il problema di performance è evidente quando si deve rispondere a richieste di tipo *snmpgetnext*. In questo caso infatti l'agent prima scorre la struttura dati per trovare il valore corrispondente all'indice passato nella richiesta e poi ordina la struttura in ordine lessicografico per cercare l'elemento successivo.

Per questo motivo abbiamo deciso di non utilizzare un iteratore, ma di implementare delle soluzioni più efficienti per rispondere alle richieste. Gran parte del codice generato in modo automatico da *mib2c* è risultato comunque utile. In particolare abbiamo utilizzato le seguenti funzioni:

```
void initialize_table_ntopTable (void);
```

e la funzione

```
int
ntopTable_handler (netsnmp_mib_handler * handler,
                   netsnmp_handler_registration * reginfo,
                   netsnmp_agent_request_info * reqinfo,
                   netsnmp_request_info * requests);
```

La prima funzione si occupa della definizione degli indici, della registrazione di un *handler* per le richieste e della registrazione della tabella. La seconda funzione si occupa della gestione delle richieste. In particolare, a partire dalla *request* passata come argomento è possibile ricavare il puntatore ad una struct di tipo *netsnmp_table_request_info* attraverso la funzione:

```
table_info = netsnmp_extract_table_info (request);
```

table_info è una struct che permette di ottenere gli indici e la colonna associata alla richiesta. In questo modo è possibile gestire in modo differente le richieste di tipo snmpget da quelle di tipo snmpgetnext.

La funzione

```
static void
processRequest (netsnmp_table_request_info * table_info,
```

```
                                        HostTraffic * traffic);
```

è utilizzata per entrambi i tipi di richiesta per inoltrare le risposte attraverso la rete. Il puntatore *table_info* serve per ottenere gli indici e la colonna richiesti, mentre il puntatorene *traffic* serve per avere i dati corrispondenti alla colonna richiesta.

Implementazione get

Per ottimizzare la snmpget abbiamo sfruttato il fatto che Ntop può trovare in modo estremamente veloce una *HostTraffic* nella sua hash dato il suo *HostSerial*. La funzione che si occupa di questo è la:

```
extern HostTraffic* findHostBySerial(HostSerial serial,
                                     u_int actualDeviceId);
```

definita in util.c.

L'idea è quindi quella di ricavare un HostSerial a partire dalla richiesta fatta per poi eseguire una *findHostBySerial* per trovare i dati di traffico relativi a quell'host. La funzione che si occupa di questo è la funzione:

```
static int getHostSerialFromIndex (netsnmp_table_request_info *
                                   table_info, HostSerial * serial);
```

dove *table_info* è un puntatore che serve per ottenere gli indici della richiesta snmp, mentre *serial* è puntatore alla struct HostSerial che deve essere riempita con i valori presi dall'indice.

Implementazione getnext

Per la gestione della get-next era necessario avere una struttura dati che contenesse i dati ordinati in maniera lessicografica in base agli oid corrispondenti alle HostTraffic. Per questo in principio si era pensato di utilizzare un albero binario di ricerca, ma poi questo è risultato inefficente dal punto di vista dell'utilizzo della memoria, infatti per ogni elemento inserito occupava qualcosa come 20 byte soltanto per puntatori. Allora si è scelto di utilizzare un'array che contenesse i puntatori alle HostTraffic mantenuti ordinati lessicograficamente grazie ad un algoritmo di tipo *insertion sort*. Naturalmente l'array deve essere autoincrementante per poter memorizzare tutti gli host che Ntop monitora.

Quando viene avviato il plugin oltre alla registrazione della tabella e degli altri elementi di snmp vengono anche inizializzate le variabili: *arrayOfHost*,*numberOfHost*, *maxNumberOfHost*. La prima è l'array dei puntatori, la seconda indica quanti host ci sono al momento nell'array e la terza indica il numero massimo di host che può memorizzare al momento l'array, quando numberOfHost sarà uguale ad maxNumberOfHost l'array verrà incrementato. Quando il plugin viene fermato verrà chiamata la funzione *resetData* che libererà tutta la memoria occupata dall'array.

Tramite la funzione *simplehandlePluginHostCreationDeletion* il plugin saprà da Ntop se ci sono nuovi host da memorizzare o da rimuovere. Se ci sono nuovi host da memorizzare viene utilizzata la funzione *addHost*, che implementa appunto l'insertion sort. Questa prima tramite una ricerca lineare cercherà la posione adatta per l'host, poi sposterà i dati da quella posizione in poi avanti di 1 posizione e inserirà nel buco il puntatore al nuovo elemento. In caso di rimozione viene invece utilizzata la funzione *removeHost* che tramite una ricerca binaria cercherà l'host da rimuovere e in seguito sposterà tutti i dati in posizioni successive indietro di 1 posizione. La funzione di ricerca binaria e la ricerca lineare sfruttano rispettivamente la funziona per il confronto tra host, la prima, e per il confronto tra oid e host, la seconda. Entrambe la funzioni di confronto hanno bisogno di costruirsi gli oid degli elementi contenuti nell'array e

questo oid viene costruito a partire dalla struttura *HostSerial* contenuta nella HostTraffic. In questa struttura è infatti descritto a che tipo appartiene l'host(ipv4, ipv6, ethernet, fibre channel) e vi si trova anche l'identificatico dell'host(indirizzo ipv4, indirizzo ipv6, indirizzo MAC, indirizzo FC). La costruzione degli oid avviene facendo una lettura di 8 bit alla volta dell'indirizzo. Una volta ottenuti gli oid questi verranno confrontati con la funzione *compare_oid*.

Quando l'agent riceve una richiesta di tipo snmpgetnext prima di tutto controlla se sono stai passati gli indici. Se non stati passati verrà risposto con il primo host nell'array che rappresenta, appunto, il primo elemento della tabella tramite la funzione *getFirstHostByOid*. Se sono stati passati gli indici verrà fatta una ricerca binaria dell'elemento richiesto. Se l'elemento è presente prima controlliamo che la richiesta non riguardi l'ultima colonna dell'ultimo elemento, in questo caso verrà risposto con un *NOSUCHOBJECT*. Se invece è stata richiesto l'ultimo elemento di una colonna che non sia l'ultima verrà preso il primo elemento e come dato verrà ritornato la colonna successiva a quella richiesta. Nel caso in cui non si tratti né dell'ultimo elemento né dell'ultima colonna verrà recuperato l'elemento successivo a quello richiesto scorrendo semplicemente l'array, e poi passato alla *processRequest* per preparare la risposta. L'accesso all'array dalle funzione è garantito essere in mutua esclusione grazie all'utilizzo di un mutex,*snmpMutex*

Da notare è l'utilizzo della funzione *snmp_set_var_objid* prima della chiamata alla *processRequest*. Questa funzione serve per inserire nella risposta l'intero oid corrispondente all'elemento richiesto.

Compilazione ed esempio di funzionamento

Per compilare il plugin è necessario modificare il file *Makefile.am* in modo che ntop possa utilizzare il plugin come libreria condivisa. Ntop utilizza infatti gli autotools della GNU. Dopo aver modificato il file con la versione distribuita occorre dare i seguenti comandi:

```
automake
autoconf
./configure
make
```

A questo punto Ntop può essere avviato con il supporto a SNMP.

Una volta fatto partire il plugin possono essere fatte le richieste snmp. Per esempio è possibile inoltrare una snmpget con il seguente comando:

```
fuscof@black:~$ snmpget -v 2c -c public localhost \
>1.3.6.1.4.1.30000.1.1.34.2.0.0.4.212.216.172.62
SNMPv2-SMI::enterprises.30000.1.1.34.2.0.0.4.212.216.172.62 = Counter64: 550
```

Capitolo 5. Sviluppi futuri

Attualmente il plugin funziona, ma non è completo. Devono ancora essere gestiti i dispositivi di tipo Fibre Channel nelle richieste di tipo snmpget e deve essere implementato il supporto a più di un device di sniffing. Per aggiungere questa funzionalità molto probabilmente sarà conveniente modificare l'ordine degli indici del MIB in modo tale che si possa fare una *snmpwalk* su un particolare device oppure su una vlan di un particolare device.

Bibliografia

[ntop] *Ntop [http://www.ntop.org]* . Deri Luca.

[netsnmp] *Net-Snmp [http://net-snmp.sf.net]* . .