Simple Network Management Protocol – SNMP

PV177

Petr Holub

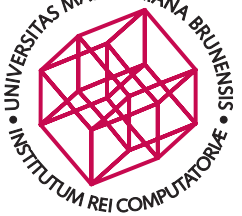
hopet@ics.muni.cz

25. 3. 2014

Historie SNMP

- Potřeba unifikace protokolů pro správu síťových prvků ve druhé polovině 80. let
- 1987 – RFC 1028, Simple Gateway Monitoring Protocol (SGMP)
- 1988 – RFC 1067, SNMPv1
- 1989 – RFC 1098, SNMPv1 (obsoletes RFC 1067)
- 1990 – RFC 1157, SNMPv1 (obsoletes RFC 1098)

Historie SNMP



- 1993 – RFC 1441–RFC 1452, SNMPv2
  - prakticky se příliš neujalo kvůli složité autentizaci
  - přineslo bulk operace
- 1996 – RFC 1901–RFC 1908, SNMPv2c
  - ostraněna složitá autentizace, stalo se *de facto* standardem SNMPv2
- 1996 – RFC 1909–RFC 1910, SNMPv2u
  - bezpečnostní mechanismus přejat do SNMPv3

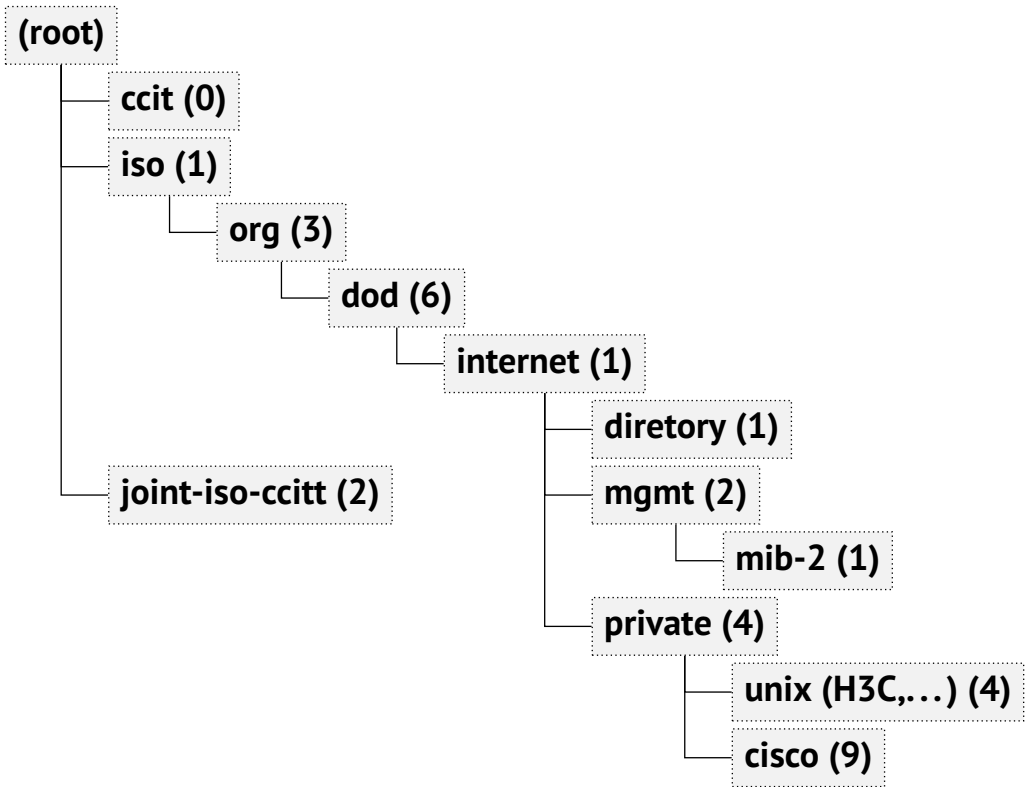
Historie SNMP

- 1999 – RFC 2570, SNMPv3
- 2002 – RFC 3410, SNMPv3 (obsoletes RFC 2570)

Management Information Base (MIB)

- Hierarchická stromová struktura objektů
  - kořen nemá identifikátor
  - důležité větve:
    - ccitt (0)
    - iso (1) – nejpoužívanější
    - joint-iso-ccitt (2)
- Objekt
  - určen pomocí OBJECT IDENTIFIER (OID)
- Vizualizace: např. `http://www.oidview.com/`

Management Information Base (MIB)



Datové typy SNMP

RFC 1151 **SMIv1**

RFC 1442 **SMIv2 – historické**

RFC 2578 **SMIv2**

- **Zápis IP adres**
  - **RFC 2851 (2000) – Textual Conventions for Internet Network Addresses**
  - **RFC 3291 (2002, obsoleted by RFC 4001) – dtto**
  - **RFC 4001 (2005) – dtto**

Datové typy SNMP

Přehled

http://www.logmatrix.com/blog/bid/69744/Understanding-SNMP-Data-Types

Typ	*SMIv1* RFC 1151	*SMIv2* RFC 2578	*Omezení přístupu*	*ASN.1 BER tag*	*Note*
INTEGER	ano (3.2.1.1)	ano (7.1.1)		0x02	32 b se znaménkem, podpora enumerace
Integer32	ne	ano (7.1.1)		0x02	32 b se znaménkem
OCTET STRING	ano (3.2.1.1)	ano (7.1.2)		0x04	
BITS	ne	ano (7.1.4)		0x04	
NULL	ano (3.2.1.1)	ano		0x05	nemá hodnotu
OBJECT IDENTIFIER	ano (3.2.1.1)	ano (7.1.3)		0x06	
IpAddress	ano (3.2.3.2)	ano (7.1.5)		0x40	
Counter	ano (3.2.3.3)	přípustný		0x41	nezáporné, neklesající, 32 b
Counter32	no	ano (7.1.6)	R/O	0x41	dtto
Gauge	ano (3.2.3.4)	přípustný		0x42	nezáporné, 32 b
Gauge32	ne	ano (7.1.7)		0x42	dtto
Unsigned32	ne	ano (7.1.11)		0x42	dtto
TimeTicks	ano (3.2.3.5)	ano (7.1.8)		0x43	dtto
Opaque	ano (3.2.3.6)	deprecated (7.1.9)		0x44	jen pro MIBy SMIv1
Counter64	ne	ano (7.1.10)	R/O	0x46	nezáporné, neklesající, 64 b

Datové typy SNMP

SMIv1

INTEGER

- 32 b celočíselný typ
- v SMIv1 nezáporný„ -2^{31} až $2^{31} - 1$ (specifikováno až v SMIv2)
- příklady rozsahů (v SMIv1 musí být nezáporné!):

```
INTEGER(0..127)
INTEGER(0..10|20|30|50..100)
```

- příklad enumerací:

```
INTEGER{true(1), false(2)}
```

Gauge

- může růst i klesat
- překročí-li definovaný rozsah, zůstane na minimu nebo maximu

Datové typy SNMP

SMIv1

Counter – neklesající, v rozsahu 0 až $2^{32} - 1$.

TimeTiks – Δt, v setinách sekundy, 0 až $2^{32} - 1$.

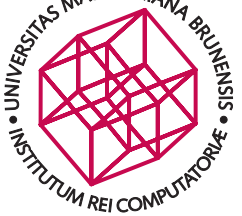
OCTET STRING – byte stream :)

– SMIv1 nespecifikuje délkové omezení, SMIv2 omezuje na 65 535 octetů

– příklad specifikace

```
OCTET STRING (SIZE(0..1023))
```

Datové typy SNMP



SMIv1

OBJECT IDENTIFIER – typ OIDu v MIBu :)

IpAddress – 4 octety, pouze IPv4

NetworkAddress – obecný typ pro adresu
- v SMIv2 nahrazen vylepšenou definicí IpAddress

Opaque – obecný byte stream, kódovaný pomocí ASN.1 BER
- v SMIv2 je zachován z důvodu kompatibility s SMIv1, jinak deprecated

Datové typy SNMP

SMIv2

INTEGER

- 32 b celočíselný typ
- v SMIv1 nezáporný, ale v SMIv2 -2^{31} až $2^{31} - 1$
- příklady rozšíření v SMIv2:

```
INTEGER{lessThan(-1),equal(0),greaterThan(1)}
```

Integer32

Unsigned32

Gauge32

Counter32

Counter64

Datové typy SNMP

SMIv2

BITS – **výčet**
– **příklad:**

```
BITS 1 (TCP), 2(Netware), 3(NetBIOS)
```

InetAddressType, InetAddress, ... – **RFC 4001**
– **typy: unknown, ipv4, ipv6, ipv4z, ipv6z, dns**
– **(z) – typ obsahující zone index (%4d, např.** `fe80::3%eth0`**)**

Datové typy SNMP

- Skalární objekty

```
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (75916362) 8 da
SNMPv2-MIB::sysContact.0 = STRING: hopet@ics.muni.cz
```

- Tabulkové objekty
  - definováno v ASN.1 pomocí konstrukce SEQUENCE OF s 0 nebo více členy
  - každý řádek tabulky je sekvence skalárních objektů
  - tabulky se nemohou vnořovat

Datové typy SNMP

Příklad definice tabulky:

- **ipIfStatsTable, OID 1.3.6.1.2.1.4.31.3**
- **statistiky per interface**
- **Cisco SNMP Object Navigator:**

  `http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en&translate=Translate&objectInput=1.3.6.1.2.1.4.31.3#oidContent`

- **definice v SMIv2:**

  `ftp://ftp.cisco.com/pub/mibs/v2/IP-MIB.my`

Datové typy SNMP

Příklad definice tabulky:

```
ipIfStatsTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF IpIfStatsEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "The table containing per-interface traffic statistics.  This
            table and the ipSystemStatsTable contain similar objects
            whose difference is in their granularity.  Where this table
            contains per-interface statistics, the ipSystemStatsTable
            contains the same statistics, but counted on a system wide
            basis."
    ::= { ipTrafficStats 3 }

ipIfStatsEntry OBJECT-TYPE
    SYNTAX     IpIfStatsEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
           "An interface statistics entry containing objects for a
            particular interface and version of IP."
    INDEX { ipIfStatsIPVersion, ipIfStatsIfIndex }
    ::= { ipIfStatsTable 1 }
```

Datové typy SNMP

Příklad použití tabulky:

`snmpwalk -v1 -cpublic IP_switche 1.3.6.1.2.1.4.31.3`

`(snmpwalk -On -v1 -cpublic IP_switche 1.3.6.1.2.1.4.31.3)`

```
IP-MIB::ipIfStatsInReceives.ipv4.631 = Counter32: 7194405
IP-MIB::ipIfStatsInReceives.ipv4.632 = Counter32: 0
IP-MIB::ipIfStatsInReceives.ipv4.1015 = Counter32: 8
IP-MIB::ipIfStatsInReceives.ipv6.631 = Counter32: 22729
IP-MIB::ipIfStatsInReceives.ipv6.632 = Counter32: 0
IP-MIB::ipIfStatsInReceives.ipv6.1015 = Counter32: 0
IP-MIB::ipIfStatsInOctets.ipv4.631 = Counter32: 58605011
IP-MIB::ipIfStatsInOctets.ipv4.632 = Counter32: 0
IP-MIB::ipIfStatsInOctets.ipv4.1015 = Counter32: 256
IP-MIB::ipIfStatsInOctets.ipv6.631 = Counter32: 1762368
IP-MIB::ipIfStatsInOctets.ipv6.632 = Counter32: 0
IP-MIB::ipIfStatsInOctets.ipv6.1015 = Counter32: 0
IP-MIB::ipIfStatsInHdrErrors.ipv4.631 = Counter32: 0
IP-MIB::ipIfStatsInHdrErrors.ipv4.632 = Counter32: 0
IP-MIB::ipIfStatsInHdrErrors.ipv4.1015 = Counter32: 0
IP-MIB::ipIfStatsInHdrErrors.ipv6.631 = Counter32: 0
IP-MIB::ipIfStatsInHdrErrors.ipv6.632 = Counter32: 0
IP-MIB::ipIfStatsInHdrErrors.ipv6.1015 = Counter32: 0
```

Základní operace v SNMP

GetRequest a SetRequest

- získání nebo nastavení hodnoty
- atomická semantika

```
snmpget -v1 -c public 1.3.6.1.2.1.17.7.1.4.3.1.2
snmpset -v1 -c private 1.3.6.1.2.1.17.7.1.4.3.1.2
```

Základní operace v SNMP

Příklad v Perlu s využitím Net-SNMP binárek

```
   my $SNMPGetCmd = "snmpget -v1 -c " . $switchauth->{'snmprwcommunity'} .
         " " . $switch->{'hostname'} . " ";
2  my $SNMPSetCmd = "snmpset -v1 -c " . $switchauth->{'snmprwcommunity'} .
         " " . $switch->{'hostname'} . " ";
   # dot1qVlanStaticEgressPorts
4  my $VLANOIDrd = "1.3.6.1.2.1.17.7.1.4.3.1.2";
   my $VLANOIDwr = $VLANOIDrd;
6  # dot1qVlanStaticUntaggedPorts
   # my $VLANOIDwr = "1.3.6.1.2.1.17.7.1.4.3.1.4";
8
   open (SNMPGET, "$SNMPGetCmd $VLANOIDrd.$VLAN |");
10 my $SNMPGetOutput = "";
   while (<SNMPGET>) {
12    chomp;
      $SNMPGetOutput .= $_;
14 }
   close (SNMPGET);
```

Základní operace v SNMP

Příklad v Perlu s využitím Net-SNMP binárek

```
$ snmpget -v1 -c public 147.251.54.18 1.3.6.1.2.1.17.7.1.4.3.1.2.54
SNMPv2-SMI::mib-2.17.7.1.4.3.1.2.54 = Hex-STRING:
AF FF FF FF 7F FE FF FF FF FB FF FF DF FF FF FF
FF FF FF FF FF FF FF FF 00 00 00 00 00 00 F0 00
00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00
```

Základní operace v SNMP

Příklad v Perlu s využitím Net-SNMP binárek

```
$SNMPGetOutput =~ s/^.*\s+=\s+Hex-STRING:\s+(\S.*\S)\s*$/$1/
      || die "Unknown format of SNMP response!\n\nOriginal command was
            :\n$SNMPGetCmd $VLANOIDrd.$VLAN\n\nOutput was:\
            n$SNMPGetOutput\n\n";
print $SNMPGetOutput . "\n";

my @portConfig = split /\s+/, $SNMPGetOutput;
my $portArray = "";
for (my $i = 0; $i <= $#portConfig; $i++) {
   $portArray .= unpack ("B8", pack ("H2", $portConfig[$i]));
}

print substr($portArray, 0, 128) . "\n";

foreach $switchport (@switchports) {
   # add/remove port from given setup
   switch ($operation) {
      case "enable"
         { substr($portArray, $switchport - 1, 1) = "1"; }
      case "disable"
         { substr($portArray, $switchport - 1, 1) = "0"; }
      else
         { die "Unknown operation requested!"; }
   }
}
```

Základní operace v SNMP

Příklad v Perlu s využitím Net-SNMP binárek

```
   print substr($portArray, 0, 128) . "\n";
2
   # set ports setup for given VLAN
4  my $portConfig = "";
   for (my $i = 0; $i < length ($portArray); $i += 8) {
6     if ($i != 0) {
         $portConfig .= " ";
8     }
      $portConfig .= unpack ("H2", pack ("B8", substr ($portArray, $i, 8))
           );
10 }
   print $portConfig . "\n";
12
   system ($SNMPSetCmd . " $VLANOIDwr.$VLAN x " . '"' . $portConfig . '"')
        ;
```

Základní operace v SNMP

Příklad v Perlu s využitím Net-SNMP binárek

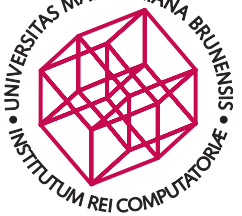
```
   # now do the verification we succeeded with our task
2  open (SNMPGET, "$SNMPGetCmd $VLANOIDrd.$VLAN |");
   $SNMPGetOutput = "";
4  while (<SNMPGET>) {
      chomp;
6     $SNMPGetOutput .= $_;
   }
8  close (SNMPGET);

10 $SNMPGetOutput =~ s/^.*\s+=\s+Hex-STRING:\s+(\S.*\S)\s*$/$1/
         || die "Unknown format of SNMP response!\n\nOriginal command was
               :\n$SNMPGetCmd $VLANOIDrd.$VLAN\n\nOutput was:\
               n$SNMPGetOutput\n\n";
12 print $SNMPGetOutput . "\n";

14 undef @portConfig;
   @portConfig = split /\s+/, $SNMPGetOutput;
16 $portArray = "";
   for (my $i = 0; $i <= $#portConfig; $i++) {
18    $portArray .= unpack ("B8", pack ("H2", $portConfig[$i]));
   }
```

Základní operace v SNMP

Příklad v Perlu s využitím Net-SNMP binárek

```
   print substr($portArray, 0, 128) . "\n";
2
   foreach $switchport (@switchports) {
4     # have we got what we wanted?
      switch ($operation) {
6        case "enable"
            { die "Enabling port $switchport for $VLAN failed!\n" if
                substr($portArray, $switchport - 1, 1) != "1"; }
8        case "disable"
            { die "Disabling port $switchport for $VLAN failed!\n" if
                substr($portArray, $switchport - 1, 1) != "0"; }
10    }
   }
12
   print "Configuration of $switch->{'hostname'} via SNMP-HP finished\n";
```

Základní operace v SNMP



GetNextRequest

- procházení MIBů (walk)

GetBulkRequest

- získání velkého množství hodnot najednou
- zavedeno v SNMPv2

Základní operace v SNMP

GetBulkRequest v Perlu pomocí Net::SNMP

```
#! /usr/local/bin/perl

# http://search.cpan.org/~dtown/Net-SNMP-v6.0.1/lib/Net/SNMP.pm

   use strict;
   use warnings;

   use Net::SNMP qw(:snmp);

   my $OID_ifTable = '1.3.6.1.2.1.2.2';
   my $OID_ifPhysAddress = '1.3.6.1.2.1.2.2.1.6';

   my ($session, $error) = Net::SNMP->session(
      -hostname    => shift || 'localhost',
      -community   => shift || 'public',
      -nonblocking => 1,
      -translate   => [-octetstring => 0],
      -version     => 'snmpv2c',
   );

   if (!defined $session) {
      printf "ERROR: %s.\n", $error;
      exit 1;
   }
```

Základní operace v SNMP

GetBulkRequest v Perlu pomocí Net::SNMP

```
   my %table; # Hash to store the results
2
   my $result = $session->get_bulk_request(
4     -varbindlist    => [ $OID_ifTable ],
      -callback       => [ \&table_callback, \%table ],
6     -maxrepetitions => 10,
   );
8
   if (!defined $result) {
10    printf "ERROR: %s\n", $session->error();
      $session->close();
12    exit 1;
   }
14
   # Now initiate the SNMP message exchange.
16
   snmp_dispatcher();
18
   $session->close();
```

Základní operace v SNMP

GetBulkRequest v Perlu pomocí Net::SNMP

```
   for my $oid (oid_lex_sort(keys %table)) {
2     if (!oid_base_match($OID_ifPhysAddress, $oid)) {
         printf "%s = %s\n", $oid, $table{$oid};
4     } else {
         printf "%s = %s\n", $oid, unpack 'H*', $table{$oid};
6     }
   }
8
   exit 0;
10
   sub table_callback
12 {
      my ($session, $table) = @_;
14
      my $list = $session->var_bind_list();
16
      if (!defined $list) {
18       printf "ERROR: %s\n", $session->error();
         return;
20    }
```

Základní operace v SNMP

GetBulkRequest v Perlu pomocí Net::SNMP

```
1       # Loop through each of the OIDs in the response and assign
        # the key/value pairs to the reference that was passed with
3       # the callback.  Make sure that we are still in the table
        # before assigning the key/values.
5
        my @names = $session->var_bind_names();
7       my $next  = undef;

9       while (@names) {
           $next = shift @names;
11         if (!oid_base_match($OID_ifTable, $next)) {
              return; # Table is done.
13         }
           $table->{$next} = $list->{$next};
15      }
```

Základní operace v SNMP

GetBulkRequest v Perlu pomocí Net::SNMP

```
      # Table is not done, send another request, starting at the last
      # OBJECT IDENTIFIER in the response.  No need to include the
      # calback argument, the same callback that was specified for the
      # original request will be used.

      my $result = $session->get_bulk_request(
         -varbindlist    => [ $next ],
         -maxrepetitions => 10,
      );

      if (!defined $result) {
         printf "ERROR: %s.\n", $session->error();
      }

      return;
   }
```

Základní operace v SNMP

Trap

- push mechanismus – notifikace změn
- možnost ztráty notifikace – SNMP běží nad UDP

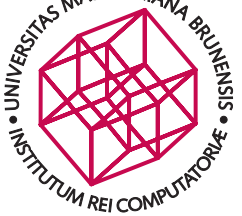
InformRequest

- obdoba Trap s potvrzením doručení

Zajímavé MIBy

- `http://www.net-snmp.org/docs/mibs/`
- `http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en`

Zajímavé MIBy



IF-MIB souhrnná informace o rozhraních

- počet rozhraní: .1.3.6.1.2.1.2.1
- tabulka: .1.3.6.1.2.1.2.2

```
$ snmptable -v1 -c public 147.251.54.18 .1.3.6.1.2.1.2.2
 ifIndex ifDescr         ifType ifMtu    ifSpeed    ifPhysAddress
       ifAdminStatus ifOperStatus   ifLastChange ifInOctets ifInUcastPkts
       ifInNUcastPkts ifInDiscards ifInErrors ifInUnknownProtos
       ifOutOctets ifOutUcastPkts ifOutNUcastPkts ifOutDiscards
       ifOutErrors ifOutQLen                ifSpecific
       1       A1 ethernetCsmacd  9216  100000000 0:1f:fe:fd:fe:ff
                        up           up 16:16:44:43.43  930867842
             1764101          105311            0          0
                              0   364368522        2474468        11666349
                      1337           0         0 SNMPv2-SMI::transmission
             .7
       2       A2 ethernetCsmacd  1500 1000000000 0:1f:fe:fd:fe:fe
                        up           up 16:15:05:28.78  385481416
             659290            1429            0          0
                              0  1278671840         970989          138502
                               0           0         0 SNMPv2-SMI::transmission
             .7
```

Zajímavé MIBy

IF-MIB souhrnná informace o rozhraních

- počet rozhraní: .1.3.6.1.2.1.2.1
- tabulka: .1.3.6.1.2.1.2.2

```
snmpwalk -On -v1 -c public 147.251.54.18 .1.3.6.1.2.1.2.2
...
IF-MIB::ifDescr.3 = STRING: A3
IF-MIB::ifDescr.4 = STRING: A4
...
IF-MIB::ifMtu.4 = INTEGER: 1500
IF-MIB::ifMtu.5 = INTEGER: 9216
...
```

Zajímavé MIBy

IP-MIB souhrnná informace o rozhraních

```
1.3.6.1.2.1 - SNMP MIB-2
1.3.6.1.2.1.4 - ip
1.3.6.1.2.1.4.21 - ipRouteTable
```

```
$ snmpwalk -v1 -c public 147.251.54.18 .1.3.6.1.2.1.4.1
IP-MIB::ipForwarding.0 = INTEGER: forwarding(1)
snmpwalk -v1 -c public 147.251.54.18 1.3.6.1.2.1.4.21
IP-MIB::ip.21.1.1.0.0.0.0 = IpAddress: 0.0.0.0
IP-MIB::ip.21.1.1.67.58.32.0 = IpAddress: 67.58.32.0
IP-MIB::ip.21.1.1.67.58.60.32 = IpAddress: 67.58.60.32
IP-MIB::ip.21.1.1.127.0.0.0 = IpAddress: 127.0.0.0
IP-MIB::ip.21.1.1.127.0.0.1 = IpAddress: 127.0.0.1
IP-MIB::ip.21.1.1.147.251.54.0 = IpAddress: 147.251.54.0
IP-MIB::ip.21.1.2.0.0.0.0 = INTEGER: 631
IP-MIB::ip.21.1.2.67.58.32.0 = INTEGER: 1015
IP-MIB::ip.21.1.2.67.58.60.32 = INTEGER: 1015
...
```

Net-SNMP

snmpget

snmpbulkget

snmptable **tabulkový výpis tabulek :-).**

snmpdelta **sledování rozdílů counterů.**

```
$ snmpdelta -c public -v 1 -Cs localhost IF-MIB::ifInUcastPkts.3 IF-MIB::ifOutUcastPkts.3
[20:15:43 6/14] ifInUcastPkts.3 /1 sec: 158
[20:15:43 6/14] ifOutUcastPkts.3 /1 sec: 158
[20:15:44 6/14] ifInUcastPkts.3 /1 sec: 184
```

snmpset

snmpwalk

snmpbulkwalk

snmptrap

snmpinform

Programování se SNMP

Poslání Trap v Perlu

```
use strict;
use Net::SNMP qw(:ALL);

my ($session, $error) = Net::SNMP->session(
   -hostname  => $ARGV[0] || 'localhost',
   -community => $ARGV[1] || 'public',
   -port      => SNMP_TRAP_PORT,      # Need to use port 162
);

if (!defined($session)) {
   printf("ERROR: %s.\n", $error);
   exit 1;
}

my $result = $session->trap(
   -enterprise   => '1.3.6.1.4.1',
   -agentaddr    => '10.10.1.1',
   -generictrap  => WARM_START,
   -specifictrap => 0,
   -timestamp    => 12363000,
   -varbindlist  => [
      '1.3.6.1.2.1.1.1.0', OCTET_STRING, 'Hub',
      '1.3.6.1.2.1.1.5.0', OCTET_STRING, 'Closet Hub'
   ]
);
```

Programování se SNMP

Poslání Trap v Perlu

```
1  if (!defined($result)) {
       printf("ERROR: %s.\n", $session->error());
3  } else {
       printf("Trap-PDU sent.\n");
5  }

7  ## A second trap example using mainly default values

9  my @varbind = ('1.3.6.1.2.1.2.2.1.7.0', INTEGER, 1);

11 $result = $session->trap(-varbindlist  => \@varbind);

13 if (!defined($result)) {
       printf("ERROR: %s.\n", $session->error());
15 } else {
       printf("Trap-PDU sent.\n");
17 }

19 $session->close();
```

Programování se SNMP

Trap receiver v Perlu

```
1   #!/usr/bin/perl

3   sub my_receiver {
        print "********** PERL RECEIVED A NOTIFICATION:\n";
5
        # print the PDU info (a hash reference)
7       print "PDU INFO:\n";
        foreach my $k(keys(%{$_[0]})) {
9         printf "  %-30s %s\n", $k, $_[0]{$k};
        }
11
        # print the variable bindings:
13      print "VARBINDS:\n";
        foreach my $x (@{$_[1]}) {
15          printf "  %-30s type=%-2d value=%s\n", $x->[0], $x->[2], $x->[1];
        }
17  }

19  NetSNMP::TrapReceiver::register("all", \&my_receiver) ||
      warn "failed to register our perl trap handler\n";
21
    print STDERR "Loaded the example perl snmptrapd handler\n";
```

Programování se SNMP

Co je v Perlu

Net::SNMP základní balík využívající knihovnu Net-SNMP.

NetSNMP::TrapReceiver přijímání Trap.

Net::SNMP::Util obal nad Net::SNMP, mírně komfortnější práce.

NSNMP::Simple pure Perl SNMP (experimentální).

Programování se SNMP

PySNMP v Pythonu – jednoduchý Get

```
1  from pysnmp.entity.rfc3413.oneliner import cmdgen

3  cmdGen = cmdgen.CommandGenerator()

5  errorIndication, errorStatus, errorIndex, varBinds = cmdGen.getCmd(
       cmdgen.CommunityData('public'),
7      cmdgen.UdpTransportTarget(('demo.snmplabs.com', 161)),
       cmdgen.MibVariable('SNMPv2-MIB', 'sysDescr', 0),
9      lookupNames=True, lookupValues=True
   )
11
   # Check for errors and print out results
13 if errorIndication:
       print(errorIndication)
15 elif errorStatus:
       print(errorStatus)
17 else:
       for name, val in varBinds:
19         print('%s = %s' % (name.prettyPrint(), val.prettyPrint()))
```

Programování se SNMP

PySNMP v Pythonu – asynchronní Get

```
from pysnmp.carrier.asynsock.dispatch import AsynsockDispatcher
from pysnmp.carrier.asynsock.dgram import udp, udp6
from pyasn1.codec.ber import encoder, decoder
from pysnmp.proto import api
from time import time

# Protocol version to use
pMod = api.protoModules[api.protoVersion1]
#pMod = api.protoModules[api.protoVersion2c]
# Build PDU
reqPDU =  pMod.GetRequestPDU()
pMod.apiPDU.setDefaults(reqPDU)
pMod.apiPDU.setVarBinds(
    reqPDU, ( ('1.3.6.1.2.1.1.1.0', pMod.Null('')),
              ('1.3.6.1.2.1.1.3.0', pMod.Null('')) )
    )

# Build message
reqMsg = pMod.Message()
pMod.apiMessage.setDefaults(reqMsg)
pMod.apiMessage.setCommunity(reqMsg, 'public')
pMod.apiMessage.setPDU(reqMsg, reqPDU)
```

Programování se SNMP

PySNMP v Pythonu – asynchronní Get

```
startedAt = time()

def cbTimerFun(timeNow):
    if timeNow - startedAt > 3:
        raise Exception("Request timed out")

def cbRecvFun(transportDispatcher, transportDomain, transportAddress,
              wholeMsg, reqPDU=reqPDU):
    while wholeMsg:
        rspMsg, wholeMsg = decoder.decode(wholeMsg, asn1Spec=pMod.Message())
        rspPDU = pMod.apiMessage.getPDU(rspMsg)
        # Match response to request
        if pMod.apiPDU.getRequestID(reqPDU)==pMod.apiPDU.getRequestID(rspPDU):
            # Check for SNMP errors reported
            errorStatus = pMod.apiPDU.getErrorStatus(rspPDU)
            if errorStatus:
                print(errorStatus.prettyPrint())
            else:
                for oid, val in pMod.apiPDU.getVarBinds(rspPDU):
                    print('%s = %s' % (oid.prettyPrint(), val.prettyPrint()))
            transportDispatcher.jobFinished(1)
    return wholeMsg
```

Programování se SNMP

PySNMP v Pythonu – asynchronní Get

```
transportDispatcher = AsynsockDispatcher()
transportDispatcher.registerRecvCbFun(cbRecvFun)
transportDispatcher.registerTimerCbFun(cbTimerFun)
# UDP/IPv4
transportDispatcher.registerTransport(
    udp.domainName, udp.UdpSocketTransport().openClientMode())
# Pass message to dispatcher
transportDispatcher.sendMessage(
    encoder.encode(reqMsg), udp.domainName, ('demo.snmplabs.com', 161))
transportDispatcher.jobStarted(1)
## UDP/IPv6 (second copy of the same PDU will be sent)
transportDispatcher.registerTransport(
    udp6.domainName, udp6.Udp6SocketTransport().openClientMode())
# Pass message to dispatcher
transportDispatcher.sendMessage(
    encoder.encode(reqMsg), udp6.domainName, ('::1', 161)
)
transportDispatcher.jobStarted(1)
# Dispatcher will finish as job#1 counter reaches zero
transportDispatcher.runDispatcher()
transportDispatcher.closeDispatcher()
```

Programování bez SNMP :-)

- Proč?
  - chyby v implementaci SNMP na daném zařízení
  - potřeba řídit zápisy konfigurace
- CLI
  - většinou přes SSH (nebo hůře Telnet)
  - využití dialogového mechanismu typu Expect (`http://www.tcl.tk/man/expect5.31/expect.1.html`)
  - využití SSH knihoven
    - C/C++: `http://www.libssh2.org/`
    - Perl: `use Net::SSH2;`
    - Python:

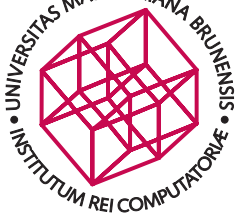
```
from twisted.conch.ssh import transport, keys, userauth,
connection, channel, session
```

SSH CLI v Pythonu


```
class SSHChannel(ssh.SSHChannel):

    name = 'session'

    def __init__(self, conn):
        ssh.SSHChannel.__init__(self, conn=conn)

        self.data = ''

        self.wait_defer = None
        self.wait_data  = None
```

Příklad z: Jensen H. T., OpenNSA

SSH CLI v Pythonu




```
    @defer.inlineCallbacks
    def sendCommands(self, commands, enable_password):
        LT = '\r' # line termination

        try:
            log.msg('Requesting shell for sending commands', debug=True, system=LOG_SYSTEM
            term = os.environ.get('TERM', 'xterm')
       winSize = (25,80,0,0)
       ptyReqData = session.packRequest_pty_req(term, winSize, '')
            yield self.conn.sendRequest(self, 'pty-req', ptyReqData, wantReply=1)
            yield self.conn.sendRequest(self, 'shell', '', wantReply=1)
            log.msg('Got shell', system=LOG_SYSTEM, debug=True)

            d = self.waitForData('>')
            yield d
            log.msg('Got shell ready', system=LOG_SYSTEM, debug=True)

            # so far so good

            d = self.waitForData(':')
            self.write(COMMAND_ENABLE + LT) # This one fails for some reason
            yield d
            log.msg('Got enable password prompt', system=LOG_SYSTEM, debug=True)
```

Příklad z: Jensen H. T., OpenNSA

SSH CLI v Pythonu


```
            d = self.waitForData('#')
            self.write(enable_password + LT)
            yield d

            log.msg('Entered enabled mode', debug=True, system=LOG_SYSTEM)

            d = self.waitForData('#')
            self.write(COMMAND_CONFIGURE + LT) # This one fails for some reason
            yield d

            log.msg('Entered configure mode', debug=True, system=LOG_SYSTEM)

            for cmd in commands:
                log.msg('CMD> %s' % cmd, debug=True, system=LOG_SYSTEM)
                d = self.waitForData('#')
                self.write(cmd + LT)
                yield d
```

Příklad z: Jensen H. T., OpenNSA

SSH CLI v Pythonu


```
            log.msg('Configuration done, writing configuration.', debug=True, system=LOG_S
            d = self.waitForData('#')
            self.write(COMMAND_WRITE + LT)
            yield d

            log.msg('Configuration written. Exiting.', debug=True, system=LOG_SYSTEM)
            self.write(COMMAND_EXIT + LT)
            # Waiting for the prompt removed by hopet - we could wait forever here! :(

        except Exception, e:
            log.msg('Error sending commands: %s' % str(e))
            raise e

        log.msg('Commands successfully send', system=LOG_SYSTEM)
        self.sendEOF()
        self.closeIt()
```

Příklad z: Jensen H. T., OpenNSA

SSH CLI v Pythonu


```
    def waitForData(self, data):
        self.wait_data  = data
        self.wait_defer = defer.Deferred()
        return self.wait_defer


    def dataReceived(self, data):
        log.msg("DATA:" + data, system=LOG_SYSTEM, debug=True)
        if len(data) == 0:
            pass
        else:
            self.data += data
            if self.wait_data and self.wait_data in self.data:
                d = self.wait_defer
                self.data       = ''
                self.wait_data  = None
                self.wait_defer = None
                d.callback(self)
```

Příklad z: Jensen H. T., OpenNSA

SSH CLI v Perlu


```
1  sub config_ssh_forceten($$$$$) {
       my $switch = shift;
3      my $switchauth = shift;
       my $VLAN = shift;
5      my $operation = shift;
       my @switchports = @_;
7
       my $switchport;
9      my $switchportname;

11     print "Configuring $switch->{'hostname'} using Force10 SSH template.\n";

13     my $ssh2 = Net::SSH2->new();
       print "Passed new\n";
15
       $ssh2->connect($switch->{'hostname'}) or die;
17     print "Passed connect\n";
```

Příklad z: Holub P., SBF

SSH CLI v Perlu


```
   if ($ssh2->auth_password($switchauth->{'sshlogin'},$switchauth->{'sshpassword'})) {
2     print "Passed auth\n";
      my $channel = $ssh2->channel();
4     $channel->shell();
      print "Got shell\n";
6     print "Putting channel to O_NONBLOCK mode\n";
      $channel->blocking(0);
8     print $channel "enable\n";
      print $channel "$switchauth->{'enapassword'}\n";
10    print $channel "conf\n";
      print $channel "int vlan $VLAN\n";
12    foreach $switchport (@switchports) {
         $switchportname = $switch->{'portmapping'}->{'port'}->{$switchport}->{'content'};
14       switch ($operation) {
            case "enable"
16             { print $channel "tagged " . $switchportname . "\n"; }
            case "disable"
18             { print $channel "no tagged " . $switchportname . "\n"; }
            else
20             { $channel->close; die "Unknown operation requested!"; }
         }
22    }
```

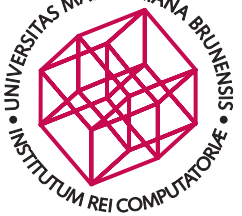
Příklad z: Holub P., SBF

SSH CLI v Perlu


```
1       print $channel "exit\n";
        print $channel "exit\n";
3       # now read and print the output we have got so far
        my $response = &get_n_prompts (\$channel, 6);
5       print $response;
        # last step is to verify that the configration has succeeded
7       foreach $switchport (@switchports) {
           $switchportname = $switch->{'portmapping'}->{'port'}->{$switchport}->{'content'};
9          print $channel "show vlan id $VLAN\n";
           $response = &get_n_prompts (\$channel, 1);
11         print $response, "\n";
           switch ($operation) {
13            case "enable"
                 { die "Enabling port $switchportname for $VLAN failed!\n" if $response !~ /
15            case "disable"
                 {
17               # this is more tricky, we need to check that we got the
                 # response at all
19               die "Disabled port $switchportname for $VLAN failed!\n" if $response !~ /^\
                 }
21         }
        }
```

Příklad z: Holub P., SBF

SSH CLI v Perlu




```
1       print $channel "logout\n";
        $channel->wait_closed;
3       print "Closing SSH channel\n";
        $channel->close;
5    } else {
        die "SSH authentication failed!";
7    }

9    print "Configuration of $switch->{'hostname'} via SSH-Force10 finished\n";
  }
```

- Součástí není záměrně `write mem` – provádí se agregovaně po celé dávce konfigurace.

Příklad z: Holub P., SBF

Problematika zápisu do zařízení

- SNMP
  - zápis do nevolatitní paměti se provádí po každé write operaci
  - v případě dlouhodobých častých změn může dojít k „dožití“ paměti
- CLI
  - záleží na sémanitce konkrétního CLI
  - většinou se zápis zadává explicitně
  - v případě velkého množství průběžných změn možnost transakčního přístupu:
    1. okamžité zadávání příkazů do zařízení bez `write mem`
    2. perzistnentní udržování informací o nezapsaných konfiguračních krocích na straně aplikace
    3. opakované provedení nezapsaných kroků v případě restartu síťového zařízení