Chương **5**

Lập trình ứng dụng SNMP với Delphi 2010

- Chuẩn bị lập trình các phần mềm SNMP
- Thiết kế phần mềm nhận trap : SNMP Trap Receiver.
- Thiết kế phần mềm giám sát lưu lượng thiết bị : SNMP Traffic Monitor
- Thiết kế phần mềm SNMP agent cho Windows server, hỗ trợ lấy các thông tin tự tạo.
- Abstract Syntax Notation One (ASN.1)

Chương này tác giả sẽ trình bày cách viết các phần mềm SNMP sử dụng ngôn ngữ lập trình Delphi phiên bản 14 (Delphi 2010). Mục đích chương này là trình bày ý tưởng từng bước trong việc viết các ứng dụng SNMP hơn là trình bày cú pháp lập trình SNMP trên Delphi. Trên các ngôn ngữ khác thì chỉ khác nhau ở các hàm khởi tạo, gửi, nhận bản tin, còn về trình tự thực hiện thì vẫn giống nhau. Các đoạn code của tác giả được viết ở mức đơn giản để có thể dễ dàng đọc hiểu và chuyển đổi, nên chúng không phải là khuôn mẫu có tốc độ cao nhất hay bẫy lỗi tốt nhất.

Source code của toàn bộ các project có thể download tại trang chủ của quyển tài liệu này.

Tại sao bạn cần phải lập trình SNMP ?

Nhiều thiết bị, ứng dụng được các hãng thiết kế mib riêng, bạn không thể giám sát chúng bằng ứng dụng snmp thông thường. Bạn có thể dùng phần mềm của chính hãng thiết bị đó để giám sát. Nhưng nếu bạn có nhiều chủng loại thiết bị khác nhau thì bắt buộc bạn phải dùng từng phần mềm riêng. Bây giờ làm thế nào để dùng một ứng dụng duy nhất để giám sát tất cả chúng ? Lúc này bạn cần biết cách lập trình ứng dụng giám sát SNMP. Cũng có một số phần mềm cho phép giám sát "custom mib" nhưng chưa hẳn chúng đã đáp ứng hoàn toàn nhu cầu của bạn.

Các thiết bị gửi các event dạng trap đến một trap host. Định nghĩa trap chuẩn chỉ có một số event rất nghèo nàn, do đó các dòng sản phẩm khác nhau đều có định nghĩa rất nhiều trap enterpriseSpecific mà phải dùng sản phẩm của chính hãng mới có thể đọc được. Nếu bạn có file mib mô tả event của các thiết bị, làm thế nào để dùng một ứng dụng duy nhất để làm host nhận event và cảnh báo cho tất cả các chủng loại thiết bị ? Lúc này bạn cần biết cách lập trình ứng dụng SNMP Trap receiver.

Giả sử bạn viết một ứng dụng nào đó, ứng dụng này chạy trên rất nhiều server. Người quản trị cần giám sát hiệu năng ứng dụng của bạn trên tất cả các server mà không cần phải truy cập vào từng server để lấy thông tin. Bạn có thể thiết kế giao thức và phần mềm giám sát riêng, nhưng nếu sử dụng SNMP thì người dùng có thể dùng các phần mềm có sẵn tính năng "custom mib" như Solarwinds để giám sát ứng dụng của bạn. Lúc này bạn cần biết cách lập trình ứng dụng SNMP Agent để bổ sung tính năng này vào ứng dụng của bạn.

Nếu bạn không phải là người phát triển ứng dụng, hoặc việc dùng các phần mềm giám sát có sẵn đã đáp ứng được nhu cầu công việc thì bạn không cần phải đọc chương này.

1. Chuẩn bị lập trình SNMP

Delphi 2010

Delphi là ngôn ngữ lập trình hướng đối tượng, cú pháp giống với Pascal. Môi trường lập trình Delphi hỗ trợ thiết kế form dạng kéo thả tương tự như Visual Studio. Các đối tượng giao diện được đóng gói gọi là VCL (Visual Component Library), tương tự như Controls trong C# hay Java Beans của Java. Có hàng trăm component trong Delphi, và chúng hỗ trợ Unicode hoàn toàn. Ứng dụng của Delphi viết ra là ứng dụng native Windows nên không sử dụng .NET Framework.

Bạn cần cài đặt Delphi 2009 hoặc 2010 để viết các ứng dụng SNMP.

Indy Project

Indy là một bộ thư viện các component hỗ trợ lập trình mạng ở mức application (layer 7 trong mô hình OSI), nghĩa là những gì ứng dụng của bạn phải xử lý là phần data sau khi tách hết các header của các giao thức lớp application. Indy hỗ trợ hầu hết các giao thức phổ biến như : TCP, UDP, IPMulticast, DNS, Echo, FTP, HTTP, IMAP4, SMTP, POP3, Telnet, ICMP, Syslog, SNMP, ... Bạn có thể viết một web server chỉ với vài dòng lệnh.

Indy là một dự án mã nguồn mở được tích hợp vào Delphi. Mã nguồn Indy được viết bằng Delphi bởi các tình nguyện viên. [1]

Nếu không sử dụng thư viện Indy để viết ứng dụng mạng, bạn có thể sử dụng các component có sẵn trong Delphi là TTCPServer, TTCPClient, TUDPSocket để thay thế. Tuy nhiên lúc này bạn phải tự viết phần mã xử lý dữ liệu ở các lớp cao hơn.

Và nếu không muốn dùng những component của Delphi nữa thì bạn có thể dùng các hàm Windows API trong thư viện Winsock.

[1] Trang chủ Indy Project : http://www.indyproject.org

Patch Indy Tiburon

Bộ Indy kèm theo Delphi 2010 có lỗi trong component IdSNMP, bạn cần update lên phiên bản mới nhất là Indy Tiburon để vá lỗi, nếu không IdSNMP sẽ hoạt động sai. Indy Tiburon có thể được download tại trang chủ quyển tài liệu này hoặc tại link gốc http://indy.fulgan.com/ZIP/

Quá trình update thực chất là xóa bản Indy trong Delphi 2010 và thay thế bằng bản Indy Tiburon, trình tự update như sau :

+ Giải nén bản Indy mới ra folder IndyTiburon.

+ Khởi động Delphi 2010.

+ Mở và biên dịch package IndyTiburon\Lib\System\IndySystem140.dpk.

+ Biên dịch package IndyTiburon\Lib\Core\IndyCore140.dpk.

+ Biên dịch package IndyTiburon\Lib\Core\dclIndyCore140.dpk.

+ Biên dịch package IndyTiburon\Lib\Protocols\IndyProtocols140.dpk.

+ Biên dịch package IndyTiburon\Lib\Protocols\dclIndyProtocols140.dpk.

+ Tắt Delphi.

+ Xóa tất cả file trong C:\Program Files\Embarcadero\RAD Studio\7.0\lib\Indy10

+ Copy tất cả các file *.dcu trong folder IndyTiburon (kể cả trong subfolder) sang C:\Program Files\Embarcadero\RAD Studio\7.0\lib\Indy10 (có khoảng 325 file).

+ Copy 5 file *.bpl trong C:\Users\Public\Documents\RAD Studio\7.0\Bpl (Win 7) hoặc C:\Documents and Settings\All Users\Documents\RAD Studio\7.0\Bpl (Win XP) và ghi đè vào các file trong folder C:\Program Files\Embarcadero\RAD Studio\7.0\bin.

+ Khởi động lại Delphi.

Lưu ý : phiên bản Indy tại thời điểm viết tài liệu này là 10.5.5, là một phiên bản vẫn chưa hỗ trợ SNMPv2c. Do đó các ứng dụng được viết chỉ hoạt động đúng với SNMPv1.

2. SNMP Traffic Monitor

Giới thiệu

SNMP Traffic Monitor là phần mềm giám sát liên tục lưu lượng của interface trên thiết bị.

Các tính năng demo bao gồm :

+ Lấy được các thông tin mô tả thiết bị (nhóm mib-2.system).

+ Lấy danh sách các interface và cho phép người dùng chọn 1 interface để giám sát.

+ Cho phép chọn các chu kỳ lấy mẫu khác nhau.

+ Vẽ lưu lượng ra biểu đồ, 2 đường lưu lượng in/out riêng, tự động zoom biểu đồ khi lưu lượng tăng.

Do chỉ là demo tập trung vào SNMP nên phần mềm có các hạn chế sau :

+ Không giám sát được cùng lúc nhiều interface.

+ Không ghi nhớ kết quả giám sát, không in được biểu đồ.

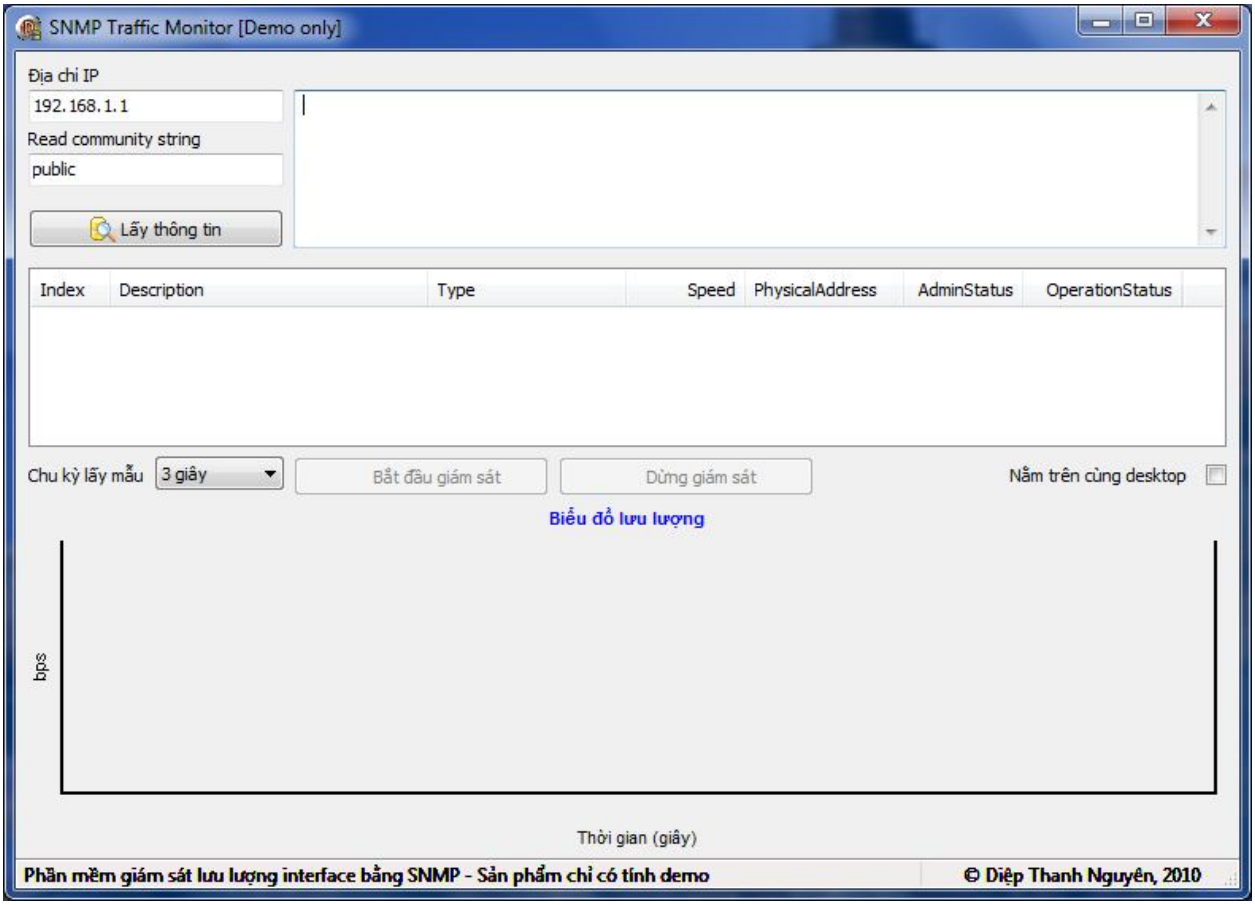
Ý tưởng thực hiện

Để lấy thông tin về hệ thống (tên, mô tả, thời gian hoạt động, ...) ta lấy tất cả OID nằm dưới .iso.org.dod.internet.mgmt.mib-2.system (.1.3.6.1.2.1.1).

Bản thân thiết bị không cung cấp thông tin về tốc độ lưu lượng của interface nên ta không thể lấy trực tiếp bằng SNMP. Ta phải lấy tổng số byte mà interface đã nhận tại OID .iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifInOctets (.1.3.6.1.2.1.2.2.1.10) và tổng số byte đã truyền tại ifOutOctets (.1.3.6.1.2.1.2.2.1.16). Việc lấy thông tin được thực hiện liên tục và tính (giá_trị_sau – giá_trị_trước)/thời_gian_giữa_các_lần_lấy_mẫu để có được tốc độ lưu lượng của interface.

Để không ảnh hưởng đến chương trình chính, phần code quét lưu lượng liên tục được đặt trong một thread.

Thiết kế giao diện



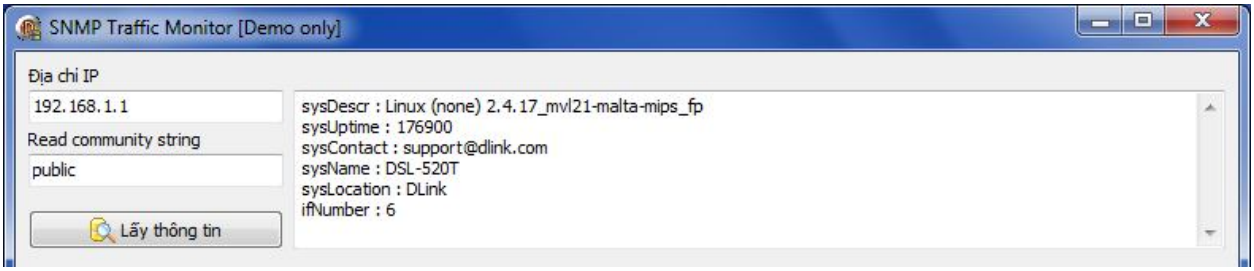
Lấy thông tin về thiết bị

Sau khi nhập IP và community string, nhấn nút "Lấy thông tin" để phần mềm lấy về các thông tin của thiết bị trong nhóm mib-2.system như name, description, contact, ….

```
mmInfo.Clear;
SNMP.Host := edHost.Text;
SNMP.Community := edCommunity.Text;
SNMP.ReceiveTimeout := 1000; // timeout = 1000 ms
// Bước 1 : dùng hàm QuickSend để lấy các thông tin thuộc group mib-2.system
// một số thiết bị không hỗ trợ sub-id nên ta lấy cả 2 object sysDescr và sysDescr.0
// hàm QuickSend sẽ trả về TRUE nếu lấy thông tin thành công, FALSE nếu timeout
// kết quả lấy về là một chuỗi chứa trong biến v
s := 'sysDescr : '; // sysDescr có oid là 1.3.6.1.2.1.1.1
if SNMP.QuickSend('1.3.6.1.2.1.1.1', SNMP.Community, SNMP.Host, v) or
   SNMP.QuickSend('1.3.6.1.2.1.1.1.0', SNMP.Community, SNMP.Host, v) then s := s + v;
mmInfo.Lines.Add(s); // xuất kết quả ra mmInfo
s := 'sysUptime : ';
if SNMP.QuickSend('1.3.6.1.2.1.1.3', SNMP.Community, SNMP.Host, v) or
   SNMP.QuickSend('1.3.6.1.2.1.1.3.0', SNMP.Community, SNMP.Host, v) then s := s + v;
mmInfo.Lines.Add(s);
s := 'sysContact : ';
if SNMP.QuickSend('1.3.6.1.2.1.1.4', SNMP.Community, SNMP.Host, v) or
   SNMP.QuickSend('1.3.6.1.2.1.1.4.0', SNMP.Community, SNMP.Host, v) then s := s + v;
mmInfo.Lines.Add(s);
s := 'sysName : ';
if SNMP.QuickSend('1.3.6.1.2.1.1.5', SNMP.Community, SNMP.Host, v) or
   SNMP.QuickSend('1.3.6.1.2.1.1.5.0', SNMP.Community, SNMP.Host, v) then s := s + v;
mmInfo.Lines.Add(s);
s := 'sysLocation : ';
if SNMP.QuickSend('1.3.6.1.2.1.1.6', SNMP.Community, SNMP.Host, v) or
```

```
    SNMP.QuickSend('1.3.6.1.2.1.1.6.0', SNMP.Community, SNMP.Host, v) then s := s + v;
  mmInfo.Lines.Add(s);
```

Các thông tin lấy được sẽ được xuất ra màn hình như ví dụ sau



Lấy số lượng interface của thiết bị

```
// Bước 2 : lấy tổng số interface đang có trên thiết bị
s := 'ifNumber : ';
if SNMP.QuickSend('1.3.6.1.2.1.2.1', SNMP.Community, SNMP.Host, v) or
   SNMP.QuickSend('1.3.6.1.2.1.2.1.0', SNMP.Community, SNMP.Host, v) then s := s + v;
mmInfo.Lines.Add(s);
// nếu số ifNumber là rỗng hoặc không phải là kiểu số thì gán = 0
if not TryStrToInt(v, ifNumber) then ifNumber := 0;
```

Lấy danh sách interface index

```
// Bước 3 : Lần lượt lấy index của các interface entry, bằng cách GetNext liên tục, bắt
đầu từ ifIndex
SNMP.Query.Host := edHost.Text;
SNMP.Query.Community := edCommunity.Text;
SNMP.Query.Port := 161;
SNMP.Query.Version := 0;  // sử dụng SNMP v1 (0=v1; 1=v2c)
SNMP.Query.PDUType := PDUGetNextRequest;
List.Clear;
// ifIndex có OID là 1.3.6.1.2.1.2.1.1, khi GetNext sẽ lấy được ifIndex của interface
đầu tiên
OID := '1.3.6.1.2.1.2.1.1';
// vì tổng số interface của thiết bị = ifNumber nên ta lặp ifNumber lần
for i := 1 to ifNumber do
begin
  SNMP.Query.MIBOID.Clear; // đầu tiên xóa sanh sách OID cần query
  SNMP.Query.MIBOID.Add(OID); // sau đó thêm ifIndex của interface cần query
  j := 1;
  while (j<=3) and (SNMP.SendQuery=False) do inc(j); // nếu fail thì cho phép lặp lại
đến lần thứ 3
  if j <= 3 then begin
    L := List.Items.Add;  // thêm 1 dòng mới vào danh sách interface
    L.Caption := SNMP.Reply.MIBValue[0]; // gán ifIndex vào Caption của ListItem
    { đặt OID = ifIndex của interface hiện tại, vòng lặp sau đó sẽ GetNext để
     lấy ifIndex của interface tiếp theo }
    OID := SNMP.Reply.MIBOID[0];
  end;
end;
```

Lấy thông tin của interface

```
// Bước 4 : lấy các thông tin khác của từng interface
SNMP.Query.PDUType := PDUGetRequest;
for i := 0 to List.Items.Count - 1 do
```

```
begin
  L := List.Items[i];
  { đưa các oid cần lấy vào query, sau đó thực hiện 1 GetRequest để lấy toàn bộ
    thông tin của 1 interface cùng lúc }
  SNMP.Query.MIBOID.Clear;
  SNMP.Query.MIBOID.Add('1.3.6.1.2.1.2.2.1.2.' + L.Caption); // ifDescr
  SNMP.Query.MIBOID.Add('1.3.6.1.2.1.2.2.1.3.' + L.Caption); // ifType
  SNMP.Query.MIBOID.Add('1.3.6.1.2.1.2.2.1.5.' + L.Caption); // ifSpeed
  SNMP.Query.MIBOID.Add('1.3.6.1.2.1.2.2.1.6.' + L.Caption); // ifPhysAddress
  SNMP.Query.MIBOID.Add('1.3.6.1.2.1.2.2.1.7.' + L.Caption); // ifAdminStatus
  SNMP.Query.MIBOID.Add('1.3.6.1.2.1.2.2.1.8.' + L.Caption); // ifOperStatus
  // nếu fail thì cho phép lặp lại đến lần thứ 3
  j := 1;
  while (j<=3) and (SNMP.SendQuery=False) do inc(j);
  if j <= 3 then
  try
    L.SubItems.Add(SNMP.Reply.MIBValue[0]); // ifDescr là kiểu OctetString
    t := StrToInt(SNMP.Reply.MIBValue[1]); // ifType là kiểu INTEGER
    if t <= Length(ifTypeArray) then L.SubItems.Add(ifTypeArray[t])
                                else L.SubItems.Add(InttoStr(t));
    // ifSpeed là kiểu GAUGE nên phải chia 100
    L.SubItems.Add(InttoStr(StrToInt(SNMP.Reply.MIBValue[2]) div 100));
    // ifPhysAddress là kiểu PhysicalAddress nên phải chuyển đổi thành chuỗi đọc được
    L.SubItems.Add(PhysAddressToStr(SNMP.Reply.MIBValue[3]));
    t := StrToInt(SNMP.Reply.MIBValue[4]); // ifAdminStatus, kiểu INTEGER
    if t <= Length(ifStatusArray) then L.SubItems.Add(ifStatusArray[t])
                                  else L.SubItems.Add(InttoStr(t));
    t := StrToInt(SNMP.Reply.MIBValue[5]); // ifOperStatus, kiểu INTEGER
    if t <= Length(ifStatusArray) then L.SubItems.Add(ifStatusArray[t])
                                  else L.SubItems.Add(InttoStr(t));
  except
  end;
  { thông thường tất cả các oid nằm trong request đều được snmp agent trả về đầy
  đủ và đúng thứ tự, đoạn code trên không kiểm tra số lượng value trả về và không
  kiểm tra thứ tự }
```

Danh sách các interface xuất ra màn hình như ví dụ sau

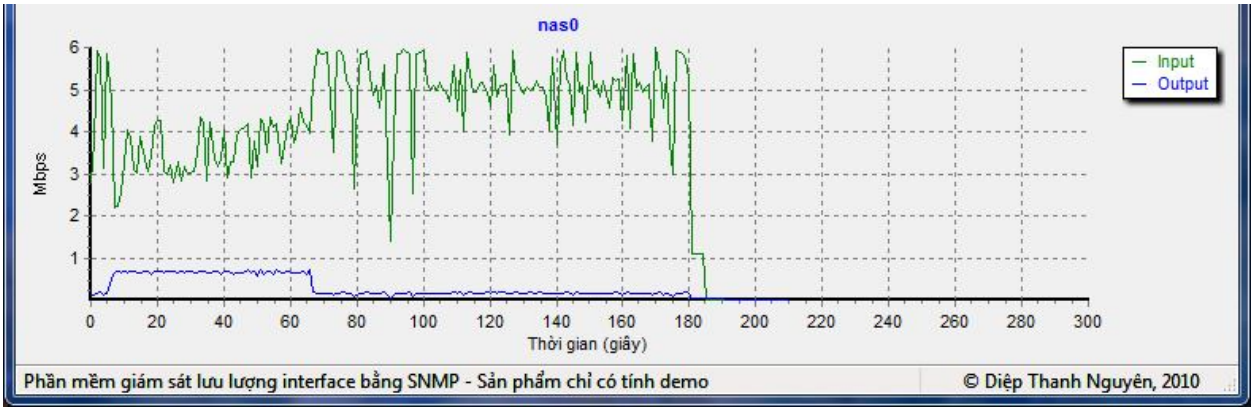
Index	Description	Type	Speed	PhysicalAddress	AdminStatus	OperationStatus
1	lo	softwareLoopback(24)	100000		up(1)	up(1)
2	eth0	ethernetCsmacd(6)	100000	0019.5B7D.0EAA	up(1)	up(1)
3	nas0	ethernetCsmacd(6)	100000	0019.5B7D.0EAA	up(1)	up(1)
4	br0	ethernetCsmacd(6)	0	0019.5B7D.0EAA	up(1)	up(1)
5	br1	ethernetCsmacd(6)	100000		up(1)	up(1)

Quét lưu lượng

Sau khi đã có danh sách các interface index, bạn click vào một interface trong danh sách và nhấn nút "Bắt đầu giám sát", phần mềm sẽ tạo snmp query lấy thông tin ifInOctets và ifOutOctets của interface index đang chọn, quét liên tục theo chu kỳ. Đoạn code như sau :

```
  snmp.Query.PDUType := PDUGetRequest;
  snmp.Query.MIBOID.Clear;
  // ifInOctets có oid là 1.3.6.1.2.1.2.2.1.10, dòng đang được chọn trong List là đối
tượng List.Selected, caption của dòng này chính là interface index
  snmp.Query.MIBOID.Add('1.3.6.1.2.1.2.2.1.10.' + List.Selected.Caption);
  snmp.Query.MIBOID.Add('1.3.6.1.2.1.2.2.1.16.' + List.Selected.Caption);
  // gửi snmp query, lấy 2 item ifInOctets và ifOutOctets cùng lúc
  snmp.SendQuery;
```

Sau khi có thông tin tổng số byte truyền và nhận theo từng chu kỳ lấy mẫu, bạn vẽ chúng ra trên biểu đồ. Cách xử lý thông tin và vẽ biểu đồ không nằm trong phạm vi quyển tài liệu này, bạn hãy xem thêm trong source code. Hình minh họa biểu đồ như sau :



3. SNMP Trap Receiver

Giới thiệu

SNMP Trap receiver là phần mềm có khả năng nhận và hiển thị các trap version 1.

Ý tưởng thực hiện

+ Dùng component IdSNMP của Delphi để nhận trap ở port 162.

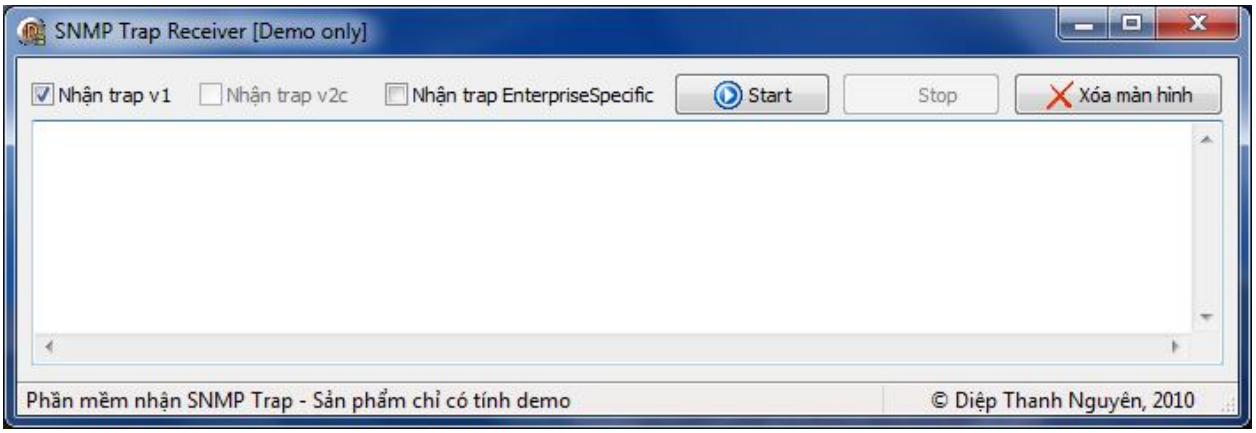
+ Khi một trap được nhận thì phần mềm sẽ xem có phải version của trap là v1 hay không, nếu là v1 thí bản tin trap sẽ được tiếp tục xử lý.

+ Bản tin trap sẽ được hiển thị ra màn hình với đầy đủ các thuộc tính. Chương 4 đã trình bày cấu trúc của một bản tin trap v1, gồm : enterprise, agent-addr, generic-trap, specific-trap, time-stamp và các variable-bindings.

+ Nếu trap có kiểu generic thì phần mềm sẽ hiển thị tên trap. Chương 1 đã liệt kê 7 giá trị generic gồm : coldStart(0), warmStart(1), linkDown(2), linkUp(3), authenticationFailure(4), egpNeighborloss(5), enterpriseSpecific(6).

+ Trong các oid có trong bản tin trap, nếu oid nào là con của mib-2.system và mib-2.interfaces thì nó sẽ được hiển thị tên thay vì oid.

Thiết kế giao diện



+ Nút [Start] để bắt đầu nhận và xử lý trap.

+ Checkbox [Nhận trap enterpriseSpecific] là tùy chọn có hiển thị các trap có generic = 6 hay không.

Cấu trúc dữ liệu

Để có thể hiển thị tên của các trap generic - ví dụ hiển thị "Generic : linkDown(2)" thay vì "Generic : 2" – thì ta cần một mảng định nghĩa chúng.

```
var
  genTrap: array[0..6] of string = (
  'coldStart(0)', 'warmStart(1)', 'linkDown(2)', 'linkUp(3)', authenticationFailure(4)',
  'egpNeighborloss(5)', 'enterpriseSpecific(6)');
```

Để có thể hiển thị tên của các oid – ví dụ hiển thị "sysDescr" thay vì ".1.3.6.1.2.1.1.1" – ta cần khởi tạo thủ tục định nghĩa chúng

```
var
  oid: TstringList;

procedure TfrmMain.InitCommonOID;
begin
  oid := TStringList.Create;
  oid.Delimiter := '=';
  oid.Add('1.3.6.1.2.1.1.1=sysDescr');
  oid.Add('1.3.6.1.2.1.1.2=sysObjectID');
  oid.Add('1.3.6.1.2.1.1.3=sysUptime');
  oid.Add('1.3.6.1.2.1.1.4=sysContact');
  oid.Add('1.3.6.1.2.1.1.5=sysName');
  oid.Add('1.3.6.1.2.1.1.6=sysLocation');
  oid.Add('1.3.6.1.2.1.1.7=sysServices');
  oid.Add('1.3.6.1.2.1.2.1=ifNumber');
  oid.Add('1.3.6.1.2.1.2.2.1.1=ifIndex');
  oid.Add('1.3.6.1.2.1.2.2.1.2=ifDescr');
  oid.Add('1.3.6.1.2.1.2.2.1.3=ifType');
  oid.Add('1.3.6.1.2.1.2.2.1.4=ifMtu');
  oid.Add('1.3.6.1.2.1.2.2.1.5=ifSpeed');
  oid.Add('1.3.6.1.2.1.2.2.1.6=ifPhysAddress');
  oid.Add('1.3.6.1.2.1.2.2.1.7=ifAdminStatus');
  oid.Add('1.3.6.1.2.1.2.2.1.8=ifOperStatus');
  oid.Add('1.3.6.1.2.1.2.2.1.9=ifLastChange');
  oid.Add('1.3.6.1.2.1.2.2.1.10=ifInOctets');
  oid.Add('1.3.6.1.2.1.2.2.1.11=ifInUcastPkts');
  oid.Add('1.3.6.1.2.1.2.2.1.12=ifInNUcastPkts');
  oid.Add('1.3.6.1.2.1.2.2.1.13=ifInDiscards');
  oid.Add('1.3.6.1.2.1.2.2.1.14=ifInErrors');
  oid.Add('1.3.6.1.2.1.2.2.1.15=ifInUknownProtos');
  oid.Add('1.3.6.1.2.1.2.2.1.16=ifOutOctets');
  oid.Add('1.3.6.1.2.1.2.2.1.17=ifOutUcastPkts');
  oid.Add('1.3.6.1.2.1.2.2.1.18=ifOutNUcastPkts');
  oid.Add('1.3.6.1.2.1.2.2.1.19=ifOutDiscards');
  oid.Add('1.3.6.1.2.1.2.2.1.20=ifOutErrors');
  oid.Add('1.3.6.1.2.1.2.2.1.21=ifOutQLen');
  oid.Add('1.3.6.1.2.1.2.2.1.22=ifSpecific');
end;
```

Nhận trap

Khi click nút [Start], chương trình sẽ thực hiện vòng lặp liên tục để nhận trap

```
procedure TfrmMain.bbtStartClick(Sender: TObject);
begin
  snmp.ReceiveTimeout := 100; // thời gian timeout cho hàm ReceiveTrap là 100ms
  snmp.Active := True;
  bbtStart.Enabled := False;
  bbtStop.Enabled := not bbtStart.Enabled;
  while snmp.Active do
  begin
    while snmp.ReceiveTrap = 1 do // =1 nếu nhận được trap trong vòng 100ms
    if (ckTrapv1.Checked and (snmp.Trap.Version = 0)) then
    begin
      ProcessTrapv1(snmp.Trap); // hàm xử lý bản tin trap v1
    end;
    Sleep(1); // nếu không tạm dừng 1ms thì vòng lặp sẽ chiếm 100% CPU
    Application.ProcessMessages;
  end;
end;
```

Hàm xử lý trap v1 như sau

```
procedure TfrmMain.ProcessTrapv1(var trap: TSNMPInfo);
var i,j: integer; s: string;
begin
  // nếu nhận bản tin trap có generic = 6 (enterpriseSpecific) mà checkbox [Nhận trap
enterpriseSpecific] không được chọn thì ngừng xử lý
  if (not ckSpecific.Checked) and (Trap.GenTrap = 6) then Exit;

  mm.Lines.Add('+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+');
  mm.Lines.Add( 'Host: ' + Trap.Host + TAB +
                'Version: ' + GetTrapVersionStr(Trap.Version) + TAB +
                'Enterprise: ' + trap.Enterprise);

  mm.Lines.Add( 'Generic: ' + genTrap[Trap.GenTrap] + TAB +
                'Specific: ' + InttoStr(Trap.SpecTrap) + TAB +
                'TimeStamp: ' + InttoStr(Trap.TimeTicks));

  // nếu bản tin trap có chứa variable-bindings (các cặp oid-value) thì ValueCount là
tổng số các oid-value chứa trong trap
  if Trap.ValueCount = 0 then mm.Lines.Add('(no values)')
  else
  for i := 0 to Trap.ValueCount - 1 do
  begin
    // nếu oid là một trong các oid nằm dưới mib-2.system và mib-2.interfaces thì hiển
thị tên của oid đó
    j := isCommonOID(Trap.ValueOID[i]);
    if j > -1 then s := Replacestr(Trap.ValueOID[i], oid.Names[j],
oid.ValueFromIndex[j])
              else s := Trap.ValueOID[i];
    mm.Lines.Add('   ' + s + TAB + '=' + TAB + Trap.Value[i]);
  end;
  mm.Lines.Add('');
  Trap.Clear;
end;
```

4. SNMP Agent

(in progress)

5. Abstract Syntax Notation One (ASN.1)

(in progress)

Tóm tắt

+