EXA CORPORATION

SNMP Tips

～第 1.0 版～

（株）エクサ

システム基盤ソリューション事業部

2003 年 4 月 1 日

目次

第1章　はじめに

本 Tips では、 ネットワーク管理者やこれから管理者になろうとする方に、 SNMP、Linux およびその周辺からテーマを選び、できるだけ実例を 引き合いに出して、基本的なテクニックやヒントについて、簡潔に かつ具体的に紹介していくことを目的としています。

この Tips をご覧になることで、

- **SNMP 管理ツールを導入したが使い方がよくわからない。**
- **SNMP でネットワーク管理しようと思っているが、具体的にどうすればよいかわからない。**
- **Linux で SNMP エージェントを動かしたい。**

などの疑問、要望を解決する一助になればと考えております。

SNMP はほとんどのネットワーク機器やワークステーションなどに実装されていますが、いざ管理ツールで情報を収集しようとしても「具体的にどうしたらいいのだろう？」とか「なぜ機器毎に違う管理ツールを使わなければならないのだろう？」など、お困りになったことはありませんか？

本Tipsは、基本的な仕組みから始めて、それらの疑問と解決のためのヒントやテクニックをなるべく簡単に解説してみようという試みです。

第2章　実用 SNMP

ここでは、SNMP の概要から、実際に SNMP によるネットワーク管理の実例まで解説してゆきます。

2.1 SNMP の概要

2.1.1 はじめに

SNMP は「Simple Network Management Protocol」の略称で、直訳すると「簡易ネットワーク管理プロトコル」となります。プロトコルですから、本来の意味としては主に「情報交換手順」のことを指しますが、「SNMP という情報交換手順で、データの送信および受信をおこなうことが可能な仕組み」のことを指す場合の方が一般的です。(「プロトコル」＝「通信規約」が正しい解釈ですね) SNMP は TCP/IP ネットワーク環境 (オープンなネットワーク環境) の管理プロトコルとして規定され、主にネットワーク中継装置 (ルータやブリッジなど) や TCP/IP プロトコルを使用したネットワーク環境の情報を収集することを目的に作られています。最近では SNMP の普及 (もう業界標準と呼んでもよいでしょう) によって単にネットワーク管理に止まらず、システム管理などを行うために SNMP を使用する場合もあります。

では、管理者は SNMP をどのように使ってネットワークの情報を収集し管理を行うのでしょうか？

繰り返しますが、SNMP はあくまでプロトコルですから管理者は通信する相手 (管理対象) が必要です。ここでは、「管理装置」のことを「マネージャ」, 管理対象のことを「エージェント」と呼ぶことにします。マネージャとエージェントの通信イメージは下図(Fig-1-1)のようになります。

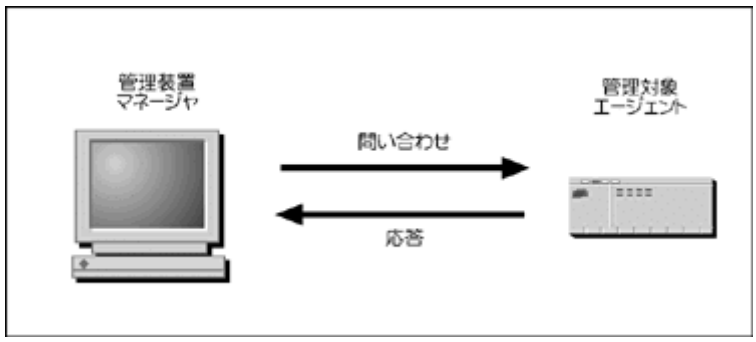


Fig-1-1

このように、マネージャはエージェントに対して情報の問い合わせを行い、エージェントはマネージャに応答します。このエージェントが持っている情報の集まりを MIB (Management Information Base)と言い、MIB に規定された個々の情報のことをオブジェクトと呼びます。マネージャはエージェントの情報 (MIB) を収集し、それらをもとにネットワークの状態を判断し管理することができます。

例えば、あるルータのインタフェース (ポート) の状態を確認したい場合は、ルータのインタフェースの状態を表す MIB 情報に対してリクエスト を行い、返り値 (応答してきた値) を判断する必要があります。

つまり、SNMP はデータ収集を行うためのプロトコルですから、管理者はエージェントの機能や、ネットワークに関する知識が必要になります。ただ、最近の SNMP マネージャソフトウェアには、GUI などを利用して機器やネットワークの状態をわかりやすく表示することができるようになっているものもあります。

今回は、SNMP の基本となる3つの取り決めとして SNMP (RFC1157) , MIB-2 (RFC1213),

SMI (RFC1155, RFC1212)について、簡単な説明をします。

2.1.2 SNMP (Simple Network Management Protocol)

ここでは、プロトコルとしての SNMP について簡単に解説しましょう（主に Ethernet 環境での解説です）。

■ SNMP はプロトコル

SNMP は、UDP というコネクションレス型プロトコルの上位にマッピングされるプロトコルです。プロトコルとしての仕様は、RFC1157 として定義されています。現在では、UDP 以外に IPX にマッピングしているケースもあるようです（RFC1298 で規定されています）。下図(Fig-1-2)は、SNMP のマッピングと MIB－オブジェクトの関係イメージです。

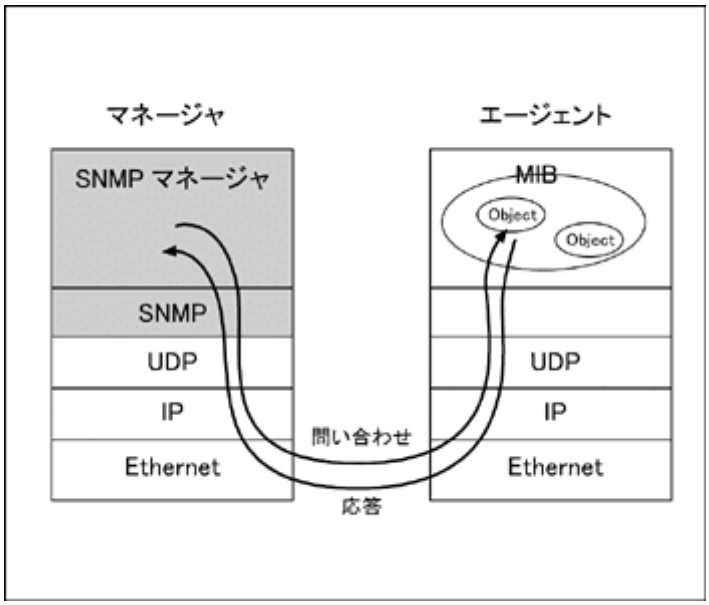


Fig-1-2

■ 情報交換手順（コマンド）は5つ

マネージャとエージェント間の情報交換手順は、以下の5つです。

［マネージャからエージェントへの問い合わせ要求および、設定要求］

・**Get Request**

・**Get Next Request**

・**Set Request**

［エージェントからマネージャへの応答および、イベント通知］

・**Get Response**

・**Trap**

マネージャからの3種類の要求に対して、エージェントはすべて Get Response で応答します。

Trap は、エージェントがマネージャに対して自らイベントを通知するためのコマンドです。

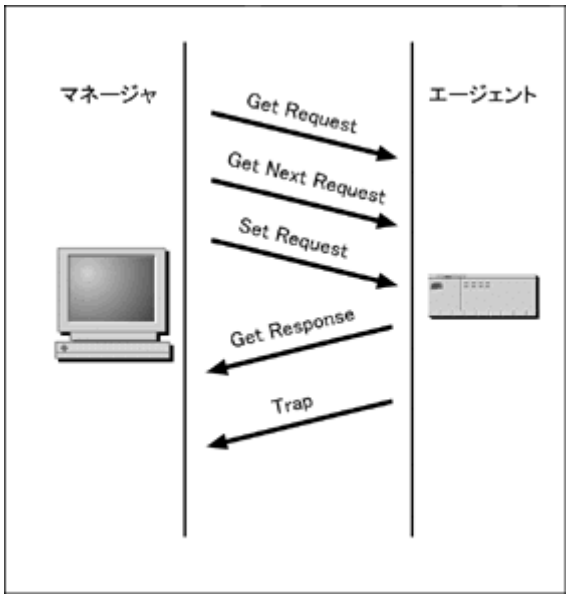


Fig-1-3

2.1.3 MIB (Management Information Base)

MIB は管理情報の集まりのことで、一種のデータベースのようなものです。個々の管理情報（MIB の構成要素）のことを**オブジェクト（Object）**と呼びます。

■ MIB の種類

MIB といっても、メーカや機種別に様々なものがありますが、「TCP/IP デバイスが実装すべき MIB」として定義してあるのが **MIB-2（RFC1213）**です。RFC では、MIB-2 以外にも、HUB や Bridge などの機種別や FDDI 用や ATM 用などの様々な MIB が定義されています。これらの、RFC で規定されている MIB のことを**標準 MIB** と言います。一方、メーカ独自の管理情報が定義されたものは**拡張 MIB** や**ベンダ MIB** などと呼ばれ、実装されている機器についての詳しい情報が定義されています。

■ オブジェクト ID

MIB の構成要素であるオブジェクトにはそれぞれ識別子（ID）が割り振られています。オブジェクト ID は必ず一意でなければいけません。たとえば、MIB-2 の sysDiscr というオブジェクトには、「1.3.6.1.4.2.1.1.1」という ID が割り振られています。オブジェクト ID の体系をツリー状に表したのが MIB ツリーと呼ばれるものです。(Fig-4) **ただし、実際の管理者にとってはオブジェクト ID やその関係より、オブジェクトの意味の方が重要です。**

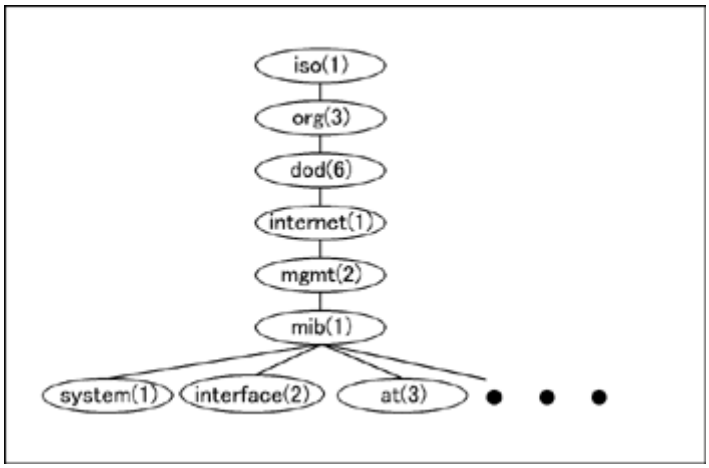


Fig-4

■ MIB の定義

SMI と ASN.1(*1)に則って記述されているドキュメントのことを MIB ファイルと呼びます。

MIB ファイルには、その MIB に含まれるオブジェクト ID やその意味などが記述されています。

SNMP で管理対象からどのような情報を収集できるかどうかは、管理対象がサポートしている MIB ファイルを見る必要があります。

(*1) ASN.1 (抽象構文記法タイプ1) : (「ISO 8824」もしくは「JIS X 5603」)

ISO が規定するデータ標記等のルール。SNMP では、そのサブセットを利用しています。

2.1.4 SMI (Structure of Management Information)

SMI は管理情報 (MIB) を定義するためのルールのようなものです。ですから、開発者以外はあまり気にする必要はありません。ただし、MIB の定義について規定されているものなので、興味のある方は目を通しておくとよいかもしれません。

最近は少なくなりましたが、メーカが提供する拡張 MIB には SMI や ASN.1 の規則違反やタイプミス (!) などの原因で、SNMP マネージャソフトウェアに MIB を追加できない場合があります。このような時は、SMI や ASN.1 を確認する必要があります。(メーカ等に問い合わせても、答えが返ってこない場合ですが. . .)

2.2 MIB-2 でネットワーク情報を見てみよう

2.2.1 はじめに

SNMP マネージャを導入しても、あらかじめ設定されているアラームや、管理対象（デバイス）の状態を表す専用のビューワーしか使っていない場合が多いのではないでしょうか？

そこで、今回はもう少し SNMP マネージャを活用するための情報として、SNMP 対応の機器がほとんどサポートしている「MIB-2 情報をどう使えばいいのか」を紹介してみようと思います。MIB-2 の全ての情報については、「特集：MIB-2 概説」を参照していただくとして、ここでは例などを交えつつ紹介していきたいと思います。

とりあえずここに記述してある情報を確認するだけでも、障害の切り分けなどに役に立つのではないかと思います。

2.2.2 デバイスが認識しているネットワークの状態

デバイスが認識しているネットワークの情報を確認することができます。

MIB-2 で定義されているオブジェクトでデバイスの状態を表す代表的なものは以下の通りです。

■ **デバイスの連続稼働時間（sysUpTime）：**

デバイスが起動してから現在（応答を返した時）までの時間を、100 分の 1 秒単位で表しています。最大値は、2^32-1（約 497 日）までカウントアップすることができます。この時間を超えると再び 0（ゼロ）にもどり加算されていきます。この値を参照することで、連続稼働時間の確認や、瞬断が発生しているかどうかを判断する目安になります。

障害判断例：

sysUpTime の値が、前回応答のあった値よりも少ない場合は、瞬断などがあったと判断する。

注意：この値が1周り回ったかどうかの判断は、SNMP マネージャの仕様などにより異なります。

ColdStart，WarmStart トラップとの併用をお勧めします。

ColdStart，WarmStart：デバイスの起動時やリブート時に送出されるトラップです。

例） sysUpTime の変化

連続稼働時間：8 時間

sysUpTime = 2880000

電源 OFF（or リスタート）

連続稼働時間：10 分

sysUpTime = 60000

■ **インタフェース (ポート) の初期設定状態 (ifAdminStatus)：**

デバイスのポートに設定している状態を表します。

状態は、up, down, testing があります。up の場合は、設定者がそのポートを使用可能な状態に設定している場合となります。

この値を参照することで、そのポートが使える状態に設定してあるかどうかを確認することができます。

ただし、一部の機種では ifOperStatus の状態と連動 (同じ値になる) する場合があるようです。

■ **インタフェース (ポート) の現在の状態 (ifOperStatus)：**

デバイスの現在のポート状態を表します。

状態は、up, down, testing があります。up の場合は、そのポートを現在使用していることを表します。この値を参照することで、そのポートを使用しているかを確認することができます。「使用しているかどうか」の判断は、データリンク層レベルでのリンクが張れているかどうかです。

障害判断例：

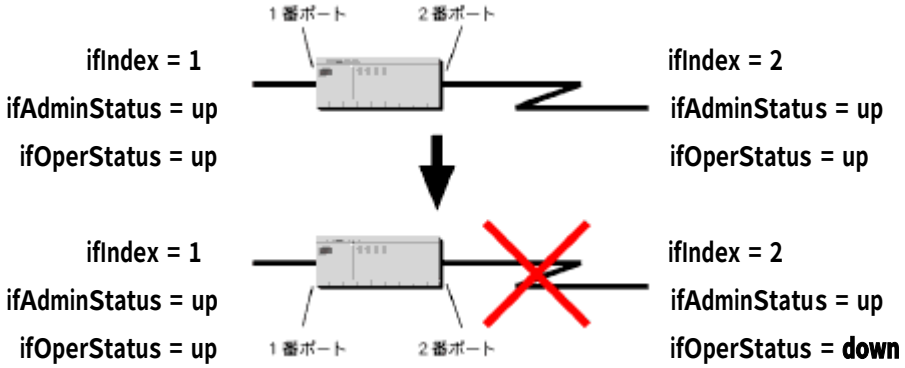
ルータの 2 番ポートが WAN 回線に接続されている。2 番ポートがダウンした場合、そこから先のネットワークに接続することができなくなります。

このような事象が発生する原因としては、以下のようなものが考えられます。

・接続先ルータなどがダウンした。

・回線や、ケーブルが切断された。

・TA などが接続されている場合は、TA がダウンしている可能性もあります。



注意：「ifOperStatus が down であった場合に障害と判断する。」という定義を行った場合、障害を正確に判断できない場合があります。例えば、複数のインタフェースをサポートしているルータなどの場合は、使用していないインタフェースに対応する ifOperStatus の値は down となるためです。

このことを、回避するためには、「ifAdminStatus が up で、かつ ifOperStatus が down であった場合に障害と判断する。」もしくは、「ifOperStatus が up から down に変化した場合に障害と判断する。」というような定義を行わなければなりません。

■ **インタフェース (ポート) の状態が変化した時間 (ifLastChange)：**

ifOperStatus の状態が変化したときの sysUpTime の値です。

この値を参照することで、いつ該当ポートの状態が変化したかを確認することができます。上記例の場合、［ sysUpTime 値－ifLastChange 値］ で、「どのくらい前に状態が変化したか」がわかります。

■ **IP アドレステーブル (ipAddrTable)：**

デバイスに設定されている IP アドレスの一覧です。

この情報を参照することで、デバイスのポート別の IP アドレスとサブネットマスクの設定などが確認できます。

■ **IP フォーワーディング状態 (ipFowarding)：**

そのデバイスが IP データグラムをフォワーディングしているかどうか、つまりそのデバイスがルータとして機能しているかどうかを確認することができます。こ の値が「not-forwarding」の場合は、ルータとして機能していないことを表します。

2.2.3 デバイスの性能

デバイスを通過するトラフィックなどの情報を確認することができます。

ネットワーク中継装置 (ルータなど) に有用な情報です。

性能情報を表す Object の値を参照する場合は、特に注意が必要です。RFC などで定義してある Object の意味は、管理対象が実装するに当たり、解釈の違いなどにより異なる場合がありますので、可能であれば想定している Object 値の意味と、実際に収集した値が同じかどうかを確認したほうがよいでしょう。

また、Object 値がカウンタの場合、最大値 (2^32-1) を超えた場合の処理 (評価方法) も確認しておく必要があります。

■ **ポート別入出力オクテット数や使用率 (ifInOctets/ifOutOctets)**

入出力オクテット数を監視することで、インタフェース (ポート) を通過するトラフィックの使用率などを計算することができます。

接続線の使用率を計算するためには以下のオブジェクト値が必要です。

ifIndex	ポート番号を特定するために必要です。
ifSpeed	接続回線容量 (スピード) を確認するために必要です。 (単位は、bit/sec で表示されます。)
ifInOctets	入力オクテット (バイト) 数のカウンタ
ifOutOctets	出力オクテット (バイト) 数のカウンタ

ifInOctets と ifOutOctets の値はカウンタですので、時系列でトラフィック量などを確認するためには、ポール間隔での差分値を計算します。

使用率計算方法：

[(ifInOctets 差分値 + ifOutOctets 差分値) × 8 ÷ (ポール間隔秒) ÷ ifSpeed × 100] %

注意：

(1) 機器によっては、ifIndex の値とポート番号が対応していない場合があります。

(2) 機器によっては、ifSpeed の値が接続している回線容量と異なる値になっている場合があります。

(3) 機器によっては、ifInOctets と ifOutOctets の値が「インタフェースを経由したトラフィック値」を表していない場合があります。

(4) ここで計算する値は、あくまでインタフェースを通過するトラフィック使用率ですので、回線やセグメントの使用率とは異なる場合があります。

■ デバイスの負荷 (ifInDicards/ifOutDiscards)

デバイスが処理しきれずに捨てたパケット数の割合等を監視することで、デバイスの負荷を監視することができます。デバイスが破棄したパケット数は、ifDiscards の値を参照します。ルータなどでこの値が増加している場合は、転送先インタフェースの回線容量が転送要求トラフィックに対し少なすぎるか、搭載メモリが少ない可能性があります。また、パケットが破棄されていますので、データの再送が頻繁に発生している可能性があります。

全体のトラフィック（パケット数）と廃棄されたパケット数の割合を計算することで、どの程度のトラフィックで破棄されるパケットが増加するかを判断する目安になります。

経験上、出力廃棄パケット数（ifOutDiscards）は入力廃棄パケット数（ifInDiscards）より非常に少ない場合が多いので、負荷の傾向を監視することが目的であれば、入力廃棄パケット数に関する情報のみを監視していればよいと思います。

負荷による廃棄パケットの割合を計算するためには以下のオブジェクト値が必要です。

ifIndex	ポート番号を特定するために必要です。
ifInUcastPkts	受信ユニキャストパケット数を表します。
ifInNUcastPkts	受信ノンユニキャストパケット(例えば、サブネ ットワークのブロードキャストやマルチキ ャストなど)数を表しま す。
ifInDiscards	正常なパケットとして受信したが、バッファ不足などで破棄されたパケット数を表します。
ifInErrors	エラーにより破棄されたパケット数を表します。
ifInUnknownProtos	不明なプロトコルなどであることが原因で、破棄されたパケット数を表します。

ifIndex 値以外のオブジェクト値は、カウンタ値ですので時系列で値の変化を確認するためには、ポール間隔での差分値を計算します。

廃棄パケット率計算方法：

入力パケット数は、以下の計算で求められます。

[ifInUcastPkts＋ifInNUcastPkts＋ifInDiscards＋ifInErrors＋ifInUnknownProtos]

したがって、負荷による廃棄パケットの割合は以下の式で求められます。

[ifInDiscards 差分値÷（入力パケット数差分値）×100] %

2.2.4　デバイスが認識しているネットワークの状態

デバイスが認識しているネットワークの情報を確認することができます。

IP ネットワークの状態を確認できます。

■ MAC アドレスと IP アドレスの対応表 (ipNetToMediaTable)

そのデバイスが認識している MAC アドレスと IP アドレスの対応表一覧です。

この値を確認することで、どの接続されている端末の「接続ポート」「MAC アドレス」「IP アドレス」がわかります。(出力結果は、コマンド arp -a とほぼ同じです。)

ipNetToMediaIfIndex	当該エントリがどのインタフェースに接続されているかを確認するために必要です。
ipNetToMediaPhysAddress	物理アドレスを表します。
ipNetToMediaNetAddress	物理アドレスに対応する IP アドレスを表します。

例：以下の表は、「ipNetToMediaTable」を Get したときのサンプル。

ipNetToMediaIfIndex	ipNetToMediaPhysAddress	ipNetToMediaNetAddress
1	0123456789ab	111.111.111.111
2	ba9876543210	111.111.112.111

■ルーティングテーブル (ipRoutingTable)

そのデバイスのルーティングテーブルを表します。

ipRouteDest	この経路の宛先 IP アドレスです。
ipRouteIfIndex	この経路で次にホップするのに通過する ifIndex の番号 (インタフェース番号) です。
ipRouteMetric1～5	当該経路のメトリックを表します。
ipRouteNextHop	当該ルートで次にホップするインタフェースの IP アドレスです。
ipRouteType	ルートのタイプです。
ipRouteProto	どのように経路を決定するかのルーティングメカニズムを表します。

ipRouteAge	当該経路が最後に更新されたか、または他の方法で正しいと決められてから の秒数です。
ipRouteMask	ipRouteDest の値と比較をする前に、宛先のアドレスと論理積をとるマスクを表 します。
ipRouteInfo	経路は、ipRouteProto に設定されているルーティングメカニズムに依って決められます。

ルーティング経路情報は、以下の様に見ることができます。

「ipRouteDest」のネットワーク向けのフレーム (IP データグラム) は、当該ルータの「ipRouteIfIndex」ポートから、「ipRouteNextHop」アドレスに転送する。

2.2.5 ネットワーク詳細情報

TCP/IP に関するエラーなどの情報を確認することができます。

■不良 IP パケット情報など (ip グループ)

ipOutNoRoutes	宛先に転送する為の経路が判明しなかった (宛先不明の) 為に廃棄された IP データ グラムの数を表します。この値が常に増加している場合は、 どこかのデバイスが常に、不正 (宛先不明) の IP アドレスに対して、何らかの要求を出している可能性があります。
ipReasmFails	IP 組み立ての過程で検出された不具合(例えば、タイムアウトエラー など)の数を表します。この値が常に増加している場合は、データの転送効率が非常に悪い可能性があります。 原因として、コリジョンが多発している場合や使用率が非常に高いセグメント経由で通信している場合などが考えられます。
ipFragFails	当該エンティティでフラグメント化する必要があったのに、フラグメント化 できなくて、捨てられた IP データグラムの数を表します。例えば、IP データグラムの"Don'tFragment"フラグがセットされていた 場合などが有ります。この値が常に増加している場合は、ルータの設定やデータ送受信元のデバイスの設定を確認したほうがよいでしょう。
ipRoutingDiscards	有効だけれども放棄されたルーティングエントリの数を表します。理 由としては、他のルーティングエントリのためのバッファスペースが足り なくなったことが考えられます。この値が常に増加している場合は、ルーティングトポロジーが当該デバイスの性能では対応できないほど複雑になっている可能性があります。

■ IP 経路不一致情報など (icmp グループ)

ICMP は、IP プロトコル内のエラー制御のためのプロトコルです。

icmpInDestUnreachs	受け取った ICMPDestinationUnreachable メッセージの数を表します。ゲートウェイがルーティングテーブル上の目的地アドレスが到達不可能であることを検出すると、ホストに対してこのメ ッセージを返します。この値が増加するのは、当該デバイスが宛先不明 IP アドレスに対して通信を行おうとした場合などです。
icmpOutDestUnreachs	送りだした ICMPDestinationUnreachable メッセージの数を表します。ゲートウェイが、ルーティングテーブル上の目的地アドレス が到達不可能であることを検出すると、ホストに対してこのメッセージを返します。この値が増加するのは、当該デバイス経由で、他のデバイスが宛先不明 IP アドレスに対して通信を行おうとした場合などです。

■ tcp コネクションの状態確認 (tcpConnTable)

tcpConnState	TCP コネクションの状態を表します。 管理ステーションがセットする値は主に deleteTCB(12)です。
tcpConnLocalAddress	TCP コネクションのローカル IP アドレスを表します。 ノードに関連するどんな IP インタフェースに対してもコネクション を受け取る listen 状態のコネクションの 場合、この値は 0.0.0.0 を使います。
tcpConnLocalPort	TCP コネクションのローカルのポートの番号を表します。
tcpConnRemAddress	TCP コネクションのリモートの IP アドレスを表します。
tcpConnRemPort	TCP コネクションのリモートのポート番号を表します。

注意：エージェントによっては、Connection の状態がしばらく listen である場合、エントリーから削除してしまうものもあるようです。また、close の状態の場合は一定時間たつとエントリから削除されます。このテーブルの情報を確認することで、当該デバイスがどの TCP サービス (ftp サーバ, telnet サーバなど) を提供しているかがわかります。

例：

tcpConnState	tcpConnLocalAddress	tcpConnLocalPort	tcpConnRemAddress	tcpConnRemPort
listen	0.0.0.0	23	0.0.0.0	0
listen	0.0.0.0	80	0.0.0.0	0

このような情報が表示されるデバイスは、telnet と http(Web ブラウザからのアクセス）サービスが動作していることを示します。

tcpConnLocalPort=23：telnet ポート
tcpConnLocalPort=80：http ポート

tcp のポート番号については、「RFC1700」として公開されています。

2.3 RMON でトラフィック管理だ

2.3.1 はじめに

最近、RMON 対応のネットワーク機器や、RMON プローブなどが色々出回っています。ただ、これらの情報も専用のツールなしでは、なかなか活用することは難しいのではないでしょうか？ そこで、 前回の MIB-2 でネットワーク情報を見てみよう」でも記述しましたが、これで何がわかるのか？また、どう評価すればいいのか？という視点で解説をしてみようと思います。
RMON-MIB には、Ethernet 用 (rfc1271 or rfc1757) と、TokenRing 用 (rfc1513) がありますが、今回は Ethernet 用の RMON-MIB について解説します。

今回は、ちょっとパワーユーザ向けです。

2.3.2 RMON とは

RMON-MIB (Remort network MONitoring MIB) とは、「リモートネットワークをモニターするための MIB」です。主に、セグメント単位のトラフィック管理などを行うために規定されています。
ここでいう「セグメント」とは、コリジョンドメインのことを指します。スイッチングハブや、ブリッジで区切られる範囲ですね。この理由は、「RMON は MAC 層 (データリンク層) の情報を定義」しているものだからです。

RMON-MIB をサポートしているデバイス (以降：プローブ) は、接続されているセグメントに流れるトラフィック情報を収集／蓄積することができます。また、セグメントに流れているフレーム (パケット) をキャプチャできるものもありますので、プロトコルアナライザのような使い方も可能です。ただし、RMON-MIB ではホストの識別は MAC アドレス (Ethernet アドレス) で行うために、セグメント (コリジョンドメイン) を超えるトラフィックの計測には向きません。

また、RMON-MIB の情報を収集するためにはプローブに対して「どのようなタイプのデータを収集するか」を設定したり、「収集したデータの表示」などを行うために、専用のマネージャソフトウェアがないと大変手間がかかります。専用のマネージャソフトウェアを利用することで、以下のようなことが容易にできます。

・セグメントに流れるトラフィックの履歴をグラフでわかりやすく表示
・セグメントで発生しているトラフィックの異常をリアルタイムに表示
・専用のテンプレートで、わかりやすいパケットキャプチャの設定
等

注意：RMON-MIB は、全ての情報 (MIB)が Optional となっていますので、プローブによって収集可能な情報が異なる場合があります。カタログなどを参照して、必要な情報が収集できるタイプかどうかをあらかじめ確認しておきましょう。

2.3.3 RMON-MIB 情報を見る

RMON-MIB は、セグメントに流れる様々なタイプのトラフィックを収集することができます。ここでは、それら収集できる情報の概要を紹介します。

今回は、Object 個々の説明はあまりしません。詳しくは、RFC や文献を参照してみてください。

■統計情報(The Statistics Group)：

このグループは、モニターしているセグメントの基本的な統計情報を提供します。このグループにはインターフェース毎に以下のような情報 (Object) が定義されています。
「パケット」と表現している部分について、Ethernet ではフレームという表現が適切なのですが、RMON-MIB の Object 名との整合を考慮してこのようにしています。

・パケット数
・オクテット数
・長さ別パケット数
　(64, 65～127, 128～255, 256～511, 512～1023, 1024～1518)
・ブロードキャストパケット数
・マルチキャストパケット数
・コリジョン数
・エラーパケット数
　(CRCAlign, Undersize, Oversize, Fragment, Jabber)

また、プローブのどのインタフェースからデータを収集しているかの情報や、そのエントリの作成者，状態などを表す情報も定義されています。

RMON-MIB 対応のデバイスは、この「統計情報」をインタフェース別にあらかじめ作成されている (エントリが作成されている) 場合が多いのですが、まれに何もエントリが作成されてなかったり、スイッチングハブの場合はインタフェース別に作成されていないものもあります。注意しましょう。

■履歴情報 (The History Group)：

このグループには、制御用の Table とデータ格納用の Table があります。

●制御テーブル (historyControlTable)
　テーブルの1行は以下の条件から構成されます。

historyControlIndex	設定番号
historyControlDataSource	データ収集インタフェース番号 ifEntry.ifIndex の値を ObjectID で設定します。
historyControlBucketsRequested	要求バケット(Buckets)数 要求されたバケットの個数です。 バケットとは1回のサンプリングで収集するデータのいれものを指し、データ格納テーブルの1行分を表わします。デフォルトの設定は 50 です。
historyControlBucketsGranted	収集可能バケット数 実際に収集可能なバケットの個数です。
historyControlInterval	サンプリング間隔 (秒)。デフォルトは 1800 (30 分)、上限は 3600 (1時間) です。

historyControlOwner	設定者
historyControlStatus	状態

この制御テーブルを使って、1つのセグメント（インターフェース）に1つ以上のサンプリングの設定を行うことができます。例えば、30 秒間隔で瞬間的な変化を調べ、30 分間隔で恒常的な稼動状況を把握するというような使い方です。

●データ格納テーブル（etherHistoryTable）
このテーブルには、historyControlTable で設定されているサンプリング間隔などの条件で、データが格納されます。上述の historyControlTable で設定された収集可能バケット数を越えた場合は、古いものから順に削除されます。1行に格納されている情報は以下の通りです。

・パケット数
・オクテット数
・ブロードキャストパケット数
・マルチキャストパケット数
・コリジョン数
・エラーパケット数
　（CRCAlign, Undersize, Oversize, Fragment, Jabber）
・使用率（Object 値の単位は、1=0.01%となっています。）

■ホスト情報（The Host Group）：

LAN 上の特定のホストに関する統計情報をとる場合にはこのグループを使います。履歴情報グループと同じように、制御用の Table とデータ格納用の Table があります。

●制御テーブル（hostControlTable）

hostControlIndex	設定番号
hostControlDataSource	データ収集インタフェース番号 ifEntry.ifIndex の値を ObjectID で設定します。
hostControlTableSize	テーブルサイズ hostTable に格納するホスト数を表わします。
hostControlLastDeleteTime	対応する hostTable のエントリを削除した sysUpTime 値。
hostControlOwner	設定者
hostControlStatus	状態

●データ格納テーブル（hostTable, hostTimeTable）

hostTable と hostTimeTable に格納される情報は全く同じですが、インデックス（ソート順）が「MAC アドレス順（hostTable）」か「検出順（hostTimeTable）」かが異なります。

これらのテーブルには以下の情報が定義されています。

・MAC アドレス
・プローブがホストを検出した番号（順番）

・入力パケット数
・出力パケット数
・入力オクテット数
・出力オクテット数
・出力エラーパケット数
・出力ブロードキャストパケット数
・出力マルチキャストパケット数

■ホストトップ N 情報 (The Host Top N Group)：

このグループは、「The Host Group」で収集した情報をもとに、任意のパラメータ順で上位"N"件のリストを作成します。

このグループには、制御用の Table と、データ格納用の Table があます。

●制御テーブル (hostTopNControlTable)

hostTopNControlIndex	設定番号
hostTopNHostIndex	トップ N を計算する元となる hostTable の hostIndex 値
hostTopNRateBase	トップ N リストの基準パラメータ ・入力パケット数 ・出力パケット数 ・入力オクテット数 ・出力オクテット数 ・出力エラーパケット数 ・出力ブロードキャストパケット数 ・出力マルチキャストパケット数 の内のいずれかを指定します。
hostTopNTimeRemaining	hostTopNTable の次回の更新までの時間
hostTopNDuration	更新間隔
hostTopNRequestedSize	要求エントリ数 (N)
hostTopNGrantedSize	実際に設定されているエントリ数 (N)
hostTopNStartTime	hostTopNTable のエントリ作成時の sysUpTime 値
hostControlOwner	設定者
hostControlStatus	状態

●データ格納用テーブル (hostTopNTable)

hostTopNControlTable で指定したパラメータ順でエントリが作成されます。

■マトリックス情報 (The Matrix Group)：

マトリックス情報を使って、セグメント内の任意のホスト間で発生したトラフィックの記録を見ることができます。このグループにも、制御用の Table と、データ格納用の Table があります。

●制御テーブル (matrixControlTable)

matrixControlIndex	設定番号

matrixControlDataSource	データ収集インタフェース番号 ifEntry.ifIndex の値を ObjectID で設定します。
matrixControlTableSize	データ格納テーブルのエントリ数
matrixControlLastDeleteTime	対応する matrixTable のエントリを削除した sysUpTime 値。
matrixControlOwner	設定者
matrixControlStatus	状態

●データ格納テーブル (matrixSDTable, matrixDSTable)
matrixSDTable と matrixDSTable に格納される情報は同じですが、インデックス (ソート順) が「SourceAddres (データ送信元) 順 (matrixSDTable)」か「DestinationAddress (データ受信先) 順 (matrixDSTable)」かが異なります。

これらのテーブルには以下の情報が定義されています。

・送信元 MAC アドレス
・送信先 MAC アドレス
・パケット数
・オクテット数
・エラーパケット数

■フィルタとキャプチャ (The Filter Group, The PacketCapture Group)：

フィルタとキャプチャを利用することで、ネットワークに流れる特定の条件 (ビットパターンでマッチング、ステータスで選択) に合致するパケットを収集し、分析することができます。
プローブに格納されたパケットは、Hex で表示されますので、専用のマネージャ で表示しないと意味がわかりにくいでしょう。

■アラームとイベント (The Alarm Group, The Event Group)：

アラームとイベントを利用することで、「The Statistics Group」で定義してある情報をキーに「しきい値」を設定し、イベントを発行することができます。
ただ、RMON-MIB で規定してあるイベントは、ログとトラップだけです。

2.4 REPEATER-MIB で HUB 稼動情報を GET

2.4.1 はじめに

今回は、HUB の情報を見てみることにしましょう。今回のテーマである REPEATER-MIB とは、HUB 用に定義されている MIB です。*似たような項目を定義している MIB として、「Ethernet-like Interface Types (RFC1398)」,「MAU-MIB (RFC1515)」があります。*

HUB の状態を管理／監視することで、接続されている端末のネットワーク使用状況を確認したりすることができます。

では、さっそく本題に入ることにしましょう。

2.4.2 REPEATER-MIB とは

REPEATER-MIB とは、マルチポートリピータ（HUB）用の MIB のことです。HUB とありますが、スイッチング HUB のことではありません。あくまで、リピータ機能の HUB のことを指します。

ご存知のことだと思いますが、リピータ（HUB）の役割は、ビットレベルの電気信号の増幅装置です。要は、アンプみたいなものですね。このように、単に HUB の機能だけを考えると、IP アドレスや MAC アドレスを割り当てる必要の無いデバイスと言えます。しかし、ただの増幅装置だとしても、HUB はネットワークの中継装置ですから異常が発生した場合に影響範囲はそれなりに大きいはずです。また、サーバなどが接続されている場合は、「ネットワーク経路の異常」か「サーバ自体の異常」かの障害を切り分けるためにも、ある程度の管理は必要だと思います。

さて、SNMP を利用して HUB の管理を行うためには、IP アドレスや MAC アドレスが必要になります。ここで注意なのですが、HUB が持つ MAC アドレスは、HUB のいずれかのポートに割り当てられている MAC アドレスではなく、管理用アドレスである場合があります。

HUB の状態を確認するだけでしたら、telnet や Web ブラウザからのアクセスで確認できるような機種もありますが、常時状態を監視するには SNMP マネージャを利用したほうが障害の早期発見もできるので便利です。

2.4.3 REPEATER-MIB 情報を見る

REPEATER-MIB には以下のような情報が定義されています。

●ベーシックグループ:

REPEATER-MIB で定義されている管理情報の中で、必ず実装しなければならない部分です。

■HUB 自体の状態
LED 付きの HUB が当該 LED に表示する HUB の稼動情報などが定義されています。

■ポートやスタックの状態

●モニタグループ:
Optional（オプション）として定義されています。実装しなくてもよい情報です。
大抵のインテリジェント HUB は実装しているようです。

■HUB 全体に流れるトラフィック情報
■HUB のグループ（ボード）別に流れるトラフィック情報
■HUB のポート別に流れるトラフィック情報

●アドレストラッキンググループ:
このグループも、Optional（オプション）として定義されています。 実装していない HUB も結構あるようですね。ポートに接続されている機器の MAC アドレスなどがわかるので、なかなか有効だと思います。

■HUB のポート別の接続先 MAC アドレス情報

2.4.4 おまけ

RMON-MIB や MIB-2 の解説でも触れましたが、性能に関する情報（モニターグループの情報）の評価は、1つのデータを収集するのではなく、時系列のログとしてある程度収集し、表計算ソフトなどでグラフ化して傾向を分析しないと、あまり役に立つデータにはなりません。

ただ、REPEATER-MIB の性能情報は、ポート単位でデータを収集できるので、全てのポート情報をログするとなると大変ですね。ですから、カスケードポートや、サーバに接続されているポートのみを監視するのが妥当でしょう。 SNMP マネージャには、ログを収集するためにポート（Index）指定できるものもありますので、是非それらの機能を使用して収集しましょう。

2.5 BRIDGE-MIB でスイッチドネットワークを見る

2.5.1 はじめに

今回のテーマは、「BRIDGE-MIB でスイッチドネットワークを見る！」ですが、ちょっと趣向を変えて、「標準 MIB でスイッチングハブの状態などをどれだけ見ることができるか」で進めていきたいと思います。

実際は、BRIDGE-MIB だけではスイッチドネットワークのコンフィグレーション状態を完全に把握することが出来ないからなのですが. . .

なぜ、スイッチングハブなのに、BRIDGE-MIB なのかというと、ちょっと強引ですが「スイッチングハブはマルチポートブリッジ」でもあると考えられるからです。

ではさっそく始めてみましょう。

2.5.2 ルータとして動作しているか？

管理対象のスイッチングハブが、ルータの機能を内蔵している場合やレイヤ 3 スイッチである場合、ルータとして動作している場合があります。
ルータとして機能しているかどうかの確認は、（連載第 2 回でも記述しましたが）MIB-2 の Object 値で確認することが出来ます。

■ルータとして動作しているか？
「MIB-2 - ip - ipForwarding」 の値が、「forwarding(1)」となっている場合は、そのスイッチングハブは「ルータとしても動作している」ことを示します。

■どのポートに IP アドレスが割り振られているか？
スイッチングハブがルータとして動作している場合は、ポート別に IP アドレスが割り振られています。ただし、全てのポートに IP アドレスが割り振られているとは限りません。

表示例：

ipAdEntAddr	ipAdEntIfIndex	ipAdEntNetMask	ipAdEntBcastAddr	ipAdEntMaxSize
10.10.1.254	1	255.255.255.0	1	0
10.10.2.254	3	255.255.255.0	1	0
10.10.3.254	6	255.255.255.0	1	0

この場合は、1, 3, 6 番ポートに IP アドレスが割り振られているのが確認できます。
また、このような時でも、2 番ポートの「ifOperStatus」の値が「up」になっている場合は、該当ポートがブリッジングポートとして動作している可能性があります。

■ルーティング経路はどうなっているか？

ルーティング経路の情報は、当然ですが「ipRoutingTable」の情報を参照することでわかります。

2.5.3 ブリッジング機能の確認

ほとんどのスイッチングハブはブリッジの機能を実装しています。スイッチングハブが複数台接続されているネットワークの場合は、ほとんどのものがスイッチングハブ間でスパニングツリーアルゴリズムを利用して経路情報を決定しています。

■スパニングツリー動作状況

スイッチングハブのスパニングツリー動作状況は、BRIDGE-MIB(RFC1493)の dot1dStp Group で確認することができます。

以下に、このグループのオブジェクトの一部について解説します。

dot1dStpProtocolSpecification	どのタイプのスパニングツリープロトコルが動作しているかを表示します。 通常は、「ieee8021d(3)」の値となりますが、独自のプロトコルを使用している場合は、「unknown(1)」となる場合があるようです。
dot1dStpTimeSinceTopologyChange	ブリッジング経路情報が変更されたときの「sysUpTime」のタイムスタンプが格納されます。 同一ブロードキャストドメインに、スイッチングハブやブリッジが追加されたりした場合に、値が変化します。
dot1dStpTopChanges	ブリッジング経路情報が変更した回数です。
dot1dStpRootCost	このブリッジ（スイッチングハブ）を通過するためのコストを数字で表しています。 この数字を参考にして、ブリッジ間でどの経路が最短の経路かを計算したりします。
dot1dStpRootPort	複数のブリッジが同一ブロードキャストドメインに存在している場合、このブリッジからルートのブリッジへの最短のポート番号を表します。

■フィルタリング設定など

フィルタの設定等は、「dot1dStaticTable」の情報で確認することができる場合もあるようです。

このテーブルには、ポート別に設定した MAC アドレスによるフィルタリングの設定が格納されている場合があります。

ただし、あくまでブリッジ機能でのフィルタリング設定を格納するためのテーブルですので、スイッチングハ

ブ独自のフィルタリングの設定は格納されていない可能性が高いです。

2.5.4 ポートの状態確認

ポート状態の確認は、「MIB-2 - interfaces - ifOperStatus」でも確認することが出来る場合もありますが、MIB の実装方法によっては、実際のポートの状態とずれが生じるものもあるようです。

■ポートの稼動状態

BRIDGE-MIB - dot1dStpPortEntry - dot1dStpPortEnable

この値が「enabled(1)」の場合は、当該ポートが使える状態になっていることを示します。MIB-2 の「ifAdminStatus」と同じような Object だと考えれば良いようです。

■ポートのブリッジング状態

BRIDGE-MIB - dot1dStpPortEntry - dot1dStpPortState

この値が、「disabled(1)」の状態であれば、そのポートは使われていない状態です。
また、「blocking(2)」の場合は、ブリッジング経路選択時にクローズ（ブロッキング）したポートであることを表します。 ただし、この状態をとる場合は、ブリッジング経路がループをする構成になっている場合です。
「forwarding(5)」の場合は、ブリッジング機能が有効なポートであることを示します。

2.5.5 おまけ

スイッチングハブに V-LAN の設定をしている場合、MIB-2 や BRIDGE-MIB では、「もうどうしようもない」と考えてしまうかもしれません。
ところが、今までシリーズで書いてあることを思い出してみると、MIB-2 の情報だけでも V-LAN 設定の概要がわかる場合があります。

■ポート別に割り当てられている IP アドレスの確認
MIB-2：ipAddressTable
V-LAN を設定している場合は、1つのポートに対して、複数の IP アドレスを設定している場合があります。このような場合でも、結局はポートに IP アドレスが割り振られているわけですから、ipAddressTable で確認することができます。

■ ポートに接続されている端末（IP＆MAC アドレス）の確認
MIB-2：ipNetToMediaTable
IP アドレスは割り振られていないポートでも、どのサブネットのメンバであるかどうかを ipNetToMediaTable で確認とることができる場合があります。
ipNetToMediaTable は、学習（収集）した IP アドレスや MAC アドレスが、どのポート番号（ifIndex 番号）に接続されているかどうかを確認することができます。
この Table の情報で、ポート番号別に収集できる IP アドレスのネットワークアドレス部で「どのサブネットのメンバか」を確認します。

2.6 SNMP をもっと使おう！

2.6.1 はじめに

「実用 SNMP」の連載もいよいよ今回が最後となってしまいました。最後ということなので、今までのおさらいを含めていこうかと思います。

2.6.2 SNMP 対応

最近の PC や UNIX コンピュータ用の OS には、SNMP エージェント機能が付属している場合があります。ものによっては、オプションという形でインストールしなければなりませんが、UNIX 系の OS では、MIB-2 を実装している SNMP エージェントがデフォルトで動作している場合が多いようです。
ベンダ拡張 MIB を入手できている場合は、サーバのプロセス（サービス）の動作状況なども確認できる場合もありますので、有効に活用できる可能性もあります。

2.6.3 イベント通知／受信

SNMP には、TRAP という手順があります。
この手順を利用し、ユーザ定義の独自情報を SNMP-TRAP イベントとして SNMP マネージャに通知する仕組みを作成することが出来ます。

TRAP を送出する仕組みは、OS に標準で SNMP トラップ送出機能が付属している場合はそれを利用できます。また、弊社商品の「AdminCenter SNMP package」を利用することもできます。

これらの機能を使うことで、シェルやバッチファイルなどでユーザ定義の独自情報（プロセスの状態など）を SNMP-TRAP として通知することが出来ます。

詳しい仕組みや方法等は、是非別の機会にでもまとめて紹介したいと思います。

2.6.4 SNMP マネージャでこれだけは管理しておきましょう。

SNMP マネージャで管理する場合は、デフォルトの設定任せだと、イベントを検知しても何が起こっているかを判断できずに結局、宝の持ち腐れになってしまう可能性があります。
せっかくですから、最低限の設定内容の把握や、設定などを行ってみましょう。

■SNMP マネージャの監視項目

SNMP マネージャの監視項目を整理してみましょう。
SNMP マネージャを利用したネットワーク監視項目としては、以下のようなものがあると思います。

ネットワーク中継機器のレスポンスの有無のチェック

サーバー機のレスポンスの有無のチェック

ネットワーク中継機器のインタフェースの状態

端末のレスポンスの有無のチェック

トラフィック情報の収集

可能であれば、管理対象のグループ分けをするなどして、障害検知時の切り分けをしやすいように設計しましょう。

実際には、運用体制や担当者の管理責任範囲によって、内容が変わってくるとは思いますが、とりあえず最低限の（一般的な SNMP マネージャが可能な）設定を確実に把握しましょう。

■ネットワーク中継機器のレスポンスの有無のチェック

ネットワーク中継装置のレスポンスの有無のチェックは、以下のような注意が必要と思います。

ルータ：

あたり前の話ですが、ルータには複数の IP アドレスが割り振られています。
レスポンスの有無をチェックする場合は、ルータに設定されている IP アドレスのうち、どの IP アドレスのレスポンスをチェックすれば良いのでしょうか？

単純に考えると「ルータが正常に動作しているかどうか」とは、設定されている経路に正しく情報が伝えられているかどうかを確認すればよいといえます。
そう考えると「ルータのレスポンスの有無のチェック」は、設定されている全ての IP アドレスのレスポンスの有無を確認した方がよいでしょう。

(1) 設定されている IP アドレスすべてのレスポンスの有無をチェックする。

ルータは、インタフェースがダウンした場合に、該当インタフェースに設定されている IP アドレスからレスポンスが無くなる場合があります。
（大半はそうなのですが...）
ただし、SNMP マネージャに論理的に一番近い経路の IP アドレスのインタフェースがダウンした場合は、ルータに設定されている全ての IP アドレスからレスポンスがなくなります。

(2) SNMP マネージャに論理的に一番近い経路の IP アドレスのみをチェックする。

この場合は、ルータに設定されている他のインタフェースの Up/Down の確認は、MIB-2 の情報を利用して確認します。この方法であれば、インタフェースダウンと本体ダウンの障害切り分けがしやすいのではないかと思います。

スイッチングハブ（ブリッジ）：

スイッチングハブ（特にレイヤ 2 スイッチの場合）の監視は、レスポンスが無い場合でも、スイッチングハブの機能は動作している場合がありますので注意が必要です。SNMP エージェント機能だけがダウンしている場合などが、それに該当します。

スイッチングハブ自体のダウンと SNMP エージェント機能のみのダウンの切り分けは、当該スイッチングハブに接続されている機器のレスポンスの有無でチェックできます。

ハブ：

ハブのレスポンスのチェックも、スイッチングハブと同様の考えで良いと思います。
ただし、（SNMP 対応の）全てのハブを監視する必要があるかどうかを検討する必要もあると思います。
もし、ハブがダウンしてしまった場合でも、HUB の設置場所や、該当 HUB に接続されている端末などの情報を把握していなければ、障害の影響範囲の特定や障害対処が時間がかかり、結局ハブを監視していない場合とくらべてもあまり監視効果が期待できないからです。
もし、ハブを監視する場合は、その設置場所や接続されている端末などの構成情報をしっかり把握しておきましょう。

■ネットワーク中継機器のインタフェースの状態

ネットワーク中継装置のインタフェースの状態も監視項目としては重要だと考えます。
インタフェースの状態を確認することで、「ある管理対象のレスポンスが無い」という状態が、インタフェース（接続回線）側にあるのか、中継装置自体が原因なのかの障害切り分けに役立つ場合があります。

この場合、ダウンしたインタフェース（ポート番号）を特定できるイベント通知の設定が可能であれば、より迅速に対応できると思います。もし、イベント通知設定が不可能であったり、面倒である場合は、最低でも**SNMP マネージャの操作方法で、どのインタフェースの状態がダウンしたか** を確認できるようにしておきましょう。

■サーバ機のレスポンスの有無のチェック

サーバ機のレスポンスの有無のチェックは、PING を利用することをお勧めします。
SNMP リクエストのレスポンス確認は、SNMP マネージャの仕様によりタイムアウトの時間が異なっている場合があります。PING であれば、（OS 別ですが）共通のタイムアウト値となっていますので（Solaris などではデフォルト 20 秒）、より実際のネットワークに誓い状態でレスポンスのチェックができます。

また、SNMP をサポートしている機器の場合に、ベンダ拡張 MIB があれば、サーバー機のディスクの状態や、プロセス（サービス）の稼動状態などを監視してもよいかも知れません。

■端末のレスポンスの有無のチェック

IP 端末のレスポンスのチェックは、サーバー機のレスポンスチェックとは別のイベントとして通知したほうが良いでしょう。

運用にもよりますが、1日に1回は終了させる業務端末などと同じ障害レベルでサーバのイベント通知を行う設定にしてあると、本当に致命的な障害（サーバーダウン）との区別がつきにくくなります。

■トラフィック情報の収集

さて、トラフィック情報の収集ですが、収集データの意味と表示／表現方法を考慮して収集しましょう。

例えば、WAN 回線に接続されているルータなどのインタフェースの入出力オクテット数を収集することで、当該回線使用量の概算値が求められる場合が多いのですが、WAN 回線が全二重で接続されている場合は、入力トラフィック量と、出力トラフィック量のそれぞれが、最大転送容量として契約回線容量が保証（FremeRelay などは別です）されます。ですから、単純に「入出力オクテット数の加算値がトラフィック量」と思っていると、とんでもない値になってしまいますので注意しましょう。

また、Ethernet 環境などのバス型 LAN（懐かしい響きですね）の場合、「セグメント上に流れるトラフィックや、コリジョンを知りたい」という場合が多いのですが、これもなかなか難しいでね。
「セグメント＝コリジョンドメイン」と考えれば、RMON プローブなどを導入することで、トラフィックデータは収集できます。
ただ「セグメント＝ブロードキャストドメイン」となると、スイッチングハブやブリッジなどが導入されているネットワークの場合、スイッチングハブやブリッジに接続されているネットワーク全体のトラフィック量を収集する機能が無い場合は、正確なトラフィック量を求めることはコストを考えるとほぼ、不可能なのではないかと思います。

コリジョンについても同様に、HUB で接続されたネットワーク（ケーブル）上では、「コリジョン」という状態は発生しません。
ネットワークに接続される端末に対して、「コリジョンが発生したようだ」ということを通知するための信号（Jam 信号）をカウントして、擬似的にコリジョンとして認識するしかないようです。
HUB 内等で発生したコリジョンについては、当該機器の MIB 情報（例：rptrPortMonitor～）でカウントするこ

とが出来ます。

2.7 関連資料

邦題：「TCP/IP ネットワーク管理入門」
原著者：M.T.ローズ
訳者：西田竹志
発行元：株式会社トッパン
書籍番号：ISBN4-8101-8521-4
原題：「The Simple Book」
題目：OpenDesign No.10
全力特集：ネットワーク管理技術のすべて
発行元：ＣＱ出版株式会社

第3章　実用 SNMP(Linux バージョン)

3.1 Linux で SNMP エージェントを動かそう!!

3.1.1 はじめに

「SNMP」ってご存知ですか? これは、Simple Network Management Protocol の略で、現在巷で良く(?)利用されているネットワーク管理プロトコルです。管理対象となる機器に SNMP エージェントと呼ばれる機能を配置し、管理する場所に SNMP マネージャと呼ばれる機能を置き、ネットワークの管理を行うのが一般的な形です。

ところで皆さんはその SNMP エージェントやマネージャを身近に感じたことはあるでしょうか?ここでは、その SNMP の概要やどうやって使うのか、またどんなことに使えるのかなどを、3 回程度に分けて、簡単に説明して見ようと思っています。

今の予定では、次のように考えています。

第 1 回	**Linux で SNMP エージェントを動かそう!!** Linux(RedHat 7J)上で、ucd-snmp(SNMP のパッケージの一つ)をとりあえず動かしてみる。
第 2 回	**SNMP エージェントの情報ってどんなもの??** ucd-snmp に付属しているコマンドを利用して、実際の情報を見てみます。
第 3 回	**SNMP を使って監視をしてみよう。** Linux マシンを SNMP を使って監視する簡単な例をいくつか紹介します。
番外編	**SNMP マネージャを利用してみよう。(1), (2)**

SNMP の詳しい内容については、SNMP Tips を参照してください

3.1.2 SNMP エージェントを動かす環境は?

Linux で SNMP を動かすのですが、Linux や SNMP エージェントと言っても、世の中にはたくさんの種類があります。この説明で利用する環境は次の通りです。

■Linux OS: RedHat 7.0

雑誌の付録に載っているもの、パソコン店、書店などで販売されているもの等から入手可能です。

- RedHat 株式会社から "REDHAT Linux 7J DELUXE"というパッケージが発売されています。(Official Red Hat Linux 7J Update CD 付き) 定価で 12,800 円程度ですが、実売価格では、1 万円前後だと思います。
- 日経 Linux 2000.12 (FTP 版) [日経 BP 社]
- Linux Japan 2000 年 12 月号 (FTP 版) [株式会社五橋研究所発行、株式会社秀和システム販売]
- UNIX USER 2000 年 12 月号 [ソフトバンクパブリッシング株式会社]

などで入手可能です。尚最近(と言ってもちょっと前ですが。。。) RedHat 7.090 Fisher [Public Beta 版] がでていましたが、これとは違います。(動作的には問題無いと思いますが、確認は行っていません。)

■SNMP Agent: ucd-snmp (net-snmp)

RedHat 7.0 J に付属しているものを利用します。GnomeRPM で、System Environment / Daemons の中に ucd-snmp4.1.2-8 という名前であります。インストールしていない場合はこのパッケージを追加してください。無い場合は NET-SNMP からダウンロード可能です。HomePage の [Distribution] からサイトを選択し ucd-snmp-4.1.2.tar.gz をダウンロードしてください。(但し、コンパイル/インストール作業が必要)
尚、ucd-snmp (net-snmp)自身の最新版のバージョンは、現時点で 4.2.1 のようです。コンパイル、インストール方法は基本的には 4.1.2 と同じようですが、構成ファイルの種類、構成などが大きく変更されているようです。(4.1.2 用に設定した構成ファイルでも動作はするようです。)このバージョンに関しては、機会があれば触れたいと思います。
一応、4.1.2 のバージョンについては、ソースコードからの簡単なインストール方法を最後に説明します。尚、コンパイルするには、GNU gcc が必要です。
(ucd-snmp の対応 OS について)

3.1.3 早速 ucd-snmp を動かしてみよう!!

さっそく ucd-snmp が提供する、SNMP エージェントを起動してみましょう。以降の説明では、Linux マシンの名前が、"host"、一般ユーザが "foo" としています。

■ 起動・停止は簡単!!

起動するためには、root になって以下のコマンドを実行します。

```
[foo@host foo]$ su
Password: ********
[root@host foo]$ /etc/init.d/snmpd start
Starting snmpd:                                            [  OK  ]
[root@host foo]$
```

これで、SNMP エージェントが起動します。SNMP エージェントはデーモンプロセスです。
停止するためには、やはり root になって、以下のコマンドを実行します。

```
[foo@host foo]$ su
Password: ********
[root@host foo]$ /etc/init.d/snmpd stop
Shutting down snmpd:                                       [  OK  ]
[root@host foo]$
```

これで SNMP エージェントが停止します。

■ 本当に動いているの???

起動したことを確認するには、以下のコマンドを実行します。

```
[foo@host foo]$ ps -ef | grep snmpd
root     1336        1 0 15:46 pts/0      00:00:01 /usr/sbin/snmpd
foo      2530    2511 0 15:47 pts/4      00:00:00 grep snmpd
[foo@host foo]$
```

上記で、"/usr/sbin/snmpd" というプロセスが SNMP エージェントです。

■ **環境はどうなっているの???**

RedHat 7J に付属する ucd-snmp は、次の場所に格納されています。

/usr/sbin	：SNMP エージェント関連（snmpd など）
/usr/bin	：SNMP 用コマンド群（SNMP エージェントから情報を取得するコマンドなど）
/etc/snmp	：構成ファイル（snmpd.conf）
/usr/share/snmp/mibs	：MIB ファイル群（SNMP エージェントが保持する情報を定義したファイルなど）
/usr/share/man	：ucd-snmp 関連のマンファイル

■ **構成ファイルの作成**

起動/停止方法は、上記で説明した通りですが、実施に値を取得するためには SNMP エージェントが、値を返すための設定が必要になります。そのために構成ファイルを編集します。
構成ファイルの設定項目にはさまざまな物がありますが、ここでは最低限必要な内容だけにします。次に示す構成ファイルを /etc/snmp にコピーしてください。

構成ファイルのサンプル ：（snmpd.local.conf）

サンプルにある内容をそのまま利用して構いません。
念のため、次のようにしてください。既に上記の snmpd.local.conf が /etc/snmp にあるとします。

```
[root@host snmp]$ mv snmpd.conf snmpd.conf.org
[root@host snmp]$ mv snmpd.local.conf snmpd.conf
```

構成ファイルの設定内容を一言でいうと、だれでも SNMP エージェントの情報にアクセスできます。って設定になっています。従って、インターネットに直接接続されているマシンなどには適用しないほうが良いです。（ローカルな環境で行ってください。）

構成ファイルの詳細に興味がある方は、man snmpd.conf を参照してみてください。

これで準備万端整いました。

3.1.4 SNMP エージェントの情報を見てみよう!!

ucd-snmp のパッケージは良くできていて、単に SNMP エージェントだけでなく、エージェントの情報を問い合わせるコマンド群や、SNMP エージェントの拡張のためのライブラリ提供など機能が豊富です。

SNMP エージェント(snmpd)から情報を取得する方法はいくつもありますが、次に簡単な方法を示します。

```
[root@host sbin]$ snmpwalk   host   public   system
system.sysDescr.0 = Linux host 2.2.16-22 #1 .....
.....
system.sysContact.0 =   foo@host
```

```
.....
[root@host sbin]$
```

上の例では、snmpwalk というコマンドを利用しています。/usr/bin にインストールされています。引数の host は SNMP エージェントの動いているマシン名、public はコミュニティ名ですがおまじないで、public とそのまま使ってください。system は、オブジェクト ID の一つですが、これもおまじないで system としてください。(system 関連の情報群です。)

また上の例で、"system.sysContact.0"と "system.sysLocation.0" という表記がありますが、ここの値("="以降の文字列)は、実は、先程コピーした構成ファイルの内容を表示しています。
コピーした /etc/snmp/snmpd.conf の 22,23 行目あたりに、"syslocation"と"syscontact"の記述があると思います。その値が現在、"xxxxxxxx", "foo@host"になっています。この部分を書き換えてみてください。書き換えが終了したら、 SNMP エージェントを再起動します。再起動には次の方法があります。

```
[root@host sbin]$ /etc/init.d/snmpd restart
Shutting down snmpd:                                       [  OK  ]
Starting snmpd:                                            [  OK  ]
[root@host sbin]$
```

再起動が終ったら、もう一度 snmpwalk を実行してみてください。sysContact / sysLocation の値が設定したものに変化してると思います。

snmpwalk だとたくさんの情報が表示されて探しにくいと思います。特定の情報だけを表示するためには、snmpget コマンドがあります。次のようにして使います。

```
[foo@host foo]$ snmpget host public system.sysLocation.0
system.sysLocation.0 = xxxxxxxx
[foo@host foo]$ snmpget host public system.sysContact.0
system.sysContact.0 = foo@host
[foo@host foo]$
```

上の例で、".0" という表現があります。これはインスタンスインデックスと呼ばれるもので、クラス(オブジェクト)とインスタンスを区別するためのものです。snmpget を使う場合には、オブジェクト ID の後ろに付くおまじない程度に思ってください。(若干異なることもあるのですが。。。オブジェクト ID に対してインスタンスを識別するための番号になります。インスタンスが 1 つの SNMP エージェントに 1 つだけのものは、"0"を付けます。複数のものは"0"以外の番号などを付けます。インスタンスインデックス(インスタンス識別子)と呼んでいます。)

system 関連の情報以外に、インターフェイスの情報、TCP の情報なども取得することができます。インターフェイスの情報(イーサーネットカードの情報だと思ってください。)

```
[foo@host foo]$ snmpwalk host public interfaces
```

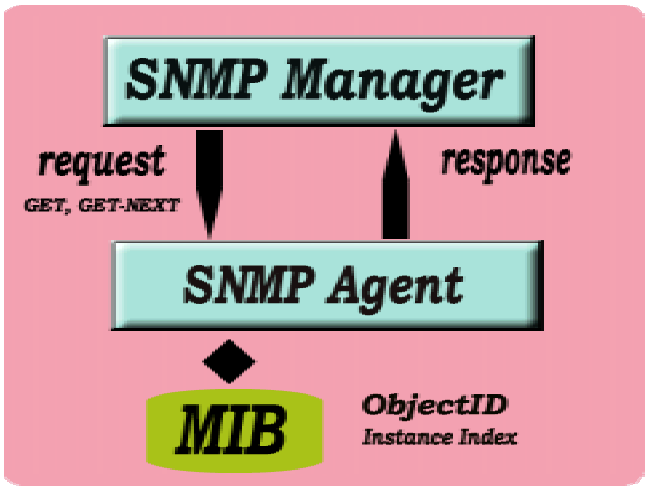
IP の情報の場合および TCP の情報の場合は次のようになります。

```
[foo@host foo]$ snmpwalk host public ip
.....
[foo@host foo]$ snmpwalk host public tcp
.....
[foo@host foo]$
```

(interfaces, ip, tcp など情報群)

ちなみに、SNMP は、UDP を利用したプロトコルです。

今までの説明で関連する単語を図にしてみます。



今回はここまでです。
次回は、ucd-snmp に含まれる他のコマンドの利用方法や、上の図に書いてある単語の簡単な説明などを行います。また情報の内容に関しても少し触れたいと思います。
御意見、御感想などあれば、御連絡ください。今後の内容に反映したいと思います。

3.1.5 ucd-snmp4.1.2 をインストールする場合!!

ucd-snmp 自身は、いろいろなコンパイルオプションがありますが、ここではまず動かしてみることを目的にしていますので、簡単な手順でできる範囲を説明します。
手順は以下の通りです。

- 入手したファイルの展開
- 環境チェック
- コンパイル
- インストール

インストールと構成ファイルの編集は root ユーザで行いますが、それいがいは作業ディレクトリに対して書き込み権があるユーザで行えば良いです。当然 root で実施しても構いませんが、重要なファイルを消したりしないように注意しましょう。以下では host というマシンの foo というユーザで作業します。

入手したファイルの展開

まずは、作業用のディレクトリを作成します。ここでは、/home/work が存在し、ダウンロードしたファイルもそこにおいてあるとして作業することにします。作業するユーザは、このディレクトリ以下に書き込み権があるユーザであれば誰でも良いです。root ユーザでも構いませんが、間違って重要なファイルを消さないようにしてください。
作業用ディレクトリの作成とファイルの展開

```
[foo@host work]$ ls
ucd-snmp-4.1.2.tar.gz
[foo@host work]$ gtar zxf ucd-snmp-4.1.2.tar.gz
[foo@host work]$ ls
ucd-snmp-4.1.2     ucd-snmp-4.1.2.tar.gz
[foo@host work]$
```

以上で、ソースコードの展開が終了しました。次はコンパイルの準備です。

環境チェック(Makefile の作成)

次に、コンパイルを行うための Makefile を作成します。といっても作業は簡単です。

```
[foo@host work]$ cd ucd-snmp-4.1.2
[foo@host ucd-snmp-4.1.2]$ ls
AGENT.txt              PORTING              agent              .........
.........
[foo@host ucd-snmp-4.1.2]$ ./configure
.......
-Press return to continue -            (ここでリターンキーを押します。)
.......
System Contact Information (foo@): foo@host
                    (作成者のメールアドレスを入れてください。[*1])
........
System Location (Unknown): Kawasaki-si   Saiwai-ku Horikawa-cho 580 ......
                    (マシンのある場所/住所などを入力してください。[*2])
........
Location to write logfile (/var/log/snmpd.log):  (ここでリターンキーを押します。)
........
Location to write persistent information (/var/ucd-snmp):  ( ここでリターンキーを押します。)
.........
creating config.h
[foo@host ucd-snmp-4.1.2]$
```

これでコンパイルの準備ができました。

[*1] [*2] SNMP のエージェントの情報の一つを設定します。この情報が構成ファイルに設定されます。後で修正可能なので、適当な文字列を設定しておいても良いです。次はコンパイルです。
デフォルトでインストール先は、/usr/local 以下になります。もしこの場所を変更したい場合は、configure コマンドを実行するときに次の用に引数を追加して実行してください。

```
[foo@host ucd-snmp-4.1.2]$ ./configure  --prefix=/usr/snmp
```

この例では、 /usr/snmp 以下にインストールすることになります。

コンパイル

いよいよコンパイルを行います。これも簡単です。

```
[foo@host ucd-snmp-4.1.2]$ make
.....
[foo@host ucd-snmp-4.1.2]$
```

以上で、コンパイルも終了です。後はインストールです。

インストール

ucd-snmp のインストールするためには、root になる必要があります。作業自体はやはり簡単です。

```
[foo@host ucd-snmp-4.1.2]$ su
Password: *******          (root のパスワードを入力してください。)
[foo@host ucd-snmp-4.1.2]$ make install
.....
[foo@host ucd-snmp-4.1.2]$
```

以上でインストール終了です。
インストール先はおおむね以下の通りです。

/usr/local/include/ucd-snmp
/usr/local/sbin
/usr/local/lib
/usr/local/bin
/usr/local/man
/usr/local/share/snmp

尚、ソースコードをコンパイルした場合、 /etc/init.d/snmpd は存在しません。
従って、起動・停止の方法は以下のようになります。

起動：　/usr/local/sbin/snmpd
停止：　kill ［ snmpd のプロセス ID ］

もしくは、起動スクリプトを /etc/init.d/ 以下にある他の起動スクリプトを真似して作っる事もできます。(ここではその説明は省略致します。)

以上で、ucd-snmp のインストールは終了です。では早速動かしてみましょう。

3.1.6 ucd-snmp の対応 OS について

ucd-snmp (NET-SNMP)が対応している OS は、FAQ の情報によると、以下の通りです。

HP-UX (9.01 to 10.20)

Ultrix (4.2 to 4.5)

Solaris (2.3 to 2.8) and SunOS (4.1.2 to 4.1.4)

OSF (4.0, 3.2)

NetBSD (1.5alpha to 1.0)

FreeBSD (4.1 to 2.2)

BSDi (4.0.1 to 2.1)

Linux (kernels 2.2 to 1.3)

AIX (4.1.5, 3.2.5)

OpenBSD (2.8, 2.6)

Irix (6.5 to 5.1)

Windows に関しては、少し問題があるようです。(詳細は ucd-snmp の FAQ で確認してください。)

3.2 SNMP エージェントの情報ってどんなもの??

3.2.1 はじめに

SNMP エージェントはいかがでしたでしょうか。きちんと動いたでしょうか。
前回は、単純にエージェントを動かす事を目的としていましたが、今回は、もう少し ucd-snmp (net-snmp)が提供しているコマンドについて、少し詳しく説明してみたいと思います。
(ucd-snmp は現在、net-snmp と呼ばれています。)

3.2.2 ucd-snmp のコマンド群

ucd-snmp は、次のコマンドを提供しています。

コマンド名	概要	説明対象	備考
snmptranslate	オブジェクト情報の変換	□	MIB 情報
snmpget	GET 操作	□	
snmpset	SET 操作	―	
snmpgetnext	GET-NEXT 操作	□	
snmptrap	TRAP 通知	―	
snmpwalk	GET-NEXT の連続表示	□	
snmptable	GET-NEXT のテーブル表示	□	
snmpdelta	GET の連続表示(数値)	―	
snmpnetstat	netstat の SNMP 版	□	
snmpstatus	エージェントの主要情報取得	□	
snmpdf	df の SNMP 版	―	*1
snmptest	SNMP 要求／応答のテスト	―	
snmpbulkget	GET-BULK 操作	―	v2c
snmpbulkwalk	GET-BULK の連続表示	―	v2c

snmpusm	SNMPv3 用ユーザセキュリティ管理用	—	v3

*1: snmpdf　は、Linux にバンドルされている ucd-snmp には含まれていないようです

ucd-snmp には、次のようなコマンドが含まれているようです。

- SNMP プロトコルを直接利用したもの
- SNMP プロトコルをベースに便利に表示できるようにしたもの
- SNMP エージェントの情報を元に、UNIX コマンドの機能を実現したもの

今回は、上記表で「説明対象」の欄に□を付けたコマンドの簡単な使い方を説明したいと思います。尚、備考欄に v2c とか v3 とかの記述があります。これは、SNMP プロトコルのバージョンを示しています。備考欄に何も記述していないコマンドは、どのバージョンにも対応しています。ちなみに、現在標準になっている SNMP プロトコルは、バージョン１(初期の SNMP プロトコル)です。SNMPv2c は、実験的な状態(EXPERIMENTAL)、SNMPv3 は、ドラフトスタンダード(DRAFT STANDARD)であり、標準にはなっていません。(これは RFC(Request for Comments：IETF 発行)の話です。)

3.2.3 snmptranslate – オブジェクト情報の変換

このコマンドは、オブジェクト ID からラベル名へまたオブジェクトのラベル名からオブジェクト ID へ変換するコマンドです。またオブジェクトに関する情報も表示できます。

細かい説明は後にして、実際にコマンドを使ってみましょう。

```
[root@host   root] # snmptranslate system
.1.3.6.1.2.1.1
[root@host   root] # snmptranslate system.sysDescr
.1.3.6.1.2.1.1.1
```

system や sysDescr は、前回も出てきたオブジェクト ID です。コマンドの実行結果の番号の並びも実はオブジェクト ID です。コンピュータシステム上でオブジェクトを一意に識別するために .1.3.6.1.2.1.1 などのように数字でオブジェクトを特定しています。system はそのラベル名になります。では次に。

```
[root@host   root] # snmptranslate -On .1.3.6.1.2.1.1
system
[root@host root] # snmptranslate -On .1.3.6.1.2.1.1.1
system.sysDescr
```

上の例は、オブジェクト ID をラベル名に変換しています。（はじめのコマンドの結果の逆）
(ucd-snmp におけるオブジェクト ID の表現方法の補足)

ここで、system と system.sysDescr のオブジェクト ID をよーく見てみましょう。その違いは最後の .1 だと分かると思います。これから、最後の .1 が sysDescr を示していることが推測できると思います。system は、その前の.1 を示します。
では、.1.3.6.1.2.1 の部分は何に対応しているのでしょうか。次のコマンドを実行してみましょう。

```
[root@host   root] # snmptranslate -Ofn .1.3.6.1.2.1.1
.iso.org.dod.internet.mgmt.mib-2.system
```

実は、.iso.org.dod.internet.mgmt.mib-2 の部分が省略されていたのです。SNMP では、MIB-II と呼ぶオブジェクト ID が定義されていて、その オブジェクト ID が、.iso.org.dod.internet.mgmt.mib-2 になります。またオブジェクト ID は、ツリー構造を利用して定義するようになっています。
オブジェクト ID のツリー構造については、テクニカル TIPS 「SNMP の概要」を参照してください。

簡単にオブジェクトIDのツリー構造を見ることもできます。次のコマンドを実行してみましょう。

```
[root@host root] $ snmptranslate -Tp system
+--system(1)
   |
   +-- -R-- String    sysDescr(1)
   |          Textual Convention: DisplayString
   |          Size: 0..255
   +-- -R-- ObjID    sysObjectID(2)
         ••••••••••
         +-- -R-- TimeTicks sysORUpTime(4)
                    Textual Convention: TimeStamp
```

これで、ツリーの構造は概略わかると思います。次のコマンドを実行してみましょう。

```
[root@host root] # snmptranslate -Td system.sysDescr
.1.3.6.1.2.1.1.1
sysDescr OBJECT-TYPE
-- FROM SNMPv2-MIB, RFC1213-MIB
-- TEXTUAL CONVENTION DisplayString
SYNTAX OCTET STRING (0..255)
DISPLAY-HINT"255a"
MAX-ACCESS read-only
STATUS current
DESCRIPTION "A textual description of the entity.   This value should
        include the full name and version identification of the
        system's hardware type, software operating-system, and
        networking software."
::= { iso(1) org(3) dod(6) internet(1) mgmt(2) mib-2(1) system(1) 1 }
```

これは、system.sysDescr オブジェクトの定義を表示しています。ツリー構造の表示部分で String と表示されているのは、SYNTAX で示されるオブジェクトの型を意味します。SIZE はその型の大きさを示します。MAX-ACCESS が示すのが、-R--などののアクセス権です。このオブジェクトがどんな値を表現しているのかは、DESCRIPTION で説明されます。

オブジェクト ID は、全て上のような OBJECT-TYPE で定義されています。その定義を MIB 定義と呼びます。

ちなみに、その MIB を定義したファイルは、ucd-snmp の環境(RedHat7.0J)では、次の場所にあります。

/usr/share/snmp/mibs

上記の system.sysDescr の定義は、このディレクトリの RFC1213-MIB.txt の中で定義されています。RFC1213-MIB.txt は、MIB-II と呼ぶ標準のオブジェクトを定義したファイルです。

このディレクトリには、他にもいろいろなオブジェクトを定義した MIB が格納されています。
これらの MIB を定義する（標準化）する団体が IETF で、定義書が RFC と呼ばれるものです。
snmptranslate の他の利用方法としては、完全なオブジェクト ID が分からないときそれを補間してくれる機能もあります。

```
[root@host root] # snmptranslate -Ofn -Ib 'sys.*'
.iso.org.dod.internet.mgmt.mib-2.system
[root@host root] # snmptranslate -Ofn -TB 'sys.*'
.iso.org.dod.internet.mgmt.mib-2.system
.iso.org.dod.internet.mgmt.mib-2.system.sysDescr
   ・・・・・・・・・・・
.iso.org.dod.internet.snmpV2.snmpModules.snmpMIB.snmpMIBConformance. ¥
        snmpMIBGroups.systemGroup
```

-Ib は、候補を１つだけ表示します。-TB は、該当する候補を全て表示します。
利用したコマンド（引数）をまとめると以下のようになります。

引数なし　指定したオブジェクトのオブジェクト ID を示す。

-On　　指定したオブジェクト ID のラベル名を示す。

-Ofn　　指定したオブジェクト ID の完全なラベル名を示す。

-Tp　　指定したオブジェクト以下のツリー構造を示す。

-Td　　指定したオブジェクトの MIB 定義を示す。

-Ofn -Ib　指定したあいまいなオブジェクト名にマッチするオブジェクトを示す。

-Ofn -TB　指定したあいまいなオブジェクト名にマッチするオブジェクトを複数示す。

3.2.4 snmpget – GET 操作

snmpget は、前回もすこし登場したコマンドで、SNMP プロトコルの GET 操作を実行するコマンドです。これは、引数で指定したオブジェクト ID に対応する値を取得するコマンドです。
使い方は簡単で次のとおりです。

```
[root@host root] # snmpget host public system.sysDescr.0
system.sysDescr.0 = Linux host 2.2.16-22 #1 Tue Aug 22 16:16:55 EDT 2000 i586
```

snmpget <ホスト名> <コミュニティ名> <オブジェクト ID>

<ホスト名>　　SNMP エージェントが動作しているマシン名を指定

<コミュニティ名>　public を指定（SNMP エージェントで定義したコミュニティ名）

<オブジェクト ID>　取得したいオブジェクトのインスタンス

前回も少し説明しましたが、オブジェクトは抽象的なものを示します。あるマシン（この場合は

host）の system.sysDescr オブジェクトの値がほしい場合は、実態であるインスタンスを指定する必要があります。基本的には、オブジェクト ID の後ろに .0 を追加すればよいです。
基本的と表現したのは、オブジェクトの中には、ひとつのマシンの中に同じタイプの情報が複数存在することがあり、その場合は .0 以外を指定することになります。1 以上の数値の場合が多いです。次のコマンド例を参考にしてください。インスタンスを指定するための番号をインスタンスインデックスと呼びます。

```
[root@host root] # snmpget host public interfaces.ifTable.ifEntry.ifDescr.1
interfaces.ifTable.ifEntry.ifDescr.1 = lo0
[root@host root] # snmpget host public interfaces.ifTable.ifEntry.ifDescr.2
interfaces.ifTable.ifEntry.ifDescr.2 = eth0
```

interfaces.ifTable.ifEntry.ifDescr は、インターフェイスを説明する値を意味します。上の例では Ethernet を示しています。lo0 は、ローカルループ、eth0 は、1 つある NIC を示しています。

オブジェクト ID の中に xxxTable という表現があるオブジェクトのインスタンスは、オブジェクト ID の後ろに .0 以外を指定します。インスタンスインデックスがわからないときには、次のコマンドで確認できます。

```
[root@host root] # snmpwalk host public interfaces.ifTable.ifEntry
interfaces.ifTable.ifEntry.ifIndex.1 = 1
interfaces.ifTable.ifEntry.ifIndex.2 = 2
interfaces.ifTable.ifEntry.ifDescr.1 = lo0
interfaces.ifTable.ifEntry.ifDescr.2 = eth0
interfaces.ifTable.ifEntry.ifType.1 = softwareLoopback(24)
interfaces.ifTable.ifEntry.ifType.2 = ethernetCsmacd(6)
      ・・・・・・・・・・・・・・・
```

表示内容をみると、きちんとインスタンスインデックスが表示されていることがわかります。上の例ではオブジェクト ID の最後にある".1"と".2"がそれぞれインスタンスインデックスになります。インスタンスインデックスが実際に何になるかは、MIB(定義)に記述するようになっています。

3.2.5 snmpgetnext – GET-NEXT 操作

snmpgetnext は、SNMP プロトコルの GET-NEXT 操作を実行するコマンドです。これは指定した OID の次に存在する OID の値を取得するコマンドです。オブジェクトの実装は SNMP エージェントに依存しています（部分的に実装しないことも可能）。そんな時に、とにかく指定したオブジェクト（インスタンス）の次に持っている値を返してくれと要求するのが効率よいことがあります。

実際のコマンドは次のように使います。

```
[root@host root] # snmpgetnext host public system.sysDescr.0
system.sysObjectID.0 = OID: enterprises.ucdavis.ucdSnmpAgent.linux
```

snmpgetnext <ホスト名> <コミュニティ名> <オブジェクト ID>

<ホスト名> SNMP エージェントが動作しているマシン名を指定

<コミュニティ名> public を指定（SNMP エージェントで定義したコミュニティ名）

<オブジェクト ID>　値がほしいオブジェクトの前のオブジェクトまたはインスタンス

コマンド引数は、snmpget とまったく同じです。但し、<オブジェクト ID>には、オブジェクトまたはインスタンスのどちらでも指定できます。例えば system.sysDescr (オブジェクト)を指定した場合は次のようになります。

```
[root@host root] # snmpgetnext host public system.sysDescr
system.sysDescr.0 = Linux host 2.2.16-22 #1 Tue Aug 22 16:16:55 EDT 2000 i586
```

このコマンドの結果はインスタンスの値ですが、system.sysDescr(オブジェクト)を指定した場合と、system.sysDescr.0（インスタンス）を指定した場合では、結果が異なります。system.sysDescr の次のインスタンスは、system.sysDescr.0 となりますが、system.sysDescr.0 の次のインスタンスは、（今回の場合）system.sysObjectID.0 となります。

system 以下のオブジェクトのツリー構造が知りたい場合には、snmptranslate -Tp system です。

3.2.6 snmpwalk – GET-NEXT の連続表示

snmpwalk は、前回も登場したコマンドで、snmpgetnext (GET-NEXT 操作)を指定したオブジェクト ID から連続して実行するコマンドです。system 以下のオブジェクトインスタンスの値を表示したい場合には、次のように実行します。

```
[root@host root] # snmpwalk host public system
system.sysDescr.0 = Linux host 2.2.16-22 #1 Tue Aug 22 16:16:55 EDT 2000 i586
system.sysObjectID.0 = OID: enterprises.ucdavis.ucdSnmpAgent.linux
・・・・・・・・・・・・・・・・
system.sysORTable.sysOREntry.sysORUpTime.9 = Timeticks: (4) 0:00:00.04
```

snmpwalk <ホスト名> <コミュニティ名> <オブジェクト ID>

<ホスト名>　　SNMP エージェントが動作しているマシン名を指定

<コミュニティ名>　public を指定（SNMP エージェントで定義したコミュニティ名）

<オブジェクト ID>　値がほしいオブジェクトの前のオブジェクトまたはインスタンス

これは、snmpgetnext コマンドと同じ引数となります。

例えば、SNMP エージェントが保持しているオブジェクトインスタンスの値全てを表示したいときには、次のようにします。(この場合オブジェクトの先頭、ツリー構造の頂点をオブジェクト ID に指定します。)

```
[root@host root] # snmpwalk host public .
system.sysDescr.0 = Linux host 2.2.16-22 #1 Tue Aug 22 16:16:55 EDT 2000 i586
system.sysObjectID.0 = OID: enterprises.ucdavis.ucdSnmpAgent.linux
      ・・・・・・・・・・・・・・・・
.iso.org.dod.internet.snmpV2.snmpModules.snmpVacmMIB.・・・・・・・・
End of MIB
```

オブジェクト ID には、"."を指定しています。これは、ツリー構造のルートを示します。

snmptranslate コマンドでオブジェクト ID を表示したときに、".1.3.6.1.・・・・・"のようにはじめに"."があったと思いますが、ルートを意味していました。

3.2.7 snmptable – GET-NEXT のテーブル表示

snmptable は、テーブル形式の情報をテーブル形式で表示するためのコマンドです。ここでテーブル形式の情報というのは、インスタンスインデックスを少し説明したときに".0"で無い値を指定するものと表現しましたが、その形式のオブジェクトを示します。次のように利用します。

```
[root@host root] # snmptable host public udp.udpTable
udpLocalAddress udpLocalPort
          0.0.0.0              111
          0.0.0.0              161
          0.0.0.0              162
           ・・・・・・・・
```

snmptable <ホスト名> <コミュニティ名> <オブジェクト ID>

<ホスト名>　　SNMP エージェントが動作しているマシン名を指定

<コミュニティ名>　public を指定（SNMP エージェントで定義したコミュニティ名）

<オブジェクト ID>　値がほしいテーブルのオブジェクト ID

突然 udp.udpTable なるオブジェクトが出てきましたが、これは UDP（IP 上のプロトコル）の情報を格納するオブジェクトです。
snmpget のところで interfaces.ifTable・・・というオブジェクトが出てきましたが、この値をテーブル形式で表示するためには次のようにします。

```
[root@host root] # snmptable host public interfaces.ifTable
ifIndex ifDescr ifType ifMtu ifSpeed ifPhysAddress ifAdminStatus
ifOperStatus ifLastChange ifInOctets ifInUcastPkts
ifInNUcastPkts ifInDiscards ifInErrors ifInUnknownProtos
ifOutOctets ifOutUcastPkts ifOutNUcastPkts ifOutDisc
ards ifOutErrors ifOutQLen ifSpecific
1 lo0 softwareLoopback 3924 10000000 up up ? 129413639
1475222
 ? ? 0 ? 129413639 1475222 ?
0 0 0 .ccitt.nullOID
2 eth0 ethernetCsmacd 1500 10000000 0:c0:4f:cb:dc:56 up up ?
1859633310 14274553
 ? ? 687 ? 612432157 5976455 ?
0 0 0 .ccitt.nullOID
```

これは、テーブルに含まれているオブジェクトが多いため表示上非常にわかりづらいですが、はじめにオブジェクトの名前があり、次にインスタンスごとの各値が表示されています。1 行の桁数を指定することもできます。

```
[root@host root] # snmptable -Cw 80 host public interfaces.ifTable
```

```
ifIndex ifDescr ifType ifMtu ifSpeed ifPhysAddress ifAdminStatus
1 lo0 softwareLoopback 3924 10000000 up
2 eth0 ethernetCsmacd 1500 10000000 0:c0:4f:cb:dc:56 up
・・・・・・・・・・
SNMP table interfaces.ifTable, part 4

ifOutDiscards ifOutErrors ifOutQLen ifSpecific
0 0 0 .ccitt.nullOID
0 0 0 .ccitt.nullOID
```

少しだけわかりやすくなります。

3.2.8 snmpnetstat – netstat の SNMP 版

snmpnetstat は、UNIX などの netstat コマンドの処理を SNMP (ucd-snmp のエージェント)の機能を利用して実現したコマンドです。よく使いそうな例を次に示します。

```
[root@host root] # snmpnetstat -i host public 1
input (lo0) output input (Total) output
packets errs packets errs colls packets errs packets errs colls
1570281 0 1570287 0 0 16689806 715 7938163 0 0
24 0 24 0 0 31 0 26 0 0
・・・・・・・・
[root@host root] # snmpnetstat -r host public
Routing tables
Destination Gateway Flags Interface
default router105 UG eth0
・・・・・・・・・
[root@host root] # snmpnetstat host public
Active Internet (tcp) Connections
Proto Local Address Foreign Address (state)
tcp localhost.localdomain.1053 localhost.localdomain.canna ESTABLISHED
tcp localhost.localdomain.1305 localhost.localdomain.5000 CLOSEWAIT
tcp localhost.localdomain.canna localhost.localdomain.1053 ESTABLISHED
・・・・・・・・・
```

オプションとして、

-i　　パケット数を表示
-o　　オクテッと数を表示
-r　　ルーティングテーブルを表示

などが利用できます。

3.2.9 snmpstatus – エージェントの主要情報表示

snmpstatus は、指定した管理対象の主要情報を取得するコマンドです。

```
[xxx.xxx.xxx.xxx]=>[Linux host 2.2.16-22 #1 Tue Aug 22 16:16:55
EDT 2000 i586] Up: 14 days, 1:18:19.82
Interfaces: 0, Recv/Trans packets: 1571613/1571613 | IP:
8082445/7315768
```

これにより簡単にマシンの稼動をチェックできます。システムのアドレス、名前、OS バージョンやシステムのインターフェイス(数)の受信・送信パケット数、IP のパケット数(平均値)がわかります。
概略の情報を取得するには便利かもしれません。
尚、Linux 版の snmpstatus では、インターフェイスの値が正常に表示できません。これは snmpstatus が要求する値の一部を SNMP エージェントが実装していないために起きるようです。（if(In|Out)NUcastPkts が取得できないため）

3.2.10 snmpdf – df の SNMP 版

Linux にバンドルされている ucd-snmp には存在しないと思われます。存在していれば次のように使います。

snmpdf <ホスト名>

しかしながらバンドルされていないコマンドは試せないので、似たような結果を表示する方法を実行してみましょう。

```
[root@host root] # snmptable -Cw 80 host public hrStorageTable
hrStorageIndex hrStorageType
1 host.hrStorage.hrStorageTypes.hrStorageFixedDisk
      ・・・・・・
SNMP table host.hrStorage.hrStorageTable, part 2

hrStorageDescr hrStorageAllocationUnits hrStorageSize hrStorageUsed
/ 4096 Bytes 442492 316662
      ・・・・・・・
SNMP table host.hrStorage.hrStorageTable, part 3

hrStorageAllocationFailures
0
```

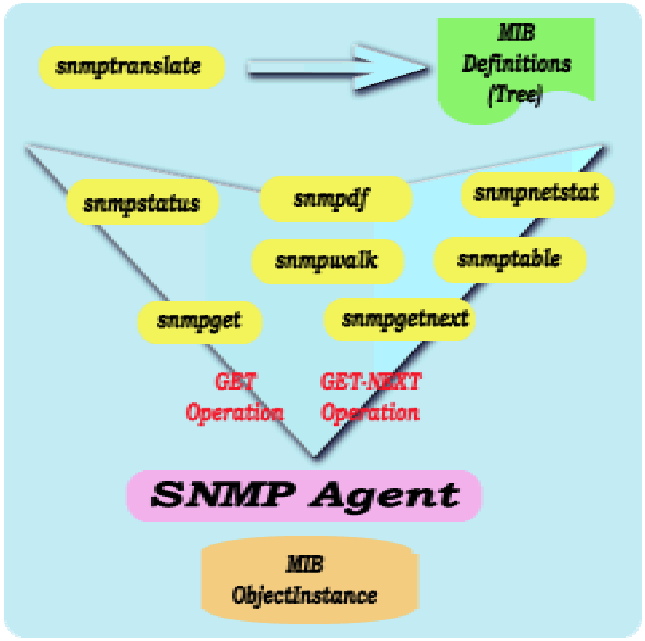
例えば、次のコマンドを実行してみると少し df の結果らしくなります。（ちょっと手抜きですが…）

```
[root@host root] # snmptable -Ib host public hrStorageTable | ¥
   awk 'BEGIN {printf("%25s %20s %12s %12s¥n", "Index", "Disk", "Size", "Used");} ¥
   $1 ~/[0-9]+/ {printf("%25s %20s %12s %12s¥n", $1, $3, $6, $7);}'
   Index                    Disk            Size         Used
      1                       /          442492       372511
      2       /proc/bus/usb       442492       442492
      3               /dev/pts                 0            0
    ・・・・・・・
```

ここで、hrStorageTable は、次のコマンドで調べることができます。

```
[root@host root] # snmptable -Ib -Td hrStorageTable
.1.3.6.1.2.1.25.2.3
hrStorageTable OBJECT-TYPE
-- FROM HOST-RESOURCES-MIB
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION "The (conceptual) table of logical storage areas on
the host.
    ・・・・・・・・・
```

以上が、コマンドの概要でした。今回利用したコマンドについて簡単にまとめます。



なんとなく ucd-snmp のコマンドの利用方法がわかっていただけたでしょうか。今回はコマンドの利用方法のみの説明だったので、実際どのように利用できるかについて、わかりにくかったと思います。

そこで次回は、今回説明したコマンドを利用して、運用に役立つ（かもしれない？？？）コマンド（スクリプト）について、ひとつ二つ考えてみようと思います。

尚、各コマンドの詳細を調べたい場合は、 man page をご利用ください。今回紹介したほかにも便利そうな利用方法（オプション）が載っています。

では。

3.2.11 ucd-snmp におけるオブジェクト ID の表現方法の補足

ucd-snmp のオブジェクト ID の表現方法は、基本的に以下のようになっています。

絶対表記の場合（先頭に"."がくる場合）

例) .iso.org.dod.internet.mgmt.mib-2.system （.1.3.6.1.2.1.1）
これは、標準的な表現です。

相対（省略）表記の場合(先頭に"."がこない場合)

例) system (1)
これは、.iso.org.dod.internet.mgmt.mib-2. が省略されたものとして解釈されます。

例えば、 オブジェクト ID を".1.3"と"1.3"と表現した場合の違いは以下のとおりです。

```
snmptranslate -On .1.3
.iso.org
snmptranslate -Ofn .1.3
.iso.org

        snmptranslate -On 1.3
        system.sysUpTime
        snmptranslate -Ofn 1.3
        .iso.org.dod.internet.mgmt.mib-2.system.sysUpTime
```

また hrStorageTable のオブジェクト ID は、
.iso.org.dod.internet.mgmt.mib-2.host.hrStorage.hrStorageTable ですので、省略形では、
host.hr.Storage.hrStorageTable と表現できます。

SNMP の標準 MIB は、.iso.org.dod.internet.mgmt.mib-2 以下のオブジェクト ID を持つので省略形で表現可能になります。これが利用できないオブジェクト ID の主なものは、ベンダー拡張 MIB と呼ばれるものです。標準の MIB は、基本的にはどの SNMP エージェントも同じように実装され（実装するベンダーによりどこまで厳密に実装しているかは別ですが…）ますが、それ以外の有用な情報を SNMP エージェントに持たせたい場合には、ベンダー拡張 MIB を新たに定義して利用します。

ベンダー拡張の MIB は、次の OID 以下に定義されることが決められています。
.iso.org.dod.internet.private.enterprises
例えば、ucd-snmp の拡張 MIB は、ucdavis(2021) として定義されています。
.iso.org.dod.internet.private.enterprises.ucdavis(2021)
ルータなどで有名な CISCO 社 の拡張 MIB は， cisco(9) として定義されています。
ちなみに弊社の拡張 MIB は、nk-exa (1030) として定義されています。

この番号は、IANA という拡張 MIB 用のオブジェクト ID 割り当て機関に申請することで取得できます。
（ [Application Forms] の [Private Enterprise Number (SNMP)]を利用。）

3.3 SNMP を使って監視をしてみよう

3.3.1 はじめに

前回の内容はいかがでしたでしょうか? きちんと動作したでしょうか。 前回は、 **ucd-snmp** が提供しているいくつかのコマンドについて snmp のプロトコルと対応付けて説明していきました。(ucd-snmp は現在、 **net-snmp** と呼ばれています)

前回までで基本的なコマンドの説明はおしまいとします。今回は、実際に SNMP を利用した Linux box の監視について説明していきたいと思います。
監視対象としては、身近に存在する Linux box として **冷蔵庫**や **家庭用ビデオゲーム機器**、 **PDA** 等が考えられますが:-)、Linux が組み込まれた機器はまだまだ一般家庭には広く浸透していないようです。 (**IPv6** が普及して、 IPv4 での各種制限が取り払われるまでは難しいかもしれませんが... でも今後成長が見込まれる分野ですね)
そこで最近の Linux ブームの中で最も多い使われ方だと思われる、 サーバやルータ(NAT box)としての Linux box の監視について説明してみたいと思います。 (周囲から、「最も多い使用目的は OS インストールだ」という声も聞こえますが:-)

3.3.2 どのようなサービスを提供しているか監視する

ここでは、SNMP agent がもっとも標準的に提供している MIB(mib-2 の system,interfaces,ip,tcp,udp グループ)を使用して、 監視対象のシステムでどのようなサービスを提供しているか監視してみます。 もっとも、このレベルの MIB で監視する場合、実際のサービスではなく、この well-known port で待受けしているようだからこのサービスが立ち上がっているのだろうと判断します。

この目的には、前回解説した **snmpnetstat** を使用します。今回の使用方法の場合、snmpnetstat は、次の 2 つのテーブルから情報 を取得してまとめて表示しています。

mib-2.tcp.tcpConnTable

mib-2.udp.udpTable

以下に使用方法とその実行結果を示します。

```
[root@host root]# snmpnetstat -v1 -a amnesia public
Active Internet (tcp) Connections (including servers) Proto ¥
                                        Local Address Foreign Address (state)
tcp *.ftp *.* LISTEN
tcp *.ssh *.* LISTEN
tcp *.telnet *.* LISTEN
tcp *.finger *.* LISTEN
tcp *.www *.* LISTEN
tcp *.sunrpc *.* LISTEN
tcp *.auth *.* LISTEN
tcp *.login *.* LISTEN
```

```
tcp *.shell *.* LISTEN
tcp amnesia.ssh lily.1024 ESTABLISHED
tcp amnesia.ssh lily.1096 ESTABLISHED .....
Active Internet (udp) Connections Proto Local Address
udp *.sunrpc udp *.snmp .....
[root@host root]#
```

上記の snmpnetstat の結果から、tcp では status が LISTEN を見ると
ftp(21),ssh(22),telnet(23),finger(79),www(80),portmapper(111),ident(113),UCB Rcmd(513,514)
がサービスを提供していることが推測できます。同様に、udp では
snmp(161),portmapper(111)
がサービスを提供していると推測できます。 (snmp リクエストに対して応答しているので、snmp サービスを提供している ことはほぼ自明ですが...)

ただし始めに書いた通り、上記は Listen 状態の port を調べているだけであり、 実際にそこで推測通りのサービスを提供している保証はありません。何らかの 意図により標準とは異なる port で待受けする運用にしている場合もあります。 しかしこれ以上のことを調べるためには、mib-2 以上が必要になります。

また上記程度のレベルなら login して netstat して確認することもできますが、 監視対象が多数になった場合大変ですし、また継続して監視を行うことで、 何らかの要因で意図しないサービスが立ち上がった場合に、これを 早期に検出することが期待できます。

3.3.3 ルータとしての使用状況を監視する

Linux は free であることから最近様々な用途に使用されています。その中でも、 ここでは、Linux をルータとして使用する場合の監視について簡単に説明します。今回は、snmp の mib-2.interfaces が主役です。

mib-2.interfaces は大きく分けて、インターフェースの数(ifNumber)と、 各インターフェースの詳細情報(ifTable)という 2 つのオブジェクトを持っています。

```
mib-2 +-interfaces
        | +-ifNumber インターフェースの数
        | +-ifTable 各インターフェースの詳細情報
            | +ifEntry
                        | +ifIndex
                        | +ifType
                        | +ifOperStatus
                        | +ifInOctets
                        | +ifOutOctets ...
```

それでは最初に interface の状態を監視してみます。これには、前回解説した **snmptable** を使用して次のように行います。

```
[root@host root]# snmptable -v1 amnesia public ifTable
SNMP table: interfaces.ifTable ifIndex ifDescr ifType ifSpeed ¥
                                        ifAdminStatus ifOperStatus ifInOctets ifOutOctets
1 lo0 softwareLoopback 10000000 up up 1432352 1432352
2 eth0 ethernetCsmacd 10000000 up up 760461785 111559824
3 eth1 ethernetCsmacd 10000000 up up 948573284 947583020 4 sit0 other 0 down down 0 0
[root@host root]#
```

上に示した結果は、紙面の都合上テーブルの列の一部抜粋になっています。

上記の結果から、このノードは 2 つの Ethernet インターフェースを持つことが 確認できます (eth0,eth1)。 ルータとして正常に働くためには、通信に使用するそれぞれの network に接続してい る interface が正常に稼働していることを監視する必要があります (もちろん、IP forwarding を許可している必要もありますが。 mib-2.ip.ipForwarding で node の状態は確認できます)。 これは、ifAdminStatus と ifOperStatus により判断できます。ifAdminStatus は、望ましい interface の状態を示します。 ifOperStatus は観測した時点の interface の状態を、 up(1),down(2)などとして報告します。(ifAdminStatus は運用者が設定した状態、ifOperStatus が実際の状態を示します。つまり、interface を up(ifAdminStatus に up を設定)したが、状態(ifOperStatus)は down という場合がある) 上記の出力を見ると、監視した時点では、ルータの 2 つの interface の状態は up であり、 正常に運用しているようだと考えられます。

ただし上の値は観測した時点での状態ですので、 実際は継続して監視する必要があります。 また、interface の状態だけではなく、実際の運用ではパケットが送受信されているか 確認する必要もあります。 これは、 router のケーブルに障害が発生するなどが考えられるのと、 回線使用量を継続的に測定しておくことで、 将来のトラフィックの予測や回線増速などの設備投資の判断材料にするためです。 定期的に観測するために手動で snmptable を実行してもいいのですが、若干大変で す。そこでこの目的のために MRTG を導入します。

MRTG(MULTI ROUTER TRAFFIC GRAPHER) とは、監視対象の node に対して snmp で polling を行い、 その結果を時系列にグラフ化してくれるツールです。またグラフだけではなく、 そのグラフを表示する html ファイルも同時に作成してくれるため、 結果を簡単に表示することができます。

■MRTG を導入する

では早速 MRTG を導入してみましょう。RPM から導入する場合は、 第 1 回の Linux で SNMP エージェントを動かそう!! の snmp エージェントの導入とほぼ同じです。MRTG の rpm ファイルを探してきて追加するだけです。今回は、mrtg-2.9.6-2.i386.rpm を使用しました。
また、自分で source から導入する場合は、先に示した MRTG の WebSite から最新の ソースを取得できます。

■早速 MRTG を動かしてみよう!!

上記でインストールが終了したら早速 MRTG を動かしてみましょう。 以降の説明では、監視対象の Linux box の名を amnesia, 監視する Linux box の名を host とします。また、監視対象では当然 ucd-snmp を稼働しておきます。

最初に監視対象マシン用に構成ファイル(mrtg.conf)を作成します。 MRTG では構成ファイル作成用ツールとして cfgmaker を提供していますので、 これを利用して作成します。

```
[root@host root]# cfgmaker public@amnesia > mrtg.conf
--base: Get Device Info on public@amnesia
--base: Vendor Id:
--base: Populating confcache
--base: Get Interface Info
--base: Walking ifIndex
--base: Walking ifType
--base: Walking ifSpeed
--base: Walking ifAdminStatus
--base: Walking ifOperStatus
 [root@host root]#
```

コマンドの引数には、監視対象の community 名とホスト名を@で区切って指定します。 結果は標準出力に出力されますので、リダイレクションして mrtg.conf に保存します。

このコマンドを実行すると対象ホストに snmp で問い合わせを行い、全 interface を 含む構成ファイルのテンプレートを作成します。 そこで監視を行いたい interface が polling 対象になるように構成ファイルを変更 します。また、この構成ファイルには polling により取得した時系列のデータや、 グラフファイル、html ファイルの保存ディレクトリ(WorkDir) が指定されていませんので、合わせて修正します。

また、作成された html ファイルを Web サーバ (**Apache** 等)で公開するためには、 このディレクトリを DocumentRoot 以下から参照できるように設定します。

ここに今回作成した構成ファイルのサンプルを示します。

構成ファイルが完成したら、データの収集を行ってみます。 データ収集を行うには、mrtg コマンドを使用します。 このコマンドを実行すると、設定ファイルに従い SNMP を利用して ifInOctets,ifOutOctets,sysUpTime,sysName の各値を収集します。

```
[root@host root]# mrtg mrtg.conf Rateup
WARNING: /usr/bin/rateup could not read the primary log file for amnesia_2 Rateup
WARNING: /usr/bin/rateup The backup log file for amnesia_2 was invalid as well Rateup
WARNING: /usr/bin/rateup Can't remove amnesia_2.old updating log file Rateup
WARNING: /usr/bin/rateup Can't rename amnesia_2.log to amnesia_2.old updating log
[root@host root]#
```

コマンドの引数には作成した構成ファイルを指定します。 最初の実行では、上記のように警告が表示されますが、 値の差分をとるための前回のデータなどが存在しないために表示されているだけなので気にする必要はありません。 必要なファイルが作成された後、出力されなくなります。

正常に動作した場合、mrtg は各インターフェース毎にトラヒック情報の時系列を 表示する html ファイルを作成します。

上記までで mrtg コマンドが正常に動作し、値を取得できることが確認できたと思います。 しかし mrtg は実行したときにしかデータの収集、グラフ作成を行いませんので、 継続して時系列の監視を行うためには定期的に手動で起動する必要があります。 これは大変ですので、最後の設定として、mrtg によるデータ収集を cron により自動化 します。

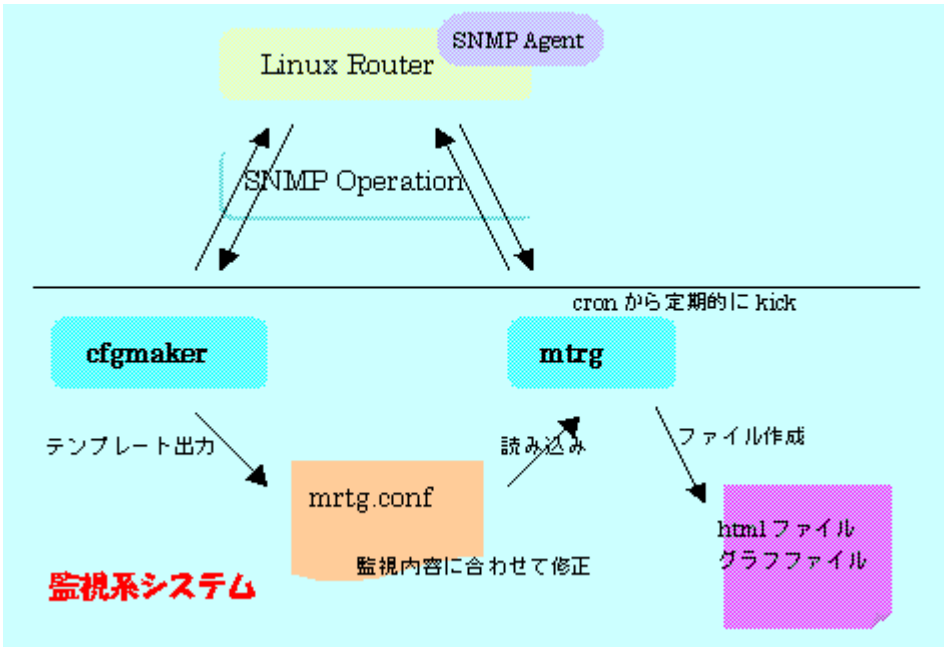
このために crontab に下記のような行を追加します。 下記は 5 分毎にデータを取得するときの例です。 構成ファイルへのパスは実際の設定に合わせて設定してください。

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * * /usr/bin/mrtg /etc/mrtg/mrtg.conf
```

cron に関する詳細は、man cron 等で確認して下さい。

これで指定したディレクトリ以下に定期的にインターフェース毎のトラフィック 情報のグラフが作成されます。作成されるグラフの例を **ここ**に示します。

また、下記に上記の流れを簡単に図にまとめて記します。



■MRTG のグラフによる継続的監視

さて、作成されたグラフを見てみましょう。 まず最初にグラフの簡単な見方を書きます。縦軸はそれぞれの時間あたりの 通信量です。横軸は時間経過で、mrtg のデフォルトでは、新たに取得した データが左側に表示され、時間と共にグラフ全体が右側にスクロールしていきます。 またこのため時間の表示が右ほど古くなります。 もしグラフを左右逆の形式(右側が新しい)にしたい場合、構成ファイルの Options の growright を設定します。(サンプルの構成ファイルでは、 コメントになっています)

先ほどの snmptable による ifTable の 監視では、観測した瞬間の状態しかわかりませんが、これにより時間軸方向での 監視が簡単にできます。例えば今回のグラフの場合、インターフェースを通過す るデータ量は通常の勤務時間帯が主で、それ以外の時間帯は大きく減少することが わかります。また、ifOperStatus が up であっても、 通常データの出入りがある時間帯にもかかわらず何もない場合、何らかの障害が 発生していることが推測できます。もちろんこのことを推測するためには、 普段から通信量のトレンドをつかんでおくことが重要になります。

また通信量の傾向をつかんでおくことで、トラフィックの増加に対する 回線容量不足にも予め対策をとっておくこともできますし、 回線増強をするときの判断基準にもなります(「なんか最近回線重い」のような定性的ではなく、 これこれこうだから重いと定量的判断ができる(ので、 上役を説得しやすい?))。

以上で今回の説明はおわりです。 かなり駆け足の説明になってしまいましたが、 なんとなくイメ

ージをつかんでいただけたでしょうか。

尚、各コマンドの詳細を調べたい場合は、man page をご利用下さい。 今回紹介した他にも便利そうな利用方法がいろいろ載っています。

さて、今回で 3 回続いた実用 SNMP(Linux バージョン)も無事最終回を迎えました。 いままでご清聴いただきありがとうございました。少しでもみなさまのご安全な ネットワーク運用に役立てていただければと思います。また何か機会をみつけて 新しい切口で切り込んでいきたいと思います。

...と上で書きましたが、早速で恐縮なのですが次回番外編を予定しています。 今までは基本的なコマンドを利用したり、 種々のツールを組み合わせての監視を紹介してきました。 番外編では、SNMP マネージャを使用した運用管理について、具体的には弊社 DualManager を利用したネットワーク管理について紹介していきたいと考えています。

第4章　特集

4.1 MIB-2 概説

4.1.1 はじめに

MIB-2 についての解説を行います。
今回の特集では、MIB-2 として定義されているオブジェクトの意味を記述します。
これらの情報は、SNMP 関連の書籍などにも記述されている内容ですが、SNMP を使用して管理を行うときの基本的な情報ですので参考になると思います。

4.1.2 概要

MIB-2(RFC1213)は、主に TCP/IP デバイスが実装すべき管理項目として定義されています。

■ the System グループ
「System グループには、全ての管理対象（SNMP エージェント実装機器）が実装すべき情報として定義されています。」
コメント：このグループは、管理対象の構成情報に関する（最小限の）基本的な情報が含まれています。

sysDescr	エンティティについてテキスト表示による説明をしています。この値は、システムのハードウェアータイプや、ソフトウェアーの OS や、ネットワー クソフトウェアを表す名前とバージョンを含んでいます。印刷可能な ASCII キャラクタだけで記述することになっています。
sysObjectID	どのようなネットワーク管理サブシステムがエンティティに含まれるかを、ベンダーを示すことによって表し た値です。この値は、SMI の enterprise サブツリー(1.3.6.1.4.1)の下に割り当てられています。これによって、どのような機器が管理されているのかを簡単で正確に知ることが出来ます。例えば、ベンダー"FlintstonesInc."が 1.3.6.1.4.1.4242 に割り当てられていたとすると、このメーカのルー タ"FredRouter"の sysObjectID は 1.3.6.1.4.1.4242.1.1 となります。
sysUpTime	システムのネットワーク管理の部分が最後に初期化されてから経過した時間を、100 分の 1 秒の単位で表した値 です。
sysContact	当該被管理ノードについて、コンタクトを取ればよい管理者の名前と、コンタクトの方法を表します。
sysName	当該被管理ノードにつけた管理用の名前です。 慣習で、このノードのフルドメイン名をつけます。
sysLocation	当該ノードが何処に有るかを表す値です。 例えば、"telephone closet, 3rd Floor"など。
sysServices	当該エンティティが主にどのようなサービスを提供するかを表します。各サービスを数値で表し、その合計が値になります。この値には最初 0 がセットされます。 このノードが処理を行うそれぞれのレイヤに対し、

	レイヤを L(1 層から 7 層の範囲)で表すとすると、2 の(L-1)乗 を加算していきます。 例えば、主にルーティング機能の処理をする(主にインターネット層のみサービスをする)ノードの値は 4 (即ち、2^(3-1))になります。これ対して、ノードがアプリケーションサービスを提供するホストである時は、72 が値に、即ち (2^(4-1)+2^(7-1))が値になります。 インターネットプロトコルスイートでは、以下のような値を使って計算します。 layerfunctionality 1 物理層(例:リピーター) 2 データリンク層、サブネットワーク(例:ブリッジ) 3 インターネット層(例:IP ゲートウェイ) 4 end-to-end(例:IP ホスト) 7 アプリケーション層(例:メールリレー) OSI のプロトコルを含むシステムでは、5 層、6 層もカウントされることがあります。

■ the Interface グループ

「Interfaces グループの実装は全てのシステムで必須です。」
コメント：このグループは、管理対象ノードのもつネットワークインタフェース（ポート）についての情報です。

ifNumber	当該システムにあるネットワークインタフェースの数を現在の状態とは関係無く表します。

ifTable

ifTable	インターフェースエントリのリストを表します。 エントリの数は ifNumber によって与えられます。
ifEntry	特定のインタフェースの下で、サブネットワーク層でのオブジェクトを持つインタフェースエントリを表 します。
ifIndex	各インタフェースに対するユニークな値を表します。 1 から ifNumber までで値を取ります。この値は、少なくとも、エンティティのネットワーク管理システムに再初期化されてから、次に初期化される まで、一定の値でなければなりません。
ifDescr	インタフェースの情報を含んだ文字列です。 会社名、製品名、ハードウェアインタフェースのバージョン を含んでいます。
ifType	プロトコルスタックのネットワーク層のすぐ下の物理／リンク層プロトコルに従って区別した、インタフェースのタイプです。other(1) regular1822(2) hdh1822(3) ddn-x25(4) rfc877-x25(5) ethernet-csmacd(6) iso88023-cmacd(7) iso88024-tokenBus(8) iso88025-tokenRing(9)

	iso88026-man(10) starLan(11) proteon-10Mbit(12) proteon-80Mbit(13) hyperchannel(14) fddi(15),lapb(16) sdlc(17) ds1(18) e1(19) basicISDN(20) primaryISDN(21) propPointToPointSerial(22) ppp(23) softwareLoopback(24) eon(25) ethernet-3Mbit(26) nsip(27) slip(28) ultra(29) ds3(30) sip(31) frame-relay(32)
ifMtu	インタフェースで send/receive できる最大のデータグラムを、オクッテト数で示したものです。ネットワークデータグラムを転送するのに使われるインタフェースでは、この値が、そのインタフェース を通じて送出できる、ネットワークデータグラムの最大数になります。
ifSpeed	インタフェースの現在のバンド幅(最大転送速度)をビット／秒で表した値です。バンド幅が異常な値だったり、正確な値が出せないインタフェースでは、この値はあまり意味を持たないで しょう。
ifPhysAddress	プロトコルスタックのネットワーク層の、すぐ下にあるプロトコル層のインタフェースアドレスを表します。このようなアドレスを持たないインタフェース(例えば、シリアルラインなど)では、このオブジェクトの値 は長さが 0 の octet 文字列になります。
ifAdminStatus	インタフェースの望ましい状態を表します。 testing 状態では通常のパケットは送出することが出来ません。up(1) down(2) testing(3)
ifOperStatus	インタフェースの、現在のオペレーショナル状態を表します。 testing 状態では、通常のパケットは送出することが出来ません。 up(1) down(2) testing(3)
ifLastChange	インタフェースが現在のオペレーショナル状態になったときの sysUpTime の値を表します。ローカルネットワーク管理サブシステムの初期化が終る前に、現在の状態になった場合、値は 0 になります。
ifInOctets	インタフェースで受け取ったオクテットの数を表します。 フレーミングキャラクタを含みます。
ifInUcastPkts	上位層から渡された、サブネットワークのユニキャストパケットの数を表

	します。
ifInNUcastPkts	上位層から渡された、ノンユニキャストパケット(例えば、サブネットワークのブロードキャストやマルチキ ャストなど)の数を表します。
ifInDiscards	上位層のプロトコルに送れないようなエラーがなかったとしても、廃棄されたインバウンドパケットの数を表します。バッファスペースが無い場合などに、そのような状況が起こります。
ifInErrors	上位層のプロトコルに送れないようなエラーがあるインバウンドパケットの数を表します。
ifInUnknownProtos	インタフェースで受け取ったが、不明かまたはサポートをしていないプロトコルである為に捨てられてしま ったパケットの数を表します。例えば、Ethernet の場合、Ethernetpacket の中の、上位のプロトコルを識別するフィールドが IP となって いない場合など。
ifOutOctets	インタフェースから送出した、フレーミングキャラクターを含むすべてのオクテットの数を表します。
ifOutUcastPkts	サブネットワークのユニキャストアドレスに転送するよう上位レベルのプロトコルから要求されたパケットの 数を表します。 廃棄されたり、送られなかったりしたパケットの数も含みます。
ifOutNUcastPkts	ノンユニキャストアドレス(サブネットワークのブロードキャストアドレスや、マルチキャストアドレスなど) に転送するよう上位レベルのプロトコルから要求されたパケットの数を表します。 廃棄されたり、送られなかったりしたパケットの数も含みます。
ifOutDiscards	送出出来ないようなエラーが無かったとしても、廃棄されたアウトバウンドパケットの数を表します。バッファスペースが無い場合などに、そのような状況が起こります。
ifOutErrors	エラーのために送出されなかったアウトバウンドパケットの数を表します。
ifOutQlen	送出パケットのキューの長さをパケット数で表します。
ifSpecific	インタフェースを構成する特別なメディアについて記述した、MIB の定義を示す値です。例えば、インタフェースがイーサーネットに繋がっているならば、このオブジェクトの値は、イーサーネッ トに特有なオブジェクトを定義した文章を示す値になります。この情報が無い場合は、値はOBJECTIDENTIFIER{00}になります。この値は、文法的に正しく、ASN.1 や BER の実装では、この値を生成したり、認識したり出来なければなりません。

■ the Address Translation (at)グループ

「Address Translation グループは全てのシステムに必須です。
しかしながら、MIB-2 ではこのグループをサポートすることはデプリケイト状態（不賛成：推奨していない）になっています。 即ち、MIB-1 を使用するノードで適用するためだけに残されています。
MIB-2 をサポートするノードでは、ほとんど 使用されないでしょう。MIB-2 やそれから後の MIB 定義では、それぞれのネットワークプロトコルグループは、それ自身のアドレス変換テーブル を

持っています。」
コメント：実際には、MIB-2 をサポートしているノードでも、このグループはサポートされている場合が多いようです。 ただし、ipNetToMediaTable のオブジェクト値と同じ内容となっているようです。

atTable	NetworkAddress を PhysicalAddress に変換するアドレス変換テーブルを表します。 アドレスを変換する変換テーブルを使用しないインタフェースも有ります(例えば、DDN の X.25 はアドレスを 変換するアルゴリズムを持っています)。全てのインタフェースがアドレス変換テーブルを使用しない場合は、アドレス変換テーブルは空になります。 つまり、エントリーを持ちません。
atEntry	各々のエントリは、1 つの NetworkAddress と、対応する PhysicalAddress を持っています。
atIfIndex	エントリが有効なインタフェースの番号を表します。 atIfIndex で参照されるインタフェースは、ifIndex で参照されるインタフェースと同じです。
atPhysAddress	メディアに依存した PhysicalAddress を表します。 このオブジェクトにヌル文字(長さが 0)を設定すると、atTable の中で、対応するエントリを無効に出来ます。 即ち、当該エントリのある変換テーブルから、そのエントリを実際に切り離します。テーブルから無効になったエントリを、エージェントが削除するようにするかどうかは実装上の問題です。従って、管理する側は現在使われなくなったエントリに関するエージェントからの表形式の情報を受け取る用 意をしておかなければなりません。このようなエントリの情報を正しく得るには、該当する atPhysAddress の値を確かめなくてはなりません。
atNetAddress	メディアに依存する PhysicalAddress に対応する NetworkAddress(例えば、IPAddress など)を表します。 NetworkAddress タイプとはいくつかの可能なプロトコルファミリのうちの 1 つのアドレスの選択を表しているが 、現在は Internet のみです。

■ the IP グループ

「IP グループの実装はすべてのシステムで必須です。」

ipForwarding	このエンティティが、受け取ったデータグラムをフォーワーディングするという意味において IP ゲートウェイ として機能するかどうかを表します。 IP ゲートウェイはデータグラムをフォーワードします。IP ホストは(ホスト経由のソースルーティングを除く)フォーワードをしません。 いくつかの被管理ノードに対しては、このオブジェクトは可能な値の一部しか取らないことが有ります。したがって、エージェントは、管理ステーションがこのオブジェクトの値を不適当な値に変えようとした時には、"badValue"の返答をするほうが良いでしょう。 forwarding(1)

	not-forwarding(2)
ipDefaultTTL	トランスポート層のプロトコルで TTL の値が与えられなかった時に、このエンティティで発生したデータグラムの IP ヘッダーの Time-To-Live フィールドに設定されるデフォルトの値を表します。IP パケットヘッダの TTL の値はゲートウェイを通過するごとに 1 減少されていきます。0 になったことをゲートウェイが知るとそのパケットは捨てられます。
ipInReceives	エラーのものを含め、インタフェースに到着した全てのインプットデータグラムの数。
ipInHdrErrors	チェックサムエラー、バージョン番号エラー、フォーマットエラー、TTL エラー、IP オプションエラーなど、IP ヘッダにエラーがある為に廃棄されたインプットデータグラムの数を表します。
ipInAddrErrors	IP ヘッダの宛先フィールドの IP アドレスが、この Entity では受け取っても意味のない値になっている インプットデータグラムの数を表します。このカウンタは、無効であるアドレス(例えば、0.0.0.0)や、サポートしていない IP アドレスクラス(例えば、クラス E)を持っているデータグラムの数も含んでいます。IP ゲートウェイでないエンティティ、つまりデータグラムをフォワードしないエンティティでは、宛先アドレスがローカルのアドレスではないために破棄されたデータグラムの数を含んでいます。
ipForwDatagrams	当該エンティティが最終の IP 宛先ではないインプットデータグラムの数を表します。データグラムを最終の宛先にフォワードする為、経路を探すことによって、当該エンティティが最終の IP 宛先ではないことが分かります。 IP ゲートウェイとして動作しないエンティティでは、この値は当該エンティティ経由のソースルーティングの パケットで、ソースルーティングオプションの処理が正常終了したものの数だけを表します。
ipInUnknownProtos	正常に目的地に到着したが、未知かまたはサポートしていないプロトコルが指定されていたために捨てられたローカルアドレスの指定してあるパケットの数を表します。
ipInDiscards	以後の処理を続けるのに問題はないが、捨てられた IP データグラム(例えば、バッファスペースが足りなかったりした時)の数を表します。データグラムの組み立て中に捨てられたデータグラムの数は含みません。
ipInDelivers	IP のユーザプロトコル(ICMP も含みます)に正常に渡されたインプットデータグラムの数を表します。
ipOutRequests	ローカルの IP のユーザプロトコル(ICMP も含みます)から、送出するために、IP に渡された IP データグラムの数 を表します。 この値には、ipForwDatagrams でカウントされたデータグラムの数は加算されません。
ipOutDiscards	送出するのに問題はないが、捨てられた(例えば、バッファスペースが足りなかったりした時)アウトプット IP データグラムの数を表します。

	ipForwDatagrams でカウントされたデータグラムの中で、このようなかたちで廃棄されることになったものも カウントします。
ipOutNoRoutes	宛先に転送する為の経路が判明しなかった為に廃棄された IP データグラムの数を表します。ipForwDatagrams でカウントされていて、"no-route"規準に当てはまるパケットもカウントされます。 全てのデフォルトゲートウェイがダウンしている為に、ホストがルーティング出来なかったデータグラムも含 みます。
ipReasmTimeout	このエンティティで、受け取ったデータグラムを組み立てるために、フラグメントを保持する最大の秒数を表 します。2 つのネットワーク A と B があり、その間にゲートウェイ G があって、A と B の最大データグラム長をそれぞれ m,n(m>n)とします。ネットワーク A から B へ長さ l(m>l>n)のデータグラムを転送する場合、G では長さ l の データグラムを、長さ n 以下のデータグラムにフラグメント化して、ネットワーク B に送出します。フラグメント化されたデータグラムはネットワーク B の目的ノードで組み立てられ、上位のプロトコルに渡され ます。ipReasmTimeout 時間内に、もとのデータグラムを組み立てるために必要なすべてのフラグメントが到着しない ときには、フラグメントすべてが捨てられます。
ipReasmReqds	受け取った IP フラグメントの中で、このエンティティで組み立てる必要のあるものの数を表します。IP ヘッダに、このデータグラムはフラグメント化されたものかどうかを示すフィールドがあり、そこで識別さ れます。
ipReasmOKs	組み立てに成功した IP データグラムの数を表します。
ipReasmFails	IP 組み立ての過程で検出された不具合(例えば、タイムアウト、エラーなどどんなものでも)の数を表します。 このカウンタの値は、捨てられた IP フラグメントの数である必要はありません。 なぜなら受け取ったフラグメントを結合し、フラグメントの数が分からなくなっても良いアルゴリズムもある からです(例えば、RFC815 など)。
ipFragOKs	当該エンティティで正常にフラグメント化された IP データグラムの数を表します。
ipFragFails	当該エンティティでフラグメント化する必要があったのに、フラグメント化できなくて、捨てられた IP データグラムの数を表します。例えば、IP データグラムの"Don'tFragment"フラグがセットされていた場合などが有ります。
ipFragCreates	当該エンティティでフラグメント化した結果、生成された IP データグラムフラグメントの数を表します。

the IP address Table

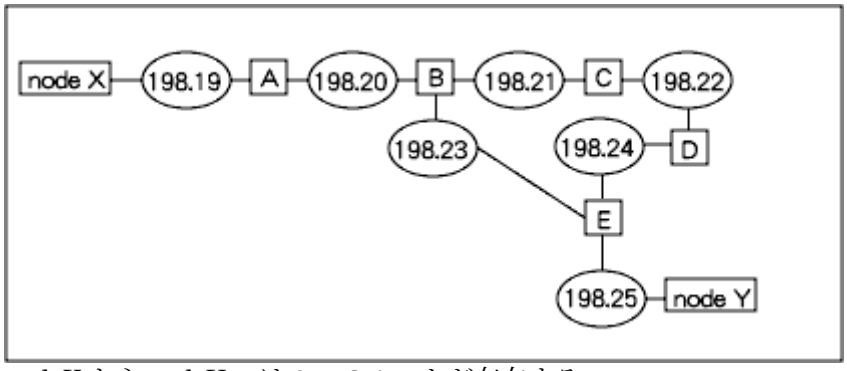
ipAddrTable	当該エンティティの IP アドレスに関するアドレス情報のテーブルです。
ipAddrEntry	当該エンティティの各 IPAddress に関するアドレス情報です。
ipAdEntAddr	IP アドレスを表します。

	当該エントリのアドレス情報は、この IP アドレスについてのものです。
ipAdEntIfIndex	このエントリが適用できるインタフェースを一意に識別する index の値です。この値で識別されるインタフェースと ifIndex の値で識別されるインタフェースは同じです。
ipAdEntNetMask	当該エントリの IP アドレスに対応しているサブネットマスクです。 このマスクは、すべてのネットワークビットを 1、ホストビットを 0 にした IP アドレスです。32 ビットの IP アドレスのうち 2 バイトをネットワーク ID、1 バイトをサブネットワーク ID、 1 バイトをホスト ID に割り当てるとすると、サブネットマスクは 255.255.255.0 となります。
ipAdEntBcastAddr	当該エンティティの IP アドレスに関連した論理インタフェースにデータグラムを送出するのに使われる IP ブロードキャストアドレスの LSB の値です。例えば、Internet 標準の、全てのビットが 1 のブロードキャストアドレスを使用する場合は、この値は 1 にな ります。 この値はこの論理インタフェースを持つエンティティによって使われるサブネットやネットワークのブロードキャストアドレスに適用されます。 Internet の場合とは違い 0 がブロードキャストに使用されることもあります。
ipAdEntReasmMaxSize	このインタフェースで受信した IP フラグメントデータグラムから、このエンティティが再編成することので きる最も大きな IP データグラムのサイズを表します。

the IP route table

ipRouteTable	当該エンティティの IP ルーティングテーブルです。
ipRouteEntry	宛先別の経路情報を表します。
ipRouteDest	この経路の宛先 IP アドレスです。 0.0.0.0 というアドレスはデフォルトルートとみなされます。一つの宛先に対して複数の経路がテーブル内にあることもあります。 しかし、そのような複数のエントリーに対するアクセスは、使用しているネットワーク管理プロトコルのテーブルアクセス機能に依存しています。
ipRouteIfIndex	この経路で次にホップするのに通過するローカルインタフェースを一意に識別するインデックスの値です。ipRouteIfIndex で識別されるインタフェースは、ifIndex で識別されるインタフェースと同じです。

<table>
<tr><td>ipRouteMetric1</td><td>ipRouteMetric1 は当該経路の最初のメトリックを表します。 このメトリックの意味はこの経路の ipRouteProto で表しているルーティングプロトコルに従って決められます。 このメトリックが使用されない場合は、値は-1 に設定されます。

例えば次のようなネットワーク間接続があったとき、
 

nodeX から nodeY へは 2 つのルートが存在する。
X->A->B->E->Y
X->A->B->C->D->E->Y
各ネットワーク間の Metric は 1 であるとすると（通常 1）、 A のルーチングテーブルは次のようになる。
<table>
<tr><th>Dest</th><th>IfIndex</th><th>Metric1</th><th>Metric2</th><th>Metric3</th><th>Metric4</th><th>NextHop</th></tr>
<tr><td>198.25</td><td>1</td><td>3</td><td>5</td><td>-1</td><td>-1</td><td>198.20</td></tr>
</table></td></tr>
<tr><td>ipRouteMetric2</td><td>当該経路のメトリックとして ipRouteMetric1 の次に採用されるものです。 このメトリックの意味はこの経路の ipRouteProto で表しているルーティングプロトコルに従って決められます。このメトリックが使用されない場合は、値は-1 に設定されます。</td></tr>
<tr><td>ipRouteMetric3</td><td>当該経路のメトリックとして ipRouteMetric2 の次に採用されるものです。 このメトリックの意味はこの経路の ipRouteProto で表しているルーティングプロトコルに従って決められます。このメトリックが使用されない場合は、値は-1 に設定されます。</td></tr>
<tr><td>ipRouteMetric4</td><td>当該経路のメトリックとして ipRouteMetric3 の次に採用されるものです。 このメトリックの意味はこの経路の ipRouteProto で表しているルーティングプロトコルに従って決められます。このメトリックが使用されない場合は、値は-1 に設定されます。</td></tr>
<tr><td>ipRouteNextHop</td><td>当該ルートで次にホップするインタフェースの IP アドレス。 ブロードキャストメディアを通ってインタフェースへバウンドする場合、この値は、そのインタフェース 上のエージェントの IP アドレスになります。</td></tr>
<tr><td>ipRouteType</td><td>ルートのタイプです。
direct(3)と indirect(4)は、IP アーキテクチャーの直接ルーティングや間接ルーティングを表します。 この値を invalid(2)に設定すると、ipRouteTable の関係するエントリが無効になります。 すなわち、当該エントリの経路から宛先を切り離すことになります。 エージェントがテーブルから無効になったエントリを除去するかどうかは実装上の問題です。 従って、管理ステーションは、エージェントから現在は使われていないエントリに関する表形式の情報が送 られて来ても、それを受け取る用意をしておかなくてはなりません。
other(1)
invalid(2)</td></tr>
</table>

<table>
<tr><td></td><td>direct(3)
indirect(4)

other：次の 3 つ以外のルート
invalid：無効であるルート
direct：この entity がネットワーク間接続されており、 NextHop で指定したノードが別のネットワークである。
(ipRouteMetric の例) indirecthost/network/sub-network</td></tr>
<tr><td>ipRouteProto</td><td>どのように経路を決定するかのルーチングメカニズムを表します。 ゲートウェイルーティングプロトコルに対応する値が有りますが、それは、ホストが、それらのプロトコルを サポートするということを意味している分けでは有りません。
<table>
<tr><td>other(1)</td><td>以下のどのプロトコルにも当てはまらないもの。</td></tr>
<tr><td>local(2)</td><td>ユーザの入力のような、プロトコルによらないもの。</td></tr>
<tr><td>netmgmt(3)</td><td>ネットワーク管理プロトコルによって設定されるもの。</td></tr>
<tr><td>icmp(4)</td><td>InternetControlMessageProtocol の RedirectMessage によって設定されるもの。 RedirectMessage はゲートウェイからホストへ送られ、 IP を送るべき正しいゲートウェイのアドレスを知らせる。</td></tr>
<tr><td>egp(5)</td><td>Exterior Gateway Protocol。AutonomousSystem （1 つの管理組織によって運営される Internetwork）間を接続するコアゲートウェ イどうしのルーティングをおこなうプロトコル。</td></tr>
<tr><td>ggp(6)</td><td>Gateway-gatewayProtocol,egp の以前に使用された。</td></tr>
<tr><td>hello(7)</td><td>HELLO プロトコルにより経路を選択する。</td></tr>
<tr><td>rip(8)</td><td>RoutingInformationProtocol,IGP(InteriorgatewayProtocol) の 1 つ。bsd の gated に実装されている。</td></tr>
<tr><td>is-is(9)</td><td>ISO で規定している管理ドメイン内でのルーティングプロトコル。</td></tr>
<tr><td>es-is(10)</td><td>ISO で規定している管理ドメイン内でのルーティングプロトコル。</td></tr>
<tr><td>ciscoIgrp(11)</td><td>CISCO 社のゲートウェイが実装しているプロトコル。</td></tr>
<tr><td>bbnSpfIgp(12)</td><td>Internet で BBN 製ゲートウェイが実装している IGP プロトコル。</td></tr>
<tr><td>ospf(13)</td><td></td></tr>
<tr><td>bgp(14)</td><td></td></tr>
</table></td></tr>
<tr><td>ipRouteAge</td><td>当該経路が最後に更新されたか、または他の方法で正しいと決められてからの秒数です。経路を決定するのに使用したルーティングプロトコルに依る以外は、"tooold"の意味は規定されていません。</td></tr>
</table>

ipRouteMask	ipRouteDest の値と比較をする前に、宛先のアドレスと論理積をとるマスクを表します。任意のサブネットマスクをサポートしていないシステムのために、エージェントは、対応する ipRouteDest の 値がクラス A、B、C のネットワークに属するかどうかを決定して、ipRouteMask の値を作ります。 その時、次の値を使用します。
ipRouteMetric5	当該経路のメトリックとして ipRouteMetric4 の次に採用されるものです。このメトリックの意味はこの経路の ipRouteProto で表しているルーティングプロトコルに従って決められます。 このメトリックが使用されない場合は、値は-1 に設定されます。
ipRouteInfo	経路は、ipRouteProto の値に有るルーティングメカニズムに依って決められます。このような経路を決定するのに使った特別なルーティングプロトコルについて記述した固有の MIB 定義を示します。 この情報が無い場合、値は OBJECTIDENTIFIER{00}に設定されます。この値は、構文上正しい値です。ASN.1 や BER の実装では、この値を生成したり、認識したり出来なくてはなりません。

the IP Address Translation table

ipNetToMediaTable	IP アドレスから物理アドレスへの変換のための IPAddressTranslationtable です。
ipNetToMediaEntry	物理アドレスに変換される IP アドレスを一つ含むエントリを表します。
ipNetToMediaIfIndex	このエントリのアドレス変換が行われるインタフェースを表します。 このインデックスで示されるインタフェースと、ifIndex で示されるインタフェースは同じです。
ipNetToMediaPhysAddress	メディアに依存した物理アドレスを表します。
ipNetToMediaNetAddress	メディアに依存した物理アドレスに対応する IP アドレスを表します。
ipNetToMediaType	このオブジェクトに invalid(2)を設定すると、ipNetToMediaTable の中の対応するエントリが無効になります。 即ち、当該エントリの変換から、当該エントリに対応するインタフェースを切り離します。エージェントが、テーブルから無効になったエントリを削除するかどうかは実装上の問題です。従って、管理ステーションは、エージェントから、現在使用されていないエントリのテーブル形式の情報を 受け取る用意が出来ていなければなりません。そのようなエントリの情報を正確に解釈するには、対応する ipNetToMediaType の値を調べる必要が有ります。 other(1),--下記以外 invaild(2),--マッピングを無効にする dynamic(3) static(4)

additional IP objects

ipRoutingDiscards	有効だけれども放棄されたルーティングエントリーの数を表します。 理由としては、他のルーティングエントリのためのバッファスペースが足りなくなったことが考えられます。

■ the ICMP グループ

「ICMP グループの実装はすべてのシステムで必須です。」

icmpInMsgs	エンティティが受け取った ICMP メッセージの総数です。これは icmpInErrors でカウントされるものも含みます。
icmpInErrors	エンティティが受け取った、ICMP エラー(ICMP チェックサムエラーや長さのエラーなど)のある ICMP メッセージの数を表します。
icmpInDestUnreachs	受け取った ICMPDestinationUnreachable メッセージの数を表します。ゲートウェイがルーティングテーブル上の目的地アドレスが到達不可能であることを検出すると、ホストに対してこのメッセージを返します。
icmpInTimeExcds	受け取った ICMPTimeExceeded メッセージの数を表します。ゲートウェイが IP データグラムの time-to-live が 0 であることを検出すると、この IP データグラムを捨て、 ソースホストにこのメッセージを送ります。
icmpInParmProbs	受け取った ICMPParameterProblem メッセージの数を表します。ゲートウェイまたはホストが IP データグラムを処理中にヘッダにエラーを発見すると、その IP データグラムは 捨てられ、ソースホストにそれが通知されます。
icmpInSrcQuenchs	受け取った ICMPSourceQuench メッセージの数を表します。ゲートウェイやホストが、バッファ不足や、IP データグラムの到着間隔が短すぎるなどの理由で処理しきれ ないとき、IP データグラムを捨てソースホストにそれが通知されます。
icmpInRedirects	受け取った ICMPRedirect メッセージの数を表します。ゲートウェイ G1 が IP データグラムを受けて、ルーティングテーブルを見て G2 に送らなければならないとき、 もし、G2 とソースホストが同じネットワーク上にあるならば、ソースホストに対して、直接 G2 に IP データ グラムを送るよう、このメッセージを送ります。
icmpInEchos	受け取った ICMPEcho(request)メッセージの数を表します。ディスティネーションアドレスの IP モジュールが活動中であるかを確認します。 このメッセージを受けたときは EchoReply メッセージで応答しなければなりません。
icmpInEchoReps	受け取った ICMPEchoReply メッセージの数を表します。
icmpInTimestamps	受け取った ICMPTimeStamp メッセージの数を表します。

	送り手は、送りだした時刻を刻印して送りだし、受け手はメッセージを受けた時刻と内部処理にかかった時間 を刻印して TimeStampReply メッセージとして返します。
icmpInTimestampReps	受け取った ICMPTimeStampReply メッセージの数を表します。
icmpInAddrMasks	受け取った ICMPAddressMaskRequest メッセージの数を表します。例えば、2 バイトのネットワーク ID、1 バイトのサブネットワーク ID、ホスト ID だとすると、この アドレスマスクは 255.255.255.0 となります。
icmpInAddrMaskReps	受け取った ICMPAddressMaskReply メッセージの数を表します。
icmpOutMsgs	エンティティが送りだした ICMP メッセージの総数です。これは icmpOutErrors でカウントされるものも含みます。
icmpOutErrors	バッファーが足りないというような ICMP で発見された問題のために、エンティティが送出しなかった ICMP メッセージの数を表します。IP がデータグラムをルーティングできないというような、ICMP の外の層で発見されたエラーは、この値には含まれません。
icmpOutDestUnreachs	送りだした ICMPDestinationUnreachable メッセージの数を表します。ゲートウェイが、ルーティングテーブル上の目的地アドレスが到達不可能であることを検出すると、ホストに対してこのメッセージを返します。
icmpOutTimeExcds	送りだした ICMPTimeExceeded メッセージの数を表します。ゲートウェイが IP データグラムの time-to-live が 0 であることを検出すると、この IP データグラムを捨て、ソースホストにこのメッセージを送ります。
icmpOutParmProbs	送りだした ICMPParameterProblem メッセージの数を表します。ゲートウェイまたはホストが IP データグラムを処理中にヘッダにエラーを発見すると、その IP データグラムは捨てられ、ソースホストにそれが通知されます。
icmpOutSrcQuenchs	送りだした ICMPSourceQuench メッセージの数を表します。ゲートウェイやホストがバッファ不足や、IP データグラムの到着間隔が短すぎるなどの理由で処理しきれないとき、IP データグラムを捨てソースホストに ICMPSourceQuench を通知します。
icmpOutRedirects	送りだした ICMPRedirect メッセージの数を表します。 ホストは、redirects メッセージを出せないので、このオブジェクトの値は常に 0 になります。ゲートウェイ G1 が IP データグラムを受けて、ルーティングテーブルを見て G2 に送らなければならないとします。もし、G2 とソースホストが同じネットワーク上にあったときは、ソースホストに対して直接 G2 に IP データグラムを送るよう、このメッセージを送ります。

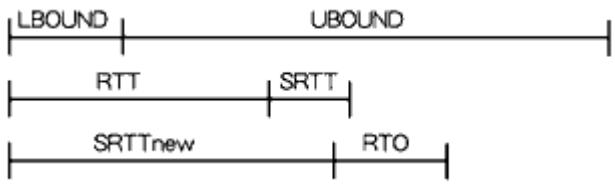
icmpOutEchos	送りだした ICMPEcho(request)メッセージの数を表します。ディスティネーションアドレスの IP モジュールが活動中であるかを確認します。 このメッセージを受けたときは EchoReply メッセージで応答しなければならなりません。
icmpOutEchoReps	送りだした ICMPEchoReply メッセージの数を表します。
icmpOutTimestamps	送りだした ICMPTimeStamp メッセージの数を表します。送り手は、送りだした時刻を刻印して送りだし、受けてはメッセージを受けた時刻と内部処理にかかった時間を刻印して TimeStampReply メッセージとして返します。
icmpOutTimestampReps	送りだした ICMPTimeStampReply メッセージの数を表します。
icmpOutAddrMasks	送りだした ICMPAddressMaskRequest メッセージの数を表します。例えば、2 バイトのネットワーク ID、1 バイトのサブネットワーク ID、ホスト ID だとすると、このアドレスマスクは 255.255.255.0 となります。
icmpOutAddrMaskReps	

■ the TCP グループ

「TCP グループの実装は、TCP を実装する全てのシステムで必須です。 特定の TCP コネクションについての情報を表すオブジェクトタイプのインスタンスは、一時的なものです。 それらは、当のコネクションが張られている間だけ存在します。」

tcpRtoAlgorithm	再送したパケットのタイムアウト値を決定するために使われるアルゴリスムです。

other	以下のどのアルゴリスムにもあてはまらないもの。
constant	コンスタンスな RetransmissionTimeOut 値。
rsre	MIL-STD-1778 で規定されているアルゴリスム。
vanj	VanJacobson のアルゴリスム。 RFC793 で参考として述べられているアルゴリスムは（rsre)、 SRTTnew=(ALPHA*SRTT)+((1-ALPHA)*RTT) RTO=min[UBOUND,max[LBOUND,(BETA*SRTTnew)]]



SRTT	SmoothedRoundTripTime ラウンドトリップタイム予測値
RTT	RoundTripTime ラウンドトリップタイム実測値

	<table><tr><td>RTO</td><td>RetransmissionTimeOut タイムアウト値</td></tr><tr><td>UBOUND</td><td>RTO の最大値(定数) 例：UBOUND=1min</td></tr><tr><td>LBOUND</td><td>RTO の最小値（定数） 例：LBOUND=1sec</td></tr><tr><td>ALPHA</td><td>新しい SRTT を算出するための factor(0-1)</td></tr><tr><td>BETA</td><td>RTT のばらつきを吸収する factor(1.3-2.0) 過去の SRTT と RTT 実測値の差から、新しい SRTT を算出し、 それにばらつきの分を加えて RTO を算出する。</td></tr></table>
tcpRtoMin	TCP の実装によって許されている最小の再送タイムアウト値をミリ秒単位で表します。このタイプのオブジェクトのより詳細な意味は、再送タイムアウトを決定するのに使われているアルゴリズム に依ります。特に、タイムアウトアルゴリズムが rsre の場合は、このタイプのオブジェクトは、RFC793 に述べてある LBOUND 量のことを意味します。
tcpRtoMax	TCP の実装によって許されている最大の再送タイムアウト値をミリ秒で表します。このタイプのオブジェクトのより詳細な意味は、再送タイムアウトを決定するのに使われているアルゴリズム に依ります。特に、タイムアウトアルゴリズムが rsre の場合は、このタイプのオブジェクトは､RFC793 に述べてある UBOUND 量のことを意味します。
tcpMaxConn	エンティティがサポート可能な最大の TCP コネクションの数を表します。最大コネクション数が変化するエンティティでは、この値は-1 になります。
tcpActiveOpens	TCP コネクションが CLOSE 状態から SYN-SENT 状態に直接状態遷移した回数を表します。
tcpPassiveOpens	TCP コネクションが LISTEN 状態から SYN-RCVD 状態に直接状態遷移した回数を表します。
tcpAttemptFails	TCP コネクションが SYN-SENT 状態かまたは SYN-RCVD 状態から CLOSED 状態へ直接状態遷移した回数と、 SYN-RCVD 状態から LISTEN 状態へ直接状態遷移した回数との和を表します。TCP コネクションの確立中に RST(Reset)ビットがセットされた "ConnectionRefused"パケットが相手側から 返ってきたとき、この状態遷移が起こり得ます。
tcpEstabResets	TCP コネクションが ESTABLISHED 状態かまたは CLOSE-WAIT 状態から CLOSED 状態へ直接状態遷移した回数を表します。上位のプロトコルから中止するよう依頼されたとき、RST ビットがセットされたセグメントが送られ、この 状態遷移が起こり得ます。
tcpCurrEstab	現在のコネクションの状態が ESTABLISHED 状態かまたは CLOSE-WAIT 状態である TCP コネクションの数を表します。
tcpInSegs	エラーであるものも含めて受け取ったセグメント数を表します。現在

	確立されているコネクションで受け取ったセグメントを含みます。
tcpOutSegs	送出したセグメントの数を表します。当該コネクション上のセグメントは含みますが、再送データのみのセグメントは含みません。
tcpRetransSegs	再送されたセグメントの数を表します。即ち、送出されたセグメントのうちで、以前１度以上送出されたこと のあるオクテットを含んだセグメントの個数を表します。

the TCP Connection table

tcpConnTable	TCP コネクションに関する情報を持ったテーブルです。
tcpConnEntry	特定の現在の TCP コネクションの情報です。 コネクションが CLOSE 状態に遷移すると、すぐに値は削除されるので、このタイプのオブジェクトは一時的なも のといえます。
tcpConnState	TCP コネクションの状態を表します。 管理ステーションがセットする値は主に deleteTCB(12)です。したがって、もし管理ステーションがこのオブジェクトに他の値をセットしようとしたら、エージェントは "badValue"レスポンスを返すのが妥当です。 もし管理ステーションがこの値に deleteTCB(12)をセットするならば、管理ノードの該当するコネクションの TCB(RFC793に定義されています)を削除し、すぐにコネクションを終了します。 実装上のオプションとして、管理ノードから RST セグメントが、TCP コネクションの他のエンドポイントに送ら れるかも知れません。

closed(1)	コネクションオープンを行い始める
listen(2)	コネクションオープンを行い始める
synSent(3)	コネクションオープン中の状態。
synReceived(4)	コネクションオープン中の状態。
established(5)	データ転送中の状態。
finWait1(6)	コネクションクローズ中の状態。
finWait2(7)	コネクションクローズ中の状態。
closeWait(8)	コネクションクローズ中の状態。
lastAck(9)	コネクションクローズ中の状態。
closing(10)	コネクションクローズ中の状態。
timeWait(11)	コネクションクローズ中の状態。
deleteTCB(12)	コネクションクローズ中の状態。

	deleteTCB(12) コネクションクローズ中の状態。 TCP のコネクションについては、ＴＣＰ接続状態遷移図を参照してください。
tcpConnLocalAddress	TCP コネクションのローカル IP アドレスを表します。ノードに関連するどんな IP インタフェースに対してもコネクションを受け取る listen 状態のコネクションの 場合、この値は 0.0.0.0 を使います。
tcpConnLocalPort	TCP コネクションのローカルポートの番号を表します。
tcpConnRemAddress	TCP コネクションのリモートの IP アドレスを表します。
tcpConnRemPort	TCP コネクションのリモートのポート番号を表します。

additional TCP objects

tcpInErrs	エラー(TCPチェックサムエラー)として受け取ったセグメントの総数を表します。
tcpOutRsts	RST フラグを含んで送出した TCP セグメントの数を表します。

■ the UDP グループ

「UDP グループの実装は、UDP プロトコルを実装しているシステムには必須です。」

udpInDatagrams	UDP のユーザに送られた UDP データグラムの総数を表します。
udpNoPorts	受け取ったが宛先ポートでアプリケーションが起動されていなかった UDP データグラムの数を表します。
udpInErrors	受け取った UDP データグラムのうち、宛先ポートでアプリケーションが起動されていないという理由以外で上位のプロトコルに渡すことができなかったデータグラムの数を表します。
udpOutDatagrams	当該エンティティから送りだした UDP データグラムの数を表します。

the UDP Listener table

udpTable	UDPlistener 情報を含むテーブルです。
udpEntry	現在の特定の UDPlistener の情報です。
udpLocalAddress	この UDPlistner に対するローカルな IP アドレスです。 そのノードに結合したどんな IP インタフェースに対してもデータグラムを受け取る UDPlistner の場合は、 0.0.0.0 が使われます。

udpLocalPort	この UDPlistner に対するローカルポートの数。

■ the EGP グループ

「EGP グループの実装は、EGP プロトコルを実装しているシステムには必須です。」
コメント：EGP グループの実装は、EGP プロトコルを実装しているシステムには必須です。 1 つの管理組織によって管理される複数のネットワークを AS:Autonomy System(自律システム)といい、AS 間を接続するゲートウェイ どうしが自分の管理しているネットワークへの到達可能性を近隣のコアゲートウェイに知らせるためのプロトコルを EGP といいます。)

egpInMsgs	エラーなしで受け取った EGP メッセージの数を表します。
egpInErrors	エラーであると判明した EGP メッセージ数を表します。
egpOutMsgs	ローカルに生成された EGP メッセージの総数を表します。
egpOutErrors	ローカルに生成されたが、EGP エンティティの資源の不足のために送り出されなかった EGP メッセージの数を表します。

the EGP Neighbor table

egpNeighTable	Neighbor として選択されたゲートウェイに関する情報のテーブルです。
egpNeighEntry	特定の EGPneighbor とこのエンティティとの関係についての情報です。
egpNeighState	このエントリの EGPneighbor に関する、ローカルシステムの EGP 状態を表します。 各 EGP 状態は､RFC904 で使用している状態の数値よりも 1 大きい値で表されています。 idle(1) acquisition(2) down(3) up(4) cease(5) EGPNeighbor の状態遷移については RFC904 に記述されています。
egpNeighAddr	このエントリの EGPNeighbor の IP アドレスを表します。
egpNeighAs	隣接 EGP の自律的なシステムを表します。 もし隣接 EGP の自律的なシステム数がわからないならば 0 が記述されます。
egpNeighInMsgs	隣接 EGP からのエラー無しで受け取った EGP メッセージ数を表します。
egpNeighInErrs	隣接 EGP からのエラー(例えば、EGP チェックサムエラー)ありで受け取ったメッセージ数を表します。
egpNeighOutMsgs	ローカルで生成して隣接 EGP に送出したメッセージの数を表

	します。
egpNeighOutErrs	ローカルで生成しても、EGP エンティティの資源の限界のために隣接 EGP に送出しなかったメッセージの数を表します。
egpNeighInErrMsgs	隣接 EGP からの EGP 定義のエラーメッセージの数を表します。
egpNeighOutErrMsgs	隣接 EGP に送出した EGP 定義のエラーメッセージの数を表します。
egpNeighStateUps	EGP の状態が隣接 EGP に対して UP 状態に遷移した数を表します。
egpNeighStateDowns	EGP の状態が隣接 EGP に対して UP 状態から他の状態に遷移した数を表します。
egpNeighIntervalHello	EGP ハローコマンドの再送間隔(1/100 秒単位)を表します。これは RFC904 で定義してある t1tltimer に相当します。
egpNeighIntervalPoll	EGP ポーリング再送間隔(1/100 秒単位)を表します。 これは RFC904 で定義してある t3timer に相当します。
egpNeighMode	EGP エンティティの、passive かまたは active のどちらかのポーリングのモードを表します。 active(1) passive(2)
egpNeighEventTrigger	オペレーターが開始した Start・Stop イベントを実行するのに使われる制御変数です。 この値が読まれる時は、いつも egpNeighEventTrigger に対してセットされた最新の値を返します。 当該ノードのネットワーク管理サブシステムに初期化された後、設定されなければ、stop 値を返します。セットされれば、RFC904 の p.8 から p.10 に記述してあるように、該当する neighbor に Start もしくは Stop の イベントを引き起こします。簡単に言うと、Start イベントによって、Idlepeer は neighbor を獲得し始め、non-Idlepeer は再度 neighbor を 獲得し始めます。 Stop イベントでは、non-Idlepeer は、egpNeighEventTrigger かまたはその他を通じて Start イベントが発生す るまで、Idle 状態になります。 start(1) stop(2)

additional EGP objects

egpAs	この EGP エンティティの自律システム数を表します。

■ the Transmission グループ

コメント：このグループは、伝送媒体固有の MIB を定義するための ID として、予約の意味で定義されています。MIB-2 では定義されていませんが、後に RFC1315（フレームリレー用 MIB）や RFC1643（イーサネット MIB）などの伝送媒体固有の MIB として、定義されています。

■ the SNMP グループ

「SNMP グループの実装は、SNMP プロトコルエンティティをサポートしているすべてのシステムに必須です。 以下に定義されるオブジェクトのいくつかは SNMP の実装によりゼロの値を取る場合があります。 これは管理エージェントとしての、または管理ステーションとしての、どちらかに特有の機能のみをサポート するように SNMP の実装が最適化されている場合です。 特に、以下に記述するオブジェクトは SNMP エンティティに関連したものであり、被管理ノード上には幾つかの SNMP エンティティがあるということなどを理解しておくべきでしょう。 (例えば、もしあるノードがホストとして動作しているなら、管理ステーションとしての動きのことなど。)」

snmpInPkts	トランスポートサービスから SNMP エンティティへ配送されたメッセージの総数を表します。
snmpOutPkts	SNMP エンティティからトランスポートサービスへ渡した SNMP メッセージの総数を表します。
snmpInBadVersions	SNMP プロトコルエンティティへ配送された未サポートの SNMP バージョンを持つ SNMP メッセージの総数を表します。
snmpInBadCommunityNames	SNMP プロトコルエンティティへ配送された、エンティティに未登録の SNMP コミュニティ名を使用した SNMP メッセージの総数を表します。
snmpInBadCommunityUses	SNMP プロトコルエンティティへ配送された、メッセージ中に SNMP コミュニティ名によって許可されていない SNMP オペレーションを記述した SNMP メッセージの総数を表します。
snmpInASNParseErrs	受け取った SNMP メッセージをデコード中に SNMP プロトコルエンティティによって検出した ASN.1 または BER のエラー総数を表します。
snmpInTooBigs	SNMP プロトコルエンティティへ配送された、エラーステータスフィールドの値が tooBig である SNMPPDUs の総数を表します。
snmpInNoSuchNames	SNMP プロトコルエンティティへ配送された、エラーステータスフィールドの値が noSuchName である SNMPPDUs の総数を表します。
snmpInBadValues	SNMP プロトコルエンティティへ配送された、エラーステータスフィールドの値が badValue である SNMPPDUs の総数を表します。
snmpInReadOnlys	SNMP プロトコルエンティティへ配送された、エラーステータスフィールドの値が readOnly である、形式的には正しい SNMPPDUs の総数を表します。エラーステータスフィールドに readOnly の値を含む SNMPPDUs を生成することはプロトコルエラーであるということに注意すべきです。このオブジェクトは、このような不正な SNMP の実装の検出をする手段として用意されています。

snmpInGenErrs	SNMP プロトコルエンティティへ配送された、エラーステータスフィールドの値が genErr である SNMPPDUs の総数を表します。
snmpInTotalReqVars	正しい SNMPGet-Request と Get-NextPDUs を受け取った結果、SNMP プロトコルエンティティによって成功裡に再生された MIB オブジェクトの総数を表します。
snmpInTotalSetVars	正しい SNMPSet-RequestPDUs を受け取った結果、SNMP プロトコルエンティティによって成功裡に変更された MIB オブジェクトの総数を表します。
snmpInGetRequests	SNMP プロトコルエンティティによって受け入れられ、処理された SNMPGet-RequestPDUs の総数を表します。
snmpInGetNexts	SNMP プロトコルエンティティによって受け入れられ、処理された SNMPGet-NextPDUs の総数を表します。
snmpInSetRequests	SNMP プロトコルエンティティによって受け入れられ、処理された SNMPSet-RequestPDUs の総数を表します。
snmpInGetResponses	SNMP プロトコルエンティティによって受け入れられ、処理された SNMPGet-ResponsePDUs の総数を表します。
snmpInTraps	SNMP プロトコルエンティティによって受け入れられ、処理された SNMPTrapPDUs の総数を表します。
snmpOutTooBigs	SNMP プロトコルエンティティによって生成されたエラーステータスフィールドの値が tooBig である SNMPPDUs の総数を表します。
snmpOutNoSuchNames	SNMP プロトコルエンティティによって生成されたエラーステータスフィールドの値が noSuchName である SNMPPDUs の総数を表します。
snmpOutBadValues	SNMP プロトコルエンティティによって生成されたエラーステータスフィールドの値が badValue である SNMPPDUs の総数を表します。
snmpOutGenErrs	SNMP プロトコルエンティティによって生成されたエラーステータスフィールドの値が genErr である SNMPPDUs の総数を表します。
snmpOutGetRequests	SNMP プロトコルエンティティによって生成された SNMPGet-RequestPDUs の総数を表します。
snmpOutGetNexts	SNMP プロトコルエンティティによって生成された SNMPGet-NextPDUs の総数を表します。
snmpOutSetRequests	SNMP プロトコルエンティティによって生成された

	SNMPSet-RequestPDUs の総数を表します。
snmpOutGetResponses	SNMP プロトコルエンティティによって生成された SNMPGet-ResponsePDUs の総数を表します。
snmpOutTraps	SNMP プロトコルエンティティによって生成された SNMPTrapPDUs の総数を表します。
snmpEnableAuthenTraps	SNMP エージェントプロセスが認証失敗(authentication-failure)トラップを生成することを許可されているかどうかを示します。このオブジェクトの値は、どのような構成情報も無視します。つまり、すべての認証失敗トラップをディセーブルにしてもよいという機能を提供しています。 このオブジェクトはネットワーク管理システムの再初期化の間、固定値として残しておくために不揮発性メモリ中に格納することを推奨しています。

参考文献：

邦題：「TCP/IP ネットワーク管理入門」
原著者：M.T.ローズ
訳者：西田竹志
発行元：株式会社トッパン
書籍番号：ISBN4-8101-8521-4
原題：「The Simple Book」

題目：OpenDesign No.10
全力特集：ネットワーク管理技術のすべて
発行元：ＣＱ出版株式会社

4.2 SNMP を利用した FreeUNIX システムの監視方法

4.2.1 はじめに

ここ最近、サーバ OS として Linux や FreeBSD などのフリーUNIX を採用する機会が増えてきているようです。

今回の「特集」では、そのフリーUNIX サーバの稼動状態を簡単に監視する仕組みを作ってみようということで進めていきたいと思います。

通常は、サーバ管理者がシェルなどを利用したプログラムを作成し障害イベントを管理している場合が多いと思いますが、今回は 「SNMP tips」ということで、SNMP を利用したイベント通知方法を紹介していきます。

通知する側（エージェント）は、Java を使用して作成してみようかと思います。Java を使う理由としては、

・プラットフォームの依存度が少ない。

・今回の目的を実現するためのプログラミングが非常に楽である。

・SNMPTrap 送出機能として、Java でプログラミングされたものを利用するから。

です。

また、すべでを Java で記述をするのではなく、条件判断などの部分は、perl か shell で記述しています。

4.2.2 概要

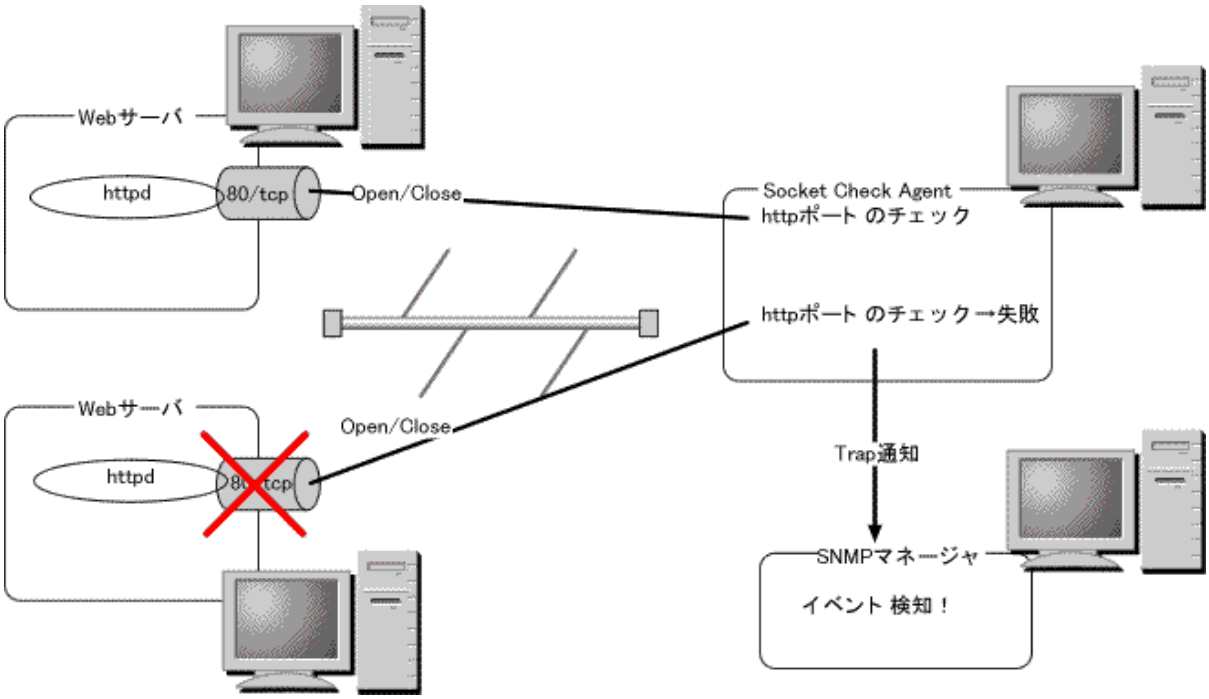
さて、今回の監視対象は、「サーバの稼動状態」ということですが、一口にサーバといっても、いろいろな役割があります。ここでいう「サーバ」とは、「ネットワークを利用してなんらかのサービスを行う仕組み」のこととしています。

身近な例としては、ファイルサーバ，プリンタサーバなど、そのサーバが提供するサービス内容別にいろいろあります。

これらのサーバのなかで、Linux などのフリーUNIX をベースに構築されているものの代表的なものは、Web サーバ（http サーバ）や、メールサーバ（POP サーバや SMTP サーバ）だと思います。

そこで、具体的なターゲットとして、Web サーバ，SMTP サーバ，POP サーバを監視してみることにしてみましょう。

稼動状態の確認方法は、それぞれのサーバが使用している TCP ソケットをリモートから定期的にチェックし、実際にそれらのサーバを利用できる状態にあるかどうかで判断します。この時、異常があると確認できた場合に、SNMP マネージャに対して TRAP を送出します。SNMP マネージャは、この TRAP 受信をすることで、障害を管理者に対して通知することができます。



監視イメージ図

4.2.3 エージェントを作ってみよう！

さて、まず最初は、サーバの TCP ソケット状態を確認する仕組みです。

具体例として、HTTP サーバをターゲットとして進めていきたいと思います。

■TCP ソケット状態の確認：

HTTP サーバは、デフォルトの設定として 80 番の TCP ソケットを利用して、サービスを提供しています。そこで、この TCP/80 番ポートをリモート（クライアント）から Open することが出来たら、「HTTP サーバが正常に稼動している可能性が高い」と判断できます。

実際には、当該ポートに対してリクエスト（ HEAD / HTTP/1.0 など）を送らないと、本当に稼動しているかどうかは判断できないのですが、簡単にするためなので今回は我慢してください。

ここでは、以下のような仕様にしてみました。

引数として、「ホスト名」と「TCP ポート番号」を指定する。

返り値は、

・正常にポートがオープンできたら「1」

・引数のホスト名の名前解決が出来なかったら「2」

・ソケットがオープンできなかったら「3」

・引数のホスト名のレスポンスがなかったら「4」

・その他のエラーが発生したら「5」

を標準出力（コンソール）に出力する。

Java のサンプルソース（ SockStat.java ）があります。（あまり Java らしくないソースですが、ちゃんと動作しますので我慢してください）これを、コンパイルして使用して下さい。コンパイルした結果の CLASS ファイル（SockStat.class）はこちらからダウンロードし、ローカルディスクに保存してください。

実行方法例）

% jre SockStat host 名（ or IPAddress） TCP ポート番号

■MIB の定義：

さて、実は一番の難関がこの「MIB の定義」です。また送出する TRAP-ID も決めなければいけないですね。

Web サーバ（HTTP サーバ）用の MIB は、1999 年 5 月時点で「Internet Draft」として定義されていますが、今回の仕様では、TCP ソケットの状態を見るだけですので、 MIB-2 を利用した方法を考えてみました。

MIB-Object の定義

ここで記述してある内容は、MIB や SNMP, に関してある程度の知識が必要です。ただし、良く分からない場合でも、「こんなものだ」ということで、記述してある通り設定すれば、目的のイベントは確認できると思います。

今回作成するイベントを、TRAP で送出させるためには、TRAP-ID を定義しなければいけません。

また、発生したイベント内容を TRAP で通知するには、VBL という形で、情報を伝える必要があります。

まず、TRAP-ID を決めましょう。今回の企画（「SNMP tips 番外編」）向けに 3 つの TRAP-ID を使えるようにしました。当然、EXA の拡張 TRAP-ID です。安心して使用しましょう。

EnterpriseID は、「1.3.6.1.4.1.1030」です。

generic-ID	specific-ID	Name
6	2300	exaEventOK
6	2301	exaEventNG
6	2302	exaEventOther

ASN.1 記述方法であれば、以下の様になります。

```
exaEventOK      TRAP-TYPE
ENTERPRISE       exa
VARIAVLES        { ????? }
DESCRIPTION      ""
::= 2300
exaEventNG      TRAP-TYPE
ENTERPRISE       exa
VARIAVLES        { ????? }
```

DESCRIPTION ""

::= 2301

exaEventOther TRAP-TYPE

ENTERPRISE exa

VARIAVLES { ????? }

DESCRIPTION ""

::= 2302

注意：本来であれば、ある TRAP-ID の「VARIAVLES」に格納される MIB-Object は一意でなければなりません。

HTTP サーバのソケット状態の確認をした内容を、TRAP の VBL に追加してみましょう。

この場合は、新規に MIB を定義する必要はないと思います。MIB-2 の「tcpConnTable」をそのまま利用できます。また、本当はやっては行けないのですが、system グループも利用してみましょう。

正常な場合：
引数の、「ホスト名（IP アドレス）」の「TCP ソケット番号」がオープンできるということですので、TCP のコネクション状態は「listen」であるということがいえます。ですから、VBL に格納する Object 値は、以下の様にします。

オブジェクト名 （オブジェクト ID）	インスタンス	値
tcpConnState （1.3.6.1.2.1.6.13.1.1）	0.0.0.0.ポート番号.0.0.0.0.0	2(listen)
tcpConnLocalPort （1.3.6.1.2.1.6.13.1.3）	0.0.0.0.ポート番号.0.0.0.0.0	ポート番号

ホスト名が解決できない場合：
引数の、「ホスト名」が名前解決できない場合、その名前をマネージャ側に伝えるために、「sysName」にその名前を入れて通知しちゃいましょう。VBL に格納する Object 値は、以下の様にします。

オブジェクト名 （オブジェクト ID）	インスタンス	値
sysName （1.3.6.1.2.1.1.5）	0	host 名_is_UNKNOWN_HOST

ソケットがオープンできない場合：
引数の、「ホスト名（IP アドレス）」の「TCP ソケット番号」がオープンできないということですので、TCP のコネクション状態は「closed」であるということがいえます。そもそも、そのポート番号は、システム上用意されていないということなのですが．．．。VBL に格納する Object 値は、以下の様にします。

オブジェクト名	インスタンス	値

tcpConnState (1.3.6.1.2.1.6.13.1.1)	0.0.0.0.ポート番号.0.0.0.0.0	1(closed)
tcpConnLocalPort (1.3.6.1.2.1.6.13.1.3)	0.0.0.0.ポート番号.0.0.0.0.0	ポート番号

レスポンスが無い場合:
引数の、「ホスト名」がからレスポンスが無い場合、その名前をマネージャ側に伝えるために、「sysName」にその名前を入れて通知しちゃいましょう。VBL に格納する Object 値は、以下の様にします。

オブジェクト名	インスタンス	値
sysName (1.3.6.1.2.1.1.5)	0	host 名_is_NoResponse

その他エラーの場合:
その他のエラーが発生した場合に、その名前をマネージャ側に伝えるために、「sysName」にその旨を入れて通知しちゃいましょう。VBL に格納する Object 値は、以下の様にします。

オブジェクト名	インスタンス	値
sysName (1.3.6.1.2.1.1.5)	0	otherError

4.2.4 エージェントを動かしてみよう！

さあ、準備が出来たところで、動かしてみることとしましょう。

■１つのコマンドにまとめる：

TCP ソケットチェック用プログラム (SockStat.class) と、SNMP トラップ送出用プログラム (sendtrap.class) を連動して動かすために、K-シェル or Perl を使います。
特に、これらの言語である必要はありません (本当は、全て Java で記述したほうがすっきりすると思います)。

サンプルソースです。
［perl バージョン］：PortCheck_pl.txt（後述）
［k-sh バージョン］：PortCheck_sh.txt（後述）
それぞれのソースは、実行時にファイル名とパーミッションの変更をしてください。
また、環境変数などは、ご使用の環境に合わせて変更してください。

MIB ファイル： rfc1213.asn1
MIB ファイルは、SNMP-TRAP を送出するときに必要になります。サンプルソースファイルの中で、MIB ファイルが存在するパスを指定してください。

実行方法例)

% PortCheck.pl host 名 TCP ソケット番号

「TCP ソケット番号」として指定する番号は、Web サーバ, POP サーバ, SMTP サーバで異なります。

監視対象	TCP ソケット番号
Web サーバを監視する場合	80
POP サーバを監視する場合	110
SMTP サーバを監視する場合	20

■定期的に実行させる:

1回1回、コマンドを入力しないと、状態を確認できないものをエージェントとと呼ぶのは忍びないですね。
そこで、定期的に実行させてみることにしましょう。
UNIX には、コマンドを定期的に実行させるために「cron」という便利なものがあります。
これを利用しない手はないですね。
ちなみに WindowsNT には同様の機能として at というものがあります。
この機能を利用しましょう。
「crontab -e」 で編集します。

crontab サンプル設定内容:

```
15 * * * * /usr/local/PCheck/PortCheck.pl hogehoge 80
```

これで、ホスト名「hogehoge」上の Web サーバの TCP ソケット状態を 15 分ごとにチェックすることができます。

4.2.5 SNMP マネージャの設定

最後は、SNMP マネージャの設定です。

エージェントが送出するイベントを受ける仕組みがないと、障害が発生したことを確認できないですね。
ただ、設定方法は、SNMP マネージャごとに異なりますので、ここでは「WebBasedManager」の設定方法を紹介します。

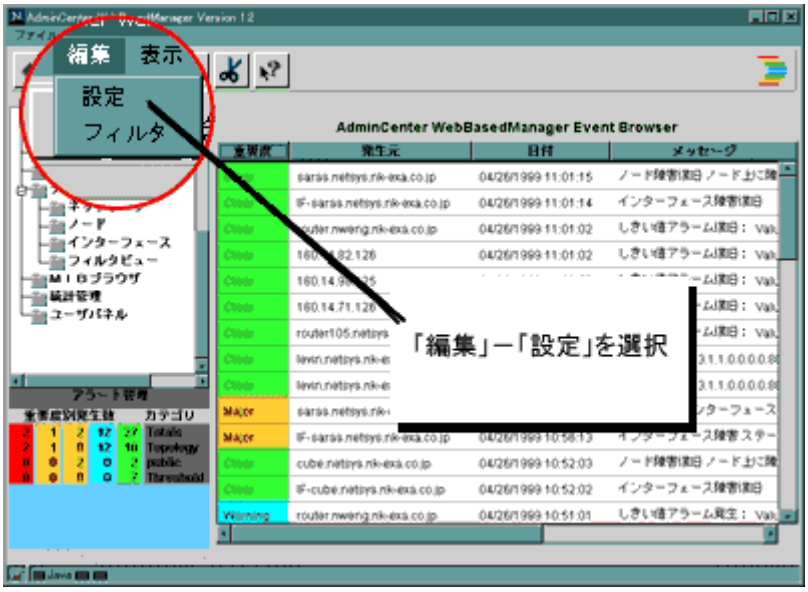
■WebBasedManager の設定：

今回作成したエージェントからのイベントは、SNMP-TRAP で送られてきますので、SNMP-TRAP を的確に受信する仕組みを設定します。
WebBasedManager は、管理対象から送られてきた SNMP-TRAP をイベント（Event）として設定することができます。WebBasedManager のイベントとして設定することで以下のようなことができます。
・ログ
・アイコンの変色
・イベント内容をメールで送信
・コマンドや、独自プログラムの実行
・異なる SNMP マネージャへの SNMP-TRAP の転送
など

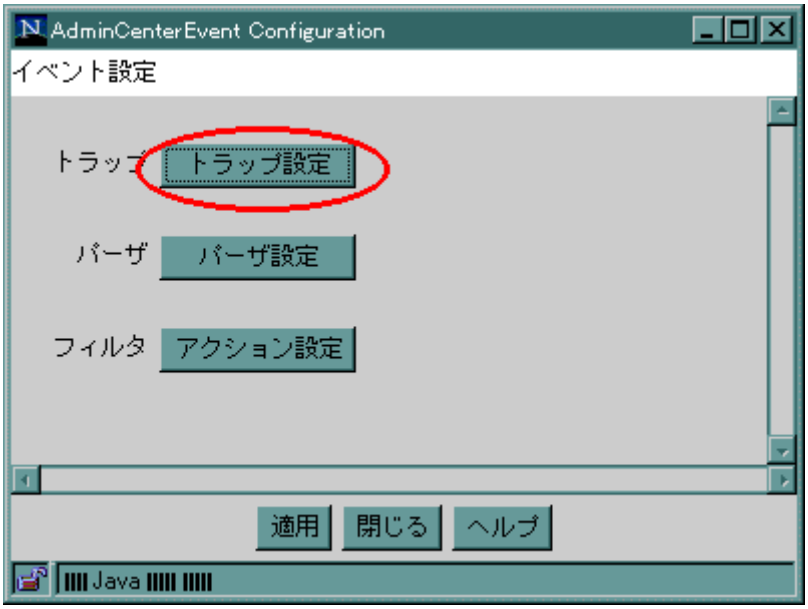
とりあえず今回は、最も簡単な「アイコンの変色」設定方法を紹介しましょう。

■SNMP-TRAP をイベントとして設定する：

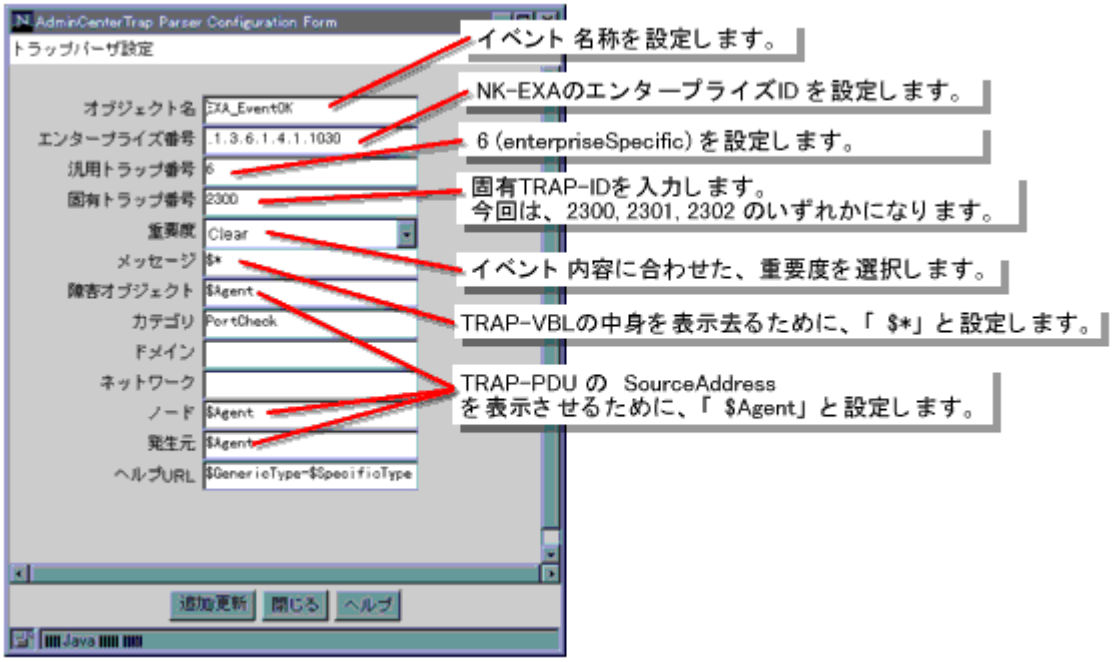
(WebBasedManager1.2J のマニュアルをお持ちの場合は、「第 6 章」も参照してください)
SNMP-TRAP のイベント設定は、イベントブラウザを通して行います。
WebBasedManager の「イベントブラウザ」－「編集」－「設定」を選択し、「イベント設定」ウィンドウを表示します。その後、「イベント設定」ウィンドウの「トラップ設定」ボタンをクリックし、「トラップ入力設定」ウィンドウを表示させます。



「イベントブラウザ」



「イベント設定」

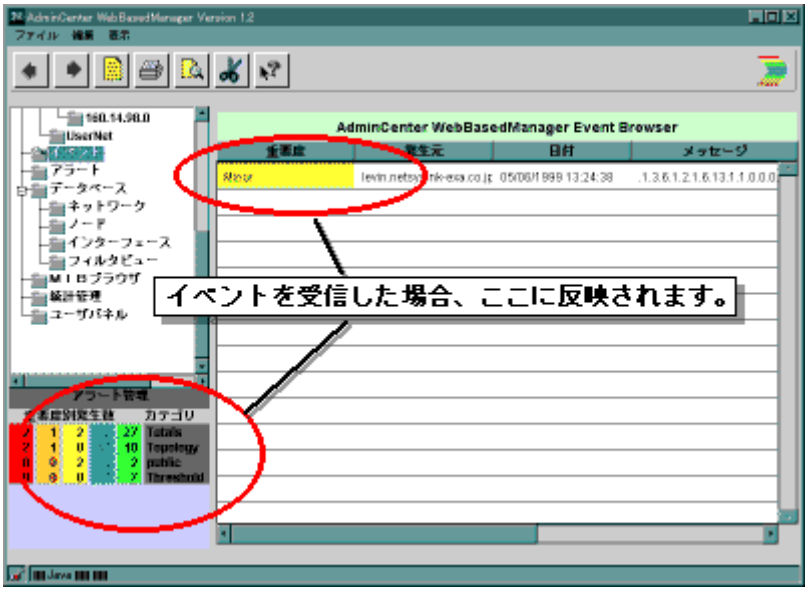


「トラップパーザ設定」

「トラップパーザ設定」ウィンドウに、上図を参考にトラップ番号（イベント内容）別に設定を行えば、WebBasedManager へのイベント設定は終了です。

■イベントの確認：

紹介した内容のイベントを、実際に受信した場合は、下図のように表示されます。



「イベントブラウザ」

このとき、「メッセージ欄」には以下のような書式で状態が通知されます。
表示例の内容については「MIB-Object の定義」を参照してください。

イベント	メッセージ表示例
正常な場合	.1.3.6.2.1.6.13.1.1.0.0.0.0.[ポート番号] .0.0.0.0 : 2, .1.3.6.1.2.1.6.13.1.3.0.0.0.0.「ポート番号」.0.0.0.0: 「ポート番号」
ホスト名が解決できない場合	.1.3.6.1.2.1.1.5.0 :「ホスト名」_is_UNKNOWN_HOST
ソケットがオープンできない場合	.1.3.6.2.1.6.13.1.1.0.0.0.0.[ポート番号] .0.0.0.0 : 1, .1.3.6.1.2.1.6.13.1.3.0.0.0.0.「ポート番号」.0.0.0.0: 「ポート番号」
レスポンスが無い場合	.1.3.6.1.2.1.1.5.0 : 「ホスト名」_is_NoResponse
その他エラーの場合	.1.3.6.1.2.1.1.5.0 : otherError

このメッセージ内容をわかりやすい形式で出力する方法もあるのですが、とりあえず「TRAP-ID」でイベントが発生したことを判断できると思います。

これでめでたく完成です。

■まとめ:

以上のような方法で、サーバ上で稼動しているプロセスが提供するサービスの動作を確認することができます。ちょっと工夫すれば、エージェント側に手を加えて「SNMP-TRAP」ではなく、イベント内容をメールで送信することも出来ることもできますね。
プログラムもそれぞれソースがありますので、どんどん使いやすいように変更してみたらいかがでしょうか？
今回作成したものは、ちょっとした変更が必要ですが WindowsNT でも動作させることも可能です。

■最後に:

今回の特集は、いかがでしたでしょうか？
Web サーバ, POP, SMTP サーバは、いまや正常に稼動しているのが当たり前という状況になっていると思います。ただ、それが「正常に動作しているかどうか」は、「ユーザが第一発見者」である場合が多いのではないでしょうか？
今回の特集が少しでも役に立つことができれば幸いです。

UNIX, Java, Perl, SNMP など、記述内容が多岐にわたっていますのでちょっとわかりにくいかもしれません。
今回のターゲットは、「UNIX 管理者向け」ということでご容赦してください。

A.1 PortCheck_pl.txt

```
#!/usr/local/bin/perl
#
#       PortCheck.pl
#               Usage: PortCheck.pl hostoname portNumber
#
#

#  Prefarence
#
#

# Java Enviroments
#
# $JAVA_HOME : jdk or jre のインストールディレクトリ
$JAVA_HOME = "/usr/jdk1\.1";

# $JRE : jdk or jre のバイナリへのパス
$JRE = "/usr/bin/jre";

# $AVSP      : AdminCenter SNMPpackage のインストールディレクトリ
$AVSP = "/export/home2/SNMPpack/AdventNetSNMPv1";

# PATH      : パスです。jdk や、jre の実行ファイルがあるディレクトリを含めてください。
$PATH = "$JAVA_HOME/bin:$PATH";

# $CLASSPATH : Java の標準の CLASS パスの他に、SNMPpackage の CLASS パスを含めてください。
$CLASSPATH =
"\.:/usr/java1\.1/lib/classes\.zip:$AVSP/classes:/export/home2/SNMPpack/AdventNetSNMPv1/applications:/export/home2/PStest/ht
tpcheck";

# $SENDTRAP : AdminCenter SNMPpackage の TRAP-SENDER クラスの指定
# $AVSP/applications ディレクトリ以下にあります。
#
$SENDTRAP = "sendtrap";

# $RFCFILE  : AdminCenter SNMPpackage がロードする MIB ファイルをフルパスで指定します。
$RFCFILE = "/export/home2/PStest/httpcheck/rfc1213\.asn1";


#
# 日付の取得
# フォーマット例：Mar 23 10:10:34
#
$DATETIME = `date`;
chop( $DATETIME );


# $MANAGER_NAME : SNMP マネージャの ホスト名 or IP-Address
$MANAGER_NAME = "support";

# 引数のチェック

if ( @ARGV != 2 ){

        die "Usage : $myName hostName portNumber \n" ;
}

$HOSTNAME = $ARGV[0];
$SOCKETNUM = $ARGV[1];

#  Socket Check Class ENV
```

```
$ESTRING = `$JRE -classpath $CLASSPATH SockStat $HOSTNAME $SOCKETNUM`;

chop( $ESTRING );

# print "$DATETIME : Start Portcheck\.pl \n";


#
#  TrapSender
#
# eventOK   : .1.3.6.1.4.1.1030 2300
# eventNG   : .1.3.6.1.4.1.1030 2301
# eventOtherq          : .1.3.6.1.4.1.1030 2302

    if( $ESTRING eq "1"){
#       OK の場合
                $S_TRAP=2300;
               $OID="tcpConnState\.0\.0\.0\.0\.". $SOCKETNUM. "\.0\.0\.0\.0\.0 2 tcpConnLocalPort\.0\.0\.0\.0\.". $SOCKETNUM.
"\.0\.0\.0\.0\.0\ ". $SOCKETNUM;

    } elsif( $ESTRING eq "2"){
#       名前解決が出来ない場合
                $S_TRAP = 2301;
                $OID="sysName\.0 ". $HOSTNAME. "_is_UNKOWN_HOST";
                $HOSTNAME = $MANAGER_NAME;

    } elsif($ESTRING eq "3"){
#       ソケットオープンが出来ない場合
                $S_TRAP=2301;
               $OID="tcpConnState\.0\.0\.0\.0\.". $SOCKETNUM. "\.0\.0\.0\.0\.0 1 tcpConnLocalPort\.0\.0\.0\.0\.". $SOCKETNUM.
"\.0\.0\.0\.0\.0\ ". $SOCKETNUM;

    } elsif($ESTRING eq "4"){
#       レスポンスがない場合
                $S_TRAP=2301;
                $OID="sysName\.0 ". $HOSTNAME. "_is_NoResponse";

    } else {
#       その他の場合
                $S_TRAP=2302;
                $OID="sysName\.0 otherError";
    }

# TrapSend ( SNMP Packege )

# print "$DATETIME :  SendTrap for $MANAGER_NAME  \, VBL is $OID \n";

# TRAP 送出プログラムの実行
system "$JRE -classpath $CLASSPATH $SENDTRAP -c public -m $RFCFILE $MANAGER_NAME \.1\.3\.6\.1\.4\.1\.1030 $HOSTNAME 6 $S_TRAP
1 $OID";

print "$DATETIME :  PortCheck\.pl Finished \n";
```

A.2 PortCheck_sh.txt

```
#!/bin/ksh
#
#    PortCheck.sh
#       Usage: PortCheck.sh hostName portNumber
#
#

#  Prefarence
#
#

# Java Enviroments
#
# JAVA_HOME : jdk or jre のインストールディレクトリ
JAVA_HOME=/usr/jdk1.1

# AVSP      : Advent SNMPpackage のインストールディレクトリ
AVSP=/export/home2/SNMPpack/AdventNetSNMPv1

# PATH      : パスです。jdk や、jre の実行ファイルがあるディレクトリを含めてください。
PATH=$JAVA_HOME/bin:$PATH

# CLASSPATH : Java の標準の CLASS パスの他に、SNMPpackage の CLASS パスを含めてください。
CLASSPATH=.:/usr/java1.1/lib/classes.zip:$AVSP/classes:/export/home2/SNMPpack/AdventNetSNMPv1/applications:/export/home2/PSt
est/httpcheck

export  PATH
export  CLASSPATH

# TRAP 送出先の SNMP マネージャのホスト名 or IP アドレス
MANAGER_NAME=levin

# RFCFILE  : AdminCenter SNMPpackage がロードする MIB ファイルをフルパスで指定します。
RFCFILE=/export/home2/PStest/httpcheck/rfc1213.asn1



# 引数のチェック／変数へ格納
if [ $# -ne 2 ]
then
        echo "Usage : $myName hostName portNumber"
        exit 1
fi

HOSTNAME=$1
SOCKETNUM=$2


#  Socket Check Class ENV

ESTRING=`jre SockStat $HOSTNAME $SOCKETNUM`

#
#  TrapSender
# EXA-TRAP-ID
#   exaEventOK                  : .1.3.6.1.4.1.1030 2300
#   exaEventNG                  : .1.3.6.1.4.1.1030 2301
#   exaEventOther      : .1.3.6.1.4.1.1030 2302

    case $ESTRING in
       # OK の場合
```

```
           "1")  S_TRAP=2300
                 OID="tcpConnState.0.0.0.0."$SOCKETNUM".0.0.0.0.0 2 tcpConnLocalPort.0.0.0.0."$SOCKETNUM".0.0.0.0.0
$SOCKETNUM"
              ;;

       # 名前解決が出来ない場合
           "2")  S_TRAP=2301
                 OID="sysName.0 "$HOSTNAME"_is_UNKOWN_HOST"
                 HOSTNAME=$MANAGER_NAME
              ;;

       # ソケットオープンが出来ない場合
           "3")  S_TRAP=2301
                 OID="tcpConnState.0.0.0.0."$SOCKETNUM".0.0.0.0.0 1 tcpConnLocalPort.0.0.0.0.80.0.0.0.0.0 $SOCKETNUM"
              ;;

       # レスポンスがない場合
           "4")  S_TRAP=2301
                 OID="sysName.0 "$HOSTNAME"_is_UNKOWN_HOST"
              ;;

       # その他の場合
           "5")  S_TRAP=2302
                 OID="sysName.0 OTHER"
              ;;
    esac

#
# TrapSend ( SNMP Packege )
#
# TRAP 送出プログラムの実行

jre sendtrap -c public -m $RFCFILE $MANAGER_NAME .1.3.6.1.4.1.1030 $HOSTNAME 6 $S_TRAP 1 $OID

exit
```