Mpd 4. 4. 1 User Manual (5-9)

Archie Cobbs, Michael Bretterklieber, Alexander Motin

訳：北　川　正　一

目　次

1～4章は前号に掲載

5 装置タイプ

5.1 モデム タイプ コマンド

この章ではモデムタイプ回線に特有のコマンドについて記述します。これらのコマンドは現在稼働中の回線に適用され, その回線が**モデム**タイプである場合にのみ有効です。

`set modem device` *devname*

モデム回線はシリアルポート上で機能します。このコマンドは mpd でこの回線のためにどのシリアルポートを使用するかを指定するために必要とされます。*devname* はシリアルデバイスのパス名, 例えば, `/dev/cuad0` 等でなければなりません。

`set modem var` *$variable string*

予めチャットスクリプト変数 *$variable* の値を *string* にセットします。チャット変数についての詳細は チャットスクリプト の章 (6) を見てください。

`set modem speed` *speed*

このコマンドはシリアル装置をオープンするときのシリアルポートの初期速度 (例えば 9600, 57600, 115200) をセットします。チャットスクリプトを利用すれば, その後のシリアルポート速度はいつでも変更できることに注意してください。

`set modem script` *connect-script*

このコマンドはこの回線のための PPP 接続を初期化するために, mpd でどのチャットスクリプトを走らせるかについて決定します。*connect-script* は `mpd.script` の中のラベルと一致しなければなりません。Mpd はシリアルポート装置をオープンした後にこのラベルへジャンプします。もしも *connect-script* がセットされていないと接続のチャット段階はスキップされます (例えば, あなたが直接のヌルモデム接続をする場合等)。

`set modem idle-script` *idle-script*

このコマンドは回線が接続されていないとき, シリアルポートに対し mpd がどのようなコマンドを実行するかについて指定します。*idle-script* が何もセットされていないと, 回線が接続されていない場合, mpd はシリアルポートをクローズされたままにします。そうでなければ, 回線が切れるとき mpd はチャットスクリプト *idle-script* を実行します。このスクリプトが**失敗**で終了した場合, mpd はスクリプトを再実行します。

そうならない (すなわち, **成功**で終了した) 場合は, 次に何をするべきかについて決定するため mpd はチャット変数 `$IdleResult` の内容を調べます。もし, それが `answer` に等しいならば mpd は着信が答えられたと仮定して, すぐに PPP 交渉に入ります。それが `ringback` に等しいならば, mpd はシリアルポートを閉じてそれを再度開き, 通常の発信接続を開始します (すなわち, *connect-script* を使用します)。

もしも `$IdleResult` が他の値となっているか, 何もセットされていない場合, mpd は**エラー**が発生したかのように振る舞い, そして単にスクリプトを再度実行します。

`set modem watch` *+/-signal ...*

Mpd は通常シリアルポートのキャリア探知シグナルを追跡し, そのシグナルが失われたとき, 接続を落とします。`set modem watch -cd` とすることで, この振る舞いを無効にすることができます。また, mpd は DSR 信号で同じことをすることができます。しかしながら, デフォルトでは DSR を無視するようになっています。DSR 信号の監視を有効にするには `set modem watch +dsr` を使用してください。

5.2 Netgraph 装置タイプ コマンド

この章では netgraph タイプ回線に固有のコマンドについて記述します。これらのコマンドは現在稼働中の回線に適用され, 現在稼働状態にある回線がタイプ `ng` を持つ場合のみ有効です。

この装置タイプには書き込みエラーを受け取ること以外に, 物理レイヤー切断 (すなわち, キャリア探知の喪失に類比したもの) を見つける直接的な方法がないことに注意すべきです。これを達成するための 1 つの方法は, 物理レイヤーが落ちるとき Netgraph フックへの接続が断たれるように保証することです。

`set ng node` *nodepath*

`set ng hook` *hook*

これらのコマンドは mpd がどの Netgraph ノード, そして, そのノードでどのフックにつながることになっているかを設定します。ノードは絶対 Negraph アドレス *nodepath* を通して見つかります。そして, そのノードは *hook* と名をつけられたフリーなフックを持っていなければなりません。これらのコマンドは両方とも必要とされます。

フックは `ng_ppp(3)` netgraph ノードのタイプの回線フックに直接接続可能でなければなりません。つまり, (圧縮されていない限り) アドレスと制御フィールドから始まり, PPP プロトコル番号と情報フィールドが続いているが, チェックサムフィールドは含んでいない PPP フレームを送受信する準備ができていなければなりません。

5.3 TCP 装置タイプ コマンド

この章では TCP タイプ回線に固有であるコマンドについて記述します。これらのコマンドは現在稼働状態の回線に適用され, 現在稼働状態の回線がタイプ `tcp` を持つ場合にのみ有効です。
ng_ksocket モジュールの制限のため, この装置タイプには 2 つの問題があります:

- 空いている着信回線がないとき, 着信を受け付けないのではなく, 新しい接続はとにかく受け付けられて, それから落とされます。
- tcp 接続状態はモニターされません。それで, 壊れた回線を探知するために'`set link keep-alive`' コマンドを使用しなくてはなりません。

`set tcp self` *ipaddr* [*port*]

TCP 接続のローカル IP アドレスとポートを設定します。ポートフィールドは受信接続を受け付けるために必要です。

`set tcp peer` *ipaddr* [*port*]

TCP 接続のピアーの IP アドレスとポートを設定します。アドレスとポートフィールドは, 発信接続のために必要です。受信接続のためには, 必要ではありませんが, 接続可能な相手の制限に利用することができます。

`set tcp enable` *option* ...
`set tcp disable` *option* ...

回線のために TCP 装置タイプオプションを有効, 無効にします。

以下のオプションがサポートされます :

`originate`

このオプションはピアーへの TCP 接続の開始を有効にします。このオプションが無効になっていると mpd はいかなる TCP 接続も開始しません。

`incoming`

このオプションは着信 TCP 接続の受諾を有効にします。このオプションが無効にされていると mpd はいかなる着信 TCP 接続も受け入れません。

5.4 UDP 装置タイプ コマンド

この章では UDP タイプ回線に固有であるコマンドについて記述します。これらのコマンドは現在稼働状態の回線に適用され, 現在稼働中の回線がタイプ `udp` を持つ場合にのみ有効です。
より良好なチャンネル活用状態を得たいならば, この回線タイプが IP 断片化を避けて使用されるために `set iface mtu` コマンドの使用が示唆されます。

注：PPP 回線がフレームを再整列させることは許されないので, 技術的にこの装置タイプは完全には正しくありません, それでも, UDP パケットは順序が乱れたまま届けられることが可能です。したがって, この回線タイプを使う場合は, マルチリンク PPP, PPP 暗号化または PPP 圧縮を可能にするべきでありません。さもなければ, 順序が乱れたフレームは落とされ, パフォーマンスの低下に至ります。

`set udp self` *ipaddr* [*port*]

UDP 接続のローカル IP アドレスとポートを設定します。何も指定されないと何らかの適切なローカル IP アドレスが使われます。ポートの指定は受信接続を受け付けるために必要です。

`set udp peer` *ipaddr* [*port*]

UDP 接続のピアーの IP アドレスとポートを設定します。アドレスとポートの指定は, 接続の開始のために必要です。受信接続のためには必要ではありませんが, 接続可能な相手の制限に利用することができます。

`set udp enable` *option* ...

`set udp disable` *option* ...

回線のために UDP 装置タイプオプションを有効, 無効にします。

以下のオプションがサポートされます:

`originate`

このオプションはピアーへの UDP 接続の開始を有効にします。このオプションが無効になっていると mpd はいかなる UDP 接続も開始しません。

`incoming`

このオプションは着信 UDP 接続の受諾を有効にします。このオプションが無効にされていると mpd はいかなる着信 UDP 接続も受け入れません。

5.5 PPTP 装置タイプ コマンド

この章では PPTP タイプ回線に固有であるコマンドについて記述します。これらのコマンドは現在稼働状態の回線に適用され, 現在稼働状態の回線がタイプ `pptp` を持つ場合にのみ有効です。

PPTP プロトコルは, 回線レイヤーメディアが, 単に偶然 IP 接続であるような一つの回線レイヤータイプとして, 非常に簡単に理解することができます。それで, 例えばモデムに対してダイヤルする電話番号を設定する代わりに, 接続する IP アドレスを設定します。上記の点以外は回線レイヤー機能は全く同じです。したがって, PPTP は IP の上に「トンネル」PPP フレームを許容します。

PPTP 接続が一方のマシンから他方に TCP 接続によって始められることに注意すべきです。そして, そのサーバーは通常 TCP ポート 1723 で待機しています (そして,

これは下記のコマンドでのデフォルトの **ポート** です)。PPTP は IP プロトコルナンバー 47 である GRE プロトコルも使います。ファイアウォールの設定で, この種の IP パケットを許容するように調整する必要があるかもしれません。
完全な PPTP ネットワークトポロジーは以下のようになります:
クライアント← ある種の回線タイプ → PAC ← PPTP トンネル → PNS
PAC は物理レベルリピーターです。それはある種のタイプの PPP 接続を受け, PPTP プロトコルを用いて PNS にそれを転送します。PNS は ppp エンドポイントです。それは PPTP トンネルを通して ppp フレームを受け取り, それらを処理します。
単純なケースでは, 物理的な転換が必要でないとき, トポロジーを単純化することができます:
クライアント (PAC エミュレーター) ← PPTP トンネル → PNS
Mpd は PAC と PNS モードの両方で動くことができます。PAC として mpd は単純なケースの PAC エミュレーターと完全な PAC トポロジーの両方をサポートします。完全な PAC は mpd のリピーター機能を使っている 2 台の物理的な装置を接続することによって設定することができます。

`set pptp self` *ipaddr* [*port*]

PPTP 接続のローカル IP アドレスとポートを設定します。

`set pptp peer` *ipaddr* [*port*]

PPTP 接続のピアーの IP アドレスとポートを設定します。このコマンドは発着信両方の接続に適用されます。発信接続では, このコマンドは接続先を指定するために必要です。着信接続では, このコマンドはオプションです; 設定されない場合, mpd はどんなホストからの着信接続でも受け入れます。それ以外の場合は指定された IP アドレス (そして, オプションであるポート) からの接続だけが許容されます。

`set pptp callingnum` *number*
`set pptp callednum` *number*

PPTP 接続を始めるときの発信, 着信用の電話番号を設定します。ほとんどの場合の VPN アプリケーションでは, これは無視されます。しかし, 特定のケースでは実際の電話番号が必要です。デフォルトは空の文字列です。

`set pptp enable` *option* ...
`set pptp disable` *option* ...

回線のための PPTP 装置タイプオプションを有効, 無効にします。

以下のオプションがサポートされます:

originate

このオプションはピアーへの PPTP 接続の開始を有効にします。このオプションが無効にされていると mpd は一切の PPTP 接続を開始しません。**set pptp peer** コマンドによりピアーの IP アドレスを設定しなければなりません。

incoming

このオプションは着信 PPTP 接続の受け入れを有効にします。このオプションが無効にされていると mpd は一切の着信接続も受諾しません。

outcall

PPTP において, 2 つの IP ホスト間の個々の PPP 接続 (いくつかあるかもしれません, そして, これらはいかなる 2 つの IP ホストの間の**単独の** TCP 接続とも混同されません) は着信か発信のどちらかとして始められます。これは, 例えば, 着信電話のアクセスサーバー (PAC) からリモート PPTP サーバー (PNS) への転送のより一般的な使用と同様に, リモートアクセスサーバー (PAC) を通して (PNS による) 電話発信を行うことを許すことになります。

このオプションが有効にされるとき mpd は発信 (PNS) を始めます; さもなければ, mpd は着信待ち (PAC) を始めます。着信待ちの方がより適切なように思えますが, マイクロソフト PPTP ダイヤルアップアダプタークライアントの挙動に合わせて, デフォルトは発信開始です。

Mpd は着信 PPTP 接続についてどちらの種類の呼び出しでも受け入れます。

delayed-ack

遅延 ACK を有効にします。これは信頼性の高い回線でのスループットを改善することができます。デフォルトはオンです。

always-ack

すでに送られているとしても常に ACK を含めます。これは信頼性の低い回線でのスループットを改善することができます。デフォルトはオフです。

windowing

プロトコルによって指定されるウィンドーイングメカニズムを有効にします。これを働かなくすることは mpd がプロトコルに違反する原因になります。そして, おそらく他の PPTP ピアーを混乱させますが, しばしばより良い性能を示します。ウィンドーイングメカニズムは PPTP プロトコルにおける設計エラーです; L2TP(PPTP の後継) では, それは取り除かれています。デフォルトは無効です。

5.5.1 PPTP セットアップ ヒント

目的がLANに接続するリモートクライアント用のPPTPサーバーのセットアップならば, 以下のような設定が適当でしょう。
設定すべき mpd.links の各エントリ:

```
#
# Act like a PPTP server allowing clients to connect
#
Pptp0:
    set link type pptp
    set pptp enable incoming
    set pptp disable originate
```

ローカルイーサネットにLANアドレスを割り当てているならば, set ipcp dns と set ipcp nbns コマンドを用いてクライアントのバンドルのためにDNSとNBNSサーバー情報を設定するとともに, 各々のインターフェイスでproxy-ARPを有効にする必要もあるかもしれません,
同時に複数のクライアントの接続を可能にしたい場合は, 各々潜在的クライアントのために1つずつ複数のバンドルを定め (そして, 各々を読み込んで実行し) なければなりません。各々のバンドルは衝突を避けるために, 異なるピアー proxy-ARP IP アドレスを割り当てるように交渉しなければなりません。例えば:

```
#
# mpd.conf: allow up to 2 clients
#
default:
    load Pptp1
    load Pptp2

Pptp1:
    new -i ng0 Pptp1 Pptp1
    ...
    set ipcp ranges 1.1.1.1/32 1.1.1.101/32
    ...

Pptp2:
    new -i ng1 Pptp2 Pptp2
    ...
    set ipcp ranges 1.1.1.1/32 1.1.1.102/32
     ...
```

その代わりにピア ツー ピア対称形のトンネリングの設置 (すなわち PPTP 上で PPP を使っている通常の IP ルーティング) を設定したいならば, 以下のようにすればよいかもしれません (ここで 1.1.1.1 はローカル IP アドレスで, 2.2.2.2 はリモートの IP アドレスです):

```
#
# Tunnelling PPP traffic over a PPTP connection with peer 2.2.2.2
#
Pptp1:
    set link type pptp
    set pptp self 1.1.1.1
    set pptp peer 2.2.2.2
    set pptp enable incoming originate
```

リモートマシンには, 自身とピアーのアドレスが反対となった同じエントリがあるでしょう。
もし NT サーバーに接続するならば, 認証名が NT ドメイン名を含まなければならない点に注意してください。例えば:

```
    set auth authname "DOMAIN\\username"
```

Windows 95 と 98 のクライアントが正常に動作するように, 以下のアップデートを行ってください。 VPN Update for Windows 98 and Dial-Up Networking 1.3 Available (`http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q191540`)

5.6 L2TP 装置タイプ コマンド

この章では L2TP タイプ回線に固有であるコマンドについて記述します。これらのコマンドは現在稼働状態の回線に適用され, 現在稼働中の回線がタイプ `l2tp` である場合にのみ有効です。
L2TP プロトコルは, IP ピアーの間で仮想トンネルを作成し, 維持するためにポート 1701 (そして, これは下記のコマンドでデフォルトの **ポート** です) で UDP データグラムを利用します。一つ以上の独立した PPP 接続 (セッション) を, このトンネル内で運用することができます。
完全な L2TP ネットワークトポロジーは以下のようなものです :
クライアント← ある種の回線タイプ → LAC ← L2TP トンネル → LNS
LAC は物理レベルリピーターで, ある種のタイプの PPP 接続を受けて, L2TP プロトコルを用いて LNS にそれを転送します。LNS は ppp エンドポイントで, それは L2TP トンネルを通して ppp フレームを受けて, それらを処理します。

単純なケースでは, 物理的な転換が必要でないときトポロジーは単純化することができます:

クライアント (LAC エミュレーター) ← L2TP トンネル → LNS

Mpd は LAC と LNS の両方のモードでの動作が可能です。LAC として mpd は単純なケースの LAC エミュレーターと完全な LAC トポロジーをサポートします。完全な LAC は mpd のリピーター機能を使っている 2 台の物理的な装置を接続することによって設定することができます。

ウィンドウズ L2TP クライアントはさらなるトンネルの安全確保のために IPSec 暗号化を用います。それで, 接続を確立させるためには MPD ルータで IPSec を設定しなければならないか, あるいはキー

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RasMan\Parameters`

でレジストリ DWORD 値 ProhibitIpSec を 1 にセットすることによって, ウィンドウズ上で IPSec を働かなくしなければなりません。

`set l2tp self` *ipaddr* [*port*]

L2TP 接続のローカル IP アドレスとポートを設定します。複数の IPs を持つルータで着信接続を受け付けていて, クライアントがこのルータで最も近いアドレスにつながっていない状況で, このオプションがセットされない場合, 既知の組み込まれたバグがあります。

`set l2tp peer` *ipaddr* [*port*]

L2TP 接続のピアーの IP アドレスとポートを設定します。このコマンドは発着信両方の接続に適用されます。発信接続では, このコマンドはどこに接続するかについて指定するために必要です。着信接続ではこのコマンドはオプションです; 設定されていないと mpd はどんなホストからでも着信接続を受け入れます。そうでない場合は指定された IP アドレス (そして, オプションであるポート) からの接続のみ受諾します。

`set l2tp hostname` *name*

L2TP トンネルローカルホスト名をセットします。サーバー側では, 待機している IP (`set l2tp self` ...) とピアー IP (`set l2tp peer` ...) のあらゆる一組に対して 1 つの一意のホスト名のみがサポートされます。いくつかのホスト名が定義されている場合, 最初に適合したものだけがすべての着信接続のために使われます。

`set l2tp secret` *secret*

L2TP トンネルシークレットを設定します。トンネル接続を認証して重要な制御パケット avpairs を暗号化するのに用いられます。サーバー側では, 待機している IP(`set l2tp self` ...) とピアー IP(`set l2tp peer` ...) のあらゆる一組に対して 1 つの一意のシークレットのみがサポートされます。いくつかのシークレットが定義されている場合, 最初に適合したものだけがすべての着信接続のために使われます。

注 : このオプション普通の PPP 認証とは関係ありません。ウィンドウズ クライアントはトンネル認証をサポートしません。

`set l2tp callingnum` *number*

`set l2tp callednum` *number*

L2TP 接続を始めるとき発信, 着信用の電話番号を設定します。ほとんどの VPN アプリケーションでは, これは無視されます。しかし, 特定のケースでは実際の電話番号が必要です。デフォルトは空の文字列です。

`set l2tp enable` *option* ...

`set l2tp disable` *option* ...

回線のための L2TP 装置タイプオプションを有効, 無効にします。

以下のオプションがサポートされます:

`originate`

このオプションはピアーへの L2TP 接続の開始を有効にします。このオプションが無効にされていると mpd は一切の L2TP 接続を開始しません。`set l2tp peer` コマンドにより, ピアーの IP アドレスを設定しなければなりません。

`incoming`

このオプションは着信 L2TP 接続の受け入れを有効にします。このオプションが無効にされていると mpd は一切の着信接続を受諾しません。

`outcall`

L2TP トンネルの内部で, 個々の PPP 接続 (いくつかあるかもしれません) は着信か発信のどちらかとして始められます。これは, 例えば, 着信電話のアクセスサーバー (LAC) からリモート L2TP サーバー (LNS) への転送のより 一般的な使用と同様に, リモートアクセスサーバー (LAC) を通して (LNS による) 電話発信を行うことを許すことになります。

このオプションが有効にされるとき mpd は発信 (LNS) を始めます; さもなければ, mpd は着信待ち (LAC) を始めます。

`hidden`

L2TP トンネルシークレットが設定されるとき, さらなる保護ためにいくつかの制御データを隠す (暗号化する) ことが可能です。

デフォルトは無効です。

`length`

デフォルトで, L2TP はデータパケットのためのでなく制御パケットのためにヘッダ Length フィールドを使用します。このオプションはデータパケットのために Length フィールドを有効にします。この機能を有効にすると, パケットにデータの埋め込みが発生するかもしれない回線で役に立つかもしれません。これを無効にすると 1 パケットにつき 2 バイト オーバーヘッドを減らします。

デフォルトは無効です。

`dataseq`

デフォルトで L2TP は制御パケットにヘッダシーケンスフィールドを必要としますが, データパケットには必要としません。このオプションはデータパケットにシーケンスフィールドを有効にします。この機能有効にすると, パケットの再配列が起こるかもしれませんが, それに耐えられない回線で役に立つかもしれません。これを無効にすると 1 パケットにつき 4 バイト オーバーヘッドを減らします。

デフォルトでは有効にされます。

5.7 PPPoE 装置タイプ コマンド

こので章は PPPoE タイプ回線に固有であるコマンドについて記述します。これらのコマンドは現在稼働中の回線に適用され, 現在稼働中の回線がタイプ `pppoe` を持つ場合にのみ有効です。

PPPoE は, 2 つの設定情報を必要とします; 使用するイーサネットインターフェイスの名前と ISP によって供給されているはずのサービス名。サービス名がない場合は, たいていは空の文字列 (デフォルト) でも十分です。

`set pppoe iface` *interface-name*

使用するイーサネットインターフェイスの名前を指定します。PPPoE はインターフェイスの通常の使用に影響を及ぼしません。

`set pppoe service` *service-name*

要請された PPPoE サービス名をセットします。着信接続が許された回線で, サービス名を「*」に設定することは, どんなサービス名が指定された着信接続要請でも受け入れを許可するということになります。

2008-01-30 以降の FreeBSD 6.3-STABLE/7.0-STABLE より, 外部への接続のために指定されたアクセスコンセントレータへのリクエストに

"AC-Name\Service-Name" シンタックスを利用することができます。

set pppoe acname *name*

クライアントに送信されるこの PPPoE アクセスコンセントレータの名前をセットします。指定されない場合はローカルホスト名が使われます。

set pppoe enable *option ...*

set pppoe disable *option ...*

回線に PPPoE 装置タイプオプションを有効, 無効にします。

以下のオプションがサポートされます:

originate

このオプションはピアーへの PPPoE 接続の開始を有効にします。このオプションを無効にすると mpd はどのような PPPoE 接続も開始しません。

incoming

このオプションは着信 PPPoE 接続の受け入れを有効にします。このオプションを無効にすると mpd はどのような着信 PPPoE 接続も受け入れません。このオプションを使用する前に **iface** と **service** パラメータを設定しなければなりません。

6 チャットスクリプト

Mpd は **モデム**タイプの回線 (すなわち非同期シリアルポート) を活用するために, 強力なチャットスクリプト言語を備えています。この言語は解釈され動的に実行されます。タイムアウトと正規表現パターンマッチ機能を有し, 完全なイベント駆動処理をサポートしています。
以下に記すことの多くは mpd に含まれる mpd.script ファイルを見ることによって, より容易に理解されます。

6.1 スクリプトファイル形式と実行

`mpd.script` の文法は他の mpd 設定ファイルと類似しています。ハッシュサイン (#) で始まっている行は, 無視されます。ラベル (**label**) はそれ自身で 1 行を占め, 最初のコラムから始まりコロン文字で終わります。
チャットコマンドは, タブ文字でインデントされ, 一行につき一コマンド記述されます。`mpd.conf` や `mpd.links` とは異なり, チャットコマンドの処理はブランク行に遭遇しても, **停止しません。**
コマンドは順番に実行されます。エラーが起きるか, あるいは以下の命令のうちの 1 つに遭遇するまで, 実行は停止することなく続行されます :

- `success` スクリプトは, 成功を返します。
- `failure` スクリプトは, 失敗を返します。
- `wait` **イベント**が起こるまで, スクリプトの実行は停止されます。

イベントは, 次のセクションで記述されます。

6.2 イベント

イベントとはタイマーの有効期限または入力のあるパターンのマッチングです。まだ起こっていないイベントは**未決定のイベント**です。未決定のイベントはそれぞれタイマーとマッチコマンドにより生成されます。
各々の未決定のイベントは, イベントと関連する (**ターゲットラベル**と呼ばれる) ラベルを持ちます。イベントが起こると, そのとき実行は ターゲットラベル で再開が始まります。イベントは `wait` コマンド実行の間に起こるだけです。
イベントは **セット**と名づけられたグループに分けられます。これらが同じ状況の交互の結果を代表するという点で, 通常同じセットのイベントは関連性があります。

一連の未決定のイベントは, それらが起きる前に**キャンセル** コマンドで, はっきりと**中止される** (すなわち, 忘れられる) かもしれません。そのセットのすべてのイベントは, 中止されます。
一連の未決定のイベントが中止されるかもしれない他の方法はセットのどれかのイベントでが起こるという, 暗黙のうちのものです。実行はそれが起こったイベントに関連したターゲットラベルで再開されます, そして, そのセットの他の全てのイベントは直ちに中止されます。

6.2.1 タイマーイベント

タイマーイベントは `timer` コマンドから生成されます。`timer` ステートメントが実行された時刻からスタートして, 指定の時間が経過したとき, タイマーイベントが起こります。更なる実行は `timer` コマンドにより指定されるターゲットラベルから開始されて続行されます。
(すべてのイベントと同様) タイマーイベントは `wait` コマンド実行の間のみ起こり得ます。

6.2.2 マッチイベント

マッチイベントは `match` と `regex` コマンドにより生成されます。`match` または `regex` コマンド中で指定されたパターンが入力とマッチしたとき, マッチイベントが起こります。文字は `wait` コマンドの実行の間のみ入力されます。パターンがマッチするために, イベントがつくられた**後** (すなわち, 以降の `wait` コマンドの間) に, パターンの最初にマッチしている文字が読み込まれなければなりません。
パターンがマッチされるとき, チャットスクリプトの実行はターゲットラベルから始まり続行されます。複数のパターンが同じ入力文字に関してマッチするとき, mpd はマッチすると定められた最初のものを選びます。
パターンはプレーンテキストか拡張正規表現 (下記参照) です。

6.3 セット

セットとは, 未決定のタイマー および/または マッチイベントの集まりです。ある一つのイベントが起こるとき, または適切なキャンセルコマンドの実行と同時に, セットのすべてのイベントは中止されます。
複数のセットはどんなときでも作動中かもしれません。一つのセットからのイベントの発生は他のセットのイベントには影響を及ぼしません。

6.3.1 特別なセットとターゲット

timer, match と regex コマンドでは, (省略することによって) **デフォルト セット**を指定してもかまいません。このセットは**どんな**イベントの発生に対しても自動的に中止されるという点をを除き, 他のどのセットとも同様のものです。それは空の文字列 (すなわち, "") を用いて明示的に名をつけられるかもしれません。
timer, match と regex コマンドでは, (省略することによって) **デフォルト ターゲット**も指定することができます。名前が空の文字列でもあるこのターゲットは, それが常に「直後に続くステートメントである次の wait ステートメント」を参照する特別な性質を持ちます。即ち, あらゆるイベントのターゲットがデフォルトターゲットであるならば, そのイベントの発生は, ちょうど次に続く wait ステートメントで開始される実行を続けさせることになります。
最終的には, all が "cancel" コマンドで使われるとき, **全**セット (すなわちすべての未決定のイベント) はキャンセルされます。

6.4 変数

変数は単純な文字列から成ります。それらは大域的で, **一時的** なものと**永続的**なものの 2 つのタイプになります。永続的変数は複数のチャットスクリプトの呼び出しに対して持続され, 一方, 一時的な変数はスクリプトが完了するたびに忘れられます。
変数はドル記号, それから文字があり, さらに文字, 数字, アンダースコアが続けられて指定されます。最初の文字が大文字であるならば, 変数は永続的なものとなります; そうでないならば, それは一時的です。変数名は, オプション的に中括弧 {} で囲まれることもあります。

例 :

```
$initString
$My_variable_234
${i_am_safely_followed_by_a_letter}
```

6.4.1 特別な変数

これらの変数は, チャットスクリプトで特別な意味を持ちます:

$Login

set auth authname で指定される認証のログイン名。

$Password

$Login に対応するパスワード。

`$modemDevice`

`set modem device` コマンドで設定される使用中のシリアルポート装置。例えば, `/dev/cuad0`。この文字列を変更しても使われているポートは変更されません。

`$Baudrate`

常に, 現在のボーレート (例えば"57600") に一致します。これは読込み/書込み変数です; これをセットするとボーレートを変更することになります。無効なボーレートをセットしようとするとスクリプトが失敗する原因になります。

`$matchedString`

`match` イベントが起きるとき, このストリングはパターンにマッチした入力文字列を含みます。

`$matchedString0`
`$matchedString1`
`$matchedString2`
...

`regex` イベントが発生するか, あるいは, `if match` または `if !match` コマンドが実行され正規表現パターンが括弧に入れられた部分表現を含むとき, これらの文字列は各々の部分表現にマッチしている部分文字列に等しくなっています。`$matchedString0` は文字列全体に等しく, 一方`$matchedString1` は最初にマッチした部分表現に対応する部分, `$matchedString2` は 2 番目にマッチした部分表現に対応する部分にそれぞれ一致する, 等となっています。

`$IdleResult`

この変数は**アイドルスクリプト**の結果を返すのに用いられます; 値が `answer` であるならば, mpd は着信が答えられたと仮定します。値が `ringback` であるならば, mpd は発信接続を始めます。詳細は モデム タイプ コマンド (5.1) についての章を見てください。

`$$`

常にドル記号 1 個に展開されます。

6.5 スクリプト コマンド

`set` *$variable-name string*

$variable-name を *string* にセットします。*string* は最初に変数展開されます。

`match` *name string label*
`match` *string label*
`match` *string*

入力に *string* があるとき, 実行を *label* から再開する新しい未決定のイベントをイベントセット *name* に加えます。マッチは厳密でなければなりません。すべての引数は変数展開されます。
もし *name* が与えられていないか, または *name* が空の文字列であるならば, 新しいイベントをデフォルトセットに加えます。
もし *label* が与えられていないか, または *label* が空の文字列であるならば, イベントは `wait` コマンドに続くステートメントから開始して, 実行が再開されます (すなわち, デフォルトターゲットを用います)。

`regex` *name pattern label*
`regex` *pattern label*
`regex` *pattern*

`match` コマンドと同様ですが, 厳密な文字列の代わりに拡張正規表現によるマッチングとなります。もし *pattern* が拡張正規表現として有効でない場合は, スクリプトは失敗します。
正規表現は行毎にマッチングが確認されます; パターンにマッチしている入力は複数の行にわたることはできません。行は改行文字かキャリッジリターン-改行文字の組 (後者はより一般的です) で終了されます。行末は, 正規表現をテストするとき mpd が入力からこれらを無視するので, 直接これらの文字へのマッチを得ようとするよりはむしろ, ドル記号を使ってマッチされなければなりません。
拡張正規表現の詳細については `re_format(7)` を見てください。

`timer` *name seconds label*
`timer` *seconds label*
`timer` *seconds*

seconds 秒が経過したら, 実行を *label* から再開する新しい未決定のイベントをセット *name* に加えます。すべての引数は変数展開されます。
もし *name* が与えられていないか, または *name* が空の文字列であるならば, 新しいイベントをデフォルトセットに加えます。
もし *label* が与えられていないか, または *label* が空の文字列であるならば, イベントは `wait` コマンドに続くステートメントから開始して, 実行が再開されます (すなわち, デフォルトターゲットを用います)。

`wait`

`wait` *seconds*

最初の形式は, 無期限になんらかのイベントが起こるのを待ちます。イベントが起こると, そのイベントのターゲットラベルから開始して実行を続行します。第2行目の形式は次と同等です:

```
timer seconds
wait
```

`if` *string1* `==` *string2 command*

`if` *string1* `!=` *string2 command*

2つの変数から展開された文字列が等しいか, 等しくない場合に *command* を実行します。

`if` *string* `match` *pattern command*

`if` *string* `!match` *pattern command*

string を正規表現 *pattern* に対してテストして, *string* がマッチするか, しないかにより, *command* を実行します。*string* と *pattern* の両方とも最初に変数展開されます。

`print` *string*

string をシリアルポートに出力します。変数は展開されます, そして, 通常のmpd Cスタイルのエスケープ文字を使うことができます。

`cancel` *name1* [*name2* ...]

セット *name1*, *name2* ... ですべての未決定のイベントを中止します。すでに空であるセットをキャンセルしてもエラーとはなりません。

`goto` *label*

ラベル *label* へジャンプします。

`call` *label*

label にあるサブルーチンを呼び出します。

`return`

サブルーチンから返ります。

`success`

成功裏にスクリプトから出ます。

`failure`

失敗した状態でスクリプトから出ます。

`log` *string*

変数 *string* を展開し, ログレベル CHAT で, それをログファイルに出力します。

6.6 エラー

どんなエラーでも, スクリプトは失敗状態で終了します。エラーの例は以下のようなものです:

- 存在しないラベルへの `call` または `goto`
- 対応する `call` のない `return`
- 未決定のイベントなしの `wait` の実行
- スクリプトの終わりから外れた実行
- 無効なボーレート値 `$Baudrate` の設定試行
- 無効な正規表現との照合試行

6.7 パッケージに含まれている mpd.script

Mpd のパッケージの中にある `mpd.script` は, 通常修正なしのそのままの状態で, 典型的 PPP アプリケーションのために使うことができます。このスクリプトは以下のラベルを定めます :

`DialPeer`

このスクリプトでは, mpd は取付けられたモデムまたは ISDN ターミナルアダプター装置を特定しようとし, 装置を設定して, リモートのピアーに対してダイヤルするようになっています。これはダイヤルアップクライアントの働きをさせたいとき, `set modem script` のために使用するラベルです。
このスクリプトにより用いられる変数は以下の通りです:

`$Telephone`

ダイヤルする電話番号。

`$DialPrefix`

外線に発信するために電話番号前に必要なダイヤル部分, 必要な場合に設定。

`$ConnectTimeout`

モデムの接続を待つ時間の秒数。デフォルトは 45 秒 です。

`$SpeakerOff`

ダイヤル中にモデムスピーカーからの音を消したい場合にこの値を `yes` にセットします。

`$Serial230K`

シリアルポートが230Kボーで動作可能な場合, これを `yes` にセットしてください。

注:通常のPCのハードウェアは230Kのスピードでは動作しません。

さらに, ISDNターミナルアダプタの設定には以下の変数が要求されます:

`$TA_Bonding`

ターミナルアダプタに2つのBチャネルマルチリンクPPPを行わせたい場合, この変数の値を `yes` にセットします。

注:Mpdはシングルリンク PPPで接続しているように見えても, マルチリンクPPP接続で動作するターミナルアダプタはそれのための設定が必要です。

`$TA_NoDoubleTelno`

`$TA_Bonding` が `yes` に設定されている場合, ダイヤル番号を2重にしないように設定します。通常, そのような状況では mpd は, `ATDT${Telephone}&${Telephone}` コマンドを使用してダイヤルします。`$Telephone` 文字列がすでに両方の番号を含んでいる場合, `$TA_NoDoubleTelno` を `yes` にセットします。

`$TA_56K`

Bチャネルを56Kに制限する必要がある場合, この変数を `yes` にセットします。北アメリカのある種のISDN回線のためにのみ必要なものです。

`$TA_VoiceCall`

音声モードコールが必要な場合, この変数を `yes` にセットします。北アメリカのある種のISDN回線のためにのみ必要なものです。すべてのターミナルアダプタがこれをサポートするわけではありません。

`$TA_AuthChap`

ある種の旧型のターミナルアダプタでは, リモート側からCHAPまたはPAP認証のいずれが要求されるのかを事前に明示することが必要となります。ターミナルアダプタにCHAPを使うように指示するためにはこの変数を `yes` にします。

`$TA_SwitchType`

ISDN回線のスイッチタイプを設定します。`Ni-1`, `DMS-100`, `5ESS P2P` または `5ESS MP` のうちの1つでなければなりません。北アメリカのISDNためにのみ必要です。

$TA_Dirno1

$TA_Dirno2

$TA_SPID1

$TA_SPID2

両方の B チャネルのために, これらを ISDN 回線のディレクトリ番号と SPID にセットしてください。北アメリカの ISDN のためにのみ必要です。

AnswerCall

これは, 着信を待って, それに答える待機のスクリプトです。このスクリプトにより用いられる変数は以下の通りです:

$ConnectTimeout

モデムの接続を待つ時間の秒数。デフォルトは 45 秒 です。

$RingTimeout

RING が接続を一度諦め, 再試行の着信を待つ時間。デフォルトは 10 分です。

Ringback

これは着信を待つ待機のスクリプトでもありますが, 呼び出しに答える代わりに, それを無視して, 発信接続を開始します。これは, リモートのダイヤルアップクライアントマシンを動かすときに役立ちます。

$RingbackTimeout

接続を諦めるまでの時間 (リセットして, 再試行します)。デフォルト : 60 分。

$RingStoppedTime

アナログモデムのために, 発信のダイヤルをしようとする前に呼び出しが止まるのを待たなければなりません, さもなければ, うかつにも着信に答えてしまうことになります。この値は, ベルを鳴らすことが止まったと判断するために待つ最小限の時間にセットされます。デフォルト : 8 秒。

7　トラブル シューティング

7.1　トラブル シューティング

トラブルシューティングに関するいくつかの情報。
更なる助言については `freebsd-net@freebsd.org` メーリングリストまたは sourceforge.net mpd フォーラムとメーリングリストに問い合わせてください。

動作しません, その原因もわかりません。

Mpd には syslog をサポートしている広範囲なログシステムがあります。ログファイルへを出力するよう指令するには, 次の行

```
!mpd
*.*                                    /var/log/mpd.log
```

を `/etc/syslog.conf` ファイルに追加し, `/var/log/mpd.log` ファイルを作成し, 設定を有効にするために syslogd への SIGHUP シグナルを送ります。
すべての可能なログの記録を有効にするため, 'log +all' コマンドを使うことができます。

パケットが流れません。

`/etc/rc.conf` で `gateway_enable="YES"` と設定してあるか, 確認してください。そのようにしないと, FreeBSD ボックスはパケットを転送しません。あるいは, 直ちに有効にするために
`sysctl -w net.inet.ip.forwarding=1`
を実行してください。
また, ファイアウォールのセッティングをチェックしてください。Mpd はファイアウォール規則に取り込まれる必要があるかもしれない新しいインターフェイスを作成します。あなたが PPTP を使用しているならば, TCP ポート 1723 と IP プロトコル 47(GRE) を許可する必要があります。

動作しません, そして, 不可解な netgraph 関連のエラーがログにあります。

すべての必要な netgraph KLD がロードされているか確認してください。
`kldstat -v | grep ng_`
とすることによって, それらをチェックすることができます。
通常, これらは要求があり次第ロードされます。そうなっていない場合, `kldload(8)` を使って, 手動でそれらをロードすることができます。

以下のノードのタイプが必要であるか, または必要かもしれません:

- ng_async
- ng_bpf
- ng_car
- ng_deflate
- ng_ether
- ng_iface
- ng_ksocket
- ng_l2tp
- ng_mppc
- ng_ppp
- ng_pppoe
- ng_pptpgre
- ng_nat
- ng_netflow
- ng_pred1
- ng_socket
- ng_tcpmss
- ng_tee
- ng_tty
- ng_vjc

Mac OS X PPTP クライアントに対する低いパフォーマンス。

4.7-RELEASE より新しいバージョンへの FreeBSD へアップグレードしてください。または, 手動で このパッチ
(`http://www.freebsd.org/cgi/cvsweb.cgi/src/sys/netgraph/ng_pptpgre.c.diff?r1=1.25&r2=1.26`)
をあててください。

設定しましたが, 期待通りに動いていないようです。

あなたの `mpd.conf` と `mpd.links` の書式が正しいことを確認してください。ラベルは左端に揃えである必要があり, 他の行はそうであってはなりません。

RADIUS サーバーからの有効な返答が得られません。

RADIUS サーバー設定と `mpd.conf` あるいは `radius.conf` の中で指定されるそれぞれの共有シークレットの両方を確認してください。RADIUS サーバーのポートをチェックしてください: そのポートは 1812(認証) と 1813(アカウンティング) でなければなりません, それらは mpd のデフォルトポートでもあります。過去に非公式のポート番号 (1645 と 1646) が RADIUS のために使われました, しかし, これらは受け入れられません。

Mpd を走らせようとすると, "Operation not permitted" というエラーが帰ってきます。

これは, しばしば
(a) kernel `config(8)` オプションによって, いろいろなモジュールをカーネルのコンパイル時に静的に組み込む代わりに, KLD モジュールの形で netgraph を使っている。
(b) カーネルセキュリティレベルを上げている (`init(8)` マニュアルページ参照)。これは KLD モジュールがロードされるのを妨げます。
を組合せていることに原因があります。問題を解決するため, これらのうちの1つを変えてみてください。

ダイアルインのために回線を設定しましたが, 時々切断した後に, mpd は何らかの無限のループに入ってしまいます。

これは mpd が"リダイヤル"を試行しているからです。そして, それはもちろんダイアルイン回線には適切でありません。リダイヤル機能を無効にするため,

```
set link max-redial -1
```

および

```
set bundle yes noretry
```

を使用してください。

Windows XP をクライアントとする PPTP サーバーとして mpd を使っています。データ量の大きいウェブサイトにアクセスするときや, 大きなディレクトリを ftp でリストするようなとき, 接続がハングしてしまうようです。

Windows XP は非常に小さい MTU(通常 1396 Bytes) を使うように設定されていますが, 大きめのパケットが回線を流れる場合, これは断片化を引き起こします。断片化は ICMP レベルで交渉されますが, そのようなパケットを落としてしまうような悪いルータがネットワーク上にあると, それはありがちなことですが, 接続はハングしているように見えるでしょう。試すことができる最初のことは mpd の MTU 値を減らしていくということです。そのためには次のようにします:

`set link mtu 1300`
および
`set bundle disable multilink`
これは, ほとんどの場合助けとなるでしょう。TCP 接続のために, TCP-MSS-Fix を有効にすることも可能です:
`set iface enable tcpmssfix`
(mpd-3.15 から利用可能)。

シスコ機に接続するときの MPPE stateless に関する問題。

あなたの装置の IOS をアップグレードして, 以下をを見てください:
`CSCdu30944 MPPE rejects stateless Fixed in 12.3(11.4)`

8　内部文書

8.1　ToDo

MPD のための未完了の仕事のリスト。あなたが活発に MPD の発展をサポートしたいならば, これは良い出発点です。仕事を完了した後に, MPD のメーリングリストに投稿してください。

コールバック (クライアントとサーバー)

現在, MPD は, 番号交渉なし, CBCP サポートなし, クライアントとしてのみ最も単純な LCP コールバックだけをサポートします。

未実装プロトコル

lcp.c はプロトコルのリスト (gLcpConfOpts) を含みます, "false" による各々のプロトコルは実装されません。

8.2　認証

この章は, MPD の実装に特有の細部について述べます。

認証プロトコル - 短い概要

現在, MPD はこれらの認証プロトコルをサポートします: PAP, MD5-CHAP, MS-CHAPv1, MS-CHAPv2 and EAP。
PAP を使うとき, パスワードはネットワークの上を平文として送られます, したがって, ピアーの回線が安全でないならば, PAP は避けるべきです。他方, PAP は, たとえパスワードが不可逆なハッシュされたフォーマットで保存されるとしても, サーバーでいかなるパスワードデータベースを使用することも許します。

伝統的な CHAP-MD5 は, サーバーに保存された平文のパスワードを必要とします。パスワードハッシュは, 次のように計算されます: md5(id + password + challenge), ここで id は各々の認証試みの後, 加算されます。challenge はサーバーで生成され, それからクライアント (ピアー) に送信されます。ピアーは, サーバーへハッシュを送り, サーバーは自分自身で平文のパスワードを使用してハッシュを生成します。両方のハッシュが同一であれば, 認証は成功します。

MS-CHAP は, サーバーに平文パスワードを必要としませんが, NT-Hash または LAN Manager-Hash(LAN Manager-Hash は弱く, 使用するべきではありません) によりハッシュ化されたパスワードを必要とします。MS-CHAPv1 はハッシュ化アルゴリズムとして DES を使用しますが, これは弱いので, 使わないでください! MS-CHAPv2 はピアー チャレンジとサーバー チャレンジを使って, ハッシュ化アルゴリズムとして SHA1 を使用しますので, MS-CHAPv1 よりずっと安全です。MS-CHAPv2 は, NT-Hash が有効であることを必要とします。

通常 UNIX システムには, パスワードのために異なる非可逆なハッシュ化アルゴリズムが使用されています, したがって, CHAP アルゴリズムを使用したい場合, 伝統的な UNIX パスワードデータベースを使うことはできません。ただし, 例外として FreeBSD verson 5.1 とそれ以後でパスワードデータベースで NT-Hash フォーマットをサポートします (login.conf に次のように設定可能:passwd_format=nth)。しかし, MPD は現在 UNIX パスワードデータベースに対する認証はサポートしていません。

EAP は, 拡張認証プロトコルです。Mpd は, 内部的に EAP-Type MD5 だけをサポートします; 他の EAP-Types が, RADIUS サーバーと連携して使われるかもしれません。

注: MPPE キーは認証からの結果を使って生成されるため, MPPE が動作するためには, MS-CHAPv1 または MS-CHAPv2 が必須です。

認証プロトコル交渉

MPD 3.14 から始められましたが, 認証プロトコルについて交渉されるとき, MPD はより知能的に振る舞います。MPD は, 相互に同意できるプロトコルが見つかるまで, 安全性の高いものから, 低いものへとプロトコルの内部リストを捜索します。回線接続が PPTP リンクである場合, MS-CHAP は最も望ましいものです。それ以外の場合では, MD5-CHAP が最も望ましいものです。

8.3 RADIUS 認証

この章では MPD の実装に特有の細部について述べます。

RADIUS 内部

Mpd は, RADIUS を利用したユーザー認証とセッションアカウンティングをサポートします。RADIUS-Accounting と RADIUS-Authentication は独立していますので, どのような組合せでも利用することが可能です。

すべての認証方法 (PAP, CHAP, MS-CHAPv1, MS-CHAPv2, EAP) は, RADIUS でサポートされます。パスワードの変更は, 現在サポートされません。

Mpd でサポートされる RADIUS 属性:

N	Name	Access		Accounting	
		Req	Resp	Req	Resp
1	User-Name	+	+	+	-
2	User-Password	+	-	-	-
3	CHAP-Password	+	-	-	-
4	NAS-IP-Address	+	-	+	-
5	NAS-Port	+	-	+	-
6	Service-Type	+	-	+	-
7	Framed-Protocol	+	-	+	-
8	Framed-IP-Address	-	+	+	-
9	Framed-IP-Netmask	-	+	+	-
12	Framed-MTU	-	+	-	-
18	Reply-Message	-	+	-	-
22	Framed-Route	-	+	-	-
24	State	+	+	+	-
25	Class	-	+	+	-
27	Session-Timeout	-	+	-	-
28	Idle-Timeout	-	+	-	-
30	Called-Station-Id	+	-	+	-
31	Calling-Station-Id	+	-	+	-
32	NAS-Identifier	+	-	+	-
40	Acct-Status-Type	-	-	+	-
42	Acct-Input-Octets	-	-	+	-
43	Acct-Output-Octets	-	-	+	-
44	Acct-Session-Id	-	-	+	-
45	Acct-Authentic	-	-	+	-
46	Acct-Session-Time	-	-	+	-
47	Acct-Input-Packets	-	-	+	-
48	Acct-Output-Packets	-	-	+	-
49	Acct-Terminate-Cause	-	-	+	-
50	Acct-Multi-Session-Id	-	-	+	-
51	Acct-Link-Count	-	-	+	-
52	Acct-Input-Gigawords	-	-	+	-
53	Acct-Output-Gigawords	-	-	+	-
60	CHAP-Challenge	+	-	-	-
61	NAS-Port-Type	+	-	+	-
85	Acct-Interim-Interval	-	+	-	-
95	NAS-IPv6-Address	+	-	+	-
99	Framed-IPv6-Route	-	+	-	-

```
    Microsoft VSA (311)
1   MS-CHAP-Response              +       -       -       -
2   MS-CHAP-Error                 -       +       -       -
7   MS-MPPE-Encryption-Policy     -       +       -       -
8   MS-MPPE-Encryption-Types      -       +       -       -
10  MS-CHAP-Domain                -       +       -       -
11  MS-CHAP-Challenge             +       -       -       -
12  MS-CHAP-MPPE-Keys             -       +       -       -
16  MS-MPPE-Send-Key              -       +       -       -
17  MS-MPPE-Recv-Key              -       +       -       -
25  MS-CHAP2-Response             +       -       -       -
26  MS-CHAP2-Success              -       +       -       -

    mpd VSA (12341)
1   mpd-rule                      -       +       -       -
2   mpd-pipe                      -       +       -       -
3   mpd-queue                     -       +       -       -
4   mpd-table                     -       +       -       -
5   mpd-table-static              -       +       -       -
6   mpd-filter                    -       +       -       -
7   mpd-limit                     -       +       -       -
154 mpd-drop-user                 -       -       -       +
```

Mpd は, RADIUS サーバーがアカウンティング スタート/更新 返答パケットのベンダー固有の mpd-drop-user 属性をゼロ以外の値にセットすることによってユーザーセッションを終了させるのを許可します。

RADIUS の ACL's

Mpd は, RADIUS サーバーから与えられるアクセス制御リスト (ACLs) を使用することができます。この ACLs は, ipfw 規則, パイプ, キューとテーブル, 更には mpd 内部のトラフィック filtering/shaping/limiting 機能を含むかもしれません。これら 2 つもセットがあるのは冗長です。ipfw は標準的で一般的な解決策として提案されますが, 一方 ng_bpf + ng_car に基づく内部のフィルタ/シェーパー/リミッターは多数のアクティブリンク動作状態ではより速く動作することが期待されます。
この機能を利用するためには, 以下のような辞書を RADIUS サーバーに加えなければなりません:

```
#-----------------------------------------------------------
# dictionary.mpd

VENDOR          mpd               12341

ATTRIBUTE       mpd-rule          1       string        mpd
ATTRIBUTE       mpd-pipe          2       string        mpd
ATTRIBUTE       mpd-queue         3       string        mpd
ATTRIBUTE       mpd-table         4       string        mpd
ATTRIBUTE       mpd-table-static  5       string        mpd
ATTRIBUTE       mpd-filter        6       string        mpd
ATTRIBUTE       mpd-limit         7       string        mpd
ATTRIBUTE       mpd-drop-user     154     integer       mpd
#-----------------------------------------------------------
```

ipfw

以下のようなRADIUS設定を記述することができます:

```
mpd-table += "1=10.0.0.1",
mpd-table += "1=10.0.0.15",
mpd-pipe += "1=bw 10Kbyte/s",
mpd-pipe += "5=bw 20Kbyte/s",
mpd-rule += "1=pipe %p1 all from any to table(%t1) in",
mpd-rule += "2=pipe %p5 all from table(%t1) to any out",
mpd-rule += "100=allow all from any to any",
```

mpdがこれらのパラメータを受け取ると, ipfw(8) を呼び出して, 10000 ('set global start...' により設定可能) から始まっている一意の番号でファイアウォール規則, パイプとキューが生成されます。各々の規則の末尾に, "via ngX" が付加され規則はクライアントの該当するネットワークインターフェイスだけに適用されます。
この例の結果では, 以下のコマンドを実行した状態を得ることができます:

```
ipfw table 32 add 10.0.0.1
ipfw table 32 add 10.0.0.15
ipfw pipe 10000 config bw 10Kbyte/s
ipfw pipe 10001 config bw 20Kbyte/s
ipfw add 10000 pipe 10000 all from any to table(32) in via ng0
ipfw add 10001 pipe 10001 all from table(32) to any out via ng0
ipfw add 10002 allow all from any to any via ng0
```

接続回線が落ちると, すべての構築された規則は取り除かれます。

内部の (ng_bpf/ng_car)

Mpd は mpd-filter と mpd-limit RADIUS 属性によって要求されるとき, 複雑なインターフェイス毎のトラフィック フィルタリング/帯域制限エンジンを netgraph の中につくることができます。

mpd-filter 属性は, mpd-limit で使うためのパケットフィルタ宣言です。mpd-filter は, 2 つの主な部品から成ります: match/nomatch 判定と条件。条件のために使われる tcpdump(libpcap) 表現構文。

`mpd-filter:` *match* | *nomatch* {*condition*}

mpd-limit 属性は, パケットに対して行われるアクションです。それは, 2 つの主な部品から成ります:フィルタとアクション。

`mpd-limit:` {*filter*} {*action*}

フィルタは, "all"(どんなパケットでも) または"fltX"(指定された mpd-filter に合致しているパケット) のいずれかになります。

`filter:` *any* | *fltX*

アクションは以下のいずれかです: "pass" (処理を止めて, パケットを通過させます), "deny" (処理を止めて, パケットを捨てます), "rate-limit" (シスコのような rate-limit を行います), "shape" (単純な RED 認識のあるトラフィックシェーピングをする)。

アクション "rate-limit" と "shape" は, このアクションを行った後に処理を止めるために, オプションの "pass" 接尾辞を指定することができます。

`action:` *pass* | *deny* | *rate-limit* {*rate(bits/s)*} [{*normal burst(bytes)*} [{*extended burst(bytes)*}]] [*pass*] | *shape* {*rate(bits/s)*} [{*burst(bytes)*}] [*pass*]

例として, RADIUS 設定に以下のように記述することができます:

```
mpd-filter += "1#1=nomatch src net 10.0.0.0/24",
mpd-filter += "1#2=match src net 10.0.0.0/10",
mpd-filter += "2#1=match dst net 10.0.0.0/16",
mpd-filter += "2#2=match dst net 11.0.0.0/8",
mpd-limit += "in#1=flt1 pass",
mpd-limit += "in#2=flt2 shape 64000 4000 pass",
mpd-limit += "in#3=all deny",
mpd-limit += "out#1=flt2 pass",
mpd-limit += "out#2=all rate-limit 1024000 150000 300000",
mpd-limit += "out#3=all pass",
```

結果として, トラフィックフィルタを実装するため, 1つの ng_bpf ノードとトラフィックシェーピングとレートを制限することためにいくつかの (この例では2つの) ng_car ノードが作成されます。10.0.0.0/24 以外の 10.0.0.0/10 からの入って来るトラフィックは通過させます, 10.0.0.0/16 と 11.0.0.0/8 へのトラフィックは 64Kbits/s に調整されます, 他のすべては拒否されます。10.0.0.0/16 と 11.0.0.0/8 への出て行くトラフィックは通過させます, 他のすべては 1024Kbit/s に制限されます。

8.4 外部の認証

この章では MPD の実装に特有の細部について述べます。
Mpd は, 外部スクリプトを呼び出すことによる認証とアカウンティングをサポートします。そのスクリプトへのパスは, set auth extauth-script ... と set auth extacct-script ... コマンドを使用して明示されなければなりません。
実行に際して, このスクリプトは標準入力から, 空行で区切られた要求属性:値 の組のセットを受け取り, 標準出力へ同じ書式で返答を生成しければなりません。

現在サポートされる要求属性：
USER_NAME
USER_PASSWORD - PAP のためのみ
NAS_PORT
NAS_PORT_TYPE
CALLING_STATION_ID
CALLED_STATION_ID
PEER_ADDR
PEER_PORT

現在サポートされる返答属性:
RESULT - SUCCESS, UNDEF（有効な USER_PASSWORD が返された場合）または FAIL のうちの1つ
USER_NAME
USER_PASSWORD
FRAMED_IP_ADDRESS
FRAMED_ROUTE
FRAMED_IPV6_ROUTE
FRAMED_MTU
SESSION_TIMEOUT
IDLE_TIMEOUT

REPLY_MESSAGE
MS_CHAP_ERROR
MPD_RULE
MPD_PIPE
MPD_QUEUE
MPD_TABLE
MPD_TABLE_STATIC
MPD_FILTER
MPD_LIMIT

実行に際して, exacct スクリプトは標準入力から, 空行で区切られた要求属性:値 の組のセットを受け取り, 標準出力へ同じ書式で返答を生成しければなりません。

現在サポートされる要求属性：
ACCT_STATUS_TYPE
ACCT_SESSION_ID
ACCT_MULTI_SESSION_ID
USER_NAME
IFACE
BUNDLE
LINK
NAS_PORT
NAS_PORT_TYPE
ACCT_LINK_COUNT
CALLING_STATION_ID
CALLED_STATION_ID
PEER_ADDR
PEER_PORT
PEER_IDENT
FRAMED_IP_ADDRESS
ACCT_TERMINATE_CAUSE
ACCT_SESSION_TIME
ACCT_INPUT_OCTETS
ACCT_INPUT_PACKETS
ACCT_OUTPUT_OCTETS
ACCT_OUTPUT_PACKETS

現在サポートされる返答属性:
MPD_DROP_USER - ゼロ以外の値はその接続回線が切断されるべきであることを意味する

大部分の属性説明のために, それらの RADIUS 代替項目を見てください。

8.5 開発者へのヒント

この章では開発者へのいくらかのヒントを記述します。

ソース-コード-スタイル

これについて言うべきことはそれほど多くありません, 既存のソースファイルに少しだけ目を通してください。
Tab-Width は 2 つのインデントによる 8 です。オペレーターとオペランドの間でスペースを挿入してください。

NgFuncGetStats

`NgFuncGetStats()` は link-stats を増やすことに関わる他の機能 (エコー 要求/返答, 帯域幅制御) があるので, 決して**明白に** true にセットされたパラメータで呼び出さないでください。Mpd は単にそれらを使用する代わりにリンク-レベルに netgraph link-stats のコピーを保持します。内部の stats-struct を更新するために `LinkUpdateStats()` を呼ぶことができます。

新しい認証-バックエンド

認証バックエンドは mpd のその他の部分から独立して走らなければなりません, すなわち, 認証プロセスはそれ自身のスレッド (スレッド-安全性について注意しなければなりません) で始まるので, どの mpd リソースにもアクセスしてはいけません。AuthData オブジェクトは, すべての必要な情報のコピーを保持する認証関数に引き渡されます。もし利用するバックエンドが, その他, MTU, IP 等のパラメータを提供するならば, これらを AuthData あるいは Auth の適切な場所に置いてください。
Mpd の内部データから読むことを避けられないならば, Giant Mutex を取得してください:

```
[...]
  pthread_mutex_lock(&gGiantMutex);
  [do whatever]
  pthread_mutex_unlock(&gGiantMutex);
[...]
```

9 参考文献

9.1 参考文献

PPP プロトコルは, 多くの RFC で文書化されています。そのいくつかを下のリストに挙げます。

RFC 1332 The PPP Internet Protocol Control Protocol (IPCP)
(http://www.ietf.org/rfc/rfc1332.txt)
RFC 1334 PPP Authentication Protocols
(http://www.ietf.org/rfc/rfc1334.txt)
RFC 1570 PPP LCP Extensions
(http://www.ietf.org/rfc/rfc1570.txt)
RFC 1661 The Point-to-Point Protocol (PPP)
(http://www.ietf.org/rfc/rfc1661.txt)
RFC 1662 PPP in HDLC-like Framing
(http://www.ietf.org/rfc/rfc1662.txt)
RFC 1877 PPP Internet Protocol Control Protocol Extensions for Name Server Addresses
(http://www.ietf.org/rfc/rfc1877.txt)
RFC 1962 The PPP Compression Control Protocol (CCP)
(http://www.ietf.org/rfc/rfc1962.txt)
RFC 1968 The PPP Encryption Control Protocol (ECP)
(http://www.ietf.org/rfc/rfc1968.txt)
RFC 1969 The PPP DES Encryption Protocol (DESE)
(http://www.ietf.org/rfc/rfc1969.txt)
RFC 1973 PPP in Frame Relay
(http://www.ietf.org/rfc/rfc1973.txt)
RFC 1974 PPP Stac LZS Compression Protocol
(http://www.ietf.org/rfc/rfc1974.txt)
RFC 1978 PPP Predictor Compression Protocol
(http://www.ietf.org/rfc/rfc1978.txt)
RFC 1979 PPP Deflate Protocol
(http://www.ietf.org/rfc/rfc1979.txt)
RFC 1990 The PPP Multilink Protocol (MP)
(http://www.ietf.org/rfc/rfc1990.txt)
RFC 1994 PPP Challenge Handshake Authentication Protocol (CHAP)
(http://www.ietf.org/rfc/rfc1994.txt)

RFC 2284 EAP
(http://www.ietf.org/rfc/rfc2284.txt)
RFC 2419 The PPP DES Encryption Protocol, Version 2 (DESE-bis)
(http://www.ietf.org/rfc/rfc2419.txt)
RFC 2427 Multiprotocol Interconnect over Frame Relay
(http://www.ietf.org/rfc/rfc2427.txt)
RFC 2433 Microsoft PPP CHAP Extensions
(http://www.ietf.org/rfc/rfc2433.txt)
RFC 2516 A Method for Transmitting PPP Over Ethernet (PPPoE)
(http://www.ietf.org/rfc/rfc2516.txt)
RFC 2548 Microsoft Vendor-specific RADIUS Attributes
(http://www.ietf.org/rfc/rfc2548.txt)
RFC 2637 Point-to-Point Tunneling Protocol (PPTP)
(http://www.ietf.org/rfc/rfc2637.txt)
RFC 2661 Layer Two Tunneling Protocol (L2TP)
(http://www.ietf.org/rfc/rfc2661.txt)
RFC 2865 Remote Authentication Dial In User Service (RADIUS)
(http://www.ietf.org/rfc/rfc2865.txt)
RFC 2866 RADIUS Accounting
(http://www.ietf.org/rfc/rfc2866.txt)
RFC 2869 RADIUS Extensions
(http://www.ietf.org/rfc/rfc2869.txt)
RFC 3579 RADIUS EAP
(http://www.ietf.org/rfc/rfc3579.txt)

9.2 功績

- EAP, RADIUS 強化, Mpd-4 のための基礎作業は SURFnet (http://www.surfnet.nl/) により後援されました。
- L2TP 強化は, Scom.ca Internet Services 社が後援しました。
- 長年のサポートと製作品テストについて Optima-Telecom に特別の感謝を捧げます。