セキュリティを考慮したブロック接続型サーバの開発

東海職業能力開発大学校　浅　井　英　史
佐々木　英　世

Development of block type server for network which considers security

Hideshi ASAI, Hideyo SASAKI

要約　近年ブロードバンドの普及にともないインターネット接続もダイアルアップ型の接続からネットワーク型の接続が増加している。しかもオフィスや家庭を問わず、場所をとらない小型ルータの需要が高まっており、中には単なるルータにとどまらずネットワークサービスを提供できるサーバタイプのものも数多く存在している。本校では、部外講師を依頼するなど兼ねてからお付き合いのある企業から、これまでのシステム・ネットワーク構築やコンピュータサポート主体の業務に加え、製品開発を行いたいとの相談を受けた。そこで前述のような状況を踏まえて、企業側がこれまでのノウハウを生かしつつ開発に取り組めるものとして、小型ネットワークサーバに、セキュリティ機能を付加したものを開発することとなった。しかし、これだけでは他社製品と変わりないため、提供するサービスごとに装置を分けて、用途に合わせて組み合わせて使用可能なブロック型とすることで新規性をアピールすることとした。

Ⅰ　はじめに

今回、事業主団体研究開発事業（F方式）による共同開発をおこなった企業は、愛知県下の情報サービス企業により構成される事業主団体に属しており、愛知県および岐阜県を活動範囲の場としてシステム・ネットワーク構築およびコンピュータ・サポート、IT教育等を手がけている。当校へも部外講師の派遣をお願いしており、兼ねてより代表者から自社製品の開発を行いたいとの相談を受けていた。話し合いを進めていく中で、もとよりネットワークサービスを提供していることと、近年小型ルータの製品化が活発であることなども踏まえ、セキュリティを考慮したルータおよびサーバで製品化を目指してみたいということで方針を立てた。具体的に、現在の顧客をベースに販売を想定し、製品のコンセプトを考えていったところ、以下のような用件を満たすことがあげられた。

○小型、省スペース、静音であること。
○ある程度カスタマイズした形で納入し、設置後の操作は極力簡略化する。
○これまでの小型ルータに無い特徴を有すること。

これらはもちろん、ルータおよびサーバの基本機能として、VPN（Virtual Private Network）やサーバセキュリティの設定がなされることを前提としての条件であるとし、加えてサーバ機能が選択できることはどうかとの提案がなされた。そこから組み合わせる仕組みについて見当し、USB（Universal Serial Bus）を用いたブロック型接続の小型ルータ兼サーバを開発することとなった。

システムを構築するにあたり、OSにはDebian Linux を採用することとした。これは企業側がこれまでに、1枚のフロッピーディスクで起動するルータを作成するなど、PC-UNIXに関しての扱いに習熟し

論文受付日　H16.1.29

ていることがあげられる。しかし現在、企業側の開発業務ではJavaがメインで、PC-UNIXのプログラム開発や組み込み関係の分野にはこれまで携わっていない。このことから、社員教育にもなるという観点でPC-UNIXを前提にした開発をおこなうこととなった。Debian Linuxのパッケージ化ツールを利用することで動作検証の開発環境が企業側と共有化できるようにした。作業は、ネットワーク関連の設定・構築を主に企業側が、本校ではその他ブロック化のための回路構成、システム構築を行うこととなった。

Ⅱ　システム構成

1　概　要

システムをCF（Compact Flash）カードに書き込んだCPUボード本体部をベースブロックとし、基本サービスはこのブロックで運用する。基本サービスとは、OSをはじめssh（Secure Shell）およびWebサーバ機能、ルータ機能である。これに対し拡張ブロックとして、その用途によりハードディスク内にFTP（File Transfer Protocol）やsamba(注1)等のサーバ機能を有し、その他ネットワークハブなどを搭載した簡易なものとなる。これらを順に積み重ねることで、目的にあったルータ兼サーバとして運用を可能とするものである。開発にあたり想定した製品のイメージは図1のとおり、機能ごとの構成は図2に示す。

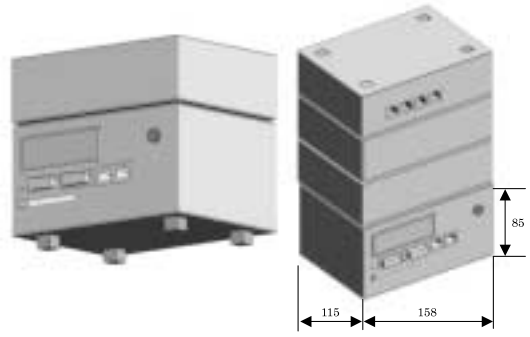

図1　製品のイメージ

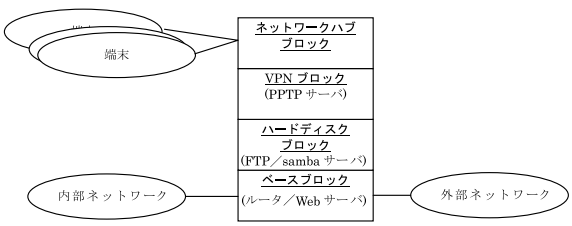

図2　ブロック単位の機能構成図

基本は4段重ね（図1右）の構成とした。上からネットワークハブブロック、VPN（セキュリティ）ブロック、ハードディスクブロックで、各役割を与える拡張ブロックとなる。一番下は、液晶表示および各インターフェースの接続口を備えた本体となるベースブロックである。これら機能を実現するためのシステムブロック図を図3に示す。各ブロック同士はUSBを介して接続され、一部機能についてはPICマイコン(注2)とシリアル通信することで制御するものである。

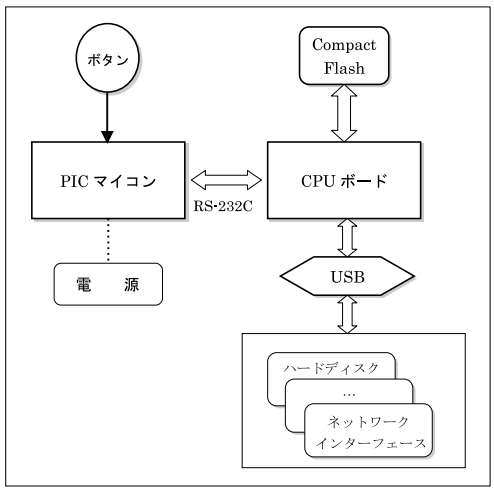

図3　システム全体の構成ブロック図

2　ベースブロックとは

本システムで使用したCPUボードはIEI社のWafer 5123というものである。選択の理由は標準でEthernetポートを2つ装備している点にある。その他、CFカードスロットを持つが、これはハードディスクのようにIDEデバイスとして起動させることも可能である。USBについては、購入当時の仕様ではUSB1.1（現行モデルではUSB2.0対応になっている）であった。また、ベースブロックの外部記憶装置にはCFカードを用いることとし、OSもそこから起動させる。CPUボードの仕様を表1に示す。その他、キーボードおよびVGAの出力端子をそなえるが、シリアルポートをコンソールとして使用できるようにした。

表1　CPUボードの主な仕様

CPU	NS Geode GX1-300MHz
Memory	256MByte
CompactFlash	96MByte
I／O	RS-232C x 2 Parallel x 1 USB1.1 x 2 FDD x 1 IDE x 1
Ethernet	Intel 82559 x 2

3　拡張ブロックとは

拡張ブロックにはハードディスクやネットワークハブを内臓し、図4に示すように、各ブロックの足の部分にUSBコネクタを配置することで、他のブロックと相互に接続できるものである。

Self Powerdデバイスを想定して、その他の足には電源供給のコネクタを収めるものとする。ネットワークサービスを追加したい場合に拡張ブロックを用いるが、これは、ハードディスクで対応することとした。

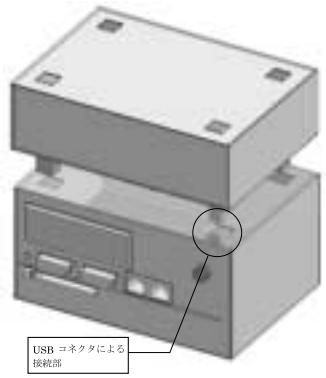

図4　接続イメージ

4　USBとマウントのしくみ

製品の基本コンセプトの中でも重点をおいたことに、拡張ブロックを積み重ねたときに、ブロック内のハードディスクを自動マウントさせる仕組みがあげられる。本システムではさらに、ブロックごとにあらかじめ設定済みのサービスを開始させることとした。これは、Linuxのカーネル2.4からPlug and Playを実現するためにHotPlug機能[(1)]が実装されるようになったことに着目したからである。この機能により、機器に必要なドライバ・モジュールのロード、およびアンロードがプラグの抜き差しに応じて自動化できることがブロック化という発想とシステムの利便性につながると考えた。そのため本システムではUSB接続による機器の使用を前提としている。使用したUSB-IDE変換基板とハードディスクおよびUSBハブ、ネットワークアダプタを図5に示す。

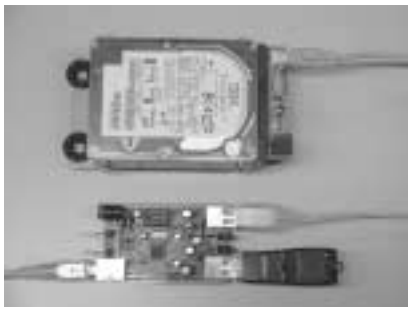

図5　USBハブとIDE変換基板

次に必要な設定を割り出すために、その動作について検証してみる。まず、カーネルはUSBデバイスのプラグインを検出すると記述してあるアプリケーション(helperプログラム)を呼び出す。実際のドライバのロードや設定処理はこのhelperプログラム（/sbin/hotplug）が行うようになっている。このhelperプログラムにはmurasaki[(2)]を使用することとした。そこで、murasakiにおける、USBデバイスでのHotPlug接続からモジュールのロードまでの流れを示す。①接続時にベンダーコードおよびプロダクトコードを読み出す。[(3)]②このときマップ（usbmap）に記述されている内容と比較して該当するデバイスの登録を探す。マップにはモジュール名またはエイリアスにつづいてベンダーコード、プロダクトコードその他が記述さている。③探し当てたモジュール名またはエイリアス名から依存関係に基づいて（murasaki.dependに記述されている）デバイスに必要なモジュールをロードする。④つづいてモジュール名、デバイス名ごとに実行させたいスクリプトがあれば、そのファイル名を指定（murasaki.callに記述）して、モジュールのロードに引き続きスクリプトを実行させることができる。

以上のことがらを、ハードディスクブロックにおける具体的な処理の流れに置き換えると次のようになる。今回使用したUSB-IDEの変換基板に関するベンダーコード／プロダクトコードは「0x411/0x2b」であったが、ドライバに識別されると「Vendor:IC25N020」および「Model:ATCS04-0」などのように認識される。このとき同時にディスクデバイスをUSB経由で接続する際はSCSI機器として扱われることもログから確認できる。これらのデバイスについて登録済みコードであればよいが、今回はマップにおいてエイリアス名“alias-sd”を与えて先のベンダーコード／プロダクトコードを追記している。

つづいてmurasaki.dependでは“alias-sd”のエイリアス名に“usb-storage”モジュールを対応させた記述をする。これによりUSBデバイスの認識とともにモジュールが読み出され使用したUSB-IDEの変換基板が認識されるためマウント可能となる。このとき、デバイスを認識するとともに/proc/scsiディレクトリ以下にはファイルが作成される。ファイル名はusb-storage-*n*であり、末尾の*n*はデバイスが追加されるごとにカウントアップされた数字が追加されるものである。murasaki.callには「モジュール名：」につづけて、マウントコマンドを含めたスクリプトファイルを指定しておく。その内容には、まず現在のマウント

状況から判断して、マウントポイントを割り当てるよう記述しておくことがあげられる。このとき、マウントポイントのディレクトリに一致するよう、blk_infoというファイルをあらかじめ拡張用ハードディスクに用意することとした。そのファイルの内容は、ハードディスクブロックに収められるサービス名を記述したものである。これにより、マウントとともにblk_infoファイルの有無を調べて、もしファイルを検出すれば、その中に記述されたサービス名に該当するデーモンプログラムを起動させるというスクリプトになっている。以上のことから、USB経由でハードディスクを接続するだけで、ハードディスク内に登録したサービスを自動実行することができた。複数のUSB-IDE変換基板にも対応するようスクリプトを改めることで、どのような順番で複数のハードディスクを接続したとしても同様にそれぞれのサービスが提供できることを確認した。

5　シャットダウンボタン

実験機の外観を図6に、電源管理回路を図7に示す。

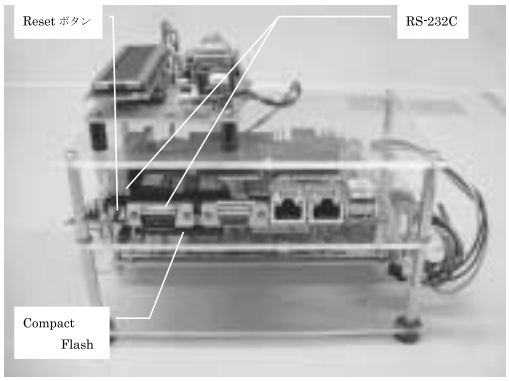

図6　実験機の外観

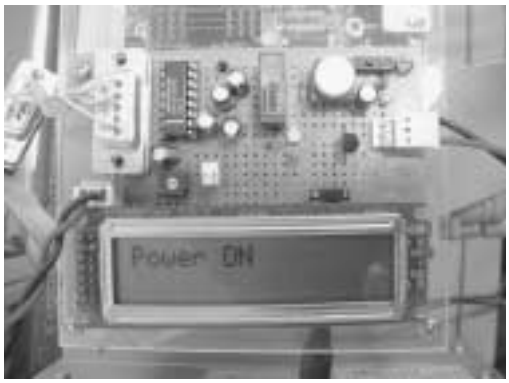

図7　電源管理回路

ベースブロックにはプッシュボタンを装備し、このボタンを押すことにより自動でシャットダウンすることとした。ボタンの押下があったことをシリアルポートを介して、CPUボード側で起動しているシリアルポート監視プログラムが受け取り、それに応じてshutdownコマンドを開始させるというものである。シャットダウン処理が終了した時点で、同じくシリアルポートを介して終了信号を電源管理を行っている回路に送信し、数秒後に本体電源を自動的に切断するようにした。

パソコンでは現在、自動的に電源が切断されることがあたりまえとなっているが、今回は工業用の5V電源を用いているため、リレー制御およびボタン操作の監視、シリアルポートとの送受信を含めた回路をPICマイコンを用いて電源管理回路を構成した。

またLinuxにおいて、シャットダウンが行われるとコンソールへの出力の最後に「Power Down」のメッセージが送られている。これをもとに、カーネルソースでシャットダウン処理を行っているところを探し、メッセージの表示後に終了信号を出すよう、カーネルソースのディレクトリにおいてsys.cの中で図8上のような行を加えた。outb関数では、シリアルポート2に対してRTS信号をオンにすることをあらわしている。これは、本システムではCPUボードのシリアルポート1（ttyS0）をコンソールに、シリアルポート2（ttyS1）を電源回路との通信に割り当てたからである。

```
case LINUX_REBOOT_CMD_POWER_OFF:
        otifier_call_chain(&reboot_notifier_list,
                                SYS_POWER_OFF, NULL);
        printk(KERN_EMERG "Power down.¥n");
        machine_power_off();
        printk(KERN_EMERG "[ttyS1] RTS ON ...¥n");
        outb(0x2, 0x2fc);
        do_exit(0);                          追加した内容
        break;
```

図8　sys.cの一部と終了画面

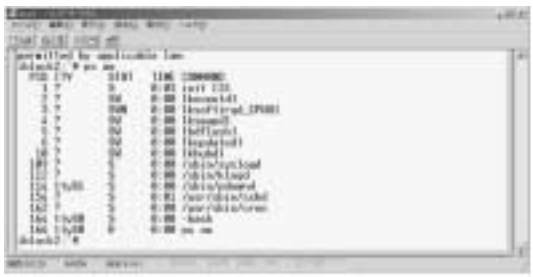

図9　プロセスの確認

このとき、シリアルポートの監視プログラム（pdownd）はデーモンプログラムとして作成し、起動時からバックグラウンドで動作させている。起動直後のプロセス実行状態を図9に示す。

このデーモンプログラムでは、シャットダウンのボタンが押されたことをPICマイコンから文字列を受け取ることで判別する。つづいてshutdownコマンドを呼び出してシステムを終了させる（図8下終了時画面）。ちなみにsys.cへの追記はシャットダウンを対象としたもので、リブート時には電源は切断されない。

6　ネットワーク機能とセキュリティ

ネットワーク機能に関しては、セキュリティを考慮してVPNを拡張ブロックで導入することで可能とした。クライアントPCにはWindowsパソコンを中心に考えているためPPTP（Point-to-Point TunnelingProtocol）によりVPNを実現させた。PPTPサーバにはPoPToPを用いたため、下記のファイルに変更が必要となる。

（1）/etc/pptpd.conf
　localip及びremoteipの設定
（2）/etc/ppp/pptpd-options
　ppp使用時のネットワーク設定
（3）/etc/ppp/chap-secrets

接続時のID及びパスワードの設定、chap-secretsファイルのserver記述の部分はpptpd-optionsファイルのservernameと合わせる必要がある。また、iptablesなどでフィルタリングされている場合は、PPTPの1723ポートおよびGRE(Generic Routing Encapsulation)のプロトコル番号47をあけておく必要がある。

WindowsパソコンからのVPN接続のためには、さらにMPPE(注3)対応のPPP(Point to Point Protocol)サーバとkernelパッチが必要であるため、事前にMPPE対応のPPPを導入し、PPPにkernelパッチを当ててからkernelを再構築した。

今回導入を見送ったものとして、セキュリティ面でのUML(User Mode Linux)の利用がある。これは外部からのアクセスに対して内部構造を隠蔽することでセキュリティを更に高めるものである。このUMLの概念を図10に示す。

ベースOS上にクライアントOS（仮想OS）を複数立ち上げ、このクライアントOSが外部からは全く別OSとして扱われるため、ベースOSを非公開とし、クライアントOSを公開することにより、ネットワークを介した攻撃に対してベースOSの存在を知られることなく、障害対策を行わせることを可能としている。

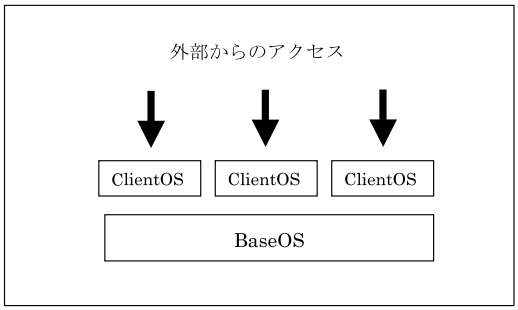

図10　UMLの概念

Ⅲ　考　察

開発期間内では、外観にあたる筐体およびUSB結合部を含む足の作成には至らなかった。結果、製品コンセプトについて実験機からの考察を以下にあげてみる。

（1）実験機では電源ボタンひとつで、終了時に自動シャットダウンさせることとしたが、USB経由でマウントしたハードディスクの取り外しなどもボタンひとつで行える利便性はある。反面、アンマウントの度にすべてのサービスが停止することになるのは運用上好ましくない。サーバとしての運用が始まれば脱着が頻繁に起こることは少ないと思われるが、対応策として、ボタンを追加してマウントを外す処理を独立させることが考えられる。そのため、現在ボタンの押下についてPICから文字コードをpdowndプログラムが受信して判断しているが、ボタンの追加とともに受信文字コードを増やし、プログラムからサービスの停止及びアンマウント処理を実行させることで可能と思われる。この点で、複数のハードディスクをマウントしている場合には選択する機能も必要と考えられる。また、複数段詰まれたブロックの内、間に挟まれるブロックについては、結局その上位のブロックについてもサービスを停止せざるを得ないなど、ブロック型であるゆえのマイナス面もあるとわかった。

（2）メンテナンスについて、ネットワークではsshで接続し、それ以外にはシリアルケーブルで接続してすぐログインきるようにした。しかし、コマンド作業が伴うなどメンテナンスの作業の簡略化に至っていない。ひとつの方法としてはベースブロックにはWebサーバ機能があるので、一般製品のようにメンテナンス用のWebページを設けるなどの対応も可能と思われる。

（3）ネットワークブロックについては、当初ルータ機能を持たせることからネットワークハブもブロックとし存在すれば、スペースの節約とネットワーク接続の集中管理になると考え追加した部分である。しかし、USB経由で接続するネットワークアダプタも数多く製品化されているため、複数のネットワークに対応したルータとしても機能させられると考えられる。

（4）実験機ではあるがその大きさについて、ベースブロック単体ではCPUボードに合わせた大きさで構成でき小型といえるが、USBコネクタを縦方向に装着することから、ハードディスク単体のブロックでもかなりの厚みができてしまうと予想された。

その他では、静音性については、電源やCPU等に冷却ファンを用いていないのでハードディスクの作動音がするくらいである。ケースがつけばさらに防音されると思われる。このことはCPUに関連しており、実験機のCPUスペックはIntel®のPentium®プロセッサ程度である。しかし、ネットワーク機能の項目で紹介したUMLについて実現するためにはかなりのCPUパワーを必要とするため、小型化には限度がある。小型化や省電力を考慮すると、日立SHプロセッサを使ったものをはじめさらに別のターゲットも該当する。さらに、そのようなCPUボードにおいても最近になってUSB2.0を搭載した製品も見受けられるので検討が必要と思われる。

Ⅳ　おわりに

今回、サーバの機能構成として、ネットワークサーバ機能、そしてセキュリティ機能について取り組んだが、それぞれがすでに単体で存在するものではある。しかし、統一的なシステムで、なおかつ、組み合わせ自由なブロック型の製品として提供できることはこの開発において十分確認できたと思われる。最終的な外観を完成させるに至らなかったため、試作品の完成を目標に今後も作業に取り組んでいくこととなった。

また本来、開発期間を短縮する意味でPC-UNIX環境が使用されることが多いが、今回は企業側の人材育成の意味も含まれたものだった。普段はただ手順として示される設定を行うばかりだが、必要に応じてソースプログラムをじっくりと解析し、よく動作を理解した上で設定を考えていくなど、この開発がスキルアップにつながったと思われる。

本取り組みにおいて、企業側は従業員数の限られたベンチャー企業であり、担当者もなかなか日ごろの業務の中では開発に専念することもできず、当初予定の期間では試作品のめどをたてるにとどまってしまった。しかし、日ごろの訓練では聞くことのない意見や疑問点を話し合いの中から見出すことができ、今後の企業人スクールの実施内容やコース開発において大変参考になった。

最後に、本取り組みにおいて多大なるご協力をいただきました有限会社インテックスの芝田社長他スタッフの方々にはこの場を借りて厚く御礼申し上げます。

［注］

（注1）sambaはUNIXおよびUNIX互換マシンをWindows互換のファイルサーバ／プリントサーバにするオープン・ソフト・ソフトウェア

（注2）PICはMicrochip Technology社のマイクロコントローラ（制御用IC）製品ファミリーの名称

（注3）MPPE（Microsoft Point to Point Encryption）はマイクロソフトのRASで使用されるポイント・ツー・ポイント用の暗号化技術

［参考文献］

（1）http://www.linux-usb.org/

（2）http://www.dotaster.com/~shuu/linux/murasaki/index_ja.html

（3）特集「USB対応機器&デバイスドライバの作成法」インターフェイス 1998年11月号，CQ出版

（4）W.Richard Stevens：詳細UNIXプログラミング［新装版］、ピアソン・エデュケーション、2001年、pp.401-409

（5）http://www.openh323.org/