教育用 OSxv6 のハイパバイザ BitVisor への移植と プロトコルスタックの実装

高井淳光[1]　吉村悠愛[1]　並木美太郎[1]

キーワード：OS,　xv6,　BitVisor,　Virtio,　lwIP

Porting educational operating systems xv6 to BitVisor and Implementation of TCP/IP Protocol stack

AKIHIRO TAKAI[†1]　YUITO YOSHIMURA[†1]
MITARO NAMIKI[†1]

Keywords: OS,　xv6,　BitVisor,　Virtio,　lwIP

1. はじめに

小型かつ軽量で実装の見通しが良いハイパバイザである BitVisor [1]はその特徴から機能拡張が他のハイパバイザと比べて容易である．拡張した機能を確認するためにゲストオペレーティングシステムが必要となるが，Linux などの OS は巨大かつ複雑であるため，その目的には適さない．そこで，実装が単純な教育用 OS である xv6 [2]の BitVisor 上での動作を考える．xv6 は PC/AT 互換機などに移植された例もあり，ビルドは容易であるが，Bitvisor の先行例はない．そこで，本稿では教育用 OS xv6 を BitVisor 上に移植し，その上で BitVisor が提供する機能の一つである Virtio を用いて物理デバイスへアクセスする．つまり今回はネットワークインターフェースカード(NIC)に BitVisor を介して操作することになる．また，オープンソース TCP/IP プロトコルスタック実装の lwIP を xv6 へ移植し，TCP/IP 通信を可能とする．この機能を用いて，カーネルには独自に定義した簡易ネットワークファイルシステム(NFS)のプロトコルに準拠したクライアントを実装し，アプリケーションに対してはソケット準拠の API を提供する．

なお，本研究は学部３年の製作実験によるものである．

2. 目標

教育用 OS の xv6 を BitVisor 上で動作させることが第一の目標である．そして xv6 に Virtio デバイス，特に virtio-net デバイスのデバイスドライバを実装する．これに加えて，オープンソースな TCP/IP プロトコルスタックの独立な実装である lwIP を移植することで xv6 を TCP/IP 通信に対応させる．これによって得られた機能を用いて我々が独自に定義する簡易的な NFS を実装する．また，ハイパバイザを用いたセキュリティ実験を行えるようにする．

3. 全体構成

システムの構成を図 1 に示す．最も下層にはハードウェアが位置しており，その上でハイパバイザ BitVisor が動作している．そのゲスト OS として xv6 が動作し，アプリケーションは xv6 の上で動作する．今回は，物理ハードウェアに NIC が含まれていることを前提とし，このデバイスに対して，BitVisor は準仮想化を行う．これによりゲスト OS には virtio-net デバイスを見せる．xv6 には BitVisor が提供する Virtio フレームワークに沿って virtio-net デバイスを操作するデバイスドライバを実装する．加えて，lwIP を xv6 へ移植し virtio-net デバイスのデバイスドライバと組み合わせることで xv6 に TCP/IP 通信の能力を実装する．また，後述する独自の簡易ネットワークファイルシステムは lwIP の API を利用して実装するものとする．lwIP の一部の API は UNIX 系 OS を参考にシステムコールとしてアプリケーションに公開する．xv6 カーネル内に実装する簡易 NFS クライアントの機能も lwIP と同様にシステムコールを介してアプリケーションに公開する．また，簡易 NFS のサーバは一般的な Linux マシン上に実装し，xv6 上のアプリケーションとの通信を行う．

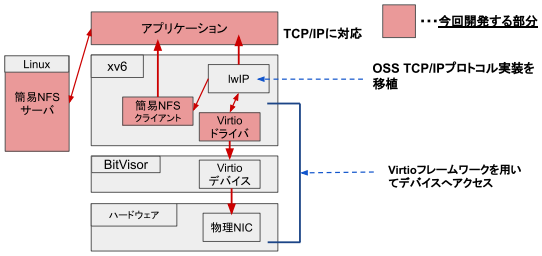

図 1　全体構成図
Figure 1 Overall Configuration Diagram

1　東京農工大学　工学部情報工学科
Department of Computer and Information Sciences,　Tokyo University of Agriculture and Technology

4.特徴

BitVisor 上で xv6 を動作させた先行例はなく，これに lwIP 等の独立した TCP/IP プロトコルスタックを移植した例は見受けられない．また，xv6 に実装するデバイスドライバは物理デバイスに直接アクセスせずに Virtio により準仮想化されたデバイスを操作することでハイパバイザを介してアクセスするという特徴を持つ．なお BitVisor 上で実行される xv6 は root mode ring0 で動作する．

5. 設計

(1) **BitVisor**

BitVisor はビルド時に NET_VIRTIO_NET フラグを有効化し virtio-net デバイスを提供できるようにした上でインストールする．BitVisor は実験用マシンのローカルに存在する HDD にインストールし，GNU GRUB を用いて起動する．

(2) **BitVisor 上の動作に伴う xv6 の修正点**

(a) **起動**

xv6 は Legacy BIOS が有効なマシンで起動することが前提とされた設計であるため，実験用コンピュータは Legacy BIOS を提供することができるものを用いる．今回は GNU GRUB から起動することを前提にしている．GNU GRUB は OS をロードする前にリアルモードからプロテクテッドモードに移行させている．しかし，xv6 はオリジナルのブートローダから起動することを前提にソースを記述している． したがって，エントリポイントのアドレスを仮想アドレスから物理アドレスへ変換を行う． GNU GRUB から起動する場合はこの処理は不要になるため，修正を行う．また，GDT（Global Descriptor Table）に設定するセグメントについても GNU GRUB は先頭のセグメントを予約しているため、xv6 にはこれとの重複を避けるように既定のセグメントをずらす修正を加える．加えて，xv6 にはハードディスクドライブ（HDD）を扱うために IDE デバイスのドライバが実装されているが，近年のコンピュータには PATA-IDE が搭載されていないためファイルを扱うことができない。これを解決するために RAM を RAM DISK の仮想 IDE として扱うように xv6 のビルド時に設定を行った．

(b) **Virtio**

Virtio の中でも特に NIC 等のデバイスを準仮想化した virtio-net デバイスに対してデバイスドライバを実装する．これにより，xv6 カーネルはネットワーク関連の機能を利用することができるようになり，加えて後述する簡易 NFS でリモートファイルを扱えるようにした．これらの機能をアプリケーションが利用することができるように，表 1 に示すシステムコールを機能拡張および新規追加する．

表 1　機能拡張と新規追加のシステムコール
Table 1　Enhancements and New System calls

機能拡張	read(int fd, void *buf, size_t len)	write((int fd, void *buf, size_t len)	close(int fd)
	open(char *path, int flugs)	mkdir(char *path)	unlink(char *path)
新規追加	socket(int family, int type, int protocol)	bind(int sockfd, u8 *addr, u16 port, size_t addrlen)	listen(int sockfd)
	connect(int sockfd, u8 *addr, u16 port, size_t addrlen)	accept(int sockfd, u8 *addr, u16 port, size_t addrlen)	select(int max, fd_set *readset, fd_set *writeset, fd_set *exset, timeval *timeout)
	send(int sockfd, void *buf, size_t length, int flag)	recv(int sockfd, void *buf, size_t length, int flag)	seek(int fd, int offset)

(3) **プロトコルスタックの実装**

xv6 に対する TCP/IP プロトコルスタックの実装には，オープンソースの独立した TCP/IP プロトコルスタック実装の一つである lwIP を移植する形で行う．lwIP は独立した TCP/IP プロトコルスタック実装として広く利用されており，xv6 への移植例もあるため，今回利用する．物理 NIC との連携は先述のとおり，xv6 カーネル内に実装する Virtio-net のデバイスドライバを用いる．lwIP が提供する API は UNIX 系 OS のネットワーク関連システムコールを参考にアプリケーションに公開する．

(4) **簡易 NFS**

BitVisor は Virtio で提供されるデバイスの中でも virtio-net のみに対応しているため，HDD などのファイルシステムの実装が別に必要となる．そこで，TCP/IP を用いて簡易 NFS を作成する．実装する簡易 NFS は読み書きを始めとする基本的な操作に対応する簡易的なものである．サーバとクライアントはメソッドのやり取りを介してデータ転送を行う．これらは，xv6 に移植した lwIP の機能を用いて実装する．リモートファイルに対しては IP アドレスとファイルパスという形で目的のファイルを指定する．簡易 NFS のサーバプログラムは一般的な Linux マシン上に実装し，xv6 上のアプリケーションとの通信を目標とする．

6. 現状と今後の課題

現在，Virtio-net のデバイスドライバを開発している．また，lwIP を xv6 のソースツリーに加えた状態でのビルドには成功している．今後はシステムコールの実装，lwIP とドライバの協調動作，簡易 NFS の実装を行う予定である．

参考文献

[1] “BitVisor: A Single-VM Lightweight Hypervisor”. https://www. bitvisor. org/
[2] “Xv6, a simple Unix-like teaching operating system”. https://pdos. csail. mit. edu/6. 828/2012/xv6. html