3P-8

ラップトップUNIXのネットワーク機能(2)

伊藤 敏美 高橋 勉

(株)東芝・青梅工場

1. はじめに

UNIX System V Release 3(以降、SVR3)ベースのラップトップ UNIX ワークステーション上に、ストリームを用いて TCP/IP プロトコルを実現し、TLI/TPI と呼ばれる SVR3 インタフェイスと BSD 系のソケット・インタフェースを同時にサポート可能とした。このことにより、ftp/telnet などの BSD 系の通信コマンドはもとより、RFS や CU/UUCPなどの SVR3 通信機能が TCP/IP 上で動作することが可能となった。

本稿では、ストリーム上に TCP/IP モジュールを実現する場合の問題点、特に TLI/TPI インタフェースやソケット・インタフェースとの整合性に関する問題点を中心に、その解決方法や実現方式の概要について述べる。

2. TLI/TPIとソケット・インタフェース

SVR3 では、ストリームと呼ばれる入出力機構がサポートされている。ストリームベースのドライバやモジュールは、(インタフェースが合致すれば)自由に組み合わせることができるため、多様な通信形態に柔軟に対応することが可能である。

一方、ストリームのこのような柔軟性を活かすには、モジュール間のインタフェースを統一しておくことが不可欠である。このため SVR3 では、トランスポート層に対する共通インタフェースとして TLI/TPI と呼ばれるインタフェースが定義されている。

TLI (Transport Level Interface) とは、具体的にはネットワーク支援ユーティリティによって提供されるライブラリ関数群によって提供されるサブルーチン・インタフェースを指し、ユーザ・レベルからトランスポート層をアクセスするためのインタフェースである。

また、TPI (Transport Provider Interface) とは、UNIX カーネル内に位置するトランスポート・プロバイダの上位インタフェースであり、ストリーム・メッセージ・インタフェースとなっている。

一方、BSD系UNIX の通信インタフェースは、socket、bind、listen などのシステム・コール群、いわゆるソケット・インタフェースである。ソケット・インタフェースは、SVR3 における TLI に対応するものである。

実際、ソケット・インタフェースを構成する各システムコールと、TLI を構成するライブラリ・ルーチンとではその名称、機能などに類似点も多い。しかし、これらのインタフェースを詳細に比較すると、以下のような差異があることがわかる。

(a) 着信シーケンス
(t_)bind、(t_)listen、(t_)acceptなどの意味が異なっており、ソケットでは listen 後は accept しなくてもコネクションは確立するが、TLI では t_accept 後に初めてコネクションが確立する。

(b) bind の省略
ソケットでは bind は省略できるが TLI では省略不可である。

(c) UDP における connect
SOCK_DGRAM のソケットでは connect が使用可能であるが、TLI ではコネクションレスの場合には、t_connect はエラーとなる。

(d) データの境界
TLI では、MOREフラグによりデータの境界/区切りがサポートされるが、SOCK_STREAM タイプのソケットではデータは単なるバイト・ストリームである。

(e) エラー番号
ベースOSの違いによりソケット関連のシステムコールから返されるエラー番号が System V では定義されていない。

3. 実現方式

(1) ソフトウェア構成

SVR3 にストリーム版 TCP/IP、およびソケット・インタフェースを導入するには例えば以下のような方式が考えられる。

(a) TLI 上にソケット・インタフェースを作成する。
(b) ストリーム上に直接ソケット・インタフェースを作成する。
(c) プロトコル・モジュールがストリーム、および非

---

*) UNIX は AT&T が開発しライセンスする OS です。

Network Facilities on Laptop UNIX Workstation (2)
Toshimi ITOH,Tsutomu TAKAHASHI
TOSHIBA Corp. OME works.

ストリームの2つのインタフェースを用意する。

本ワークステーションでは処理のオーバーヘッドやカーネル基本部への影響などを考慮して(b)の方式とした。

(2) ストリーム・モジュール構成

TCP/IP とはプロトコルの総称であり、細かく見るとTCP、UDP、IP、ICMP、ARP などのプロトコルから構成される。当初、これらの各プロトコル毎にストリームモジュール化する案が検討されたが、

(a) 不必要にモジュールの細分化を行うとシステムオーバーヘッドが増すこと

(b) これらの各プロトコルの関係が今後変化する可能性が少ないこと（例えば、TCP の下位に OSI の CLNP を使用するなど）

などの点から1つのストリーム・モジュールの中で複数プロトコルをサポートする方式とした。

インターネット通信プロトコルの中核をなす TCP、UDP、IP、および ICMP の各プロトコルはn入力、m出力のストリームドライバ（マルチプレクサ）として実現した。一方、ARP、トレイラ・プロトコルなどは LAN 通信特有のものであるので、将来の拡張性に備えて別モジュールとし、取りはずし可能な構成とした。

(3) トランスポート・サービス

TCP の場合、コネクション強制解放のための RST（リセット）フラグの他に、順次解放を行うための FIN フラグが使用できるので T_COTS_ORD に対応する。

なお、UDP は T_CLTS であり、オプション設定要求(T_OPTMGMT_REQ) によりサービス・クラスを選択できるようになっている。

(4) 拡張された TPI

TLI とソケット・インタフェースの差は原則としてライブラリで吸収するが、必要なものについてはプロバイダで対応する方式とした。

すなわち、ソケット・ライブラリからアクセスした場合には使用するサービス・プリミティブは TLI からアクセスした場合と同じであるが、許されるシーケンス（状態遷移マトリクス）が拡張されており、例えばバインド要求 T_BIND_REQ を行なっていなくても接続要求 T_CONN_REQ を受入れるなどの対応を行なった。

(5) MOREフラグ

TLI/TPI では、データの送受信において継続データの有無を示す MORE フラグが使用可能である。この機能がサポートされないと、アプリケーションによっては TCP/IP 上では動作しないことも考えられる。

本システムでは図に見るように、MORE = 0 のデータと TCP における PUSH フラグを対応させることにより MORE フラグのサポートを行なった。なお、ソケット・インタフェースからアクセスした場合には常に MORE = 1 としている。

T_BIND_REQ	⟶		⟵	T_BIND_REQ
T_BIND_ACK	⟵		⟶	T_BIND_ACK
T_CONN_REQ	⟶	SYN	⟶	T_CONN_IND
T_OK_ACK	⟵			
T_CONN_CON	⟵	SYN + ACK	⟵	T_CONN_RES
	⟶	ACK	⟶	T_OK_ACK
T_DATA_REQ	⟶	ACK	⟶	T_DATA_IND
(MORE = 1)				(MORE = 1)
T_DATA_REQ	⟶	ACK + PUSH	⟶	T_DATA_IND
(MORE = 0)	⟵	ACK	⟵	(MORE = 0)
T_ORDREL_REQ	⟶	FIN	⟶	T_ORDREL_IND
	⟵	ACK of FIN	⟵	
T_ORDREL_IND	⟵	FIN	⟵	T_ORDREL_REQ
	⟶	ACK	⟶	
T_UNBIND_REQ	⟶		⟵	T_UNBIND_REQ
T_OK_ACK	⟵		⟶	T_OK_ACK

TPI(T_COTS_ORD) の場合のシーケンス例

(6) エラー番号

System V と BSD とではシステム・エラーコードが異なっており、ソケットのサポートに伴い関連するエラーコードを導入する必要がある。本システムでは、エラーや例外事象は TLI/TPI エラーコードによって通知され、ソケット・ライブラリの中で BSD エラーコードへの変換を行なう方式とした。これにより read/write システム・コールを使用した場合でも System V Interface Definition を維持することができる。

4. むすび

ストリーム上に TCP/IP モジュールを実現し、TLI/TPI とソケット・インタフェースを両立させることにより、RFS などの SVR3 通信機能と BSD 系の通信機能を同時に使用することが可能になった。

今後の課題として、ラップトップの特徴の1つである可搬性とネットワーク機能との問題について検討してみたい。