Implementing SIGTRAN for Linux Fast-STREAMS

Design for Linux

Brian F. G. Bidulock[*]
OpenSS7 Corporation

June 16, 2007

Abstract

1 Background

UNIX networking has a rich history. The TCP/IP protocol suite was first implemented by BBN using Sockets under a DARPA research project on 4.1aBSD and then incorporated by the CSRG into 4.2BSD [MBKQ97]. Lachmann and Associates (Legent) subsequently implemented one of the first TCP/IP protocol suite based on the Transport Layer Interface (TLI) [TLI92] and STREAMS [GC94]. Two other predominant TCP/IP implementations on STREAMS surfaced at about the same time: Wollongong and Mentat.

1.1 STREAMS

STREAMS is a facility first presented in a paper by Dennis M. Ritchie in 1984 [Rit84], originally implemented on 4.1BSD and later part of the *Bell Laboratories Eighth Edition UNIX*, incorporated into *UNIX System V Release 3* and enhanced in *UNIX System V Release 4* and further in *UNIX System V Release 4.2*. STREAMS was used in SVR4 for terminal input-output, pseudo-terminals, pipes, named pipes (FIFOs), interprocess communication and networking. STREAMS was used in SVR3 for networking (in the NSU package). Since its release in *System V Release 3*, STREAMS has been implemented across a wide range of UNIX, UNIX-like and UNIX-based systems, making its implementation and use an ipso facto standard.

STREAMS is a facility that allows for a reconfigurable full duplex communications path, *Stream*, between a user process and a driver in the kernel. Kernel protocol modules can be pushed onto and popped from the *Stream* between the user process and driver. The *Stream* can be reconfigured in this way by a user process. The user process, neighbouring protocol modules and the driver communicate with each other using a message passing scheme. This permits a loose coupling between protocol modules, drivers and user processes, allowing a third-party and loadable kernel module approach to be taken toward the provisioning of protocol modules on platforms supporting STREAMS.

On *UNIX System V Release 4.2*, STREAMS was used for terminal input-output, pipes, FIFOs (named pipes), and network communications. Modern UNIX, UNIX-like and UNIX-based systems providing STREAMS normally support some degree of network communications using STREAMS; however, many do not support STREAMS-based pipe and FIFOs[1] or terminal input-output[2] without system reconfiguration.

UNIX System V Release 4.2 supported four Application Programming Interfaces (APIs) for accessing the network communications facilities of the kernel:

Transport Layer Interface (TLI). *TLI* is an acronym for the *Transport Layer Interface* [TLI92]. The *TLI* was the non-standard interface provided by SVR3 and SVR4, later standardized by *X/Open* as the *XTI* described below. This interface operated differently than the XTI in subtle ways, and is now deprecated.

X/Open Transport Interface (XTI). *XTI* is an acronym for the *X/Open Transport Interface* [XTI99]. The *X/Open Transport Interface* is a standardization of the *UNIX System V Release 4*, *Transport Layer Interface*. The interface consists of an Application Programming Interface implemented as a shared object library. The shared object library communicates with a transport provider *Stream* using a service primitive interface called the *Transport Provider Interface*[TPI99].

While *XTI* was implemented directly over STREAMS devices supporting the *Transport Provider Interface (TPI)* [TPI99] under SVR4, several non-traditional approaches exist in implementation:

Berkeley Sockets. Sockets uses the BSD interface that was developed by BBN for the TCP/IP protocol suite under DARPA contract on 4.1aBSD and released in 4.2BSD. BSD Sockets provides a set of primary API functions that are typically implemented as system calls. The BSD Sockets interface is non-standard, operated differently from the POSIX interface in subtle ways, and is now deprecated in favour of the POSIX/SUS standard Sockets interface.

POSIX Sockets. Sockets were standardized by X/Open, later the OpenGroup,[3] and IEEE in the POSIX standardization process. They appear in XNS 5.2 [XNS99], SUSv1 [SUS95], SUSv2 [SUS98] and SUSv3 [SUS03]. POSIX/SUS Sockets is now the common application environment for accessing networking, deprecating the XTI for TCP/IP networking applications.

[*]bidulock@openss7.org

1. AIX, for example.
2. HP-UX, for example.
3. *http://www.opengroup.org/*

On systems traditionally supporting Sockets and then retrofitted to support STREAMS, there is one approach toward supporting *XTI* without refitting the entire networking stack:[4]

XTI over Sockets. Several implementations of STREAMS on UNIX utilize the concept of *TPI* over Sockets. Following this approach, a STREAMS pseudo-device driver is provided that hooks directly into internal socket system calls to implement the driver, and yet the networking stack remains fundamentally BSD in style.

Typically there are two approaches to implementing XTI on systems not supporting STREAMS:

XTI Compatibility Library. Several implementations of XTI on UNIX utilize the concept of an XTI compatibility library.[5] This is purely a shared object library approach to providing *XTI*. Under this approach it is possible to use the *XTI* application programming interface, but it is not possible to utilize any of the STREAMS capabilities of an underlying *Transport Provider Interface (TPI)* stream.

TPI over Sockets. An alternate approach, taken by the *Linux iBCS* package was to provide a pseudo-transport provider using a legacy character device to present the appearance of a STREAMS transport provider.

Conversely, on systems supporting STREAMS, but not traditionally supporting Sockets (such as SVR4), there are four approaches toward supporting BSD and POSIX Sockets based on STREAMS:

Compatibility Library Under this approach, a compatibility library (`libsocket.o`) contains the socket calls as library functions that internally invoke the TLI or TPI interface to an underlying STREAMS transport provider. This is the approach originally taken by SVR4 [GC94], but this approach has subsequently been abandoned due to the difficulties regarding fork(2) and fundamental incompatibilities deriving from a library only approach.

Library and cooperating STREAMS module. Under this approach, a cooperating module, normally called `sockmod`, is pushed on a Transport Provider Interface (TPI) Stream. The library, normally called `socklib` or simply `socket`, and cooperating `sockmod` module provide the BBN or POSIX Socket API. [VS90] [Mar01]

Library and System Calls. Under this approach, the BSD or POSIX Sockets API is implemented as system calls with the sole exception of the `socket`(3) call. The underlying transport provider is still an *TPI*-based STREAMS transport provider, it is just that system calls instead of library calls are used to implement the interface. [Mar01]

System Calls. Under this approach, even the socket(3) call is moved into the kernel. Conversion between POSIX/BSD Sockets calls and TPI service primitives is performed completely within the kernel. The sock2path(5) configuration file is used to configure the mapping between STREAMS devices and socket types and domains [Mar01].

1.1.1 Standardization.

During the POSIX standardization process, networking and Sockets interfaces were given special treatment to ensure that both the legacy Sockets approach and the STREAMS approach to networking were compatible. POSIX has standardized both the XTI and Sockets programmatic interface to networking. STREAMS networking has been POSIX compliant for many years, BSD Sockets, POSIX Sockets, TLI and XTI interfaces, and were compliant in the *SVR4.2* release. The STREAMS networking provided by *Linux Fast-STREAMS* package provides POSIX compliant networking.

Therefore, any application utilizing a Socket or Stream in a POSIX compliant manner will also be compatible with STREAMS networking.[6]

1.2 Linux Fast-STREAMS

The first STREAMS package for Linux that provided SVR4 STREAMS capabilities was the *Linux STREAMS (LiS)* package originally available from GCOM [LiS]. This package exhibited incompatibilities with SVR 4.2 STREAMS and other STREAMS implementations, was bugger and performed very poorly on Linux. These difficulties prompted the OpenSS7 Project [SS7] to implement an SVR 4.2 STREAMS package from scratch, with the objective of production quality and high-performance, named *Linux Fast-STREAMS* [LfS].

The OpenSS7 Project also maintains public and internal version of the *LiS* package. The last public release was *LiS-2.18.3*; the current internal release version is *LiS-2.18.6*. The current production public release of *Linux Fast-STREAMS* is *streams-0.9.3*.

2 Objective

3 Description

4 Method

5 Results

6 Analysis

7 Conclusions

8 Future Work

9 Related Work

References

[GC94] Berny Goodheart and James Cox. *The magic garden explained: the internals of UNIX System V Release 4, an open systems design / Berny Goodheart & James Cox.* Prentice Hall, Australia, 1994. ISBN 0-13-098138-9.

[LfS] Linux Fast-STREAMS – A High-Performance SVR 4.2 MP STREAMS Implementation for Linux. http://www.openss7.org/download.html.

4. This approach is taken by True64 (Digital) UNIX.

5. One was even available for Linux at one point.

6. This compatibility is exemplified by the `netperf` program which does not distinguish between BSD or STREAMS based networking in their implementation or use.

[LiS] Linux STREAMS (LiS). http://www.openss7.-org/download.html.

[Mar01] Jim Mario. Solaris sockets, past and present. *Unix Insider*, September 2001.

[MBKQ97] Marshall Kirk McKusick, Keith Bostic, Michael J. Karels, and John S. Quaterman. *The design and implementation of the 4.4BSD operating system.* Addison-Wesley, third edition, November 1997. ISBN 0-201-54979-4.

[Rit84] Dennis M. Ritchie. A Stream Input-output System. *AT&T Bell Laboratories Technical Journal*, 63(8):1897–1910, October 1984. Part 2.

[SS7] The OpenSS7 Project. http://www.openss7.-org/.

[SUS95] Single UNIX Specification, Version 1. Open Group Publication, The Open Group, 1995. http://www.opengroup.org/onlinepubs/.

[SUS98] Single UNIX Specification, Version 2. Open Group Publication, The Open Group, 1998. http://www.opengroup.org/onlinepubs/.

[SUS03] Single UNIX Specification, Version 3. Open Group Publication, The Open Group, 2003. http://www.opengroup.org/onlinepubs/.

[TLI92] Transport Provider Interface Specification, Revision 1.5. Technical Specification, UNIX International, Inc., Parsipanny, New Jersey, December 10 1992. http://www.openss7.org/docs/-tpi.pdf.

[TPI99] Transport Provider Interface (TPI) Specification, Revision 2.0.0, Draft 2. Technical Specification, The Open Group, Parsipanny, New Jersey, 1999. http://www.opengroup.org/-onlinepubs/.

[VS90] Ian Vessey and Glen Skinner. Implementing Berkeley Sockets in System V Release 4. In *Proceedings of the Winter 1990 USENIX Conference.* USENIX, 1990.

[XNS99] Network Services (XNS), Issue 5.2, Draft 2.0. Open Group Publication, The Open Group, 1999. http://www.opengroup.org/onlinepubs/.

[XTI99] XOpen Tranport Interface (XTI). Technical Standard XTI/TLI Revision 1.0, X Programmer's Group, 1999. http://www.opengroup.-org/onlinepubs/.