USB OVER TCP/IP

Marek Vavruša
Bachelor Degree Programme, FIT BUT
E-mail: xvavru00@stud.fit.vutbr.cz

Supervised by: Zdeněk Vašíček
E-mail: vasicek@fit.vutbr.cz

ABSTRACT

USB *(Universal Serial Bus)* has become major peripheral bus in modern computers. However, devices connected to the USB are usually bound to a single computer. But it is highly desirable to extend their functionality over heterogenous IP *(Internet Protocol)* networks, preferably without any modification of existing environment. In this paper, a new method for the USB virtualization and device sharing is presented and evaluated.

1 INTRODUCTION

Peripheral device sharing is often desirable in network environments due to a limited number of physically available devices, mobile workplaces or a centralized network architecture. Recently, a lot of effort has been invested into the design of a device sharing over IP networks in form of a *Linux* kernel module[2]. Even though this approach supports wider variety of devices, it also bears several drawbacks in comparison with the proposed solution such as non-multiplatform solution, problematic kernel module installation and ad-hoc usage.

The proposed approach operates completely in userspace and provides an efficent solution for ad-hoc device sharing with low overhead.

2 PROPOSED APPROACH

The presented method intercepts function calls to a low-level USB library and transports them over IP network with open and lightweight version of RPC (Remote Procedure Call) protocol. This approach ensures data integrity and type-safe data in payload. I have chosen *libusb*[1] as the intercepted library for it's wide use and compatibility, notably for FTDI-based devices, drivers for digital cameras, security systems etc.

2.1 INTERCEPTING SHARED LIBRARY SYMBOLS

Shared libraries are loaded into memory during an executable runtime, results in reducing executable[3] size and further conserves memory by sharing the library instance between multiple processes. The most important feature of the shared libraries is that the executable linked against the shared library resolves imported symbols during runtime, thus allows us to intercept

symbols by preloading custom library with matching ABI *(Application Binary Interface)*. Since the symbol intercepting is supported on all major platforms (i.e. Linux/BSD and Windows), the proposed approach can be used to intercept *libusb* function calls.

2.2 NETWORK PROTOCOL

In order to achieve the good performance and maintain protocol extensibility *(unlike in USB/IP[2])*, I have designed a binary protocol, based on a subset of open ASN.1 and BER standards[4]. The network protocol is designed as byte-order independent and, unlike USB/IP, type-safe with emphasis on low overhead and low parsing complexity.
Metadata size is *2 - 6* octets per encoded value *(table 1)*.

Octets:	1	2 - 5	N
Value:	Type	Packed size	Value data

Table 1: Encoded value definition.

The proposed protocol is capable of transporting a raw binary data, offers size packing and automatically handles byte-order conversions in a heterogenous networks.

3 IMPLEMENTATION

The implemented library provides a C and C++ object-oriented API, both utilizing BSD sockets.

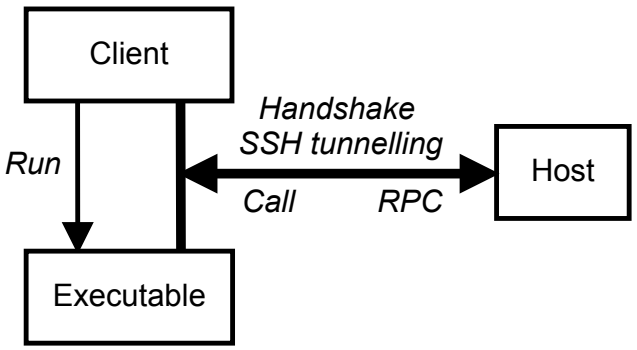

Figure 1: Network scheme.

Client wrapper is designed to preload a custom library and run the wrapped executable. The wrapper itself supports an optional SSH public-key authentication and data encryption. Remote socket descriptor is passed to the executable in SHM *(Shared Memory)* and closed on the executable exit. Calls to the wrapped functions are serialized by a global mutex.

Host daemon works as a multi-client RPC service with the ability to bind to the localhost only. This way, a remote client public-key authentication is enforced. The daemon can be extended with the custom handlers to enable rapid development for other wrapped libraries. Incoming events are handled synchronously to prevent race conditions.

4 EXPERIMENTS

The project implementation was tested in Linux and FreeBSD, but it is easily portable to the OS X/Darwin and Microsoft Windows. The concept of *shared library symbol intercepting* works without any environment modification and it is extensible to any shared library. The proposed network protocol is byte-order independent and supports an optional authentication and data encryption with SSH *(Secure SHell)*. Performance tests *(table 2)* proved a very low overhead, especially in tests transferring data in larger blocks (test 4). An extra overhead in the third test is caused by the sequential character mode transfers, multiplied by the network latency.

	Probing USB	*Terminal*	*Flashing FITkit*	*Downloading digital camera*
Native [s]	0.48	1.41	11.91	4.95
Over TCP/IP [s]	0.52	1.41	13.82	4.98
Overhead	8.33%	0%	16.04%	0.61%

Table 2: Performance tests (unbuffered, flushed disk cache, repeated).

5 RESULTS

I have designed a completely new approach for accessing the USB devices over IP networks and I have also implemented a working prototype. The main advantages of the proposed approach are: ad-hoc usage, extensible type-safe network protocol and secure connection, while keeping very good performance. The proposed approach is naturally multiplatform and supports a wide number of devices. Future research will be focused on working Windows and Darwin/OS X clients, interoperability with USB/IP[2] and a protocol extension to other libraries.

Acknowledgement: This work was partially supported by the BUT FIT grant FIT-S-10-1 and the research plan MSM0021630528.

REFERENCES

[1] Johannes Erdfelt and others.: libusb - User-space USB library (LGPL 2.1).
http://www.libusb.org

[2] Takahiro Hirofuchi , Eiji Kawai , Kazutoshi Fujikawa , Hideki Sunahara, USB/IP: a peripheral bus extension for device sharing over IP network.
Proceedings of the annual conference on USENIX Annual Technical Conference, p.42-42, April 10-15, 2005, Anaheim, CA

[3] Daniel S. Myers and Adam L. Bazinet: Intercepting Arbitrary Functions on Windows, UNIX, and Macintosh OS X Platforms
Center for Bioinformatics and Computational Biology, Institute for Advanced Computer Studies, University of Maryland, 2004.
http://lattice.umiacs.umd.edu/files/functions_tr.pdf

[4] ASN.1 (ITU-T Rec. X.680).
http://www.itu.int/ITU-T/asn1/