Tun2socks

A fast and cross-platform implementation with C++ and boost.asio

Source: https://github.com/wtdcode/tun2socks

If you find it interesting or useful, don’t hesitate to give it a star :).

Contents

- Background
- Reasons for reinventing the wheel
- Tuntap device and SOCKS5 protocol
- Framework
- Three difficulties
- Demo

Background

SSTAP
Windows
Tap Driver and ss/socks5

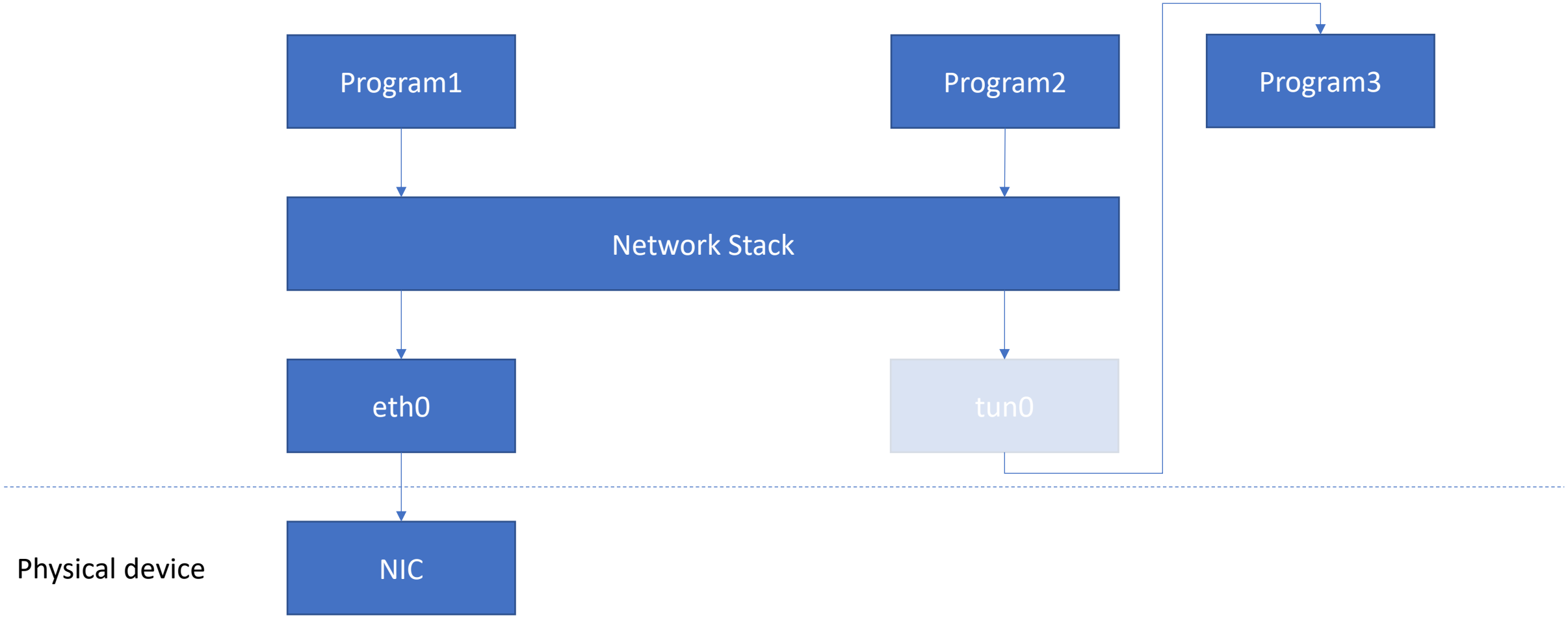
Shadowsocks
Android
badvpn(including tun2socks)

Kitsunebi & V2RayN
Android
go-tun2socks

Why reinvent the wheel

- For full flexibility.
  - e.g. Select the specific tap device.
- Better performance with C++ and boost.asio.
  - e.g. SSTAP consumes a large amount of memory when the network speed reaches 200Mbps
- To learn C++, cmake, boost, system programming.

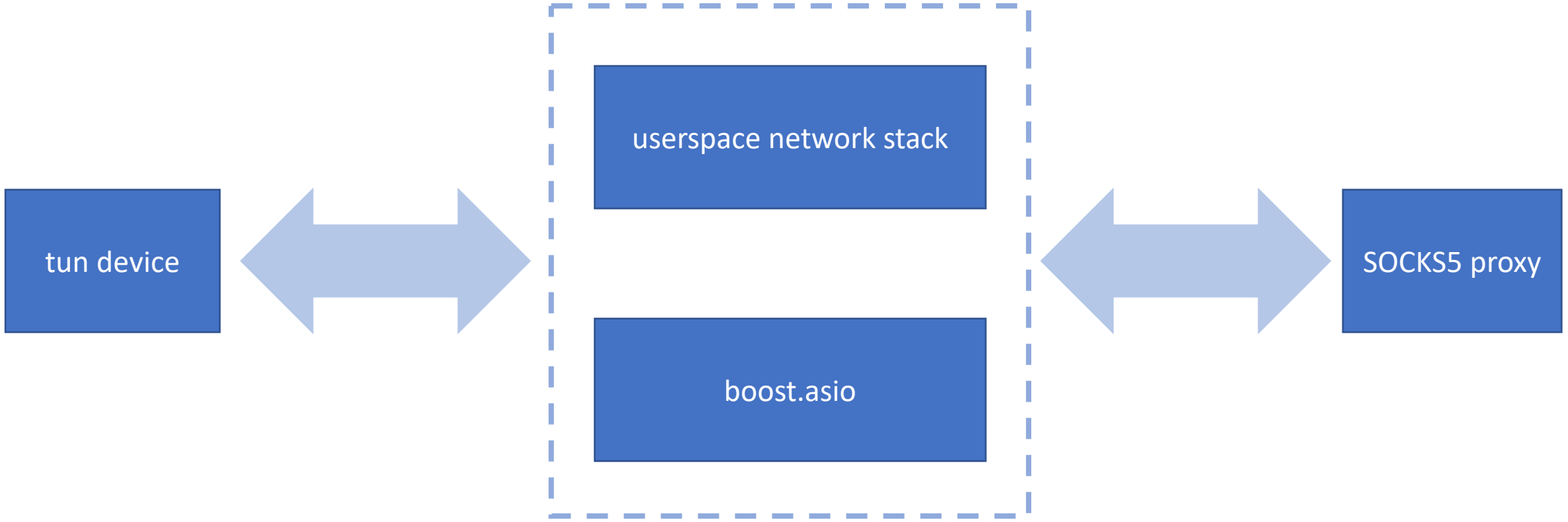
TUN/TAP device



SOCKS5

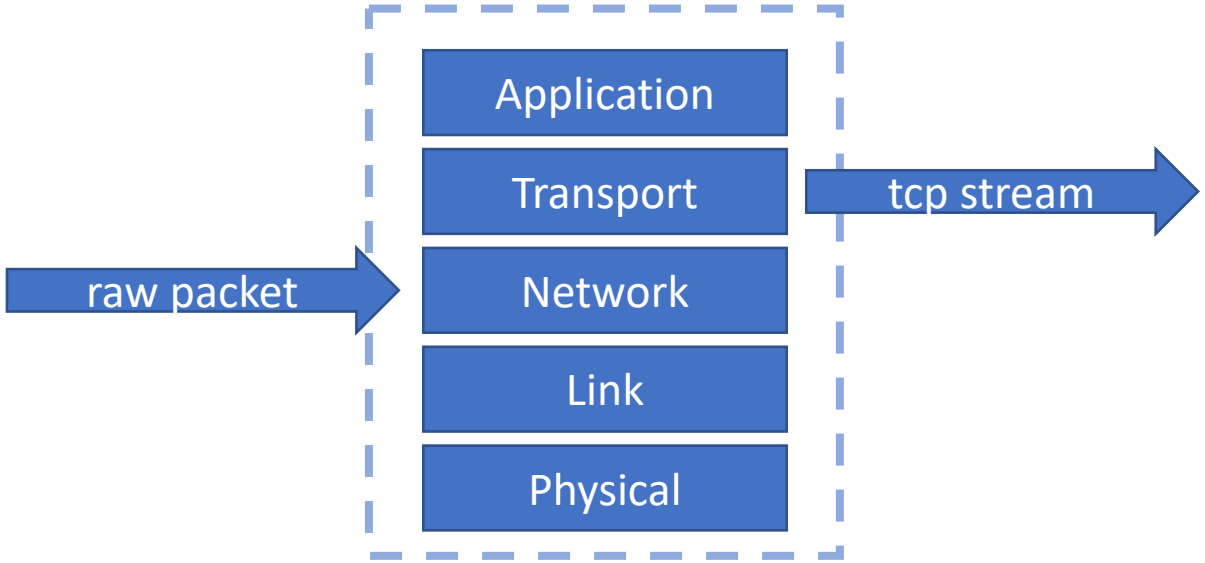
- A simple proxy protocol defined in RFC1928
- Support both TCP and UDP
- Easy to implement

Framework


tun device
userspace network stack
boost.asio
SOCKS5 proxy

Difficulty 1: Reassemble

- TUN device works at network layer but socks5 works at transport layer.
- lwIP is a robust implementation of network stack in pure C.
- Many modifications are made to accomplish the goal.



Difficulty 2: UDP

- UDP is stateless so we can’t track it like TCP.
- Modify the lwIP to create a UDP state machine like TCP.
- A global timeout is set for each UDP connection.
  - Something like how NAT is implemented

Difficulty 3: Cross-platform

- Work is much easier thanks to cmake and boost.
- TUN/TAP device is platform-dependent.
  - takes lots of time to design and implement the API
- Only support Linux and Windows for now.

Demo

Q & A

Thanks

Source: https://github.com/wtdcode/tun2socks

If you find it interesting or useful, don’t hesitate to give it a star :).