Multiport: Layer 4 Network Multiplexer and Demultiplexer

Overview

The **multiport** software consists of two pieces of software, a **Muxer** which accepts socket connections over a number of bound ports to be "multiplexed" over one TCP port, and a **Demuxer** which accepts connections over one TCP port and "de-multiplexes" the socket connections to target any number of target ports to a single target endpoint.

The multiport software allows multiple TCP and UDP ports to be transported over one TCP port making the Multiport project ideal for mapping over a remote.it connection.

Quick Start

1. Download DE/MUX from TBD

2. Install DE/MUX on macOS TBD

3. Install DE/MUX on RPi TBD

4. Configuration files needed? TBD

5. Example console input and expected output TBD

Operational Overview

Figure 1 shows a system that accepts sockets on the Muxer side, transports the sockets and data over a single TCP port to Demuxer. Demuxer then targets a host with those socket connections and data.

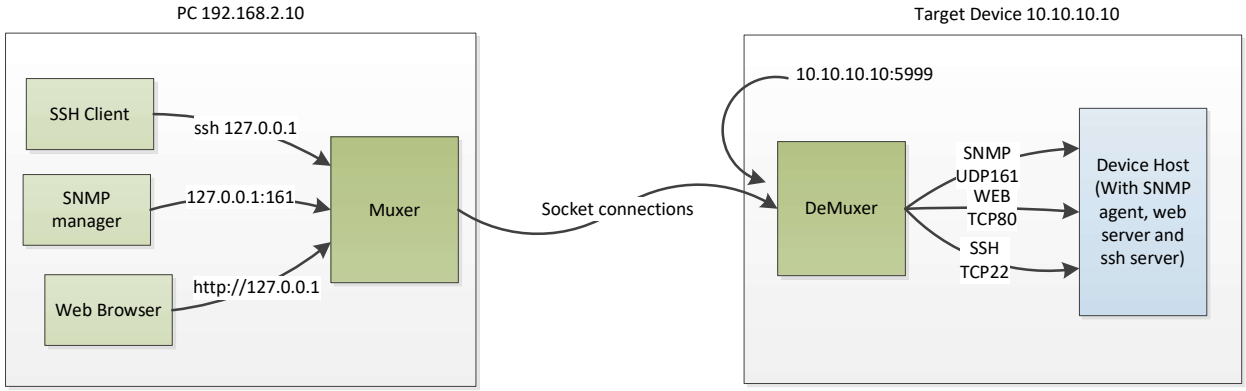


Figure 1) Muxer / Demuxer overview

The configuration of Muxer on the PC side defines which ports should be handled, in this case TCP ports 22, 80 and UDP port 161. With this configuration Muxer will be looking for traffic on these ports and tunnel the traffic to Demuxer target on TCP port 5999 at IP endpoint 10.10.10.10. Demuxer will take these TCP socket connections to TCP port 5999 and decode the proper port and target port on the

target device (in this case itself). Any reply from the target port will be transferred back through Demuxer to Muxer and back to the originating client/manager/web browser.

multiport Usage

Both pieces of the Multiport project can be driving by the command line, a configuration file or a combination of both. The Muxer and Demuxer can also be run as a background service on a Unix systems.

The following section will discuss the usage and setup of Muxer and Demuxer.

Demuxer Setup

Demuxer has two important configurations, one is the bind IP:port and the other is the target host. The bind IP:port is the endpoint that Demuxer listens to for incoming TCP socket connections that contain the socket connection requests to send to the target host. The target host is a IP address or hostname that will be targeted by the Demuxer software.

By default, if no bind IP:port is specified, it will default to 127.0.0.1:5999. If no target host is specified demuxer will target 127.0.0.1. These settings are ideal for when Demuxer is connected to by a remote.it connection that wishes to access the device that is running remote.it **connectd** and the Demuxer software.

The full command line for Demuxer is as follows:

demux [-h][-v(erbose)][-f config_file][-d][-p pidfile][-u user_2_run_as][-b bindip:port][-e <udp_expire_time>][-t <target_host>]

-h produces help output
-v produces verbose output, adding a second -v produces debug output also
-f use configuration file before parsing command line options
-d run as daemon (non windows only)
-p generate specified pidfile if daemon (non windows only)
-u user to run as as (non windows only, must start as root, drop privs to this user)
-b bind/listen on ip:port (default 127.0.0.1:5999)
-e udp expire time in seconds (default 120)
-t target_host (default 127.0.0.1)

Another option that Demuxer can accept is the specifying of a configuration file (-f). The configuration file can configure all the commands on the command line and also the port filter that can restrict access to having the connecting Muxer connect to certain ports. The configuration file format will be covered more below in its own section.

The UDP expire time option lets a user configure the UDP socket timeout to a value other than its default 120 seconds. This allows the user to configure UDP sockets that stay alive longer or shorter that have no traffic flowing on them.

The (-d) run as daemon, (-p) create a pidfile and (-u) run as user are used when running the demuxer software as a daemon. The (-p) and (-u) option are only valid if the (-d) option is specified. By default if the Demuxer software is run as root it will drop privileges after startup to user 'nobody' or the user specified by the (–u) parameter. The (-u) parameter is only relevant of the Demuxer software was started as root.

The (-v) option can be used multiple times to increase the verbosity of the Demuxer software console output and can be useful for debugging or configuration tuning.

Demuxer Configuration File

The demuxer configuration file is optional but can allow a user to configure the demuxer to not only setup the parameters available on the command line, but create TCP and UDP port restrictions of which resources that the connecting Muxer software is allowed to access.

The following is an example configuration file for the Demuxer software:

```
# verbose 0,1,2 2=maximux verbosity
verbose 2
#bind/listen to endpoint for muxer connections on:   (default to 127.0.0.1:5999)
bind_port 65535
bind_ip 0.0.0.0
#
# Target Host   (default 127.0.0.1) can be IP or hostname
#
target_host 127.0.0.1
#
# UDP port expire time in seconds (default 120) UDP mappings expire if no packets flow in this many seconds
#
udp_expire_time 240
#
# Filter TCP, exucted in order
#
filter_tcp 22 allow
filter_tcp 80 allow
filter_tcp 0-1024 deny
filter_tcp 2224 deny
#
# Filter UDP
#
filter_udp 5959 allow
filter_udp 5958 allow
filter_udp 5957 allow
filter_udp 5959 allow
filter_udp 0-65535 deny
```

The filter_tcp and filter_udp options are read and then processed in order and can be a port or a port range that is either allowed or denied.

By default all ports are allowed unless specifically set to deny in the configuration file.

Use Cases

~~Port Forwarding~~

~~*Overview*~~

~~In the port forwarding case, there may be times when you want to access multiple machines on multiple ports in the same method; this would not be possible with port forwarding since you can only port forward one port to one machine. If you wanted to access an IP camera on port 80 and 443 you could only port forward 80 and 443 to one camera, if you needed to setup another camera you would have to use ports other than 80 and 443, this may or may not work depending on the camera UI.~~

~~The muxer project would allow you to connect to devices with the same method using an arbitrary port for each device wished to share, and access them in the same method on the client side.~~

~~*Setup*~~

~~Each target device you wish to share on the internal network must have a Demuxer pointed at it.~~

remote.it

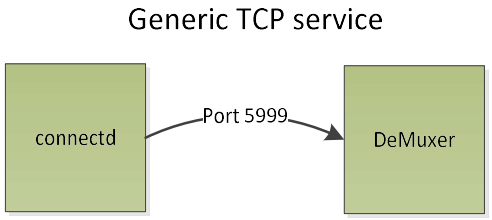
Remote.it is a great user case as one target service can allow access to any port on the target device and the desired ports need not be determined ahead of time.

In the typical remote.it installation the user configures each “service” it wishes to access on a device and each “service” is connected to individually. If a user wants to add a target port/service, the user must setup another “service” on the target remote device. In many cases this is optimal, but in other cases it can cause confusion and overhead.

Setup

To setup remote.it to use Muxer a target remote.it service must be pointed at the Demuxer software. Ideally the Demuxer software should be started up as a service at boot time and running on a known port (default 5999) and targeting the desired host if not localhost.

A generic TCP service can be used to target Demuxer, assuming Demuxer is running on TCP port 5999, the following setup can be used:



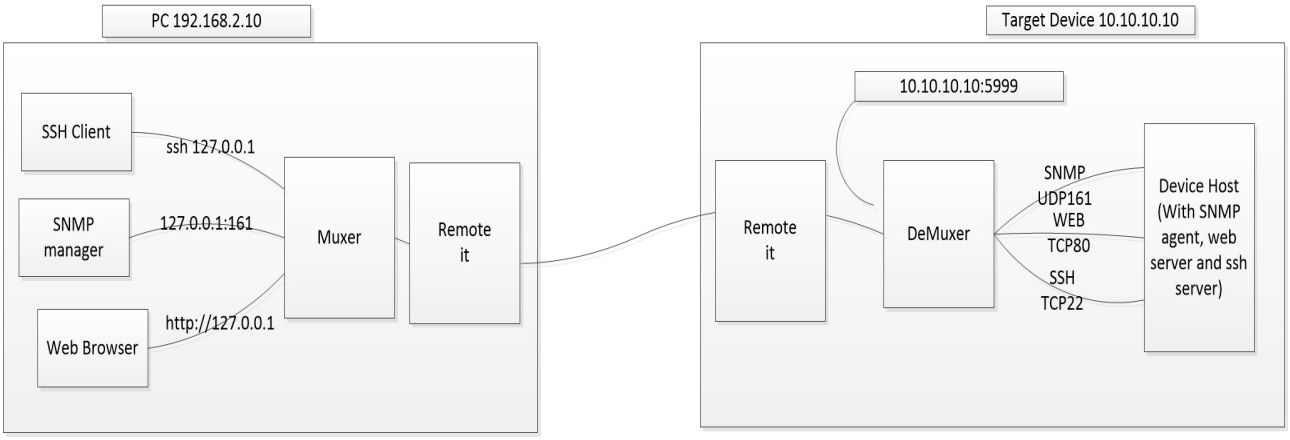
multiport Usage

network multiport protocol muxer / demuxer

This repository contains 2 projects that interoperate with each other, Muxer and Demuxer. One the Muxer can listen to a number of UDP and/or TCP sockets and tunnel them over TCP connections that target one TCP port on the Demuxer. The Demuxer will take these connections and create equivelent TCP and UDP connections to its specified target.

Typically muxer and demuxer will be used in conjunction with remote.it.

Below is an example of how it could be used.



Muxer

Muxer is the initiator/host side of the connection. Muxer accepts socket connections and send them to the target Demuxer.

Demxer

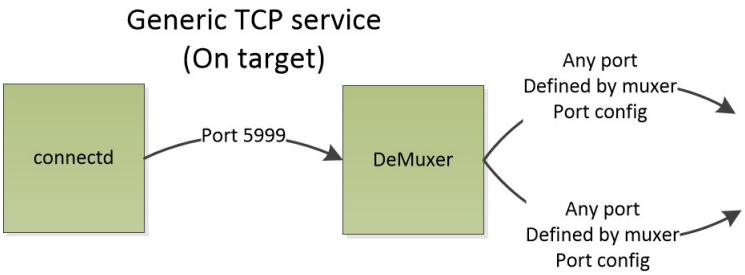
Demuxer is the target side of the connection. Demuxer accepts connections from Muxer and targets the appropriate TCP or UDP port on the Demuxer Target.

More information

For more information please see the document directory and the root directory of Muxer and Demuxer.

Demuxer Usage

Demxer is the target software that accepts connections from Muxer on a single TCP port and creates the requested TCP and UDP sockets to the specified target.



~~Build~~

~~Build can be done with the make file in the src directory for unix, or either windows dev studio 2010 or 2017. This project can also build with the buildall script in the root directory that builds all architectures in available in the tools directory.~~

Demux command format:

```
demux [-h][-v(erbose)][-f config_file][-d][-p pidfile][-u user_2_run_as][-b bindip:port][-e <udp_expire_time>][-t <target_host>]

-h produces help output
-v produces verbose output, adding a second -v produces debug output also
-f use configuration file before parsing command line options
-d run as daemon (non windows only)
-p generate specified pidfile if daemon (non windows only)
-u user to run as as (non windows only, must start as root, drop privs to this user)
-b bind/listen on ip:port (default 127.0.0.1:5999)
-e udp expire time in seconds (default 120)
-t target_host (default 127.0.0.1)
```

Listen IP:Port

The Demuxer will listen on an IP:Port for connections from the muxer (default 127.0.0.1:5999). Typically this will be through a remote.it connection. A remote.it service will target the Demuxer listen IP:Port.

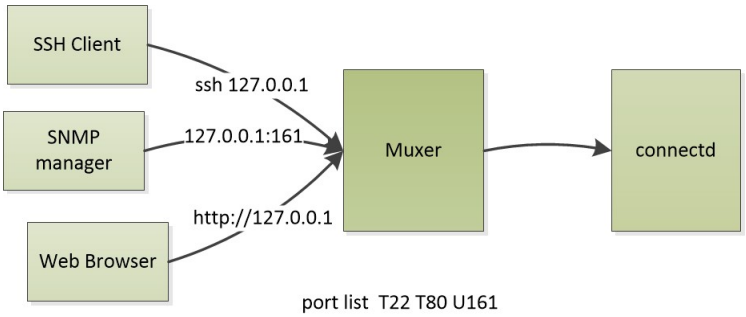
Target host

The Demux Target Host is the host that all connections will target. The target host by default is 127.0.0.1, the target host can be overridden with the -t option. Either an IP address or a resolvable hostname can be used for the target host.

Muxer will map ports from its side which will be in listen mode to ports on the target host which also should be in listen mode for TCP and bound for UDP.

Muxer Usage

Muxer is the initiator/host side of the connection. Muxer accepts socket connections and send them to the target Demuxer.



In the above example TCP ports 22 and 80, along with UDP port 161 are forwarded to the Demuxer target via remote.it

~~**Build**~~

~~Build can be done with the make file in the src directory for unix, or either windows dev studio 2010 or 2017. This project can also build with the buildall script in the root.~~

Muxer command format

```
muxer [-h] [-v(erbose)] [-d][optinal pid file] [-f config_file][-u user_2_run_as] [-t ] <bind/listen port list>
-h produces help output

-v produces verbose output, adding a second -v produces debug output also
-d runs the program as a daemon (unix only)
-p creates the specified pid file if run as daemon (unix only)
-u drop privlages to this user when damonized (unix only, defaults to nobody)
-f use configuration file before parsing command line options
-t demuxer target ip and port (defaults to 127.0.0.1:5999 if not specified)
```

Following the command options is the bind/listen list of sockets to accept traffic from and their target ports. Each entry is in the following format:

[protocol U or T][listen port][:optional target port]

Example TCP port 80 targeting port 8080 demuxer side

```
T80:8080
```

Listens on port 80, connects on port 8080 on target side.

Example TCP port 80 targeting port 80 demuxer side

```
T80
```

Listens on port 80, connects on port 80 on target side.

Example UDP port 5000 targeting port 5000 demuxer side

```
U5000
```

Example UDP port 5000 targeting port 3333 demuxer side

```
T5000:3333
```

Example sending TCP port 80 and 443 and local UDP port 3845 to 3800 demuxer side

```
T80 T443 U3845:3800
```

Full Example:

```
muxer -v T80 T443 U3845:3800
```

Full Example with non-default muxer target:

```
muxer -v -t 127.0.0.0:5998 T80 T443 U3845:3800
```

This targets port 5998 instead of the default 5999 on localhost.

Muxer configuration file

The muxer configuration file is parsed before any command line arguments. Command line arguments will override configuration file options.

Sample Muxer Configuration File

```
# this is a comment, any line that starts with '#' is a comment and will be ignored.
#
#target demuxer address
target_host 127.0.0.2
target_port 5998

#verbose level 0,1,2
verbose 2

#if daemon, user to run as
#run_as_user nobody
run_as_user mike

# sockets to bind
socket_list T2223:22 T8080:80 T2229

#IP to bind to
bind_ip 127.0.0.1
```