



基于OpenWrt的网络路由器流量监测——交叉编译与路由器配置

▼ type	Post
▼ status	Published
📅 date	@2025/06/23
☰ tags	Network 嵌入式 开发
▼ category	网络开发



根据已有文件进行环境配置与交叉编译，最终实现将监测程序挂载到实际路由器H3C Magic NX30 Pro上运行。

前置工作

安装编译所需工具和依赖

在ubuntu终端上输入以下命令，完成更新和安装。

```
#更新软件包索引
sudo apt update
#下载编译工具链
sudo apt install build-essential clang flex bison g++ gawk \
gcc-multilib g++-multilib gettext git libncurses5-dev libssl-dev \
python3-setuptools rsync swig unzip zlib1g-dev file wget
```

build-essential	编译工具包的元宝，包含make、gcc
clang	c/c++等语言的编译器前端
flex	生成词法分析器
bison	生成语法分析器
g++	GNU的c++编译器
gawk	文本处理工具，可以实现提取特征字段、计算数据等功能
gcc/++ -multilib	允许用户在64位系统上编译32位的程序
gettext	实现消息文本不同语言的翻译
libncurses5 -dev	提供图形化界面的开发库
libssl -dev	可以支持OpenSSL的加密功能，如 SSL/TLS 协议、加密算法
python3 - setuptools	构建和安装Python包，提供自动生成setup脚本、管理依赖关系等
rsync	用于在本地或远程主机之间同步文件和目录
swig	提供将现有的 C/C + + 库封装成可以被其他语言调用的包
unzip	解压工具
zlib1g - dev	压缩库的开发包





这部分完成后，后续编译会使用到。

Linux环境下加速访问github

由于后续SDK是在github仓库获取更新，直接访问的连接不太稳定，且下载速度较慢，故利用Clash配置VPN，节省下载时间。

下载安装Clash

clash下载地址：[下载地址](#)，ubuntu22.04以及以上版本选择如下：

 clashpremium-nightly-windows-arm64.exe	17.7 MB	Nov 15, 2024
 clashpremium-release-linux-amd64-v3.tar.gz	7.12 MB	Nov 15, 2024
 clashpremium-release-linux-amd64.tar.gz	7.13 MB	Nov 15, 2024
 clashpremium-release-linux-armv5.tar.gz	6.7 MB	Nov 15, 2024

解压安装

```
# 解压文件
tar -zxvf clashpremium-release-linux-armv8.tar.gz

# 给予权限
chmod +x CrashCore

# 执行 mkdir ~/clash; cd ~/clash 在用户目录下创建 clash 文件夹。
mkdir ~/clash
cd ~/clash

# 改名Clash并移动
mv CrashCore clash

# 查看版本
./clash -v

# 输出： Clash 2023.08.17 linux amd64 with go1.21.0 Thu Aug 17 15:07:25 U
TC 2023
```

编写配置文件

配置文件需要修改为 `config.yaml` 后放到 `~/config/clash/config.yaml`，根据自己的订阅网址找到配置文件，复制粘贴即可。

终端代理

默认终端不会启用代理，需要手动配置：

```
echo -e "export http_proxy=http://127.0.0.1:7890\nexport https_proxy=htt
p://127.0.0.1:7890" >> ~/.bashrc
```

或者临时配置代理：仅当前运行窗口有效：

```
export http_proxy=http://127.0.0.1:7890/  
export https_proxy=http://127.0.0.1:7890/  
export all_proxy="socks5://127.0.0.1:7891"
```

网络代理

ubuntu打开设置→网络，更改“网络代理”为“手动”，填写代理配置信息：

```
# HTTP 代理  
127.0.0.1:7890  
  
# HTTPS 代理  
127.0.0.1:7890  
  
# Socks 主机  
127.0.0.1:7891
```

代理管理

浏览器进入 <http://clash.razord.top/#/proxies> 页面可以进行代理管理，图形化界面中可以切换和详细配置，查看日志等。

Openwrt虚拟机运行程序编译

固件编译

除了直接使用官方提供的镜像外，还可以通过源码自行编译镜像，在终端输入配置以下命令：

```
#注意：这里默认前面“前置工作”的工具链全部下载完成  
  
#拉取OpenWrt源码  
git clone https://git.openwrt.org/openwrt/openwrt.git  
  
# 进入目录并切换至所需的版本  
cd openwrt  
git pull
```

```
git checkout v24.10.0
```

```
# 更新下载相关依赖
```

```
./scripts/feeds update -a
```

```
./scripts/feeds install -a
```

```
# 进行相关编译设置
```

```
make menuconfig
```

- 如果下载速度缓慢，打开网络代理
- `make menuconfig` 命令执行后进入一个图形化窗口，主要关注 Target System , Subtarget 和 Target Profile 这三项：

```
Target System (x86) --->
Subtarget (x86_64) --->
Target Profile (Generic x86/64) --->
Target Images --->
[ ] Enable experimental features by default
Global build settings --->
[ ] Advanced configuration options (for developers) ----
[ ] Build the OpenWrt Image Builder
[ ] Build the OpenWrt SDK
[ ] Package the OpenWrt-based Toolchain
[ ] Image configuration --->
Base system --->
Administration --->
Boot Loaders --->
Development --->
Extra packages ----
Firmware --->
Kernel modules --->
Languages --->
Libraries --->
LuCI --->
Network --->
Utilities --->
```

比如在我的openwrt虚拟机上，选择为x86_64架构，Generic x86_64。

保存配置后退出，开始编译镜像，

- 如果是ubuntu执行 `make` ;
- WSL则需要声明环境变量 `PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin make`
- 待编译完成后在 `./bin/target` 下可以找到对应镜像，文件可以直接写入到路由器的存储设备中（如闪存）或者VM中来安装和运行 OpenWrt 系统



一般来说不会直接编译整个openwrt的包，因为对配置和时间要求比较高，大部分情况下建议直接下载官方镜像，安装openwrt系统。

这里仅作演示使用，在我实际开发过程中还是使用的编译后的镜像。

SDK编译源程序

- OpenWrt的SDK (Software Development Kit, 软件开发工具包) 是专门为开发和编译OpenWrt额外软件包而准备的一套集成工具和环境。(更侧重于用户的应用开发)
- SDK包含交叉编译工具链、头文件、Makefile 框架等，用户可以在本地直接编译 OpenWrt 支持的软件包。
- SDK版本和 OpenWrt 固件版本是配套的，编译出来的包能确保兼容现在的系统，避免因 ABI 不一致导致的各种“无法安装”问题。

对于实际Openwrt虚拟机的SDK，我的为x86_64架构：[SDK下载地址](#)

编译打包

下载后解压：

```
tar -I zstd -xvf openwrt-sdk-24.10.1-x86-64_gcc-13.3.0_musl.Linux-x86_64.tar.zst
```

解压后将项目源代码的文件包放入SDK的package目录, 得到以下目录结构：

```
package/
├── lab2/
│   ├── src/
│   ├── openwrt.c
│   └── Makefile
```

其中Makefile文件规定了编译规则，内容如下：

```
#引入OpenWrt的全局构建规则和变量
include $(TOPDIR)/rules.mk
```

```

# 定义软件包元数据
PKG_NAME:=lab2
PKG_VERSION:=1.0
PKG_RELEASE:=1

#引入 OpenWrt 的软件包构建框架，包括一系列函数和变量（SDK根目录下）
include $(INCLUDE_DIR)/package.mk

define Package/lab2
    SECTION:=utils
    CATEGORY:=Utilities
    TITLE:=OpenWrt Traffic Monitor
    DEPENDS:=+libpcap#依赖项libpcap库
endef

define Package/lab2/description
    Traffic monitoring program for OpenWrt
endef

# 编译前准备，将源码复制到编译专用的路径下
define Build/Prepare
    mkdir -p $(PKG_BUILD_DIR)
    $(CP) ./src/* $(PKG_BUILD_DIR)/
endef

#Compile阶段，指定如何编译源码
define Build/Compile
    $(TARGET_CC) $(TARGET_CFLAGS) -o $(PKG_BUILD_DIR)/traffic_monitor \
        ./src/openwrt.c -lpcap#链接上libpcap
endef

#安装打包的规则
define Package/lab2/install
    $(INSTALL_DIR) $(1)/usr/sbin
    $(INSTALL_BIN) $(PKG_BUILD_DIR)/traffic_monitor $(1)/usr/sbin/

```

```
endif
```

```
$(eval $(call BuildPackage,lab2))
```



注意Makefile文件一定要用TAB，而不是两个空格，否则编译时会报错；同时中文注释的编码格式也有可能引起报错。

下来回到目录 `openwrt-sdk-24.10.0-mediatek-filogic_gcc-13.3.0_musl.Linux-x86_64/` 下，执行命令：

```
#准备依赖项,非常重要，缺少时可能导致工具链错误！
```

```
./script/feeds update -a
```

```
./script/feeds install -a
```

```
# 编译打包
```

```
make package/helloworld/compile V=s
```

执行make后会弹出一个类似 `make menuconfig` 的图形化配置窗口，注意global settings,advanced configuration options,libraries,base system配置。

```
Global build settings
(or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y>
[ ] excluded <M> module < > module capable

[*] Select all target specific packages by default
[ ] Select all kernel module packages by default
[ ] Select all userspace packages by default
[*] Cryptographically sign package lists
*** General build options ***
[ ] Compile with support for patented functionality
[ ] Compile with full language support
*** Package build options ***
[ ] Compile packages with debugging info
*** Stripping options ***
Binary stripping method (ssstrip) --->
```


Advanced configuration options (for developers)
(or empty submenus ---). Highlighted letters are hotkeys. Pressing
[] excluded <M> module < > module capable

- [] Show broken packages
- () Download folder
- () Local mirror for source packages
- [*] Automatic rebuild of packages
- [*] Automatic removal of build directories
- [] Use ccache
- [] Enable log files during build process
- [] Enable package source tree override

Base system
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ---). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [] excluded <M> module < > module capable

```
[*] base-files..... Base filesystem for OpenWrt
(M) block-mount..... Block device mounting and checking
<M> blockd..... Block device automounting
< > bridge..... Bridge forwarding accelerator (NEW)
+* busybox..... Core utilities for embedded Linux
[ ] Customize busybox options
< > busybox-selinux... Core utilities for embedded Linux with SELinux support (NEW)
[*] ca-bundle..... System CA certificates as a bundle
< > ca-certificates..... System CA certificates (NEW)
[*] dnsmasq..... DNS and DHCP server
< > dnsmasq-dhcpv6..... DNS and DHCP server (with DHCPv6 support) (NEW)
< > dnsmasq-full (NEW)
[*] dropbear..... Ssh client/server
Configuration --->
< > e2fsprogs..... Emergency Access Daemon (NEW)
< > firewall..... OpenWrt C Firewall (NEW)
[*] firewall4..... OpenWrt 4th gen firewall
+* fstools..... OpenWrt filesystem tools --->
+* fwtool.....
+* getrandom..... OpenWrt getrandom system helper
+* jsonfilter..... OpenWrt JSON filter utility
<M> libatomic..... Atomic support library
+* libc..... C library
+* libgcc..... GCC support library
<M> libgomp..... OpenMP support library
+* libpthread..... POSIX thread library
[*] librt..... POSIX.1b Realtime extension library
<M> libstdc++..... GNU Standard C++ Library v3
[*] logd..... OpenWrt system log implementation
[*] mtd..... Update utility for trx firmware images
+* netifd..... OpenWrt Network Interface Configuration Daemon
+* openwrt-keyring..... OpenWrt Developer Keyring
[*] opkg..... opkg package manager ---
+* procd..... OpenWrt system process manager
Configuration --->
+* procd-seccomp..... OpenWrt process seccomp helper + utrace
< > procd-selinux..... OpenWrt system process manager with SELinux support (NEW)
[*] procd-ujail..... OpenWrt process jail helper
v(+)
```

<select> < Exit > < Help > < Save > < Load >

Libraries
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ---). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [] excluded <M> module < > module capable

```
<(-)>
< > libfdt..... a utility library for reading and manipulating dtb files (NEW)
< > libgmp..... GNU multiprecision arithmetic library (NEW)
< > libiconv-full..... Character set conversion library (NEW)
< > libintl-full..... GNU Internationalization library (NEW)
< > libiw..... Library for manipulating Linux Wireless Extensions (NEW)
+* libiwinfo..... Generalized Wireless Information Library (iwinfo)
+* libjson-c..... javascript object notation
< > libltdl..... A generic dynamic object loading library (NEW)
< > liblua..... Lua programming language (libraries) (NEW)
< > liblua5.3..... Lua programming language (version 5.3) (libraries) (NEW)
+* liblucihttp..... LuCI HTTP utility library
< > liblucihttp-lua..... Lua binding for the LuCI HTTP utility library (NEW)
+* liblucihttp-ucode..... ucode binding for the LuCI HTTP utility library
< > libmd..... Message Digest functions from BSD systems (NEW)
+* libmnl..... Minimalistic user-space library for Netlink
< > libmount..... mount library (NEW)
< > libmpfr..... GNU MPFR library (NEW)
(M) libncurses..... Terminal handling library (Unicode)
< > libnetfilter-conntrack (NEW)
< > libnettle..... GNU crypto library (NEW)
< > libnftnl..... Low-level netlink library for the nf_tables subsystem
< > libnl..... Full Netlink Library (NEW)
< > libnl-cli..... CLI Netlink Library (NEW)
< > libnl-core..... Core Netlink Library (NEW)
< > libnl-genl..... Generic Netlink Library (NEW)
< > libnl-nf..... Netfilter Netlink Library (NEW)
< > libnl-route..... Routing Netlink Library (NEW)
+* libnl-tiny..... netlink socket library
< > libopcodes..... libopcodes (NEW)
[*] libpcap..... Low-level packet capture library --->
< > libpcap2..... A Perl Compatible Regular Expression library (NEW)
< > libpcap2-16..... A Perl Compatible Regular Expression library (16bit support) (NEW)
< > libpcap2-32..... A Perl Compatible Regular Expression library (32bit support) (NEW)
< > libpopt..... A command line option parsing library (NEW)
< > libpri..... libpri Primary Rate ISDN implementation (NEW)
< > libreadline..... Command lines edition library (NEW)
< > libselinux..... Runtime SELinux library (NEW)
v(+)
```

<select> < Exit > < Help > < Save > < Load >

截屏工具
屏幕截图已复制到剪贴板
已自动保存到屏幕截图文件夹。

编译打包时注意最好使用 `-j=1` 单线程，避免多个线程之间相互依赖报错，造成报错信息混乱。

成功编译后，在 `/bin/package/x86_64/base` 里面找到相应 `.ipk` 文件：

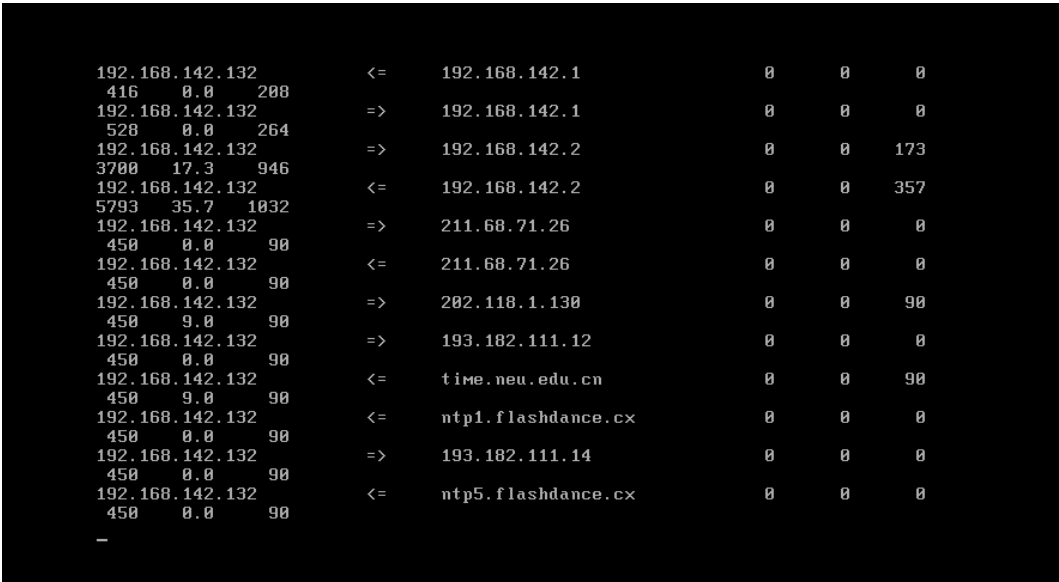


上传运行

通过共享文件夹方式，上传 `lab2` 和 `libpcap` 程序包到 Openwrt 虚拟机，然后安装：

```
#注意先安装libpcap
opkg install libpcap_xxx.ipk
opkg install lab2_xxx.ipk

#运行程序
lab2
```



选择对应端口后可以看到，程序终端正确执行了流量监测的任务。

路由器H3C Magic NX30 Pro中运行程序编译

固件下载

在官网<https://firmware-selector.openwrt.org/>下搜索设备对应的固件



下载得到如下文件，即图中1，2，4项目：

- 1 openwrt-24.10.0-mediatek-fillogic-h3c_magic-nx30-pro-initramfs-recovery.itb
- 2 openwrt-24.10.1-mediatek-fillogic-h3c_magic-nx30-pro-bl31-uboot.fip
- 4 openwrt-24.10.1-mediatek-fillogic-h3c_magic-nx30-pro-squashfs-sysupgrade.itb

将文件 openwrt-24.10.0-mediatek-fillogic-h3c_magic-nx30-pro-initramfs-recovery.itb 重命名为 openwrt-mediatek-fillogic-h3c_magic-nx30-pro-initramfs-recovery.itb

连接路由器

该款路由器（H3C Magic NX30 Pro）默认开启了telnet服务。
首先进入路由器管理网页设置登录密码。随后通过该密码与用户名 H3C 使用telnet服务进入路由器后台（telnet端口为99）

开启SSH服务

按照以下步骤开启 SSH：

```
curl -o /tmp/dropbear.ipk https://downloads.openwrt.org/releases/package
s-
19.07/aarch64_cortex-a53/base/dropbear_2019.78-2_aarch64_cortex-a53.i
pk
opkg install /tmp/dropbear.ipk
/etc/init.d/dropbear enable
/etc/init.d/dropbear start
```

固件备份

查看分区信息：

```
root@OpenWrt:/dev# cat /proc/mtd
dev: size erasesize name
mtd0: 00100000 00020000 "BL2"
mtd1: 00080000 00020000 "u-boot-env"
mtd2: 00200000 00020000 "Factory"
mtd3: 00200000 00020000 "FIP"
mtd4: 04000000 00020000 "ubi"
mtd5: 00600000 00020000 "pdt_data"
mtd6: 00600000 00020000 "pdt_data_1"
mtd7: 00100000 00020000 "exp"
mtd8: 02580000 00020000 "plugin"
```

重点备份ubi、factory、pdt_data、plugin这几个分区

```
dd if=/dev/mtd5 of=/tmp/backup.img
```

备份后使用scp命令传出并删除文件，避免存储空间不足。

OpenWrt 刷入

将前文下载的文件 openwrt-24.10.1-mediatek-filogic-h3c_magic-nx30-pro-bl31-uboot.fip 传入路由器中，使用 winscp 工具。

写入uboot：

```
mtd write ./openwrt-24.10.1-mediatek-filogic-h3c_magic-nx30-pro-bl31-ub
oot.fip
```

FIP

随后需要将其刷入 kernel。先将路由器断电，按住背后 Reset 按钮不放，再插电，等待 10s 左右进入uboot。

接着使用 PC 机用网线连接到路由器的 LAN 口。在网络设置中将本机的IP设置为 192.168.1.X，网关设置为 192.168.1.1。

交叉编译配置

下载SDK并构建项目目录

在[下载地址](#)中下载该路由器版本的cortex_a53架构SDK，解压后进入目录，传入之前写好的lab2源程序包，依然得到如下结构：

```
package/  
└─ lab2/  
    ├── src/  
    ├── openwrt.c  
    └─ Makefile
```

交叉编译软件包

先更新并安装依赖项：

```
./script/feeds update -a  
./script/feeds install -a
```

编写的Makefile文件如下：

```
include $(TOPDIR)/rules.mk  
  
PKG_NAME:=lab2  
PKG_VERSION:=1.0  
PKG_RELEASE:=$(shell date +%Y%m%d)  
  
include $(INCLUDE_DIR)/package.mk  
  
define Package/lab2
```

```

SECTION:=utils
CATEGORY:=Utilities
TITLE:=OpenWrt Traffic Monitor
DEPENDS:=+libpcap#依旧加上依赖
endif

define Package/lab2/description
  Advanced traffic monitoring for MediaTek Filologic routers
endif

define Build/Prepare
  mkdir -p $(PKG_BUILD_DIR)
  $(CP) ./src/* $(PKG_BUILD_DIR)/
endif

#定义编译配置
define Build/Compile
  $(TARGET_CC) $(TARGET_CFLAGS) -O2 \
    -o $(PKG_BUILD_DIR)/traffic_monitor \
    $(PKG_BUILD_DIR)/openwrt.c \
    $(TARGET_LDFLAGS) -lpcap
endif

#定义安装包构建规则
define Package/lab2/install
  $(INSTALL_DIR) $(1)/usr/sbin
  $(INSTALL_BIN) $(PKG_BUILD_DIR)/traffic_monitor $(1)/usr/sbin/
  $(INSTALL_DIR) $(1)/etc/traffic_monitor
endif

$(eval $(call BuildPackage,lab2))

```

在/openwrt-sdk-24.10.0-mediatek-filologic_gcc-13.3.0_musl.Linux-x86_64/feeds/base/package/boot/uboot-mediatek文件目录下找到自带的Makefile，修改第838行的目标设备，指定为本型号路由器：

```

BOOT_TARGETS := \
mt7981_h3c_magic-nx30-pro

```



这是Openwrt的软件包BUG，如果不修改指定对应设备，编译时会一直尝试查找设备，进入如下死循环：

```
make[4]: Entering directory '/home/kiwit/LAB2/openwrt-sdk-24.10.0-mediatek-filologic_gcc-13.3.0_musl.Linux-x86_64/feeds/base/package/boot/uboot-mediatek'
Checking 'python3-dev'... ok.
Checking 'python3-setuptools'... ok.
Checking 'swig'... ok.
make[4]: Leaving directory '/home/kiwit/LAB2/openwrt-sdk-24.10.0-mediatek-filologic_gcc-13.3.0_musl.Linux-x86_64/feeds/base/package/boot/uboot-mediatek'
```

然后执行交叉编译：

#进入SDK根目录

```
cd ~/LAB2/openwrt-sdk-24.10.0-mediatek-filologic_gcc-13.3.0_musl.Linux-x86_64
```

设置工具链路径

```
export STAGING_DIR=$PWD/staging_dir
```

```
export PATH=$PATH:$STAGING_DIR/toolchain-aarch64_cortex-a53_gcc-13.3.0_musl/bin
```

验证编译器可用

```
aarch64-openwrt-linux-musl-gcc --version
```

#这时输出：aarch64-openwrt-linux-musl-gcc (OpenWrt GCC 13.3.0 r28427-6df0e3d02a) 13.3.0 Copyright (C) 2023 Free Software Foundation, Inc. This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

#安装libpcap

```
sudo apt-get install libpcap-dev
```

#进入源代码目录

```
cd package/lab2/src
```

然后使用完整路径编译

```
aarch64-openwrt-linux-musl-gcc -O2 -I$STAGING_DIR/target-aarch64_cortex-a53_musl/usr/include \
```

```
-o traffic_monitor openwrt.c \
```

```
-L$STAGING_DIR/target-aarch64_cortex-a53_musl/usr/lib -lpcap
```

编译后查看对应文件属性，确认是我们想要的架构的文件格式：

```
slu@kali:~/Virtual-Platform:~/LAB2/openwrt-sdk-24.10.0-mediatek-filogic-gcc-13.3.0_musl-linux-x86_64/package/lab2/src$ aarch64-openwrt-linux-musl-gcc -O2 -I$STAGING_DIR/target-aarch64-cortex-a53_musl/usr/include \
-o traffic_monitor openwrt.c \
-L$STAGING_DIR/target-aarch64-cortex-a53_musl/lib -lpcap
slu@kali:~/Virtual-Platform:~/LAB2/openwrt-sdk-24.10.0-mediatek-filogic-gcc-13.3.0_musl-linux-x86_64/package/lab2/src$ file traffic_monitor
traffic_monitor: ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-musl-aarch64.so.1, with debug_info, not stripped
```

内存优化与传输运行

通过ubuntu和windows的共享文件夹和以及Openwrt和windows间的scp，上传对应编译后的软件包到/tmp（临时）目录：

```
PS C:\Users\administrator Cheng\Desktop> scp "C:\share_2\traffic_monitor"root@192.168.1.1:"/tmp/"
root@192.168.1.1's password:
traffic_monitor 100% 73KB 7.1MB/s 00:00
```

尝试安装libpcap时发现操作系统内存不足：

```
root@OpenWrt:~# opkg update
Downloading https://downloads.openwrt.org/releases/24.10.1/targets/mediatek/filogic/packages/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_core
Downloading https://downloads.openwrt.org/releases/24.10.1/targets/mediatek/filogic/packages/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/releases/24.10.1/packages/aarch64-cortex-a53/base/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_base
Downloading https://downloads.openwrt.org/releases/24.10.1/packages/aarch64-cortex-a53/base/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/releases/24.10.1/targets/mediatek/filogic/kmods/6.6.86-1-6ace983a14b769f576fe9c4c7961bd89/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_kmods
Downloading https://downloads.openwrt.org/releases/24.10.1/targets/mediatek/filogic/kmods/6.6.86-1-6ace983a14b769f576fe9c4c7961bd89/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/releases/24.10.1/packages/aarch64-cortex-a53/luci/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_luci
Downloading https://downloads.openwrt.org/releases/24.10.1/packages/aarch64-cortex-a53/luci/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/releases/24.10.1/packages/aarch64-cortex-a53/packages/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_packages
Downloading https://downloads.openwrt.org/releases/24.10.1/packages/aarch64-cortex-a53/packages/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/releases/24.10.1/packages/aarch64-cortex-a53/routing/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_routing
Downloading https://downloads.openwrt.org/releases/24.10.1/packages/aarch64-cortex-a53/routing/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/releases/24.10.1/packages/aarch64-cortex-a53/telephony/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_telephony
Downloading https://downloads.openwrt.org/releases/24.10.1/packages/aarch64-cortex-a53/telephony/Packages.sig
Signature check passed.
root@OpenWrt:~# opkg install libpcap
Installing libpcap1 (1.10.5-r2) to root...
Collected errors:
 * verify_pkg_installable: Only have 0kb available on filesystem /overlay, pkg libpcap1 needs 330
 * opkg_install_cmd: Cannot install package libpcap.
root@OpenWrt:~# cd /tmp
root@OpenWrt:/tmp# cd ~
root@OpenWrt:~# df -h
Filesystem      Size      Used Available Use% Mounted on
/dev/root        4.5M        4.5M         0 100% /rom
tmpfs           117.4M        4.4M    113.0M   4% /tmp
/dev/ubi0_8     11.8M       11.7M         0 100% /overlay
overlayfs:/overlay 11.8M       11.7M         0 100% /
tmpfs           512.0K         0     512.0K   0% /dev
root@OpenWrt:~#
```

通过查看并删除不必要的大型文件，实现内存优化：

```
#查找大于100K的文件
find /overlay -type f -size +100k
/overlay/upper/usr/lib/libssl.so.3
/overlay/upper/usr/lib/sftp-server
/overlay/upper/usr/lib/sshd-session
/overlay/upper/usr/lib/libevdev.so.2.3.0
/overlay/upper/usr/lib/libfido2.so.1.14.0
```



```
/overlay/upper/usr/lib/libcrypto.so.3
/overlay/upper/usr/lib/libz.so.1.3.1
/overlay/upper/usr/libexec/ssh-keygen-openssh
/overlay/upper/usr/sbin/sshd
/overlay/upper/root/lab2
/overlay/upper/root/.vscode-server/vscode-cli-18e3a1ec544e6907be1e944
a94c496e302073435.tar.gz
/overlay/upper/root/libpcap.so.1
/overlay/upper/root/libpcap.a
/overlay/upper/root/libpcap.so.1.10.4

#删除不必要文件
rm -f /overlay/upper/root/.vscode-server/vscode-cli-*.tar.gz
rm -f /overlay/upper/root/libpcap.*#重复的libpcap
rm -f /overlay/upper/usr/lib/sftp-server /overlay/upper/usr/lib/sshd-session
/overlay/upper/usr/libexec/ssh-keygen-openssh#未使用的SSH组件
```

然后重新安装，移动可执行程序到系统目录usr/sbin/中稳定运行：

```
#安装libpcap
opkg update
opkg install libpcap

# 将程序放在 /usr/sbin/
mv /tmp/traffic_monitor/usr/sbin/

#给予权限并运行
chmod +x /usr/sbin/traffic_monitor
/usr/sbin/traffic_monitor
```

Power shell上通过SSH登录Openwrt终端（实际上重启后仍可以），选择端口，最终实现在路由器上稳定流量监测：

IPaddress	direction	IPaddress	3s	5s	10s	total	10sAverage	Max				
192.168.31.149	=>	23.202.34.82	0	0	0	429	0.0	363				
192.168.31.149	=>	122.14.236.77	0	0	0	198	0.0	66				
fe80::722a:d7ff:fe88:cb4e	=>	ff02::16	0	0	0	170	0.0	170				
192.168.31.149	=>	49.7.47.42	0	0	0	264	0.0	66				
192.168.31.149	<=	122.14.236.77	0	0	0	186	0.0	62				
192.168.31.149	<=	49.7.47.42	0	62	62	250	6.2	64				
192.168.31.149	<=	183.47.102.193	0	0	0	716	0.0	582				
192.168.31.149	=>	183.47.102.193	0	0	0	281	0.0	227				
192.168.31.149	=>	8.8.4.4	0	0	0	66	0.0	66				
192.168.31.149	=>	103.121.210.210	0	0	0	66	0.0	66				
192.168.31.149	=>	101.101.101.101	0	0	0	66	0.0	66				
192.168.31.149	<=	public1.alidns.com	0	0	0	60	0.0	60				
192.168.31.149	=>	223.5.5.5	0	0	0	54	0.0	54				
192.168.31.149	<=	a23-202-34-82.deploy.static.akamaitechnologies.com	0	0	0	0	0	0	0	487	0.0	487
192.168.31.149	=>	202.89.233.100	0	0	0	54	0.0	54				
192.168.31.149	<=	202.89.233.100	0	0	0	120	0.0	120				
192.168.31.149	=>	192.168.31.1	0	0	0	1287	0.0	498				
192.168.31.149	<=	192.168.31.1	0	0	0	1018	0.0	1018				
192.168.31.149	=>	122.14.236.76	0	0	0	4135	0.0	4135				
192.168.31.149	<=	122.14.236.76	0	0	0	587	0.0	587				
fe80::722a:d7ff:fe88:cb4e	=>	fe80::5e02:14ff:fefa:df28	0	0	0	3852	0.0	3440				
192.168.31.149	<=	43.141.70.102	0	0	0	12781	0.0	11201				
192.168.31.149	=>	43.141.70.102	0	0	0	4705	0.0	4505				
192.168.31.149	<=	20.50.73.4	0	0	0	1171	0.0	1111				
fe80::722a:d7ff:fe88:cb4e	<=	fe80::5e02:14ff:fefa:df28	0	0	0	3853	0.0	3853				
192.168.31.149	=>	175.27.45.252	0	0	0	2338	0.0	2338				
192.168.31.149	=>	43.141.131.149	0	1588	1588	4400	158.8	2812				
192.168.31.149	<=	43.141.131.149	0	1394	1394	1514	139.4	1394				
192.168.31.149	<=	222.192.187.172	0	0	0	145	0.0	145				
192.168.31.149	=>	222.192.187.172	0	0	0	54	0.0	54				
192.168.31.149	=>	20.50.73.4	0	0	0	14393	0.0	14393				

未来的改进方向

开发可视化界面

利用前后端分离技术，实现可视化图表的流量监测统计，优化展示界面。

利用USB拓展系统内存

目前的开发基于单型号路由器，由于路由器和Openwrt存储空间有限，如果后续增加应用功能会导致编译和烧录时的限制；

可以考虑挂载USB存储，再创建overlay的符号链接，拓展overlay分区，然后使用外部存储运行程序；

这样可以最大限度地解决存储空间限制这一问题。



参考文章

- [Clash For Linux配置](#)



有关Notion安装或者使用上的问题，欢迎您在底部评论区留言，一起交流

~