МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по производственной практике НИР за 3-й семестр

Тема: Разработка BSP для оптического коммутатора на базе Intel Tofino

Студент гр. 8310 Слепов А. Э.

Руководитель Лисс А. А.

Санкт-Петербург

2023

ЗАДАНИЕ

НА НАУЧНО-ИССЛЕДОВАТЕЛЬСКУЮ РАБОТУ

Студент Слепов А. Э.

Группа 8310

Тема НИР: Разработка BSP для оптического коммутатора на базе Intel Tofino

Задание на НИР:

- Исследовать устройство аппаратной платформы, интерфейсы связи и протокол общения устройств
- Разработать базовые программные средства управления платформой

Сроки выполнения НИР: 01.09.2023 – 19.12.2023

Дата сдачи отчета: 20.12.2023

Дата защиты отчета: 26.12.2023

Студент гр. 8310 _____ Слепов А. Э.

Руководитель _____ Лисс А. А.

АННОТАЦИЯ

Задачей НИР является улучшение (в частности рефакторинг, добавление нового функционала и разработка системы сборки и доставки) ПО для оптического 16-портового коммутатора, разработанного в рамках весеннего семестра и разработка пакета поддержки платформы для 32-портового агрегатора балансировщика с функцией IPMI.

SUMMARY

The task of the research is to improve (in particular refactoring, adding new functionality and developing a build and delivery system) Software for an optical 16-port switch developed during the spring semester and the development of a platform support package for a 32-port load balancer aggregator with IPMI function.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

BSP – Board Support Package – пакет поддержки платформы — интегрированный пакет драйверов и/или модулей операционной системы, реализующий поддержку определенной аппаратной платформы [1].

BMC – Baseboard Managment Controller – это специализированный сервисный процессорный модуль в формате 2xM2 , который управляет питанием, отслеживает физическое состояние аппаратной платформы с помощью датчиков (термометры, АЦП) и взаимодействует с модулем управления.

Intel Tofino – это программируемая интегральная схема коммутатора данных, разработанная компанией Barefoot (Intel) для работы в высокоскоростных сетях (100-400 *GbE*).

CLI – Command Line Interface – интерфейс командной строки — это текстовый интерфейс, позволяющий пользователю взаимодействовать с операционной системой или приложением, используя команды в командной строке.

IPMI – Intelligent Platform Management Interface – интеллектуальный интерфейс управления платформой, предназначенный для автономного мониторинга и управления функциями, встроенными непосредственно в аппаратное и микропрограммное обеспечения серверных платформ.

Bypass-коммутатор - это программно-аппаратный комплекс, который обеспечивает порт отказоустойчивого доступа для встроенного в сеть активного устройства безопасности (например, системы предотвращения вторжений (IPS), межсетевой экран нового поколения (NGFW)).

Агрегатор-балансировщик – программно-аппаратный комплекс агрегации и балансировки трафика, предназначенный для фильтрации, агрегации, зеркалирования и маршрутизации высокоскоростного трафика в сетях передачи данных. Обеспечивает высокую пропускную способность, упрощает горизонтальное масштабирование и сетевую инфраструктуру.

ВВЕДЕНИЕ

Развитие современных технологий и возрастание потребностей в сетевых коммуникациях привело к необходимости создания более мощных и функциональных сетевых коммутаторов: коммутаторов-байпасов и агрегаторов балансировщиков трафика. BSP (Board Support Package) – это набор программных средств, необходимых для настройки и работы аппаратной части устройства. В данной работе мы рассмотрим основные принципы разработки BSP для сетевого коммутатора и ознакомимся с необходимыми инструментами и программным обеспечением.

1. ПОСТАНОВКА ЗАДАЧИ

Цель работы – провести рефакторинг, добавить новый функционал и организовать систему сборки и доставки ПО BSP коммутатора-байпаса на основе Intel Tofino, а также разработать ПО IPMI-контроллера агрегатора-балансировщика.

Объект исследования – программные средства управления аппаратной платформой сетевого коммутатора

Предмет исследования – коммутатор трафика на основе конвейера Intel Tofino и языка P4.

Задачи работы:

1. Проведение рефакторинга кода BSP для 16-портового коммутатора-байпаса, разделение функций по отдельным программным модулям.
2. Подготовка системы сборки и доставки ПО, сборка собственного образа Linux с BSP 16-портового коммутатора-байпаса
3. Уточнение требований и разработка BSP для 32-портового агрегатора-балансировщика с IPMI-контроллером.

Практическая ценность: результаты работы применены при серийном производстве BYPASS SP100G4M в г. Санкт-Петербург для организации пуско-наладки и стендов выходного контроля, в г. Долгопрудный ПО применяется для организации производства линейки из 15 типов модулей BYPASS SPM. Также ПО используется в сервисном в г. Москва.

2. РЕЗУЛЬТАТЫ РАБОТЫ В ОСЕННЕМ СЕМЕСТРЕ

2.1. Заявленный план работ на осенний семестр

1. Оформить набор разработанных скриптов в единый программный BSP на языке C++. Это так же потребует разработку классов для доступа к I2C-шине устройства, для взаимодействия с switchd, взаимодействия с BMC-платформы.
2. Подготовить инфраструктуру сборки, хранения и доставки ПО для платформы. В настоящее время подготовка загрузочного диска с ОС для пусконаладки девайсов занимает большое время и содержит большее количество ручных рутинных операций. Необходимо упростить и ускорить этот процесс, рассматривается вариант сборки минимального Linux с помощью Buildroot и его настройка таким образом, чтобы ОС сама получала актуальную версию ПО верхнего уровня, а также всего Firmware (встраиваемого программного обеспечения).

2.2. Описание рефакторинга ПО и новых функций

После разработки большого набора скриптов по управлению аппаратной платформой стала ясна необходимость рефакторинга и разделения ПО BSP на программные модули в соответствии с областью ответственности. Это было обусловлено:

- низким качеством кода, который разрабатывался в больших объемах в сжатые сроки с постоянно изменяющимися требованиями.
- необходимостью поддержки кода, а также расширением его функционала.

Таким образом было принято решение не переписывать код на C++, а сохранить кодовую базу на bash, оформив все программные модули в процедурном стиле и использовать механизм экспортирования функций в переменные среды.

Ключевыми новыми функциями BSP стали:

- поддержка конфигурации даты релиза firmware. Аппаратная платформа помимо ПО верхнего уровня требует для своей работы бинарные файлы: x86 bios, Tofino firmware, BMC STM firmware и STM32 Controlpass firmware. В ходе жизненного цикла продукта каждый бинарный файл получает новые релизы, которые необходимо оперативно обновлять. Это обеспечивается отдельных репозиторием бинарников, на которые ссылается BSP.

- линейка модулей Signalpass насчитывает 10 видов девайсов. Модель определяется поддерживаемой скоростью трансиверов, (10/40/100GBit/s), их количеством (2, 4 шт) и типом (SR, ER, LR, CM) (рис.1). В BSP была добавлена поддержка каждой конфигурации с учетом ее особенностей. Также был переработан интерфейс командной строки для управления модулями (рис. 2).

Артикул	Наменование
SPM100G-LR4SR4	Модуль BYPASS 100G LR4/SR4
SPM100G-SR4SR4	Модуль BYPASS 100G SR4/SR4
SPM100G-CM4SR4	Модуль BYPASS 100G CM4/SR4
SPM100G-ER4SR4	Модуль BYPASS 100G ER4/SR4
SPM40G-LR4SR4	Модуль BYPASS 40G LR4/SR4
SPM40G-SR4SR4	Модуль BYPASS 40G SR4/SR4
SPM40G-ER4SR4	Модуль BYPASS 40G ER4/SR4
SPM10G2-LRSR	Модуль BYPASS 2x10G LR/SR
SPM10G2-SRSR	Модуль BYPASS 2x10G SR/SR
SPM10G-ERSR	Модуль BYPASS 10G ER/SR

Рис. 1 – Модельный ряд модулей SignalPass

```
START Module 10G_ERSR TEST
DISABLE POWER CONTROLPASS #all
Enabled Signalpass power
POWER CONTROLPASS #1
POWER CONTROLPASS #4
Physical bypass: LINK (13/0 <-> 16/0) UP SUCCESS
MON 1: LINK (2/0 <-> 14/0) UP SUCCESS
MON 2: LINK (3/0 <-> 15/0) UP SUCCESS
OK: Mode changed.

NET 1: LINK (1/0 <-> 13/0) UP ERROR
NET 2: LINK (4/0 <-> 16/0) UP ERROR
       ****************************************

bfshell> ucli
Cannot read termcap database;
using dumb terminal settings.
bf-sde.pm> pm
error: unknown command 'pm'
bf-sde.pm> show
-----+----+---+----+-------+----+--+--+---+---+---+--------+---------------+---------------+-
PORT |MAC |D_P|P/PT|SPEED  |FEC |AN|KR|RDY|ADM|OPR|LPBK    |FRAMES RX      |FRAMES TX      |E
-----+----+---+----+-------+----+--+--+---+---+---+--------+---------------+---------------+-
1/0  |23/0|128|3/ 0|10G    |NONE|Au|Au|YES|ENB|DWN|  NONE  |              0|              0|
2/0  |20/0|152|3/24|10G    |NONE|Au|Au|YES|ENB|UP |  NONE  |              0|              0|
3/0  |18/0|168|3/40|10G    |NONE|Au|Au|YES|ENB|UP |  NONE  |              0|              0|
4/0  |16/0|184|3/56|10G    |NONE|Au|Au|YES|ENB|DWN|  NONE  |              0|              0|
13/0 |30/0|180|3/52|10G    |NONE|Au|Au|YES|ENB|DWN|  NONE  |              0|              0|
14/0 |28/0|164|3/36|10G    |NONE|Au|Au|YES|ENB|UP |  NONE  |              0|              0|
15/0 |26/0|148|3/20|10G    |NONE|Au|Au|YES|ENB|UP |  NONE  |              0|              0|
16/0 |24/0|132|3/ 4|10G    |NONE|Au|Au|YES|ENB|DWN|  NONE  |              0|              0|
bf-sde.pm> exit
bfshell> exit
 Controlpass  test ERROR
```

Рис. 2 – Обновленный интерфейс управления модулями

- Программа switchd управления чипом Intel Tofino была перенесена в systemd и для ее API разработаны функции-обертки, доступные непосредственно из командной строки Linux.

Результатом работы по рефакторингу стал набор утилит управления платформой, которые встраиваются в CLI Linux (рис. 3).

Рис. 3 – Итоговый набор разработанных утилит, доступных пользователю из командной строки

2.3. Описание инфраструктуры сборки ПО платформы

Для системы сборки и доставки ПО рассматривались 2 варианта:

- Сборка дистрибутива с помощью Buildroot – системы сборки встраиваемых Linux-систем. Он позволяет разработчикам создавать минимальные, оптимизированные и настраиваемые образы Linux для различных аппаратных платформ. Buildroot предоставляет набор скриптов и конфигурационных файлов, которые позволяют автоматизировать весь процесс сборки, включая загрузку и настройку необходимых компонентов операционной системы. Однако, минимальный образ Linux тяжело настроить для Barefoot SDE, так как она большое количество высокоуровневых зависимостей, каждую из которых было бы необходимо тоже собирать в Buildroot.
- Донастройка Ubuntu 20.04. Этот дистрибутив официально заявлен как поддерживаемый в Barefoot SDE, а также он был использован в первой итерации проекта. Также Ubuntu имеет менеджер пакетов и в целом достаточно прост в

использовании как пользователям (в случае оборудования – системным администраторам), так и разработчикам.

Для сборки операционной системы был выбран 2-ой вариант. В качестве основы операционной системы был использован Docker-образ Ubuntu 20.04 (в приложении А представлен Dockerfile). При подготовке Docker-образа были выполнены:

- Установка apt-пакетов с ядром ОС, initrd и systemd
- Установка программ для разработки: gcc, make, cmake; программ для коммуникации с аппаратным обеспечением: usbutils, pciutils, i2c-tools
- Были сгенерированы ssh-ключи и настроен root-доступ
- Добавлены systemd сервисы для switchd, rc-local compability и getty.

Для получения образа файловой системы был запущен контейнер из описанного образа, а затем его содержимое было экспортировано командой *docker export* в 10 ГБ .img файл. Затем в этот .img добляется загрузчик syslinux (аппаратная платформа имеет особенности bios, которые ограничивают выбор вторичных загрузчиков)

Для подготовки образа ОС, который будет прошиваться в SSD также необходимо сгенерировать MBR-часть. Это 512-байтная часть с исполняемым кодом MBR (440 байт) + таблица партиций (в данном случае один 10ГБ раздел).

После подготовки MBR и партиции с файловой системой с установленным syslinux-загрузчиком их необходимо совместить. Алгоритм генерации образа ОС представлен в Приложении Б.

2.4. Описание аппаратной и программной специфики 32-портового агрегатора-балансировщика трафика

Как было заявлено в весеннем отчете НИР, 16-портовый коммутатор-байпас послужил основой для разработки 32-портового 100-Гигабитного агрегатора-балансировщика трафика. Такое устройство обладает большим функционалом (в сравнении с коммутатором-байпасом) и к его отказоустойчивости и производительности применяются более серьезные требования. Поэтому такой программно-аппаратный комплекс дополнительно оборудован IPMI-контроллером. IPMI-контроллер на российском процессорном

Q7-модуле с Baikal-T [5], который контролирует и управляет Intel Core I5 Host-PC. Baikal выполняет следующие функции:

- Включение/выключение/перезагрузка хоста
- Переключение разъема USB на передней панели (переключение к Байкалу или Intel)
- Доступ к SPI-flash с Bios Intel
- Доступ к SATA-диску Intel (посредством переключения между Байкал и Intel).
- Доступ к консоли хоста по сети (технология Serial-over-Lan)
- Взаимодействие с STM32 BMC платы Tofino по интерфейсу UART.

Структурная схема агрегатора-балансировщика представлена на рисунке 4.

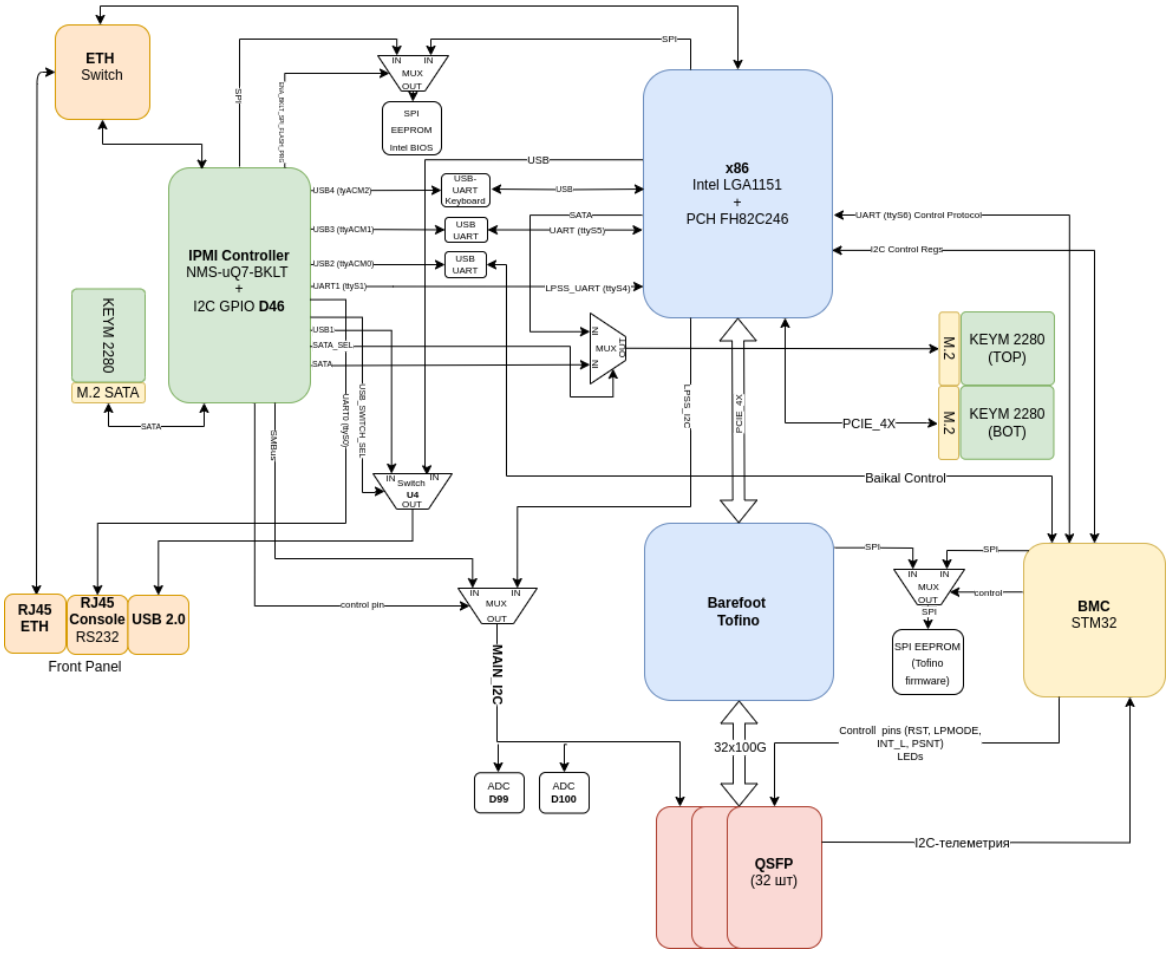

Рис. 4 – Структурная схема Агрегатора-балансировщика

Для реализации функций IPMI-контроллера на Байкал-T требуется подготовить специальный образ операционной системы Linux, которая будет учитывать аппаратные особенности платформы, а также предоставлять веб-службы (Serial-Over-Lan, непосредственно IPMI-сервер, веб-интерфейс).

Такую задачу решает опенсорсный проект OpenBMC, который предоставляет Yocto систему-сборки Linux с нужными ядрами, деревьев устройств и высокоуровнего ПО [3]. На изучение OpenBMC и попытки его запуска было потрачено 3 недели, но разработать минимально жизнеспособный продукт не вышло. Это связано со специфической архитектурой процессора Baikal-T – MIPS32. Большинство современных высокоуровневых приложений на JavaScript (с использованием V8, Node.js), C++ (boost) нативно не компилируются под архитектуру MIPS32. Также OpenBMC имеет очень глобальную, абстрактную архитектуру разбор и доработка которой заняла бы неоправданно много времени и ресурсов. Поэтому для удовлетворения нужд заказчиков разработки было принято сделать минимально-жизнеспособный продукт на Buildroot[4] Linux с веб-интерфейсом на C++ Framework Witty.

В web-интерфейсе IPMI-контроллера реализованы следующие функции:

- Вывод общей информации
- Задание IP адреса BMC
- Управление питанием x86 модуля
- Обновление прошивки BMC BAIKAL-T1000
- Обновление BIOS x86 модуля
- Обновление образа операционной системы x86 модуля на SATA-диске
- Смена пароля
- Перезагрузка BMC модуля

Вид web-интерфейса представлен на рисунке 5.

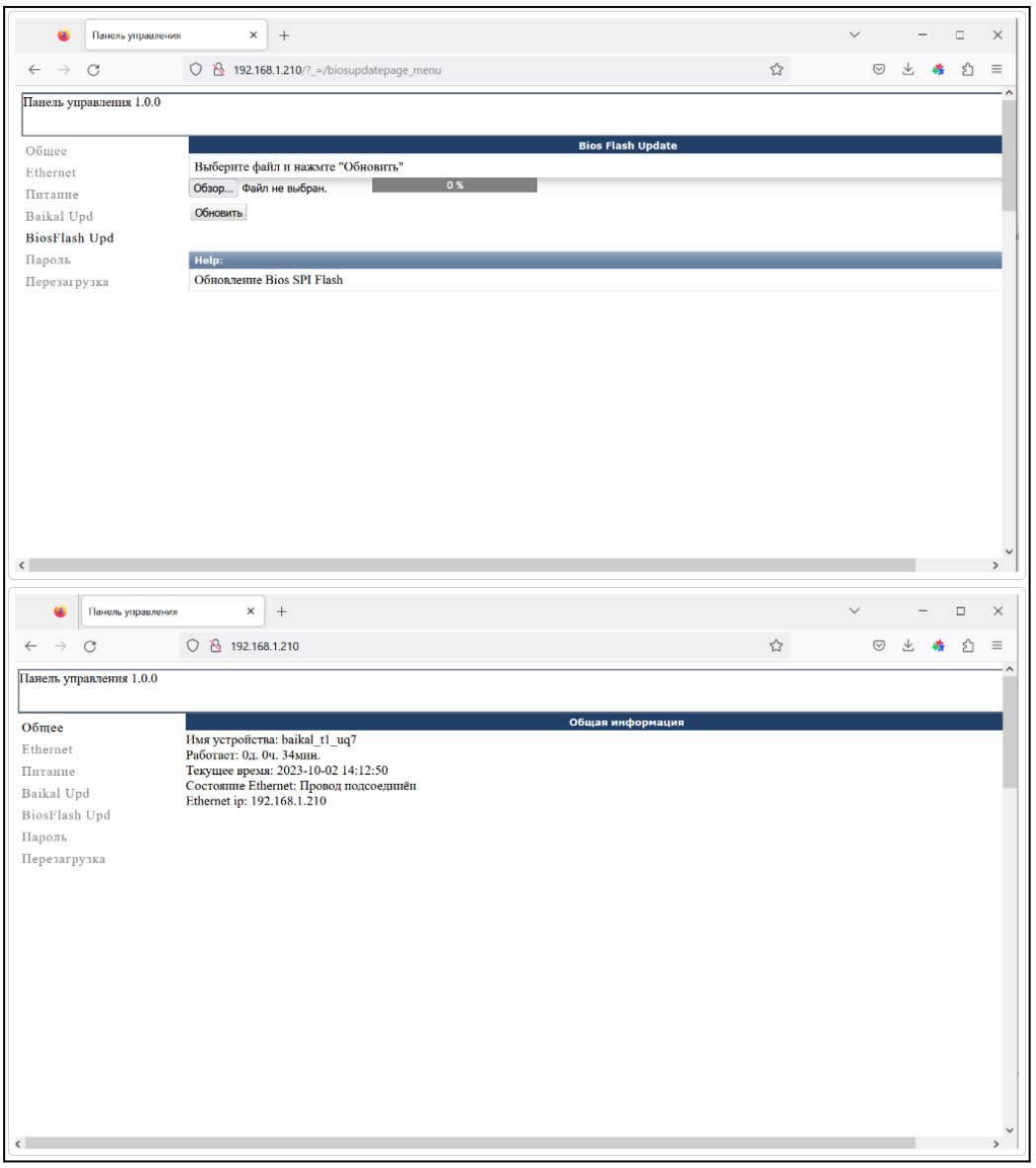

Рисунок 5 – Web-интерфейс BMC IPMI-контроллера.

Также на IPMI-контроллере реализован Serial-over-LAN (SOL) Service. Сервис на порту 2222 IPMI-контроллера предоставляет SSH-доступ к консольному порту Intel. Такой способ позволяет обеспечить полный удаленный доступ к x86 модулю (в том числе и Bios/U-boot) . На рисунке 6 представлен пример работы с SOL.

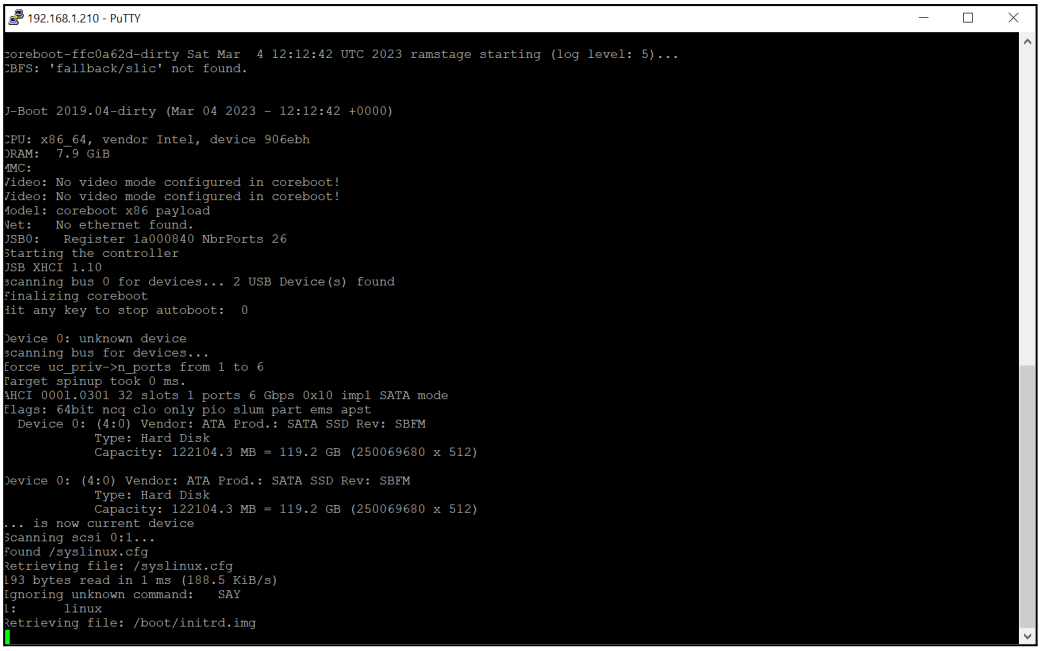

Рис. 6 – Работа сервиса Serial-Over-LAN

3. ОПИСАНИЕ ПРЕДПОЛАГАЕМОГО МЕТОДА РЕШЕНИЯ

На основании результатов работы весеннего и осеннего семестра можно описать следующую архитектуру решения:

Разрабатывается BSP для двух видов аппаратных платформ - коммутатора-байпаса и агрегатора-балансировщика. BSP представляет собой сборку операционной системы с интегрированными утилитами командной строки для управления состоянием аппаратной платформы. Обе платформы представляют собой коммутационные матрицы Intel Tofino (16 или 32 порта по 100ГБит/с), которые управляются x86 Intel CPU. На x86 собственный образ ОС Linux, собранный из Docker на основе Ubuntu 20.04. ПО пространства пользователя предоставляет CLI на Bash для управления состоянием аппаратной платформы. Также для реализации функций, которые не представляется возможным описать с помощью Bash, CLI использует утилиты собственной разработки на C++.

Платформы имеют следующие особенности, которые учитываются в пользовательском ПО:

- Коммутатор-байпас оснащен съемными модулями Signalpass (линейка из 10 разновидностей модулей), поэтому CLI должен учитывать поддерживать протокол управления модулями. Более подробно эта тема освещена в рамках весеннего НИР
- Агрегатор-балансировщик имеет IPMI-контроллер на процессорном модуле Baikal-T1000. Для него необходимо оформить свою систему сборки операционной системы Linux под архитектуру MIPS32. ОС система содержит веб-интерфейс управления платформой, Serial-over-LAN сервис и IPMI-сервер.

4. ПЛАН РАБОТЫ НА ВЕСЕННИЙ СЕМЕСТР

На весенний семестр поставлены следующие цели и задачи:

- Необходимо закончить оформление user-space BSP x86 для агрегатора балансировщика. Платформа имеет аппаратные отличия от коммутатора-байпаса, которые необходимо включить в ПО
- Оформить систему сборки IPMI-контроллера агрегатора-балансировщика. Для этого нужно подготовить загрузчик U-boot, Kernel , Device Tree и rootfs для процессорного модуля с Baikal-T1000.
- Добавить в сборку IPMI-контроллера IPMI-сервер [4], который обрабатывал бы запросы утилиты *ipmitool.* Для выполнения этого пункта предполагается адаптация IPMI-сервера с открытым исходным кодом из архитектуры OpenBMC

ЗАКЛЮЧЕНИЕ

В ходе работы в осеннем семестре была закончена работа по разработке user-space части BSP 16-портового коммутатора-байпаса. Были уточнены требования и потребности заказчиков, расширена линейка поддерживаемых обходных модулей SignalPass. Также была разработана система сборки операционной системы для аппаратных платформ коммутатора-байпаса и агрегатора-балансировщика трафика. Был проведен анализ аппаратной составляющей агрегатора-балансировщика и разработан минимально-жизнеспособный вариант IPMI-контроллера платформы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. P. Raghavan, Amol Lad, Sriram Neelakandan. Chapter 3. Board Support Package // Embedded Linux System Design and Development. — CRC Press, 2005. — ISBN 978-1-4200-3161-4. (дата обращения 15.10.2023)
2. Документация к Docker [Электронный ресурс] – URL – https://docs.docker.com/ (дата обращения 09.11.2023)
3. Документация к Yocto [Электронный ресурс] – URL – https://docs.yoctoproject.org/ (дата обращения 26.09.2023)
4. Intelligent Platform Management Interface Specification Second Generation [Электронный ресурс] – URL– https://www.intel.ru/content/www/ru/ru/servers/ipmi/ipmi-second-gen-interface-spec-v2-rev1-1.html (дата обращения 23.10.2023)
5. База-знаний компании разработчика аппаратной платфомы [Электронный ресурс] – URL – https://wiki.inmys.ru/ (дата обращения 25.05.2023)

ПРИЛОЖЕНИЕ А

Листинг Dockerfile для rootfs

```
FROM ubuntu:20.04

RUN \
      apt-get update -y \
      && apt-get -y install linux-image-virtual systemd-sysv \
      && apt-get -y install net-tools build-essential i2c-tools
isc-dhcp-client cmake\
            openssh-server nano dnsutils gparted iputils-ping
cloud-utils netcat \
            stm32flash picocom flashrom git pciutils usbutils \
      && apt-get -y install libthrift-0.13.0 libjudy-dev
libpcap0.8:amd64 \
      && echo root:root | chpasswd

RUN sed -i '/PermitRootLogin
prohibit-password/c\PermitRootLogin yes' /etc/ssh/sshd_config

ADD syslinux.cfg /

ADD boot/initrd.img boot/vmlinuz /boot/

ADD boot/5.4.0-153-generic /usr/lib/modules/5.4.0-153-generic

ADD etc/locale /etc/default/locale
ADD etc/hostname /etc/hostname
ADD etc/hosts /etc/hosts
ADD etc/rc-local.service /etc/systemd/system/
ADD etc/tofin-service.service /etc/systemd/system/

ADD etc/rc.local /etc/rc.local
ADD keys/* /root/.ssh/

#ADD opt/* /opt/
#ADD tofino-test-suite /
RUN echo "source /tofino-test-suite/set_env_vars.sh" >>
/root/.bashrc

CMD ["bash"]
```

ПРИЛОЖЕНИЕ Б

Алгоритм генерации образа операционной системы

```
docker run -it bypass_os
#generate rootfs part
dd if=/dev/zero of=p1.img bs=1 count=0 seek=10G
mkfs.ext4 -F p1.img
sudo mount -o loop p1.img /mnt/d4
docker ps -a | head #top: 474cf6834fb9
docker export ${CONT_ID} -o rootfs.tar.gz
sudo tar -xf rootfs.tar.gz -C /mnt/d4
sudo tar -xf bf-sde-9.7.1.tar.gz -C /mnt/d4
sudo cp -r additon/opt/* /mnt/d4/opt/
sudo cp -r additon/tofino-test-suite/ /mnt/d4/
sudo cp -r additon/bypass-binaries/ /mnt/d4/
sudo extlinux --install /mnt/d4
dd if=/dev/zero of=mbr.img bs=512 count=1
dd if=/usr/lib/syslinux/mbr/mbr.bin of=0.img bs=440 count=1
conv=notrunc
./genpart -t 0x83 -c -b 1 -s 20971520 | dd of=mbr.img bs=1
seek=446 conv=notrunc
echo -n -e '\x55\xaa' | dd of=0.img bs=1 seek=510 conv=notrunc
cat mbr.img p1.img > bypass_os.img
#for qemu test
qemu-system-x86_64-drive
file=ubuntu.img,index=0,media=disk,format=raw -m 4096
#for ssd flash
sudo dd if=bypass_os.img of=/dev/sda status=progress
```