ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»
ПО ПРОФИЛЮ «ИНЖЕНЕРНОЕ ДЕЛО»

38202

регистрационный номер

Секция: Системы обработки информации (ИУ5)

Унификация элементов систем обработки информации и передачи данных

Автор: **Андреев Герман Андреевич**

Инженерная школа №1581

Научный руководитель: **Львов Евгений Викторович, 17 Центральный проектный институт связи, к.т.н., к.в.н., доцент, заместитель Генерального директора по специальным проектам**

Москва - 2024

Аннотация

Целью работы является разработка унифицированного программного обеспечения (далее – ПО) для различных типов маршрутизаторов (роутеров).

Задачи, решаемые в данной работе следующие:

инженерное и экономическое обоснование трансфера технологий гражданского сектора экономики в военную сферу деятельности;

унификация функций управления маршрутизаторами различных производителей;

определение требований к аппаратным средствам и общему программному обеспечению;

разработка специального программного обеспечения роутеров;

создание прототипа рабочей системы;

проведение тестирования и апробация.

Актуальность работы обусловлена потребностями гражданского сектора экономики и военной сферы деятельности в разработке унифицированных элементов локальных-вычислительных сетей (далее – ЛВС). Унификация позволяет повысить эффективность развертывания типовых ЛВС, минимизировать ошибки конфигурирования и администрирования, уменьшить совокупные финансовые затраты при большой повторяемости, повысить скорость первоначального развертывания ЛВС, минимизировать затраты на обучение специалистов.

Предметной областью работы являются системы обработки информации и передачи данных. В целях реализации работы были применены следующие методы исследований: анализ публикаций по теме, классификация, сравнение, эксперимент.

Практическим результатом работы является разработка алгоритма и унифицированной программы для различных маршрутизаторов. В целях апробации основных решений в рамках работы создан рабочий прототип.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Когда требуется многократно построить однотипную локальную-вычислительную (далее – ЛВС) сеть, то целесообразно использовать унифицированное оборудование и инженерные подходы в ее создании. Задача построения типовых ЛВС стоит перед инженерами достаточно часто, например, при создании ЛВС сетей метрополитенов, при предоставлении услуг связи МГТС и в других случаях. Сейчас такая задача наиболее остро стоит перед военными инженерами в зоне проведении специальной военной операции (далее – СВО). Данный факт обусловлен объективными причинами, и прежде всего недостаточным финансированием и отсутствием производственных мощностей российских оборонных предприятий, уничтожением или выходом из строя большого количества оборудования ЛВС в ходе боевых действий [1]. В свою очередь недостаточное количество средств передачи информации приводит к уменьшению оперативности системы управления и в конечном счете к большим потерям в личном составе и технике. Особенно актуальна проблема для самых нижних органах военного управления в построении ЛВС, которые находятся на передовой на линии боевого соприкосновения с противником [2].

Решить проблему можно допустив, разумеется, с ограничениями, использование в военных целях элементов передачи информации, применяемых для построения ЛВС гражданского сектора экономики. Например, как это делают вооруженные силы Украины, используя терминалы Starlink для организации ЛВС и передачи информации.

Известно, что к элементам сетей передачи данных военного назначения применяются повышенные требования, их производство сопряжено с дополнительными затратами. Однако сегодня ситуация при проведении СВО такова, что нужно пожертвовать или упростить отдельные требования к оборудованию и обеспечить военных нужными техническими средствами в нужном объеме. По сути, необходимо обосновать и обеспечить трансфер

технологий из гражданского сектора в военный, что является составной частью научных и производственных задач МГТУ им.Баумана.

Одним из основных элементов сетей передачи данных являются маршрутизаторы(роутеры), которых в большом объеме как раз и не хватает для организации ЛВС систем управления в СВО. Другими словами, существует колоссальная потребность в роутерах при существенном ограничении средств для их создания или приобретения.

С другой стороны, на рынке компьютерной техники существует практически неограниченное количество различных домашних роутеров, которые можно адаптировать под цели двойного назначения.

Использование обычных домашних роутеров в военных целях будет сопряжено с проблемой отсутствия большого количества однотипных устройств с единообразной системой управления. При этом найти различные маршрутизаторы, разных производителей, не является сложной задачей, их много на рынке, причем относительно маленькой ценой. Организовать закупку возможно на пожертвования граждан или силами волонтёрских организаций. Например, главный редактор RT Маргарита Симоньян на пожертвования потратила более полумиллиона рублей, а этого хватило бы на 2.5 тысячи роутеров стоимостью 2 тысячи рублей.

Таким образом остается проблема с неунифицированной и неадаптированной под цели использования системой управления. Решить проблему можно путем создания унифицированного специального программного обеспечения для широкого круга роутеров.

Роутеры с открытой архитектурой позволяют без серьезных затрат, путем замены программного обеспечения, реализовать необходимые требования, за исключением специальных проверок и специальных исследований[1].

[1] специальная проверка и специальное исследование – исследование технических средств на предмет отсутствия недекларированных возможностей и закладок

Таким образом целью данной работы ставится разработка программного обеспечения (далее -ПО) для домашних роутеров, которое позволит использовать домашние роутеры различных производителей в создании унифицированных ЛВС, а так же, как продукцию двойного назначения.

В целях логической компоновки отчётных материалов работа декомпозирована по трем разделам.

В первом разделе работы дается обоснование разработки унифицированной программы для роутеров.

Во втором разделе определяется место разрабатываемой прикладной программы в системе, а также определяется состав основных компонентов прикладной программы. Выбирается язык программирования и среда разработки программного обеспечения для разработки прикладной программы.

В третьем разделе описывается создание тестового образца – прототипа системы, выполняется анализ достигнутого результата.

1 Обоснование разработки программы

ЛВС является одним из основных типов сетей, которые используются внутри групп пользователей для обмена данными между компьютерами и устройствами. Проектирование ЛВС требует тщательного планирования и учета различных факторов, таких как топология сети, выбор сетевого оборудования и обеспечение безопасности. Для ЛВС военного назначения применяются дополнительные требования: скрытность передачи данных и развертывания, надежность, ремонтопригодность, энергопотребление и другие.

При создании решения для построения типовых ЛВС необходимо унифицировать каждый элемент. Известно, что в состав ЛВС входят различные элементы: коммутационное оборудование, источники питания, кабели связи, усилители сигнала, антенны. Например, ЛВС может быть построена с использованием коаксиальных кабелей, кабелей на основе экранированной и неэкранированной витой пары и оптоволоконных кабелей, а может использовать в качестве среды передачи данных электромагнитные волны различных частот – КВ, УКВ, СВЧ. Если рассматривать значимость унификации каждого элемента, то наибольшую значимость в унификации всей ЛВС будет играть унификация коммутационного оборудования, поскольку кабели связи, источники питания и другие элементы уже достаточно унифицированы.

Таким образом построение типовой ЛВС будет сводиться к унификации коммутационного оборудования: системы управления этим коммутационным оборудованием. Существуют различные типы коммутационного оборудования, но наиболее универсальным является роутер. Следовательно, унифицировать управление ЛВС можно через создание однотипной системы управления роутерами. В свою очередь унификация системы управления роутерами может осуществляться за счет создания собственного уникального программного обеспечения для

различных типов роутеров. Другими словами, используя одинаковое ПО можно унифицировать функции управления различных типов роутеров, в том числе разных производителей.

Основная функция роутера – пересылка пакетов между различными сегментами сети на основе правил. Роутер может связывать разнородные сети, например, домашний роутер, связывает внутренние домашние устройства с сетью Интернет. Для принятия решений о пересылке пакетов роутером используется информация о топологии сети и определённые правила, которые задает администратор сети и роутера. Кроме основной функции у роутера есть вспомогательные функции, которые зачастую и определяют его потребительские свойства.

По областям применения маршрутизаторы делятся на несколько классов [3]:

– магистральные маршрутизаторы предназначены для построения центральной сети корпорации. Центральная сеть может состоять из большого количества локальных сетей, разбросанных по разным зданиям и использующих самые разнообразные сетевые технологии, типы компьютеров и операционных систем. Магистральные маршрутизаторы — это наиболее мощные устройства, способные обрабатывать несколько сотен тысяч или даже несколько миллионов пакетов в секунду, имеющие большое количество интерфейсов локальных и глобальных сетей.

– маршрутизаторы региональных отделений соединяют региональные отделения между собой и с центральной сетью. Сеть регионального отделения, так же, как и центральная сеть, может состоять из нескольких локальных сетей. Такой маршрутизатор обычно представляет собой некоторую упрощенную версию магистрального маршрутизатора.

– маршрутизаторы удаленных офисов соединяют, как правило, единственную локальную сеть удаленного офиса с центральной сетью или сетью регионального отделения по глобальной связи. В максимальном

варианте такие маршрутизаторы могут поддерживать и два интерфейса локальных сетей.

– маршрутизаторы локальных сетей предназначены для разделения сетей на локальные подсети. К маршрутизаторам локальных сетей относятся маршрутизаторы для использования в домашних условиях, обычно такие маршрутизаторы называются «домашний роутер».

Если пользователю необходимо решение для организации домашней сети или небольшого офиса и доступом в Интернет с минимальным объемом средств на приобретение и количеством усилий на создание сети, то обычно выбираются присутствующие на рынке домашние роутеры с установленным производителем роутера программным обеспечением. Если нужно построить несколько однотипных локальных сетей, то инженеры стараются использовать типовое решение, основанное на унифицированном роутере.

Унифицированные роутеры для построения типовых ЛВС имеют обширную сферу применения в гражданском секторе экономики, но наиболее актуальны для развертывания ЛВС в зоне СВО, так как их катастрофически не хватает. При этом необходимо учесть особые требования к таким роутерам, имеется ввиду максимальное упрощение механизмов первоначальной настройки, создание функции дистанционной настройки и администрирования роутеров.

Таким образом существует актуальная задача разработки программы для «домашнего» роутера, обеспечивающей унификацию управления и выполнение дополнительных требований для возможности осуществления трансфера технологии в военную сферу. Готовая прикладная программа не позволит решить задачу целостно, нужно дополнительно выполнить следующее: определить перечень требуемых характеристик роутеров, установить операционную систему на роутер, установить и запустить прикладную программу на роутере, а также убедиться в работоспособности системы, проведя тестирование.

Основными характеристиками роутеров являются следующие:

– скорость передачи данных;

– максимальное количество подключаемых устройств;

– максимальную дальность покрытия для Wi-Fi роутеров;

– напряжение питания. Данная характеристика особенно актуальна, когда источником питания для роутера служит обычный аккумулятор с напряжением 12 вольт;

– ток потребления. Так же важный параметр для работы в автономных полевых условиях от аккумуляторов. Чем меньше ток потребления, тем дольше будет работать роутер от аккумулятора;

– установленное на роутер программное обеспечение, которое в ИТ еще называется прошивкой.

Система управления роутеров различных моделей и производителей отличается друг от друга набором и последовательностью выполнения функций, интерфейсами администратора и другими параметрами. В случае настройки большого количества роутеров разного типа у неопытных пользователей могут возникать ошибки, что приведет к снижению эффективности организации сети передачи данных.

Как правило прошивка состоит из операционной системы, драйверов, веб-сервера и программы для администрирования устройства через Web-браузер. Программа для администрирования выполняет ряд функций:

– просмотр статуса устройства: включено/выключено;

– определение подключенных устройств, их скорости и частоты работы;

– изменения названия локальной сети и пароля доступа к устройству;

– диагностирования неисправностей для выявления и устранения сетевых проблем.

Вместе с тем для реализации целей работы домашний роутер должен обладать еще одной характеристикой – открытой архитектурой. Под открытой архитектурой понимается возможность менять программное

обеспечение роутера, например, обеспечивать установку сторонней операционной системы, прошивки.

Среди наиболее популярных производителей домашних роутеров с открытой архитектурой в настоящее время на рынке представлены следующие [4]:

– Asus компания известна своими быстрыми и надежными роутерами. Они имеют высокие показатели скорости Wi-Fi, а также поддерживают новейшие технологии, такие как Wi-Fi 6. Кроме того, у роутеров Asus есть простой и понятный интерфейс управления, который даже новички могут легко освоить.

– TP-Link –этот бренд известен своей доступностью и простотой в использовании. Роутеры TP-Link имеют отличную производительность и множество функций, включая защиту от вредоносных программ и возможность управления родительским контролем.TP-Link производитель компьютерного и телекоммуникационного оборудования. Главный офис компании расположен в Китае, в городе Шэньчжэнь.

– Xiaomi бренд, который предлагает высококачественные роутеры по доступной цене. Они имеют отличный дизайн и поддерживают самые новые технологии Wi-Fi. Кроме того, роутеры Xiaomi имеют функцию управления с помощью мобильного приложения, что делает их очень удобными в использовании. Компания Xiaomi зарегистрирована на Каймановых островах, штаб-квартира находится в Пекине.

– Netgear бренд известен своими роутерами высокого класса, которые подходят для больших домов и офисов. Они имеют высокие показатели скорости и мощности, а также множество дополнительных функций, таких как защита от DDoS-атак и возможность создания гостевой сети.

– D-Link бренд, который предлагает широкий выбор роутеров, от простых до более продвинутых моделей. Роутеры D-Link имеют хорошую производительность и поддерживают самые новые технологии Wi-Fi. Они также имеют защиту от вредоносных программ и родительский контроль.

– Huawei компания из Китая, которая предлагает высококачественные роутеры по доступной цене. Они имеют хорошую производительность и множество функций, такие возможность создания гостевой сети, а также поддержку технологии Wi-Fi 6. Кроме того, у роутеров Huawei есть функция управления с помощью мобильного приложения, что делает их удобными в использовании.

– Keenetic компания разрабатывает и продает проводное и беспроводное сетевое оборудование, в частности маршрутизаторы и точки доступа.

– Zyxel международная компания со штаб-квартирой на Тайване, производитель сетевого оборудования для среднего и малого бизнеса, промышленных предприятий и домах. Для реализации задачи трансфера технологий подходят абсолютно все, но наиболее предпочтительны компании-производители из Китая, так как имеют наименьшие риски попасть под ограничение в поставках оборудования в Российскую Федерацию в период санкционного давления.

Следующая задача, которую нужно решить заключается в выборе операционной системы для домашнего роутера. Существует целый ряд встраиваемых в роутеры операционных систем [5]. Выбраны для оценки использования из них следующие:

– OpenWRT[6] – встраиваемая операционная система, основанная на ядре Linux, предназначенная, в первую очередь, для домашних маршрутизаторов (роутеров, от англ. router). Основные компоненты включают OpenWRT в себя ядро Linux, util-linux, uClibc или musl и BusyBox. Размер всех компонентов минимизирован в связи с тем, что в большинстве роутеров сильно ограничен объём памяти. Кроме того, в OpenWRT входит пакетный менеджер opkg с репозиторием, в котором более 3000 пакетов программ и утилит.

– FreeWRT – дистрибутив Linux для встраиваемых систем, таких как беспроводные маршрутизаторы фирмы Linksys и Asus. Особенность -

возможность собирать(компилировать) прошивку на том же устройстве, где она будет установлена.

– DD-WRT – популярная прошивка маршрутизатора с открытым исходным кодом на базе Linux, предоставляет расширенные функции, такие как VPN, VLAN и QoS, и может быть установлен на широкий спектр маршрутизаторов. DD-WRT подходит как для домашнего, так и для корпоративного использования, для тех кто не хочет изучать настройку конфигурационных файлов и устанавливать дополнительные программы.

– pfSense – Платформа с открытым исходным кодом на базе FreeBSD. Она предоставляет широкий спектр функций, включая VPN, балансировку нагрузки, формирование трафика и многое другое. Платформу можно использовать для создания целого ряда сетевых устройств, от небольших домашних маршрутизаторов до брандмауэров крупных предприятий.

– VyOS – основана на Debian Linux. Она предоставляет широкий спектр функций маршрутизации и безопасности, включая VPN, брандмауэр, NAT и многое другое. VyOS можно использовать для создания целого ряда сетевых устройств, от небольших домашних маршрутизаторов до брандмауэров крупных предприятий.

– RouterOS – разработана MikroTik. Она предоставляет широкий спектр функций, включая VPN, брандмауэр, точку доступа и многое другое. RouterOS можно использовать для создания целого ряда сетевых устройств, от небольших домашних маршрутизаторов до брандмауэров крупных предприятий.

– LibreWRT – вободная прошивка от Фонда свободного программного обеспечения, которая «отпочковалась» от OpenWRT и практически ничем, кроме отсутствия проприетарных драйверов, от последней не отличается. Примечательна тем, что из-за нее Фонд свободного программного обеспечения немного изменил свои принципы: если до этого одним из условий «свободы» была необходимость иметь возможность компиляции

приложения на том же устройстве, на котором оно запускается, то теперь это необязательно.

Для реализации задачи работы подойдет любая из приведенных выше операционных систем. Кроме того, можно самостоятельно собрать образ операционной системы на имеющийся роутер, но на практике выбор операционной системы определяется наличием уже готовых решений для конкретного роутера.

Следующая задача, разработка прикладной программы. Первым этапом разработки любых прикладных программ является выбор языка программирования. Как правило для создания программного обеспечения роутеров используются языки программирования Си и Ассемблер, которые предназначены для реализации высокопроизводительных систем. В отдельных случаях используется Phyton или Perl, например, когда роутеру не требуется обеспечивать высокую производительность или достаточно оперативной памяти для выполнения программы.

Учитывая, что в школьном курсе информатики изучается Phyton, то выбор сделан в пользу именно этого языка разработки программы.

Phyton обладает огромным перечнем специализированных библиотек, которые покрывают практически все потребности в создании программ под различные цели. Вместе с тем, при неоспоримом преимуществе Phyton в кроссплатформенности, у него есть определенный недостаток – Phyton относится к интерпретируемым языкам программирования. Следовательно, кроме операционной системы для исполнения программы на домашнем роутере потребуется наличие еще и интерпретатора Phyton, установленного на роутер. Интерпретатор Phyton имеется для всех перечисленных выше операционных систем роутеров, что не ограничивает выбор операционной системы для роутера.

2 Проектирование программы, разработка алгоритма и программы

Разработка прикладной программы начинается с определения ее места в системе и декомпозиции программы на основные подсистемы(модули). Место разрабатываемой прикладной программы в системе показано на рисунке 2.1.

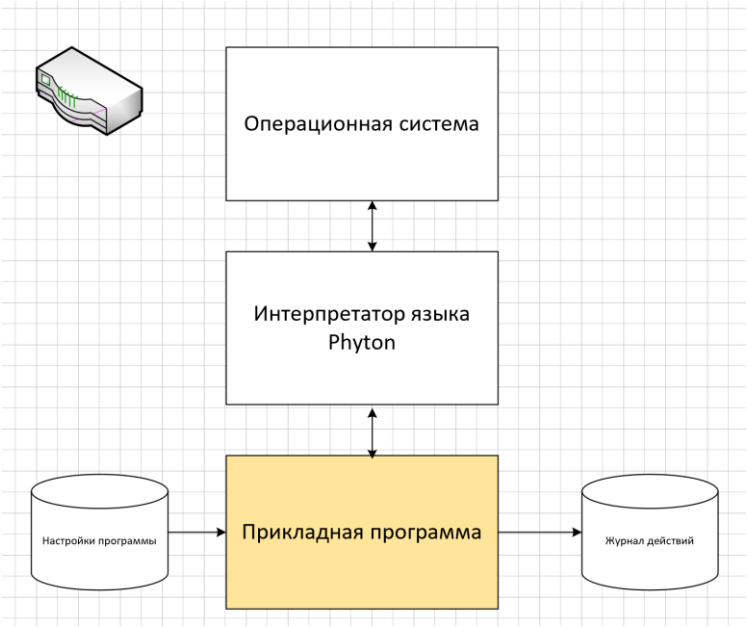

Рисунок 2.1 – Основные элементы системы

В целях эффективной разработки прикладная программа декомпозирована на следующие подсистемы: подсистема взаимодействия с мессенджером Telegram, подсистема чтения конфигурационных файлов, подсистема администрирования пользователей, подсистема журналирования, подсистема безопасности.

Основной подсистемой программы является подсистема взаимодействия с мессенджером Telegram, которая реализует функцию информационного взаимодействия с мессенджером Telegram через его открытый программный интерфейс (API). Мессенджер Telegram широко известен и представляет собой облачный сервис обмена сообщениями и другой информацией. Подсистема взаимодействия с мессенджером Telegram позволяет администратору обеспечивать управление из любой точки сети.

При использовании роутера для военных целей возникает вопрос с использованием мессенджера Telegram. По сути, в схеме информационного взаимодействия имеется третья сторона, которая не принадлежит Российской Федерации, и уровень конфиденциальности, обеспечиваемый данным ресурсом не совсем ясен. Однако существует множество фактов использования мессенджера Telegram для передачи конфиденциальной информации и на сегодня отсутствует военная альтернатива такой системе. Кроме того, информация на поле боя имеет актуальность не более 1 суток, что связано с высокой динамикой действий и скоростью устаревания информации в таких условиях. Таким образом предлагаемый подход вполне отвечает требованиям безопасности информации.

Подсистема чтения конфигурационных файлов позволяет загружать настройки программы из конфигурационных файлов, реализует функцию «мини базы данных». Конфигурационные файлы имеют строгий формат.

Подсистема администрирования пользователей позволяет добавлять и удалять права пользователей для доступа, администрировать топологию сети.

Подсистема журналирования предназначена для ведения журналов событий. По журналу событий в последующем можно определить, когда пользователь подключился в сеть Интернет и когда отключился.

Подсистема безопасности обеспечивает защиту информации при информационном взаимодействия роутера и мессенджер Telegram.

Обобщенный алгоритм работы программы приведен на рисунке 2.2.

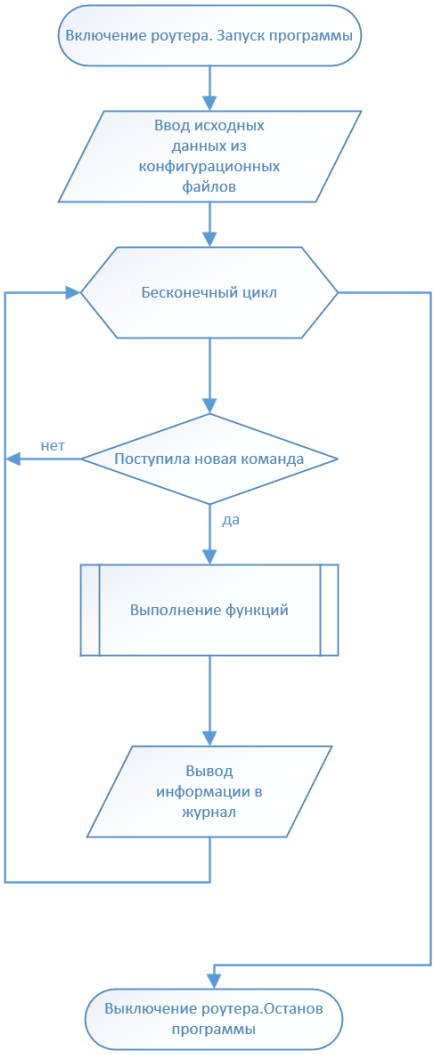

Рисунок 2.2 – Обобщенный алгоритм работы программы

Программа начинает выполнение автоматически при старте роутера и завершает свое выполнение только в момент выключения роутера. Для роутеров, которые имеют функцию аппаратной перезагрузки, программа перезагружается и начинает выполнятся заново в момент аппаратной перезагрузки.

Для разработки программы на языке Phyton можно использовать различные среды разработки или текстовые редакторы, но лучше использовать среды со статическим анализатором, который рассматривает исходный код и предупреждает о потенциальных ошибках[7], например, PyCharm или IntelliJ IDEA Поскольку на роутере может быть установлена

только определенная версия интерпретатора Phyton, то для разработки прикладной программы целесообразно использовать такую среду, в которую можно загрузить нужный интерпретатор языка Phyton. Такими возможностями обладает среда разработки IntelliJ IDEA. В работе на первом этапе использовался PyCharm, а потом IntelliJ IDEA для проверки требований по работе программы в различных версиях интерпретатора Phyton.

3 Создание рабочего прототипа и проведение тестирования

Разработанную программу для домашнего роутера можно протестировать только на реальном роутере с подключением в сеть Интернет. В качестве роутера для тестирования программы выбран домашний маршрутизатор ASUS RT-AC51U[8].

Первым шагом в создании действующего прототипа системы является установка операционной системы на роутер. Для каждого типа и версии роутера разрабатывается свой релиз операционной системы. В случае выбора неверной версии или релиза операционной системы, роутер может неправильно работать, а в отдельных случаях даже физически выйти из строя. По этой причине к выбору релиза операционной системы стоит подходить максимально внимательно.

В качестве операционной системы выбрана OpenWRT. С сайта https://openwrt.org загружен двоичный файл openwrt-23.05.0-ramips-mt7620-asus_rt-ac51u-squashfs-sysupgrade.bin с операционной системой для роутера. В названии файла закодировано семейство чип-сетов (MediaTek MT7620A), то есть микроконтроллера на котором реализован роутер, конфигурация ядра «generic», название и версия аппаратной части роутера, тип файловой системы и для какой именно цели предназначен образ. Минимальный состав прошивки обозначается «factory», а обновление существующей OpenWRT - «sysupgrade». Выбран образ с обновлением.

После загрузки операционной системы на роутер необходимо установить Phyton, для это используется команда «opkg install python-light python-pip». В команде указывается необходимость установки менеджера пакетов Phyton для последующей загрузки дополнительных библиотек нужных прикладной программе.

Проверка того, что интерпретатор загружен и работает осуществляется командой «python –version».

После выполнения текущих проверок, необходимо загрузить прикладную программу на роутер. Прикладная программа устанавливается в попку /root/telebot/bot2.

Особенностью работы всех программ на роутере является то, что все они должны автоматически выполнятся на роутере и загружаться в момент его включения или аппаратной перезагрузки.

Операционная система стартует автоматически на роутере, а для автоматического запуска прикладной программы была использована программа «cron job», настройка которой следующая: «cron job (* * * * * /root/telebot/bot2/tg_check_down.sh & >> /etc/crontabs/root). Программа «crone job» запускает скрипт «tg_check_down.sh», который проверяет список процессов на наличие в нем запущенной прикладной программы, и если не находит нужной запущенной, то запускает ее. Дополнительно скрипт записывает информацию в журнал «log_down» о запуске.

Для оценки времени готовности роутера к работе было выполнено 5 тестов. Роутер выключался из сети 220 вольт, а затем включался. Среднее время готовности роутера составило 9 секунд с точностью +/– 1 секунда.

После полной настройки роутера, его необходимо подключить к сети Интернет через WAN-порт. Обычно он выделен определенным цветом и надписью «WAN» или «INTERNET». Используются три основные типа подключения роутеров:

– Ethernet, самый распространенный вариант названия «витая пара» или «патч-корд»;

– ADSL, обеспечивает подключение к сети Интернет по телефонной линии;

– GPON, позволяет подключаться к сети Интернет с помощью оптоволоконного кабеля.

Для соединения устройств в локальной сети с роутером используется LAN-порты. Сколько портов - сколько устройств одновременно могут быть подключены в локальную сеть. LAN-порты могут организовываться

посредством радиоканала, такая технология называется Wi-Fi. В ASUS RT-AC51U можно подключить к LAN по проводной связи 4 устройства и до 10 по Wi-Fi.

Один из вариантов организации ЛВС с применением унифицированного роутера представлен на рисунке 3.1.

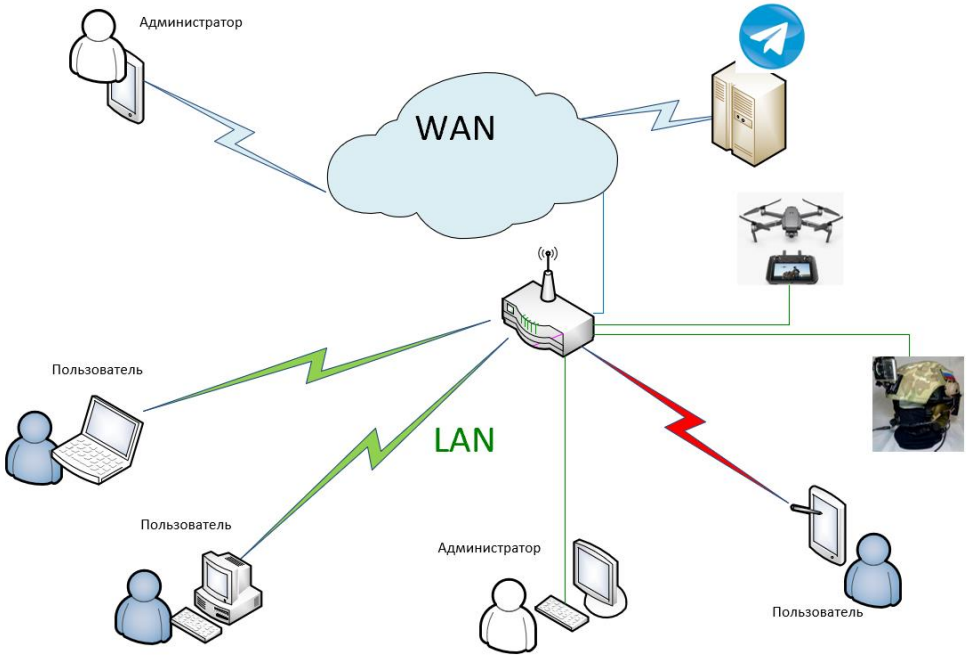

Рисунок 3.1— Вариант организации ЛВС с применение унифицированного роутера

Любая программа для использования должна обладать инструкцией. Инструкция администратора приведена в приложении А, листинг – в приложении Б.

В целях подтверждения возможности создания унифицированного программного обеспечение была отработана процедура установки прикладной программы на другой роутер более старой модели ZyXel Keenetic Giga III. В ходе эксперимента, прикладная программа была установлена на роутер, все функциональные возможности соответствовали возможностям программы установленной на ASUS RT-AC51U.

Заключение

В результате работы разработан алгоритм и прикладная программа для различных типов роутеров, создан рабочий образец и проведено тестирование программы на роутере ASUS RT-AC51U и ZyXel Keenetic Giga III. Тестирование подтвердило предположения и характеристики разработанной программы. В результате работы доказано, что разработанное ПО можно использовать на различных типах роутеров, а это значит, что для построения типовых ЛВС достаточно обновить ПО на роутере и в итоге получается унифицированная система управления роутерами и ЛВС.

Совокупность разнотипных роутеров с обновленным программным обеспечением может использоваться при построении типовых ЛВС. Затраты на создание унифицированных решений являются минимальными. Роутеры с обновленным ПО могут быть использованы как продукции двойного назначения.

В дальнейшем предполагается использовать полученные результаты работы для выполнения доработки ПО в целях создания самоорганизующихся [9] сетей передачи данных на основе унифицированных решений, в том числе как продукции двойного назначения.

Список использованных источников

1 Официальный сайт КиберЛенинка [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/obosnovanie-oblika-postroeniya-perspektivnyh-kompleksov-i-sredstv-svyazi-na-osnove-opyta-organizatsii-svyazi-pri-provedenii (дата обращения: 05.02.2024).

2 Официальный сайт Военное обозрение [Электронный ресурс]. – Режим доступа: https://topwar.ru/219980-samaja-kriticheskaja-problema-nashih-vooruzhennyh-sil-v-svo-svjaz.html (дата обращения: 05.02.2024).

3 Официальный сайт «Studfile» [Электронный ресурс]. – Режим доступа: https://studfile.net/preview/7391139/page:45/ (дата обращения: 05.01.2024).

4 Официальный сайт «Техногрант» [Электронный ресурс]. – Режим доступа: https://tekhnogarant.com/rejting-wi-fi-routerov (дата обращения: 05.01.2024).

5 Официальный сайт «Wikipedia» [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/List_of_router_and_firewall_distributions (дата обращения: 06.01.2024).

6 Официальный сайт OpenWRT [Электронный ресурс]. – Режим доступа: https://openwrt.org/ (дата обращения: 06.01.2024).

7 Свейгарт Э. Phyton. Чистый код для продолжающих.СПб.:Питер, 2022.

8 Официальный сайт ASUS [Электронный ресурс]. – Режим доступа: https://www.asus.com/ (дата обращения: 06.01.2024).

9 Официальный сайт КиберЛенинка [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/analiz-sostoyaniya-i-perspektivy-razvitiya-samoorganizuyuschihsya-setey (дата обращения: 05.02.2024).

Приложение А. Инструкция администратора.

Для запуска программы необходимо:

1. Перед первым запуском нужно скопировать исполнительный код в файловую систему
2. Далее нужно создать нового бота через Чат-бот Telegram «Bot Father» и получить токен
3. Полученный токен нужно записать в файл config.py, запись должна иметь вид «TOKEN=********»
4. Далее нужно настроить файрволл, в него добавляем правила, которыми будет управлять программа роутера, в данное правило заносим устройства которые нужно блокировать и устанавливаем уровень “REJECT”
5. В файле bot2.py нужно указать id пользователя Telegram, который будет иметь права администратора
6. Далее нужно запустить скрипт install, который установит все нужные пакеты и добавить задачу для crontabs, далее перезапустить роутер

Приложение Б. Листинг программы

(Диск CD с программой передается отдельно).

```
import os
import keyboards2 as kb
from config import TOKEN
import telebot
from telebot import types
import on_off as on
import user_action as ua
import binascii
import re
from datetime import datetime
import edit_firewall as ef

time = 0;
admin_id = "139050906"

bot = telebot.TeleBot(TOKEN)

def access(message):
   if ua.check_user(str(message.from_user.id)) or message.from_user.id ==
int(admin_id):
      return True
   else:
      return False

def add_ssh_pub_to_tmp(key):
   f = open("/root/telebot/bot2/ssh.tmp", "w+")
   f.write(key)
   f.close

def check_add_ssh_pub():
   return os.popen("sh /root/telebot/bot2/check_add_ssh_pub.sh").read()

@bot.message_handler(commands=['start'])
def process_start_command(message):
   global new_user_id
   global new_user_first_name
   if access(message):
      bot.send_message(message.from_user.id, text = u'Вы авторизованы')
   else:
      bot.send_message(message.from_user.id, text = u'Вы не
авторизованы.\nВаш запрос отправлен администратору!!!')
      bot.send_message(admin_id, text = u'Добавить пользователя "%s"
(id:%d)?' % (message.from_user.first_name, message.from_user.id),
reply_markup=kb.add_new_user(message.from_user.id,
message.from_user.first_name))

help_str_auth = u'/help - Список функций\n\
/start - Авторизация\n\
/inet_on - Включить доступ\n\
/inet_off - Отключить доступ\n\
/ssh_add_pub_key - Добавить публичный ключ для ssh\n\
/status - Узнать статутс доступа\n\
/time_left - Узнать отсаток времени\n'

help_str = u'Вы не авторизованы!!!\n/start - Авторизация\n'
help_str_root = u'/help - Список функций\n\
/start - Авторизация\n\
```

```
/inet_on - Включить доступ\n\
/inet_off - Отключить доступ\n\
/list - Вывод списка пользователей\n\
/del_user - Удалить пользователй\n\
/ssh_add_pub_key - Добавить публичный ключ для ssh\n\
/status - Узнать статутс доступа\n\
/time_left - Узнать отсаток времени\n\
/clients_list - Список подключенных клиентов\n\
/black_list - Список наказанных пользователей\n\
/add_user_black_list - Добавить пользователя в список наказанных
пользователей\n\
/delete_user_black_list - Удалить пользователя из списка наказанных
пользователей\n'



@bot.message_handler(commands=['help'])
def process_help_command(message):
    if message.from_user.id == int(admin_id):
       bot.send_message(message.from_user.id, text = help_str_root)
    elif access(message):
       bot.send_message(message.from_user.id, text = help_str_auth)
    else:
       bot.send_message(message.from_user.id, text = help_str)


@bot.message_handler(commands=['list'])
def process_help_command(message):
    if message.from_user.id == int(admin_id):
       string = ""
       for item in range(0, ua.get_count_user()):
          string = u'%s%d.\t%s\t(id:%s)\n' % (string, item + 1,
ua.get_user_list()[item][1], ua.get_user_list()[item][0])
       bot.send_message(message.from_user.id, text = string)
    else:
       bot.send_message(message.from_user.id, text = u'Вы не root!!!')


@bot.message_handler(commands=['status'])
def process_help_command(message):
    if access(message):
       check = on.check(0)
       if check == 0:
          bot.send_message(message.from_user.id, 'Доступ включен')
       else:
          bot.send_message(message.from_user.id, 'Доступ выключен')
    else:
       bot.send_message(message.from_user.id, text = u'Вы не
авторизованы!!!')


@bot.message_handler(commands=['ssh_add_pub_key'])
def process_help_command(message):
   if access(message):
         bot.send_message(message.from_user.id, text = u'Скопируйте ваш
публичный ключ, затем вставьте сюда и поставьте вначале "/"')
   else:
      bot.send_message(message.from_user.id, text = u'Вы не авторизованы!!!')


@bot.message_handler(commands=['ssh-rsa'])
def process_help_command(message):
   if access(message):
```

```
      key = re.split(r'\s{1,}', message.text)
      if len(key) == 3:
         bot.send_message(admin_id, text = u'Пользователь %s (id:%s) хочет
добавить свой публичный ssh ключ!!!' % (message.from_user.first_name,
message.from_user.id), reply_markup = kb.add_ssh(message.from_user.id,
message.from_user.first_name))
         string = key[0]
         for i in range(1, 3):
            string = u'%s %s' % (string, key[i])
         string = u'%s\n' % (string[1:])
         print(string)
         add_ssh_pub_to_tmp(string)
   else:
      bot.send_message(message.from_user.id, text = u'Вы не авторизованы!!!')


@bot.message_handler(commands=['inet_on'])
def process_start_command(message):
   if access(message):
      bot.send_message(message.from_user.id, text = u'Выберите время работы
доступа!', reply_markup = kb.hours)
   else:
      bot.send_message(message.from_user.id, text = u'Вы не авторизованы.')

@bot.message_handler(commands=['inet_off'])
def process_help_command(message):
   if access(message):
      if on.check(0) == 0:
         bot.send_message(message.from_user.id, text = u'Доступ принудительно
выключен')
         os.system("/bin/sh /root/telebot/managment/en_block")
         atq = "atq | awk '{print $1}'"
         answ = os.popen(atq).read()
         atrm  = "atrm " + str(answ)
         os.system(atrm)

         badList = ef.getListMacsFromFirewall("option name 'Gera'\n");
         # badList = ef.getListMacsFromFirewall("\toption name 'Gera'");
         UBUS = "ubus call hostapd.wlan1 del_client \"{\'addr\':'%s',
\'reason\':5, \'deauth\':false, \'ban_time\':10000}\""

         for item in badList:
            print("UBUS" + item.lower())
            print(UBUS % item.lower())
            os.system(UBUS % item.lower())


      else:
         bot.send_message(message.from_user.id, text = u'Доступ уже
выключен')
   else:
      bot.send_message(message.from_user.id, text = u'Вы не авторизованы.')


@bot.message_handler(commands=['del_user'])
def process_help_command(message):
   if message.from_user.id == int(admin_id):
      if not (ua.get_count_user()) == 0:
         bot.send_message(message.from_user.id, text = u'Вы в root!!!')
         bot.send_message(message.from_user.id, text = u'Выберите
пользователя котогрого хотите удалить!!!', reply_markup =
kb.generate_buttons(ua.get_count_user(), ua.get_user_list()))
      else:
```

```
            bot.send_message(message.from_user.id, text = u'Список пользователей
пуст!!!')
      else:
         bot.send_message(message.from_user.id, text = u'Убрать
пользователя!!!')


@bot.message_handler(commands=['time_left'])
def process_help_command(message):
      if access(message):
         if on.check(0) == 0:
            format = '%H:%M:%S'
            time = os.popen("atq | awk '{print $5}'").read()
            date = os.popen("date | awk '{print $4}'").read()
            stop = datetime.strptime(time.strip('\n'), format)
            start = datetime.strptime(date.strip('\n'), format)
            delta = stop - start
            bot.send_message(message.from_user.id, u'Доступ выключиться через
%s' % (delta))
         else:
            bot.send_message(message.from_user.id, u'Доступ выключен')
      else:
         bot.send_message(message.from_user.id, text = u'Вы не авторизованы.')

@bot.message_handler(commands=['clients_list'])
def process_get_network_clients_list_commadn(message):
      if message.from_user.id == int(admin_id):

         bot.send_message(message.from_user.id, text = u'Вы в root!!!')

         string = u'Список подключенных пользователей:\n'

         connectionsList = ef.getConnections()
         for i in range(0, len(connectionsList)):
            item = connectionsList[i]
            string = u'%s%d. %s - %s - %s\n' % (string, i + 1, item[0].upper(),
item[1], item[2])

         bot.send_message(message.from_user.id, text = string)

      else:
         bot.send_message(message.from_user.id, text = u'Не давать доступ!!!')

@bot.message_handler(commands=['black_list'])
def process_get_network_clients_list_commadn(message):
      if message.from_user.id == int(admin_id):
         bot.send_message(message.from_user.id, text = u'Вы в root!!!')

         string = u'Список плохих пользователей:\n'

         listMacsFromFirewall = ef.getListMacsFromFirewall("option name
'Gera'\n")
         # listMacsFromFirewall = ef.getListMacsFromFirewall("\toption name
'Gera'")
         for i in range(0, len(listMacsFromFirewall)):
            item = listMacsFromFirewall[i]
            string = u'%s%d: %s\n' % (string, i + 1, item)
         bot.send_message(message.from_user.id, text = string)
      else:
         bot.send_message(message.from_user.id, text = u'Не давать доступ!!!')

# @bot.message_handler(commands=['add_user_black_list'])
# def process_get_network_clients_list_commadn(message):
```

```
#  if message.from_user.id == int(admin_id):
#     bot.send_message(message.from_user.id, text = u'Вы в root!!!')



#  else:
#     bot.send_message(message.from_user.id, text = u'Пшел отсюда!!!')


@bot.message_handler(commands=['delete_user_black_list'])
def process_get_network_clients_list_commadn(message):
   if message.from_user.id == int(admin_id):
      bot.send_message(message.from_user.id, text = u'Вы в root!!!')


      list1 = ef.getConnections()
      # list2 = ef.getListMacsFromFirewall("\toption name 'Gera'")
      list2 = ef.getListMacsFromFirewall("option name 'Gera'\n")


      print("1:")
      print(list1)
      print("2")
      print(list2)


      newList = []

      for i in list1:
         for j in list2:
            if i[0].upper() == j:
               newList.append(i)

      print(newList)


      # if not (ua.get_count_user()) == 0:
      # listMacsFromFirewall = ef.getListMacsFromFirewall("option name
'Gera'\n")
      listMacsFromFirewall = ef.getConnections()
      bot.send_message(message.from_user.id, text = u'Вы в root!!!')
      bot.send_message(message.from_user.id, text = u'Выберите пользователя
котогрого хотите удалить!!!', reply_markup =
kb.generate_buttons(len(newList), newList))

      # else:
         # bot.send_message(message.from_user.id, text = u'Список
пользователей пуст!!!')

   else:
      bot.send_message(message.from_user.id, text = u'Не давать доступ!!!')


@bot.message_handler()
def process_digt_command(message):
   if access(message):
      if message.text.isdigit():
         global time
         time = int(message.text)
         bot.send_message(message.from_user.id, text = u'Выберите измерение
времени!', reply_markup=kb.time_format)
      else:
         bot.send_message(message.from_user.id, text = u'Повторите ввод')
```

```
   else:
      bot.send_message(message.from_user.id, text = u'Вы не авторизованы.')


@bot.callback_query_handler(lambda call: True)
def callback_worker(call):
   global new_user_id
   if access(call):
      global time
      check = on.check(0)
      if call.data == '1hour':
         if check == 0:
            bot.send_message(call.from_user.id, u'Доступ уже включен!')
         else:
            bot.send_message(call.from_user.id, u'Доступ будет включен на 1
час')
            bot.send_message(admin_id, u'Пользователь %s включил доступ на 1
час' % (call.from_user.first_name))
            os.system("/bin/sh /root/telebot/managment/block 1 hours")
      elif call.data == '2hour':
         if check == 0:
            bot.send_message(call.from_user.id, u'Доступ уже включен!')
         else:
            bot.send_message(call.from_user.id, u'Доступ будет включен на 2
часа')
            bot.send_message(admin_id, u'Пользователь %s включил доступ на 2
часа' % (call.from_user.first_name))
            os.system("/bin/sh /root/telebot/managment/block 2 hours")
      elif call.data == '3hour':
         if check == 0:
            bot.send_message(call.from_user.id, u'Доступ уже включен!')
         else:
            bot.send_message(call.from_user.id, u'Доступ будет включен на 3
часа')
            bot.send_message(admin_id, u'Пользователь %s включил доступ на 3
час' % (call.from_user.first_name))
            os.system("/bin/sh /root/telebot/managment/block 3 hours")
      elif call.data == 'other':
         if check == 0:
            bot.send_message(call.from_user.id, u'Доступ уже включен!')
         else:
            bot.send_message(call.from_user.id, u'Введите число')
      elif call.data == 'minutes':
         if check == 0:
            bot.send_message(call.from_user.id, u'Доступ уже включен!')
         else:
            bot.send_message(call.from_user.id, u'Доступ будет включен на ' +
str(time) + u' минут(ы)')
            bot.send_message(admin_id, u'Пользователь %s включил доступ на %d
минут(ы)' % (call.from_user.first_name, time))
            os.system("/bin/sh /root/telebot/managment/block " + str(time) +
"minutes")

      elif call.data == 'hours':
         if check == 0:
            bot.send_message(call.from_user.id, u'Доступ уже включен!')
         else:
            if time >= 24:
               bot.send_message(call.from_user.id, u'Доступ нельзя включить
больше чем на 24 часа')
            else:
               bot.send_message(call.from_user.id, u'Доступ будет включен на
' + str(time) + u' часа(ов)')
```

```
                bot.send_message(admin_id, u'Пользователь %s включил доступ на
%d часа(ов)' % (call.from_user.first_name, time))
                os.system("/bin/sh /root/telebot/managment/block " + str(time)
+ "hours")
      elif call.data.split(' ')[0] == 'add_approve':
         ua.add_user(str(call.data.split(' ')[1]), call.data.split(' ')[2])
         bot.send_message(call.data.split(' ')[1], u'Доступ предоставлен!!!')
         bot.send_message(call.from_user.id, u'Доступ предоставлен
пользователю %s (id:%s)!!!' % (str(call.data.split(' ')[1]),
call.data.split(' ')[2]))
         new_user_id = 0
         new_user_first_name = ""
      elif call.data.split(' ')[0] == 'add_decline':
         bot.send_message(call.data.split(' ')[1], u'Отказано в доступе!!!')
      elif call.data.split(' ')[0] == 'del':
         id_del = call.data.split(' ')[1]
         user_del = call.data.split(' ')[2]
         ua.del_user(id_del)
         bot.send_message(admin_id, u'Пользователь %s (id:%s) - УДАЛЕН' %
(user_del, id_del))
         bot.send_message(id_del, u'Пользователь %s (id:%s) - УДАЛЕН' %
(user_del, id_del))
      elif call.data.split(' ')[0] == 'ssh_approve':
         if check_add_ssh_pub() == 'True\n':
            bot.send_message(admin_id, u'SSH ключ пользователя %s (id:%s)
добавлен' % (call.data.split(' ')[2], call.data.split(' ')[1]))
            bot.send_message(call.data.split(' ')[1], u'Ваш ключ добавлен и
Вам предоставлен доступ')
         else:
            bot.send_message(admin_id, u'Ключ не верный')
            bot.send_message(call.data.split(' ')[1], u'Ключ не верный')
      elif call.data.split(' ')[0] == 'ssh_decline':
         bot.send_message(admin_id, u'Пользователь %s (id:%s) отказано в
добавлении SSH Ключа' % (call.data.split(' ')[2], call.data.split(' ')[1]))
         bot.send_message(call.data.split(' ')[1], u'Вам отказано в
добавлении SSH Ключа')
   else:
      bot.send_message(message.from_user.id, text = u'Вы не авторизованы.')


if __name__ == '__main__':
   bot.polling(none_stop = True, interval = 0)
```