Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра автоматизации вычислительных комплексов

Ермакова Татьяна Ивановна

Метод и средства передачи данных в реальном времени в программно-конфигурируемых сетях

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Научный руководитель:
к.ф.-м.н с.н.с
В.В.Балашов

Москва 2019

Аннотация

В данной работе реализован предложенный автором подход к передаче данных в условиях реального времени в программно-конфигурируемых сетях (ПКС). Подход основан на схеме управления трафиком в ПКС, описанной в предыдущих работах автора, и предложенном в данной работе алгоритме реконфигурации систем виртуальных каналов. Реализация подхода выполнена в виде программного средства, выполняющего функции мониторинга сети и её реконфигурации в случае возникновения сбоев с использованием предложенных алгоритма и схемы. Программное средство представляет собой приложение для ПКС-контроллера RUNOS. Проведённое экспериментальное исследование продемонстрировало работоспособность предложенного подхода в условиях реального времени.

Содержание

Введение

Специфика передачи данных в вычислительных системах реального времени (ВС РВ) заключается в повышенных требованиях к надежности и наличии жестких ограничений на время выполнения обмена сообщениями между абонентами. При построении ВС РВ предпочтение отдается коммутируемым средам обмена, которые в отличие от каналов вида точка-точка позволяют сократить число физических кабелей и гибко распределить пропускную способность.

Для избегания конфликтов между потоками данных в коммутируемых средах обмена используется механизм виртуальных каналов. Для удовлетворения потребностей ВС РВ при помощи этого механизма к виртуальным каналам предъявляются требования на ограничение следующих характеристик:

- пропускной способности;
- задержки передачи данных;
- флуктуации задержки (джиттера).

Выполнение данных требований обеспечивается назначением виртуальным каналам ряда параметров, состав которых зависит от используемой схемы построения ВС РВ. Параметры виртуальных каналов в совокупности с их маршрутами образуют конфигурацию системы. Существует несколько стандартов построения коммутируемых сетей ВС РВ на основе виртуальных каналов:

1. Avionics Full Duplex Ethernet (AFDX) [1].
2. FC-AE-ASM-RT на основе Fibre Channel [2].

Недостатком данных решений является статичность системы виртуальных каналов. Существующие стандарты предусматривают наличие ограниченного набора заранее заданных конфигураций, изменение которого невозможно в ходе работы ВС РВ. Переключение между такими конфигурациями не дает возможности гибко реагировать на сбои в работе системы.

Решением данной проблемы могут служить программно-конфигурируемые сети (ПКС) [3]. ПКС является перспективным видом компьютерных сетей, главной особенностью которых являются широкие возможности реконфигурации. Использование ПКС

для передачи данных в реальном времени даст возможность производить расчет и применение новых конфигураций сети в ходе работы системы. Это позволит перераспределять потоки данных при возникновении сбоев или смене режима работы. На текущий момент не существует готовых решений, допускающих использование ПКС в составе ВС РВ при контроле полного состава параметров управления трафиком, принятых для сетей на основе виртуальных каналов.

Данная работа посвящена созданию методов и средств, позволяющих использовать ПКС в системах реального времени с учётом предъявляемых требований качества обслуживания.

1 Постановка задачи

Целью работы является разработка подхода к использованию программно-конфигурируемых сетей в составе вычислительных реального времени. В основу подхода положена ранее разработанная автором схема управления потоками данных в ПКС на основе виртуальных каналов. Для достижения этой цели должны быть решены следующие задачи:

1. Разработка алгоритма динамической реконфигурации виртуальных каналов.
2. Реализация приложения для контроллера ПКС, выполняющего функции мониторинга сети и её реконфигурации при выявлении отказов.
3. Экспериментальное исследование полученного решения по критерию работоспособности в условиях реального времени.

2 Управление трафиком в ВС реального времени с использованием виртуальных каналов

2.1 Схема управления трафиком

В основе подхода лежит согласованная работа оконечных систем (абонентов) и коммутаторов:

- Оконечные системы при помощи планировщика обеспечивают корректную выдачу данных в сеть. Планировщик формирует трафик для виртуальных каналов и мультиплексирует их для выдачи на физическую линию.
- Коммутатор осуществляет контроль трафика при помощи алгоритма текущего ведра независимо для каждого виртуального канала. При превышении заданной пропускной способности кадры данного канала начинают сбрасываться.

Для алгоритма текущего ведра вводится ряд параметров виртуального канала:

- BAG – минимальный интервал времени вежду началами выдачи последовательных кадров на данном виртуальном канале.
- L_{max} – максимальная длина передаваемого кадра.
- J_{max} – джиттер, максимальная задержка от начала временного слота до начала передачи кадра.

Для контроля трафика вводится кредит на количество передаваемых байт, максимальное значение которого равно $AC_{max} = L_{max}(1 + \frac{J_{max}}{BAG})$. Значение кредита увеличивается с течением времени пропорционально величине $\frac{L_{max}}{BAG}$.

При поступлении кадра (или пакета) значение кредита уменьшается на его размер в случае, если это возможно. Иначе кадр (пакет) сбрасывается. Такая схема позволяет контролировать пропускную способность и джиттер. Работа алгоритма показана на рисунках 1 и Рис. 2.

2.2 Реализации схемы управления трафиком

В данном разделе описываются индивидуальные особенности следующих схем управления трафиком:

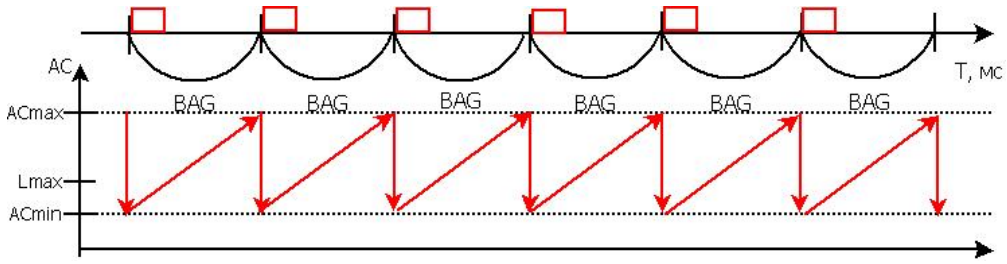

Рис. 1: Работа алгоритма текущего ведра при нулевом джиттере.

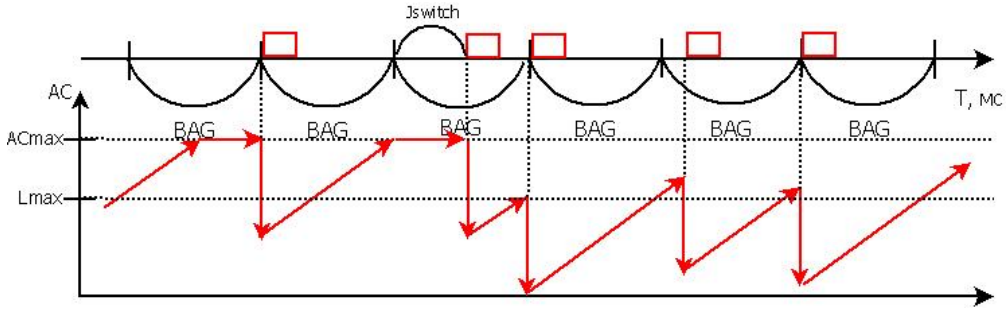

Рис. 2: Работа алгоритма текущего ведра при ненулевом джиттере.

- AFDX.
- FC-AE-ASM-RT.
- Предложенная автором схема управления трафиком в ПКС.

Стандарты AFDX и FC-AE-ASM-RT определяют построение сетей ВС РВ на основе виртуальных каналов. Маршрутизация в таких сетях производится коммутаторами при помощи статических таблиц, которые настраиваются заранее. Контроль трафика осуществляется с использованием алгоритма текущего ведра на уровне кадров. Алгоритм встроен в специализированные коммутаторы этих сетей.

В ПКС маршрутизация производится коммутаторами на основе динамических таблиц, задаваемых контроллером. Предложенная автором схема основывается на механизме meter-таблиц [4], показанных на рисунке 3 и являющихся частью стандарта OpenFlow1.3 [5]. Такие таблицы также управляются контроллером и могут быть изменены в процессе работы системы. Каждая запись meter-таблицы задаёт измеритель, контролирующий скорость привязанных к нему потоков. Схема, по которой осуществляется данный контроль, не закреплена в стандарте и зависит от конкретных реализаций.

Программный коммутатор Ofsoftswitch13 [6] реализует контроль трафика измерителями при помощи алгоритма текущего ведра и поэтому лег в основу схемы управления трафиком в ПКС.

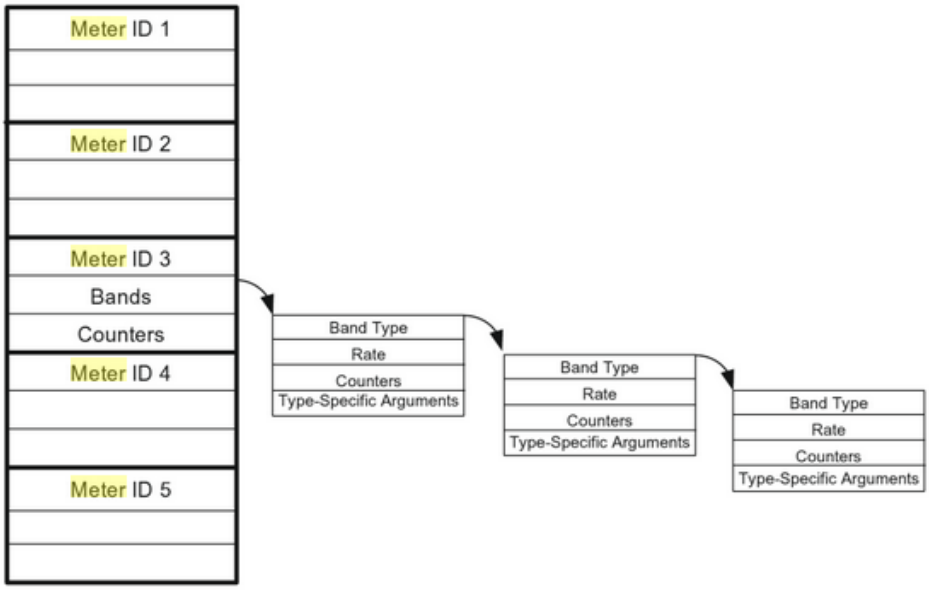

Рис. 3: Структура meter-таблицы.

Схема управления трафиком:

- Для каждого потока задается свой измеритель, на который средствами OpenFlow, посылается максимальное значение кредита и скорость его роста.
- Проверка допустимости трафика производится при помощи алгоритма текущего ведра отдельно для каждого потока на уровне пакетов.

2.3 Сравнение реализаций по критерию поддержки реконфигурируемости

Конфигурацией сети на основе виртуальных каналов назовем набор параметров виртуальных каналов в совокупности с их маршрутами. Каждой конфигурации соответствует набор таблиц маршрутизации, определяющих действия коммутаторов по передаче данных и контролю трафика в соответствии с данной конфигурацией.

Стандарт AFDX предусматривает наличие одной фиксированной таблицы маршрутизации на каждом коммутаторе. Такой набор таблиц маршрутизации соответствует единственной доступной в сети AFDX конфигурации.

Стандарт FC-AE-ASM-RT имеет возможность поддерживать на коммутаторах несколько таблиц маршрутизации, задаваемых до начала работы системы. Это позволяет задать несколько конфигураций сети. Переход между такими конфигурациями осуществляется при помощи контроллера конфигураций. Им является дополнительная оконечная станция. При смене конфигурации контроллер сообщает коммутаторам

сети идентификатор новой конфигурации. На каждом коммутаторе находится внутренняя оконечная станция, обрабатывающая информацию о смене конфигурации. Так как контроллером передается только идентификатор, весь набор конфигураций должен быть заложен в коммутаторы до начала работы системы. На время процесса перехода, который может длиться до 40 мс, все коммуникации в системе временно приостанавливаются, так как происходит полная смена таблиц коммутации. Таким образом процесс смены конфигурации нарушает циркуляцию трафика всех потоков данных системы.

В ПКС таблицы коммутаторов управляются контроллером при помощи сообщений, регламентируемых протоколом OpenFlow. В предложенной автором схеме модификация параметров виртуальных каналов производится сообщениями FlowMod и MeterMod протокола OpenFlow1.3. За счёт этого таблицы маршрутизации, равно как и meter-таблицы, могут быть изменены любым образом в ходе работы системы. Это позволяет создавать множество конфигураций, размер которого ограничен лишь возможностями физической среды передачи данных. Процесс изменения таблиц коммутаций не затрагивает абсолютно все, содержащиеся в них правила. Это даёт возможность не нарушать движение трафика по виртуальным каналам, маршруты которых не проходили через элементы сети, попавшие под сбой. Для виртуальных каналов, пути следования которых были подвержены поломке, имеется возможность сохранить часть пакетов в случае совпадения участков старого и нового маршрутов.

Динамическая реконфигурация сетей ВСРВ требуется в случаях:

1. смены режимов работы;
2. выхода из строя вычислителей.

Как было показано выше стандарт AFDX не поддерживает реконфигурацию в ходе работы системы. Возможностей же FC-AE-ASM-RT недостаточно для произведения гибкой реконфигурации, так как все сбойные режимы должны быть заранее заложены в у. Для единичного отказа число таких режимов является достаточно большим и оценивается количеством физических элементов сети. Расчет всех случаев множественного отказа является ещё более нетривиальной задачей, а хранение всех таких конфигураций в памяти коммутаторов не представляется возможным. Использование ПКС в ВС РВ позволяет производить расчет и применение новой конфигурации сети непосредственно при возникновении поломки. Такой вариант реконфигурации позволяет системе частично продолжать функционировать даже в процессе перехода. Всё это

даёт возможность гибко реагировать на множественные сбои и восстанавливать работу системы в полном объеме в тех случаях, когда это позволяют физические ресурсы.

3 Актуальность задачи

Динамическая реконфигурация сети востребована в ВС РВ. Возможность перераспределить потоки данных в ходе работы системы необходима при возникновении сбоев или желании заложить несколько режимов работы в систему.

В разделе 2 было продемонстрировано, что в ПКС имеются возможности для гибкой реконфигурации сети при соблюдении требований к качеству обслуживания, принятых в сетях ВС РВ. Для осуществления реконфигурации в случае возникновения сбоев необходимо:

1. Определить, в какой части сети произошел сбой.
2. Произвести расчет новой конфигурации с учетом вышедших из строя элементов сети.
3. Произвести переход на полученную конфигурацию.

Данная работа посвящена вопросу идентификации сбоев и алгоритму расчета новой конфигурации сети. Процедура смены конфигураций была ранее реализована и апробирована автором на предмет пригодности в рамках подтверждения работоспособности схемы управления трафиком в ПКС.

4 Алгоритм реконфигурации виртуальных каналов

4.1 Общая схема алгоритма

В процессе разработки алгоритма реконфигурации был изучен ряд работ, посвященных построению маршрутов систем виртуальных каналов addlinks. Во всех перечисленных работах формирование конфигурации ведётся при помощи последовательного применения процедуры поиска маршрута к каждому из виртуальных каналов с использованием жадных стратегий.

В основу предлагаемого алгоритма лёг описанный в работе addlinkВдовин подход к построению систем виртуальных каналов для сетей AFDX. Однако этот подход не может быть позаимствован целиком, так как применяется для начальной инициализации сети и не удовлетворяет ограничениям, накладываемым в данной работе. Разрабатываемый алгоритм реконфигурации систем виртуальных каналов функционирует в условиях реального времени, а потому должен быть направлен на минимизацию:

- времени расчёта новой конфигурации;
- времени применения новой конфигурации в сети, а значит и количества изменений в таблицах коммутации;
- числа пакетов, потерянных в процессе реконфигурации.

Для достижения этих целей алгоритм разделён на два этапа – базовый и дополнительный. На базовом этапе рассматриваются только те виртуальные каналы, которые необходимо переложить, так как они проходят через сбойные элементы сети. В случае если невозможно произвести реконфигурацию, рассматривая только это множество виртуальных каналов, для восстановления работы сети производится дополнительный этап, на котором затрагивается более широкое множество виртуальных каналов. Общая схема алгоритма показана на рисунке 4

Применение новых маршрутов в сети осуществляется только после успешного завершения работы алгоритма. Поведение приложения реконфигурации при неуспешном завершении алгоритма в рамках данной работы не рассматривается. Однако предполагается, что в этом случае будет предпринята попытка переложить часть виртуальных каналов в зависимости от их приоритета. Приоритет виртуального канала в такой модели будет определяться исходя из критичности наличия потока данных для работы системы.

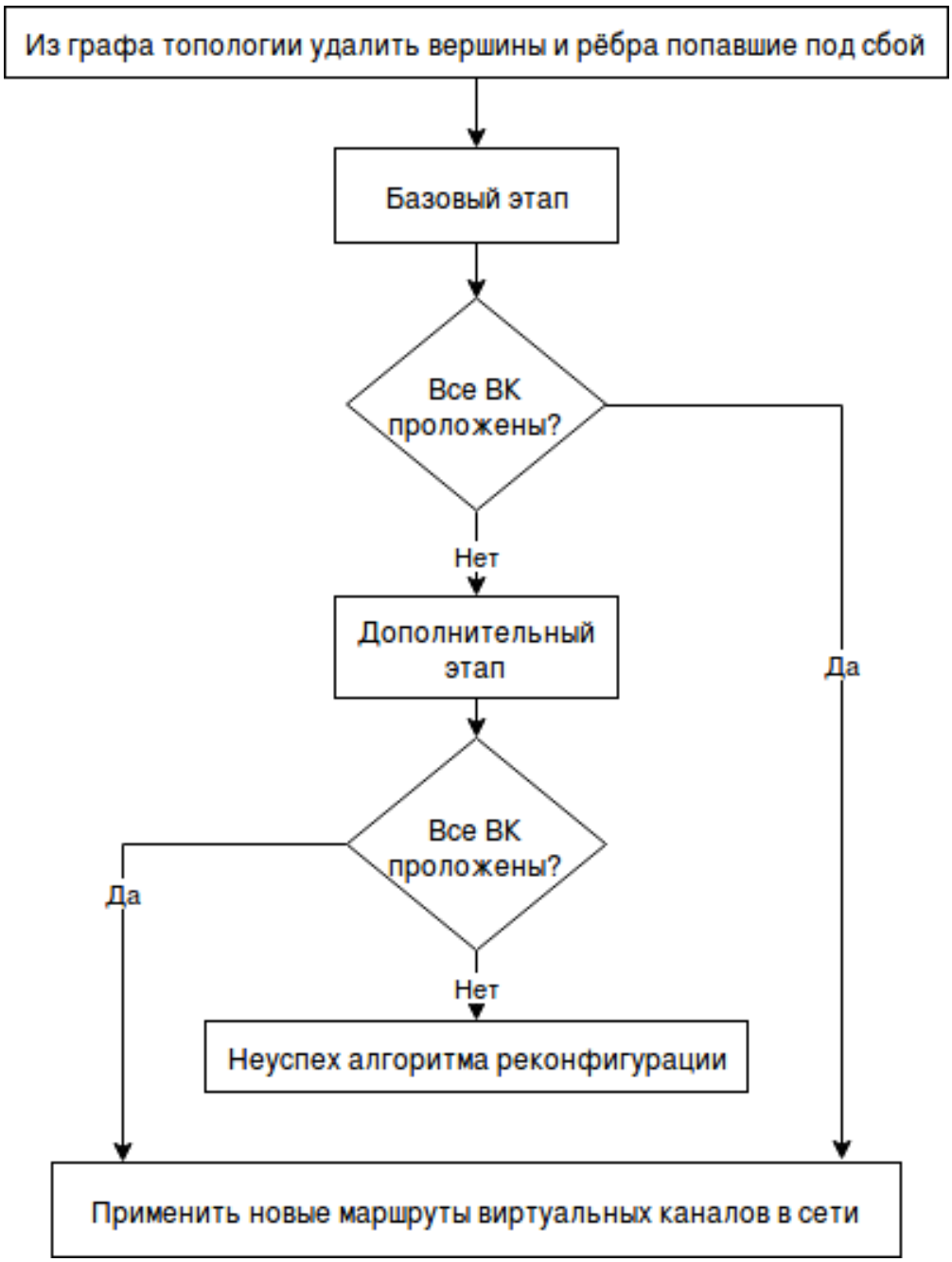

Рис. 4: Общая схема алгоритма реконфигурации.

4.2 Процедуры поиска нового маршрута виртуального канала

В основе каждого этапа алгоритма реконфигурации лежит поиск нового маршрута виртуального канала, который может осуществляться несколькими способами. Опишем эти способы прежде, чем приступить к описанию этапов алгоритма.

Введём обозначения:

- S – узел-отправитель для данного виртуального канала.
- R – узел-получатель для данного виртуального канала.

- $Node_S$ – один из узлов, между которыми прервалась связь, ближний к S.
- $Node_R$ – один из узлов, между которыми прервалась связь, ближний к R.

Процедуры поиска нового маршрута виртуального канала основываются на жадных стратегиях. Их отличие между собой заключается в глубине и скорости поиска. На базовом этапе алгоритма будет выполнен более быстрый вариант процедуры, так как цель базового этапа попытаться произвести реконфигурацию за как можно более короткий срок.

Первый вариант процедуры поиска:

1. Используя алгоритм Дейкстры addlink, найти кратчайший путь между $Node_S$ и R.
2. Добавить в него путь от S к $Node_S$.
3. Удалить циклы.

Такая процедура поиска позволяет сохранить участок пути от S к $Node_S$, когда это возможно. За счёт этого часть пакетов, отправляемых в процессе реконфигурации, не будет потеряна и будет перенаправлена по новому маршруту от $Node_S$ к R. Процедура направлена на минимизацию потерь, возникающих во время построения и применения новой конфигурации.

Второй вариант процедуры поиска:

1. Используя алгоритм k-кратчайших путей addlink, найти кратчайший путь между S и R.
2. Выбрать путь по критерию совпадения наибольшего со старым маршрутом.

Второй вариант процедуры даёт возможность найти большее количество альтернативных маршрутов. Для ограничения глубины поиска было выбрано значение $k = 3$. Ограничение на сохранение участка пути от $Node_S$ к R снимается, но при этом выбор итогового маршрута зависит от исходного маршрута виртуального канала, что позволяет минимизировать число изменений в таблицах коммутации.

4.3 Описание базового этапа алгоритма

На данном этапе производится попытка найти новые маршруты для виртуальных каналов, проходивших через сбойные элементы сети. Остальные виртуальные каналы

при этом не затрагиваются. Такой подход при успешном завершении данного этапа позволяет произвести реконфигурацию с минимальными изменениями в сети. Ограниченный набор виртуальных каналов и быстрая процедура поиска нового маршрута дают возможность минимизировать время работы базового этапа.

Обозначим за (A) подмножество всех виртуальных каналов, проходивших через отказавший элемент сети. Пропускная способность вычисляется из параметров виртуального канала, описанных в разделе 2 и равна:

$$bw = \frac{L_{max}}{BAG}$$

Базовый этап алгоритма реконфигурации:

1. Для каждого виртуального канала из набора (A) в порядке убывания пропускной способности выполнить:
    1.1. Удалить все ребра, на которых не хватает пропускной способности для данного виртуального канала.
    1.2. Запустить первый вариант процедуры поиска нового маршрута виртуального канала, описанный в разделе 4.2.
    1.3. В случае неуспеха перейти к пункту 2.
    1.4. Актуализировать значения пропускных способностей в графе сети с учётом построенного маршрута.
2. Завершить этап.

4.4 Описание дополнительного этапа алгоритма

На данном этапе задействуется более широкий набор виртуальных каналов, чем на базовом этапе. Сначала осуществляется перепрокладка всех виртуальных каналов, проходивших через сбойные элементы сети. Это позволяет изменить ситуацию в сети и найти новые маршруты для сломанных виртуальных каналов. Более глубокая процедура поиска нового маршрута, используемая на данном этапе также способствует вычислению доступной конфигурации сети. Целью данного этапа является осуществить реконфигурацию, несмотря на то, что провал базового этапа показал сложность данной процедуры.

Пусть bw_{max} – максимальная пропускная способность виртуальных каналов из множества (A). Обозначим за (B) подмножество всех виртуальных каналов, которые не входят в (A) и пропускная способность которых не больше bw_{max}.

Базовый этап алгоритма реконфигурации:

1. Для каждого виртуального канала из набора (A) в порядке убывания пропускной способности выполнить:
    1.1. Удалить все ребра, на которых не хватает пропускной способности для данного виртуального канала.
    1.2. Запустить второй вариант процедуры поиска нового маршрута виртуального канала, описанный в разделе 4.2.
    1.3. В случае неуспеха перейти к пункту 3.
    1.4. Актуализировать значения пропускных способностей в графе сети с учётом построенного маршрута.
2. Для каждого виртуального канала из набора (B) в порядке убывания пропускной способности выполнить:
    2.1. Удалить все ребра, на которых не хватает пропускной способности для данного виртуального канала.
    2.2. Запустить второй вариант процедуры поиска нового маршрута виртуального канала, описанный в разделе 4.2.
    2.3. В случае неуспеха перейти к пункту 3.
    2.4. Актуализировать значения пропускных способностей в графе сети с учётом построенного маршрута.
3. Завершить этап.

5 Структура и функции приложения реконфигурации

5.1 Структура приложения реконфигурации

Приложение реконфигурации для ПКС-контролера, реализующего предлагаемый подход, должно:

- реализовывать функции мониторинга для обнаружения сбоев;
- реализовывать функции реконфигурации при возникновении сбоев, а именно расчета новой конфигурации и её применения в сети;
- производить расчет новой конфигурации с использованием алгоритма, предложенного в разделе 4;
- применять изменения в сети на основе схемы, описанной в разделе 2.

Общая структура приложения реконфигурации показана на рисунке 5.

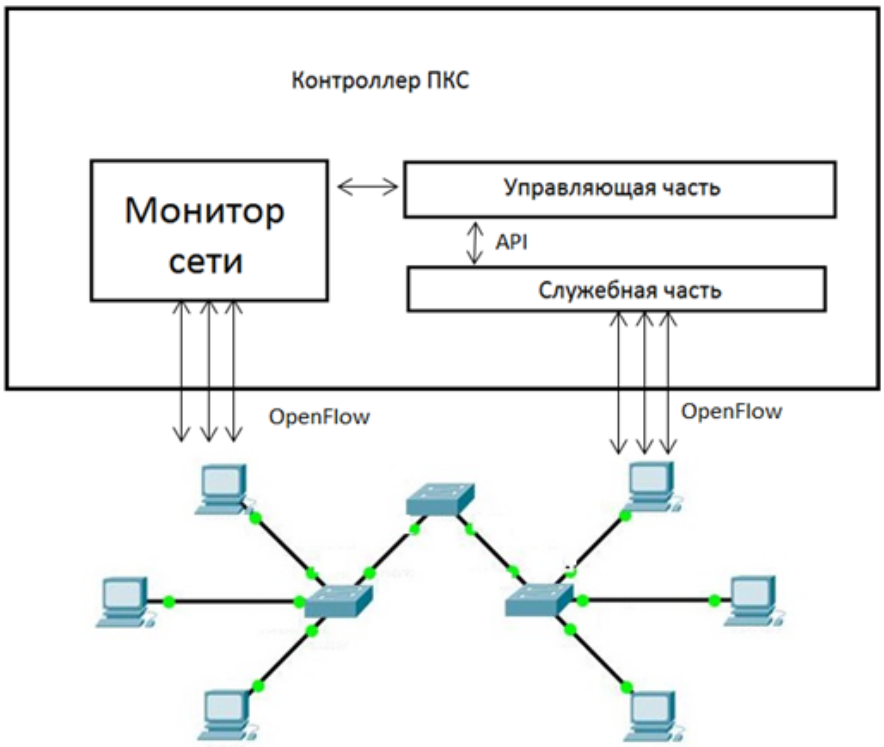

Рис. 5: Структура приложения реконфигурации.

5.2 Описание функций служебной части

Служебная часть реализует в себе описанную в разделе 2 схему управления трафиком в ПКС на основе виртуальных каналов.

Служебная часть должна предоставлять управляющей:

- информацию о потоках данных, которыми обмениваются абоненты сети;
- информацию о проложенных в сети виртуальных каналах и их параметрах;
- интерфейс для добавления/удаления/модификации виртуальных каналов с определенными параметрами качества обслуживания.

5.3 Описание функций управляющей части

Управляющая часть реализует в себе описанный в разделе 4 алгоритм реконфигурации систем виртуальных каналов.

Управляющая часть:

- отвечает за построение начальных маршрутов виртуальных каналов;
- отвечает за построение новых маршрутов виртуальных каналов при выходе из строя компонентов сети;
- использует API служебной части для применения новых маршрутов в сети после завершения работы алгоритма реконфигурации;
- использует информацию, полученную от монитора, для инициализации алгоритма реконфигурации.

5.4 Описание функций монитора

Процедура мониторинга содержит в себе три составляющие:

1. Мониторинг состояния коммутаторов.
2. Мониторинг состояния физических линий связи.
3. Мониторинг состояния оконечных систем.

Мониторинг состояний коммутаторов и физических соединений между коммутаторами производится стандартным для ПКС способом, описанным в addlinks. Ниже приведено краткое описание этих процедур. В силу специфики задачи, а именно необходимости совместной работы отправителей и коммутаторов, работа оконечных систем может быть предопределена. Особенности их функционирования задаются в момент проектирования системы. Поэтому на них может быть возложена функция отправки диагностических пакетов. Это позволяет использовать схему мониторинга состояния оконечных систем, в которой нагрузки на сеть данных ниже в сравнении с со стандартной схемой addlink.

Мониторинг состояния коммутаторов производится на уровне OpenFlow при помощи сообщений EchoReq и EchoRes. Раз в период коммутатор и контроллер обмениваются ими для подтверждения живости друг друга.

Мониторинг состояния оконечных систем производится по следующей схеме:

1. Оконечная система раз в период посылает пакет на ближайшие коммутаторы.
2. На коммутаторах инициируется сообщение Packet-in, сообщающее контроллеру о поступлении неизвестного пакета.
3. Контроллер не получив раз в период ни одного Packet-in, в котором в поле отправителя содержится идентификатор данной оконечной системы, считает её отключённой.

Описание мониторинга состояния физических линий можно разделить на две составляющие:

- мониторинг линий между коммутаторами;
- мониторинг линий между коммутаторами и оконечными системами.

Любой тип физической линии считается вышедшим из строя в случае если с коммутатора поступило сообщение, свидетельствующее об отключении соответствующего порта. В рамках OpenFlow для этого используется сообщение PortStatus, его отправка инициируется самим коммутатором при изменении статуса порта.

Мониторинг линии между коммутаторами осуществляется следующим образом:

1. Контроллер отправляет инкапсулированный в Packet-out LLDP-пакет (Link Layer Discovery Protocol addlink) на соответствующий порт коммутатора. В Packet-out

содержится информация о пересылке LLDP-пакета по исследуемой физической линии.

2. Коммутатор отправляет LLDP-пакет по линии на другой коммутатор.

3. Второй коммутатор, получив LLDP-пакет, инкапсулирует его в Packet-In и отправляет контроллеру.

4. Получив обратно пакет, контроллер делает вывод о живости физической линии.

Мониторинг линий между коммутаторами и оконечными системами производится совместно с мониторингом самих оконечных систем. Разница заключается в том, что каждый Packet-in отвечает за свою физическую линию.

Монитор сети реализует описанную схему мониторинга, а также:

- отвечает за построение начальной топологии сети;
- сообщает управляющей части о выходе из строя компонентов сети.

6 Программная реализация

6.1 Выбор базового программного обеспечения сети

6.2 Структура программного средства

Основные программные компоненты служебной части:

- Sla - класс характеристик виртуальных каналов.
- Vl - класс, отвечающий за хранение параметров виртуальных каналов (включая их маршруты). Также формирует набор сообщений для применения соответствующих параметров в сети.
- VlSet - набор виртуальных каналов.

Основные программные компоненты управляющей части:

- Netcontrol - класс, отвечающий за запуск алгоритма реконфигурации. Является классом приложения Runos.
- Algorithm - класс, реализующий алгоритмы построения маршрутов виртуальных каналов.

Основные программные компоненты монитора:

- NetTopology - класс, отображающий текущее состояние топологии сети. Использует вспомогательные классы NetLink, NetSwitch и NetHost для хранения информации о каналах, коммутаторах и абонентах соответственно.
- BandwidthInfo - хранит текущую доступную на физических каналах пропускную способность. Начальные значения задаются в конфигурации приложения.
- HostManager - приложение Runos, анализирующее состояние абонентов. Приложение написано автором работы.
- SwitchManager - приложение Runos, анализирующее состояние коммутаторов. Используется стандартное приложение, входящее в состав контроллера.
- LinkDiscovery - приложение Runos, анализирующее состояние физических каналов. Используется стандартное приложение, входящее в состав контроллера.

7 Экспериментальное исследование

7.1 Цель исследования

Целью исследования является подтверждение характеристик предложенного в работе алгоритма реконфигурации виртуальных каналов, предназначенного для использования в рамках описанного подхода к передаче данных в ПКС в условиях реального времени. Для достижения поставленной цели необходимо:

- провести исследование свойств алгоритма на различных классах входных данных;
- провести апробацию работы алгоритма в составе приложения, реализующего функции реконфигурации и мониторинга.

Свойства алгоритма, подлежащие исследованию:

- успешность и этап завершения;
- скорость работы.

Изучаемые в рамках апробации подхода характеристики системы:

- успешность восстановления трафика в сети после завершения работы алгоритма реконфигурации;
- скорость применения изменений в сети;
- характеристики качества обслуживания для трафика абонентов (задержка и джиттер).

7.2 Классы исходных данных

Параметры для формирования классов данных:

1. Распределение каналов по пропускным способностям.
2. Общая загруженность сети.
3. Топология сети.

Расчёт пропускной способности виртуального канала производится в соответствии с формулой:

$$bw = \frac{L_{max}}{BAG}$$

Выделим типы виртуальных каналов в зависимости от пропускной способности:

- легковесные $(bw = k)$;
- средние $(bw = 2k)$;
- тяжёлые $(bw = 4k)$.

Классы исходных данных, сформированные на основе процентного содержания виртуальных каналов каждого типа, показаны в таблице 1. Данные классы выбраны для исследования случаев преобладания каждого из типов виртуальных каналов. Распределение пропускных способностей входных потоков данных влияет на работу алгоритма, так как поиск альтернативного маршрута виртуальных каналов затрудняется с ростом их пропускной способности.

	Легковесные	Средние	Тяжёлые
1	90%	7%	3%
2	10%	80%	10%
3	33%	34%	33%
4	5%	15%	80%

Таблица 1: Классы данных на основе пропускной способности

Расчёт общей загруженности сети производится при помощи формулы, отражающей утилизацию физических линий в зависимости от количества маршрутов виртуальных каналов, которые могут быть проложены через данную линию.

Для графа сети, в котором ребра соответствуют физическим линиям, введём вес ребра e:

$$p_e = \sum_{vl} \frac{bw_{vl} * k_{vl_e}}{k_{vl}}$$

где

- vl – множество виртуальных каналов;
- bw_{vl} – пропускная способность виртуального канала;
- k_{vl_e} – число кратчайших путей, построенных для данного канала, проходящих через это ребро;

- k_{vl} – количество найденных кратчайших путей для данного виртуального канала.

Тогда общая загруженность сети определяется формулой:

$$p = \max_{e} \frac{p_e}{bw_e}$$

где bw_e – пропускная способность физической линии.

Два класса исходных данных, формирующиеся на основе данного параметра, показаны в таблице 2. Общая загруженность сети влияет на работу исследуемого алгоритма. С ростом загруженности возрастает сложность поиска альтернативных маршрутов для виртуальных каналов.

	Загруженность сети (p)
1	60%
2	80%

Таблица 2: Классы данных на основе загруженности сети

‘

Для апробации выхода из строя различных элементов сети и демонстрации множества вариантов поиска альтернативного маршрута было выбрано три топологии:

1. «Ромб» (Рис. 6).
2. «Двойное резервирование» (Рис. 7).
3. «add Название» (Рис. 8).

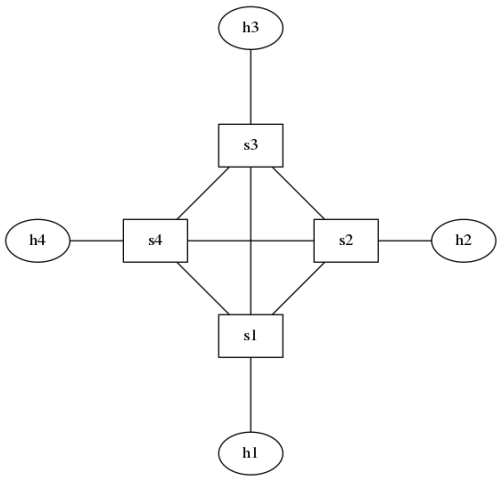

Рис. 6: Топология «Ромб».

В исследовании используется рекомендованное экспертами в области вычислительных реального времени число потоков данных равное 100.

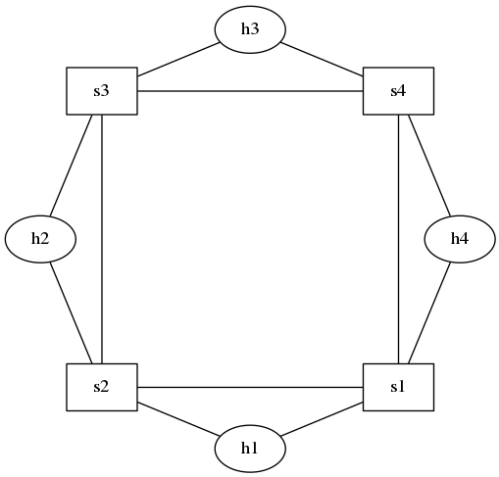

Рис. 7: Топология «Двойное резервирование».

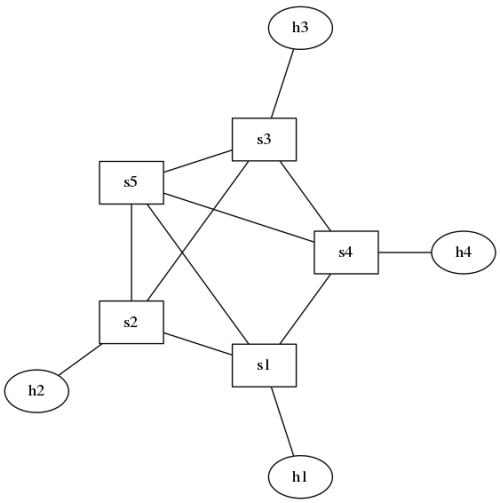

Рис. 8: Топология «add Название».

7.3 Накладываемые ограничения

Процесс реконфигурации сети влечёт за собой потери пакетов, увеличение задержек и джиттеров. Чем быстрее будет производится смена конфигурации, тем меньше эффекта этот процесс окажет на трафик абонентов бортовой системы. В рамках данного экспериментального исследования будем опираться на временные ограничения, принятые в вычислительных системах реального времени. Как упоминалось в разделе 2, в сетях FC-AE-ASM-RT на смену конфигурации отводится 40мс. При этом будем учитывать оценки на время, прошедшее с момента отправки последнего управляющего правила до момента его применения в таблице коммутации:

С учётом вышесказанного в качестве итогового времени, отводимого на реконфигурацию примем addмс.

7.4 Исследование свойств алгоритма

7.4.1 Методика исследования

Исследование свойств алгоритма производится по следующей схеме:

1. Для каждого класса данных генерируется соответствующий ему набор виртуальных каналов.
2. Для каждого набора виртуальных каналов:
   2.1. Запускается приложение, содержащее алгоритм реконфигурации.
   2.2. Запуск инициирует построение начальных маршрутов виртуальных каналов.
   2.3. Собирается статистика по количеству виртуальных каналов, проходящих через каждый элемент сети.
   2.4. Выбирается элемент сети, через который проходит наибольшее число виртуальных каналов, и имитируется его поломка.
   2.5. Поломка инициирует запуск алгоритма реконфигурации.
   2.6. По завершению алгоритм выдаёт информацию об успешности и этапе завершения, а так же времени работы.
3. Предполагается, что время работы алгоритма занимает не более 70% от всего времени, заложенного на реконфигурацию.
4. По полученным характеристикам делается вывод о возможности использования алгоритма в рамках подхода к передаче данных в реальном времени.

7.4.2 Результаты

Характеристики компьютера, на котором производилось исследование:

- Процессор – Intel Core i7, тактовая частота 1.90 ГГц.
- Оперативная память – 4 ГБ.

Результаты исследования свойств алгоритма реконфигурации виртуальных каналов приведены в addПриложение (см. таблицу addТаблица).

Максимальное время работы алгоритма – addмс. Данные показатели времени достигаются, если алгоритм вынужден выполнить дополнительный этап. Следует отметить,

что выполнение дополнительного этапа произошло в add% случаев. Как правило алгоритм завершает работу на базовом этапе. При таком варианте выполнения время завершения алгоритма не превышает addмс. Время работы алгоритма, не превысило 70% от времени, отведённого на реконфигурацию. То, как данная величина соотносится со временем, затрачиваемым на применение полученных изменений в сети, будет продемонстрировано в разделе 7.5.

Алгоритм не смог построить новую систему виртуальных каналов при add. Это объясняется тем, что add.

7.5 Апробация разработанного подхода динамической реконфигурации сети

7.5.1 Конфигурация экспериментальной системы

Система апробации представляет собой виртуальную среду, имитирующую функционирование:

1. Контроллера сети.
2. Набора коммутаторов.
3. Абонентов сети, формирующих потоки данных с заданными характеристиками.

Виртуальная среда разворачивается на основе операционной системы Ubuntu 16.04, на которой установлены следующие компоненты:

1. Runos – ПКС-контроллер [7].
2. Ofsoftswitch13 – программный коммутатор, поддерживающий OpenFlow 1.3 [6].
3. Mininet – средство имитирующее работу сети [8].

7.5.2 Сценарии апробации

Для апробации предложенного подхода к передаче данных в реальном времени был выбран набор сценариев, основывающийся на том, какие элементы сети могут выйти из строя.

Рассматривается выход из строя:

1. Коммутатора.

2. Физической линии, соединяющей два коммутатора.

3. Физической линии, соединяющей абонента с коммутатором.

4. Порта абонента или коммутатора.

Дополнительно к этому рассматриваются сценарии кратных последовательных во времени выходов из строя элементов сети.

Сценарии разрыва физической линии между коммутаторами и выхода из строя соответствующих портов производятся на всех топологиях. Оставшиеся сценарии могут быть выполнены только на топологиях «Двойное резервирование» и «add Название» (см. Рис. 7 и Рис. 8). Отказ портов практически ничем не отличается от отказа соответствующих им физических линий. Поэтому в таблице результатов данные по таким сценариям представлены не будут.

7.5.3 Методика апробации

Апробация работы алгоритма реконфигурации в рамках предложенного подхода состоит из трёх этапов:

1. Формирование набора тестовых прецедентов.

2. Проведение экспериментов на тестовых прецедентах.

3. Анализ результатов.

Тестовые прецеденты формируются выбором:

- сценария апробации;

- класса исходных данных с доступной для выбранного сценария топологией сети.

На этапе проведения экспериментов производится запуск каждого тестового прецедента:

1. Формируется соответствующая классу данных конфигурация сети.

2. Запускается приложение реконфигурации.

3. Запускается виртуальная тестовая среда, имитирующая поведение сети.

4. Запуск приложения инициирует построение начальных маршрутов виртуальных каналов.

5. Запуск тестовой среды инициирует отправку и прием потоков данных абонентами, а так же сбор статистики по этим потокам данных.

6. Средствами тестовой среды запускается сценарий апробации.

7. Соответствующая сценарию поломка инициирует запуск алгоритма реконфигурации.

8. По завершению алгоритм выдаёт информацию о времени, затраченном на применение изменений в сети.

9. Происходит сворачивание тестовой среды, инициирующее обработку собранной статистики по потокам данных.

На этапе анализа результатов для каждого тестового прецедента:

1. Проверяется факт восстановления трафика по завершению выполнения сценария.

2. Анализируется время, затраченное на применение изменений в сети.

3. Анализируется число потерянных в ходе реконфигурации пакетов.

4. Задержки и джиттеры, полученные в период реконфигурации сравниваются с эталонными задержками, полученными в штатном режиме.

5. Делается вывод об успешности работы приложения реконфигурации в рамках предложенного сценария.

7.5.4 Результаты

Результаты апробации работы алгоритма в рамках приложения, реализующего подход к передаче данных в условиях реального времени в ПКС, приведены в addПриложение (см. таблицу addТаблица).

Джиттеры и задержки потоков данных, виртуальные каналы которых участвовали в реконфигурации, отличались от эталонных не более, чем на addмс и addмс соответственно. Данное отклонение лежит в рамках допустимого. Джиттеры и задержки прочих потоков данных не изменились.

Максимальное время затраченное на процесс реконфигурации удовлетворяет наложенным в разделе 7.3 ограничениям. Потери пакетов составили add. При этом потоки,

маршруты виртуальных каналов которых не проходили через сбойные элементы, затронуты не были.

Отказы от построения новых виртуальных каналов были получены на тех же классах данных, что и при исследовании свойств алгоритма.

Было проверено несколько сценариев кратного последовательного во времени выхода из строя элементов сети. Максимальное число последовательных поломок, после которых восстанавливалась работоспособность системы, составило add. Отказы от восстановления работы сети после очередной поломки были обусловлены исключительно отсутствием путей между абонентами или нехваткой пропускных способностей.

7.6 Выводы

Проведенное экспериментальное исследование продемонстрировало, что:

- основные характеристики алгоритма соответствуют ожидаемым;
- алгоритм успешно функционирует в рамках приложения реконфигурации, работающего в виртуальной среде, имитирующей сеть ПКС;
- соблюдаются ограничения на характеристики качества обслуживания абонентских потоков данных при реконфигурации сети с использованием предложенного алгоритма.

Исследование и апробация показали, что предложенный алгоритм реконфигурации виртуальных каналов способен работать в составе приложения, реализующего подход к передаче данных в реальном времени в ПКС. Алгоритм успешно осуществляет поиск новых маршрутов для виртуальных каналов при выходе из строя различных элементов сети. Апробация показала работоспособность разработанного приложения, осуществляющего функции реконфигурации и мониторинга, и реализуемого им подхода. Время, затраченное на осуществление реконфигурации, не превосходит ограничений, заданных в существующих системах реального времени.

Заключение

текст

Список литературы

[1] AFDX / ARINC 664 Tutorial (1500-049) // Condor Engineering, Inc. – 2005.

[2] Fibre Channel Arbitrated Loop (FC-AL) // working draft proposal American National Standard for Information Technology. 1995. 98 p.

[3] Software-Defined Networking: The New Norm for Networks // Open Networking Foundation. – 2012. [PDF] (https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf)

[4] Efimushkin, T. Ledovskikh, D. Korabelnikov, D. Iazykov. Perfomance assurance in Software-Defined Networks // « Intellect Telecom» JSC.

[5] ONF OpenFlow Switch Specification, Version1.3.0 [PDF] (https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf)

[6] CPqD F. Openflow 1.3 software switch [HTML] (http://www.cpqd.github.io/ofsoftswitch13/).

[7] Shalimov A. et al. The Runos OpenFlow Controller // Software Defined Networks (EWSDN), 2015 Fourth European Workshop on. – IEEE, 2015. – C. 103-104.

[8] Oliveira R. L. S. Using mininet for emulation and prototyping software-defined networks // Communications and Computing (COLCOM) – 2014. – P. 1-6