УДК 004.04

[1]Жуков И. А., д-р техн. наук,
[1]Клименко И. А.,
[2]Аленин О. И.

ОРГАНИЗАЦИЯ МНОГОПУТЕВОЙ МАРШРУТИЗАЦИИ СРЕДСТВАМИ MPLS

[1]Институт компьютерных технологий Национального авиационного университета
[2]Национальный технический университет Украины «КПИ»

Предложен модифицированный способ разделения потоков пакетов на граничном узле сети MPLS, и алгоритм управления таблицами FEC на основе этого способа, позволяющий повысить эффективность работы сети MPLS в целом.

Введение

Основными требованиями, предъявляемыми к магистральной сети, являются высокая пропускная способность, малое значение задержки и высокая масштабируемость. Традиционные способы и протоколы маршрутизации, основанные на обработке пакетов на сетевом уровне, не позволяют эффективно решать эти задачи в современных магистральных сетях.

Поэтому в настоящее время используется архитектура многопротокольной коммутации по меткам (*MPLS, MultiProtocol Label Switching*) [1], которая обеспечивает построение магистральных сетей, имеющих практически неограниченные возможности масштабирования, высокую скорость обработки трафика и достаточную гибкость с точки зрения организации дополнительных сервисов и поддержки требуемого качества обслуживания (*QoS*).

В современных компьютерных сетях с высокоскоростными каналами передачи данных задержка при обработке пакетов на маршрутизаторах составляет значительную часть от общего времени задержки при передаче пакетов от отправителя к получателю, и ее уменьшение может существенно повысить производительность сети.

Кроме того, в сетях с многопротокольной коммутацией по меткам основная нагрузка по принятию решений о выборе подходящего пути для пакета ложится на граничный маршрутизатор, и повышение его производительности является важной задачей.

Таким образом, разработка алгоритма распределения трафика на граничном маршрутизаторе сети *MPLS*, учитывающего пропускную способность каналов связи и при необходимости балансируя трафик между несколькими путями, является актуальной задачей.

Многопротокольная коммутация по меткам

В основе архитектуры *MPLS* лежит принцип обмена меток. Передаваемый пакет ассоциируется с тем или иным классом сетевого уровня (*FEC, Forwarding Equivalence Class*), каждый из которых идентифицируется определенной меткой. [2] Значение метки уникально лишь для участка пути между соседними узлами сети *MPLS*, которые называются маршрутизаторами, коммутирующими по меткам (*LSR, Label Switching Router*). Метка передается в составе каждого кадра, причем способ ее привязки к кадру зависит от используемой технологии канального уровня.

Маршрутизатор *LSR* получает топологическую информацию о сети, участвуя в работе алгоритма маршрутизации *OSPF* [3], который используется в качестве алгоритма внутренней маршрутизации в сети *MPLS*. Затем он начинает взаимодействовать с соседними маршрутизаторами, распределяя метки, которые в дальнейшем будут применяться для коммутации. Обмен метками может производиться с помощью как специального протокола распределения меток (*LDP, Label Distribution Protocol*) [4, 5], так и модифи-

цированных версий других протоколов сигнализации в сети (например, незначительно видоизмененных протоколов резервирования ресурсов *RSVP* [6]).

Распределение меток между маршрутизаторами *LSR* приводит к установлению внутри домена *MPLS* путей с коммутацией по меткам (*LSP*, *Label Switching Path*). Каждый маршрутизатор *LSR* содержит таблицу, которая ставит в соответствие паре «входной интерфейс, входная метка» тройку «префикс адреса получателя, выходной интерфейс, выходная метка». Получая пакет, *LSR* по номеру интерфейса, на который пришел пакет, и по значению привязанной к пакету метки определяет для него выходной интерфейс. Значение префикса применяется лишь для построения таблицы и в самом процессе коммутации не используется. Старое значение метки заменяется новым, содержавшимся в поле «выходная метка» таблицы, и пакет отправляется к следующему устройству на пути *LSP*.

Вся операция требует лишь однократной идентификации значений полей в одной строке таблицы. Это занимает гораздо меньше времени, чем сравнение *IP*-адреса отправителя с наиболее длинным адресным префиксом в таблице маршрутизации, которое используется при традиционной маршрутизации.

Сеть *MPLS* делится на две функционально различные области — ядро и граничную область (рис. 1).

К ядру относятся маршрутизаторы *LSR1...LSR6*, а к граничной области – маршрутизаторы *LER1...LER8*. Сети site1…site14 представляют сети или отдельные узлы, не использующие технологию коммутации по меткам, но использующие сеть *MPLS* как опорную (обычно это узлы или сети пользователей).

Ядро образуют устройства, минимальным требованием к которым является поддержка *MPLS* и участие в процессе маршрутизации трафика для того протокола, который коммутируется с помощью *MPLS*. Маршрутизаторы ядра занимаются только коммутацией. Функции классификации пакетов по различным *FEC* классам, а также реализацию таких дополнительных сервисов, как фильтрация, явная маршрутизация, выравнивание нагрузки и управление трафиком, берут на себя граничные *LSR*-маршрутизаторы, называемые также маршрутизаторами *LER* (*Label Edge Router*). В результате интенсивные вычисления приходятся на граничную область, а высокопроизводительная коммутация выполняется в ядре, что позволяет оптимизировать конфигурацию устройств *MPLS* в зависимости от их местоположения в сети.

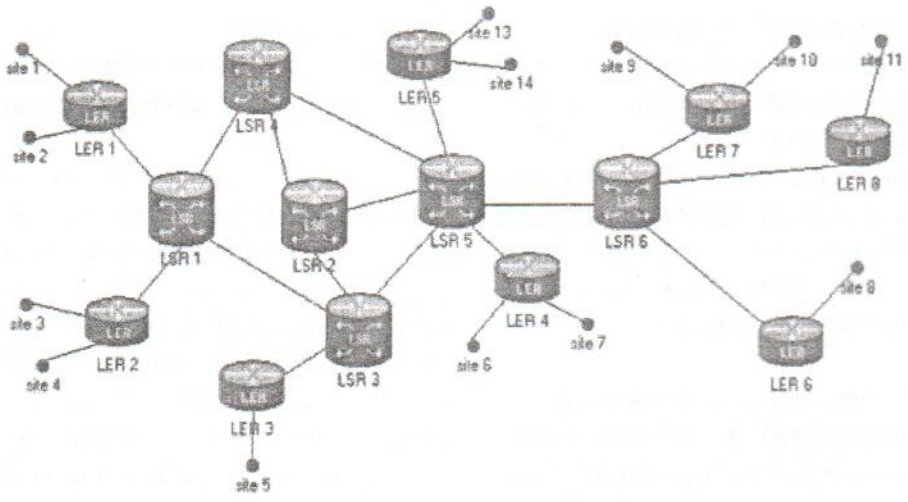

Рис. 1. Сеть *MPLS*

Таким образом, главная особенность технологии *MPLS* — отделение процесса коммутации пакета от анализа *IP*-адресов в его заголовке, что открывает ряд дополнительных возможностей. Очевидным следствием описанного подхода является тот факт, что очередной сегмент *LSP* может не совпадать с очередным сегментом маршрута, который был бы выбран при традиционной маршрутизации.

Поскольку на установление соответствия пакетов определенным *FEC* классам могут влиять не только *IP*-адреса, но и другие параметры, нетрудно реализовать, например, назначение различных *LSP* пакетам, относящимся к различным потокам *RSVP* или имеющим разные приоритеты обслуживания. Конечно, подобный сценарий удается осуществить и в обычных маршрутизируемых сетях, но решение на базе *MPLS* оказывается проще и к тому же лучше масштабируется.

Каждый из классов *FEC* обрабатывается отдельно от остальных — не только потому, что для него строится свой путь *LSP*, но и в смысле доступа к общим ресурсам (полосе пропускания канала и буферному пространству). В результате технология *MPLS* позволяет достаточно эффективно поддерживать требуемое качество обслуживания, не нарушая предоставленных пользователю гарантий. Применение в *LSR* таких механизмов управления буферизацией и очередями, как *WRED*, *WFQ* или *CBWFQ*, обеспечивает возможность контролировать распределение ресурсов и изолировать трафик отдельных пользователей.

Использование явно задаваемого маршрута в сети *MPLS* свободно от недостатков стандартной *IP*-маршрутизации от источника, поскольку вся информация о маршруте содержится в метке и в пакете не требуется хранить адреса промежуточных узлов, что способствует улучшению управления распределением нагрузки в сети.

Как уже отмечалось, метка должна быть уникальной лишь в пределах соединения между каждой парой логически соседних *LSR*. Поэтому одно и то же ее значение может использоваться *LSR* для связи с различными соседними маршрутизаторами, если только имеется возможность определить, от какого из них пришел пакет с данной меткой. Другими словами, в соединениях «точка-точка» допускается применять один набор меток на интерфейс, а для сред с множественным доступом необходим один набор меток на модуль или все устройство.

Перед включением в состав пакета метка определенным образом кодируется [1]. В случае использования протокола *IP* она помещается в специальный «тонкий» заголовок пакета, инкапсулирующего *IP* (рис. 2). Для пакетов протокола *IPv6* [7] метку можно разместить в поле идентификатора потока.

расположение метки в кадре:

MAC заголовок	Метка	*IP* заголовок	Данные

метка:

Значение метки (20 бит)	Class of Service (3 бита)	Признак конца стека (1бит)	TTL (8 бит)

Рис. 2. Формат метки *MPLS*

В рамках архитектуры *MPLS* вместе с пакетом разрешено передавать не одну метку, а целый их стек [1]. Операции добавления/изъятия метки определены как операции на стеке (*push/pop*). Результат коммутации задает лишь верхняя метка стека, нижние же передаются прозрачно до операции изъятия верхней. Такой подход позволяет создавать иерархию потоков в сети *MPLS* и организовывать туннельные передачи.

Преимущества и недостатки MPLS по сравнению с традиционной маршрутизацией

Основные преимущества технологии многопротокольной коммутации по меткам по сравнению с традиционной маршрутизацией заключаются в следующем:

- возможность предоставления широкого спектра дополнительных сервисов при сохранении масштабируемости сети, за счет отделения выбора маршрута от анализа *IP*-адреса;
- увеличение скорости обработки пакетов в узлах, так как не требуется извлекать пакет из кадра, определять *IP*-адрес и производить поиск в таблице маршрутизации наиболее подходящего адресного префикса;
- гибкая поддержка качества обслуживания (*QoS*), интегрированных сервисов и виртуальных частных сетей;
- возможность использовать для принятия решения о маршрутизации не только адреса получателя, но и других данных (например, адреса отправителя, требований к качеству обслуживания и т.д.);
- разделение функциональности между ядром и граничной областью сети, за счет чего может быть упрощена структура маршрутизаторов ядра сети и повышено их быстродействие, кроме того, политика управления трафиком может быть реализована только на граничных маршрутизаторах, что облегчает настройку сети.

К недостаткам следует отнести то, что технология многопротокольной коммутации по меткам предполагает перенос основных функций по определению пути прохождения пакета через сеть на граничные маршрутизаторы, что усложняет их реализацию и приводит к необходимости создания эффективных алгоритмов для обработки и распределения пакетов на этих узлах.

Требования к алгоритму разделения трафика на граничных узлах

Отметим некоторые особенности разрабатываемого алгоритма, и требования, которым он должен соответствовать:

- алгоритм является распределенным алгоритмом балансировки нагрузки;
- не требуется нового аппаратного обеспечения;
- не требуется нового программного обеспечения на промежуточных узлах (узлах *LSR*), изменяется только алгоритм обработки пакетов на граничных узлах сети *MPLS* (узлах *LER*);
- не изменяется протокол обмена метками;
- минимизируется переупорядочивание пакетов *TCP*-соединений;
- матрица трафика не должна быть известна заранее.

При использовании алгоритма предполагается, что существуют несколько установленных явных *LSP* путей (например, вручную или с помощью протокола *RSVP-TE*) между входным узлом и выходным узлом в сети *MPLS*. Такая конфигурация часто встречается в крупных корпоративных сетях или сетях провайдеров, когда между двумя узлами существует более одного пути, возможно с разными метриками. Одной из задач управления трафиком на входящем узле является оптимальное распределения трафика по этим нескольким путям, например такое, что нагрузка на каналы передачи данных будет равномерной.

Облако сети *MPLS* состоит из маршрутизаторов коммутации по меткам и физических каналов связи между ними. В сети может использоваться любой протокол канального уровня, а также любой протокол сетевого уровня. Множество путей *LSP* и маршрутизаторов *LSR* представляют собой виртуальную топологию, организованную поверх физической сети. При этом различные пути *LSP* могут разделять одни и те же физические каналы передачи данных между маршрутизаторами *LSR*. Поэтому при разработке алго-

ритма необходимо учитывать при определении загруженности каналов кроме потоков данных, передающихся непосредственно от выбранного входного узла к выходному узлу, еще и трафик, проходящий через разделяемые каналы и узлы от других входных узлов.

На рис. 3 представлена функциональная схема блока распределения трафика, реализованного на граничном маршрутизаторе.

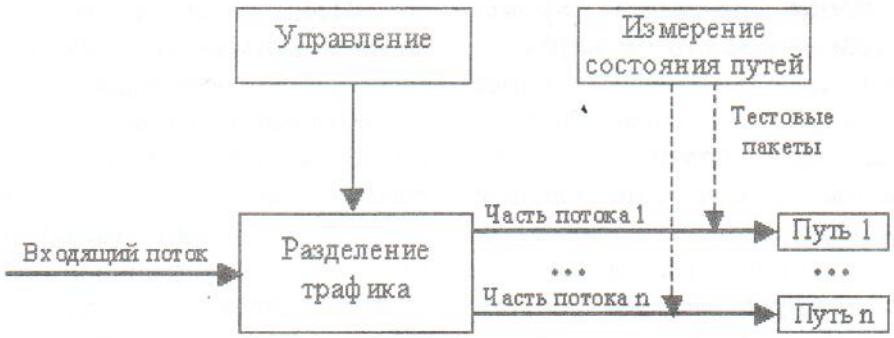

Рис. 3. Функциональная схема блока распределения трафика

Входящий трафик проходит процедуру фильтрации и распределения, целью которой является упрощение переключения трафика среди *LSP* способом, уменьшающим возможности прихода пакетов к получателю не по порядку. Данному механизму не требуется статистика требований трафика или информация о состоянии потока. Процедура распределения трафика определяет, как и когда переключать потоки между маршрутизаторами *LSP*, используя тестовые пакеты. Процедура распределения трафика состоит из двух этапов: этап мониторинга и этап балансировки нагрузки. На этапе мониторинга в случае обнаружения значительного и постоянного изменения состояния сети выполняется переход на этап балансировки. На этапе балансировки алгоритм выравнивает нагрузку среди *LSP*. Как только нагрузка выравнивается, алгоритм переходит на этап мониторинга и весь процесс повторяется.

Способы разделения трафика

С целью обеспечения равномерной загрузки сети используются различные способы распределения трафика. В качестве критерия распределения трафика используется выражение (1) [8].

$$\min_x C(x) = \sum_l C_l(x^l), \qquad (1)$$

при условии $\sum_{p \in P_S} x_{sp} = r_s$ для всех $s \in S$,

где P_S – множество всех путей *LSP*, r_S – поток трафика между входным и выходным узлом, x_{SP} – часть трафика, передаваемая по пути p, $C(x)$ – функция стоимости использования канала.

Сумма потоков данных, проходящих по каналу l, определяется выражением:

$$x^l = \sum_{s \in S} \sum_{l \in p, p \in P_S} x_{SP}. \qquad (2)$$

Способы разделения трафика, поступающего на входной маршрутизатор сети *MPLS*, могут быть различными. Для получения полностью равномерной (или требуемой) загрузки каналов передачи данных, общий поток пакетов необходимо разделить на множество более мелких потоков, вплоть до разбиения на отдельные потоки данных для каждого соединения между прикладными процессами на узле отправителя и узле получателя. Однако такой уровень гранулярности приведет к необходимости поддерживать множество записей в памяти граничного маршрутизатора и существенно увеличит время выбора соответствующего пути

LSP для поступившего пакета. Целесообразным является агрегирование множества потоков в один (например, основываясь на адресе сети отправителя и получателя).

Рассмотрим способы разделения трафика.

Разделение трафика случайным образом на уровне пакетов

Данный способ предусматривает разделение поступающих пакетов для передачи по одному из подходящих путей *LSP* случайным образом. При большой интенсивности поступления пакетов и правильной реализации генератора случайных чисел загрузка каждого из путей (и соответствующих физических каналов передачи) будет равномерной. Главным преимуществом данного способа является его простота, и, как следствие этого, возможность реализации на его основе алгоритма с небольшой временной сложностью. Основным недостатком является то, что пакеты *TCP*-потока, могут быть направлены по разным путям. При этом происходит переупорядочивание пакетов, принадлежащих одному *TCP*-потоку.

Разделение в соответствии с циклической дисциплиной

Этот способ предполагает поочередное направление поступающих пакетов для передачи через подходящие пути *LSP*. Как и при случайном разделении, данный способ прост в реализации и алгоритм на его основе имеет малую временную сложность. Его недостатками является переупорядочивание пакетов одного потока, а также некоторая вероятность неравномерной загрузки каналов передачи данных при определенных условиях (например, при существовании нескольких интенсивных входных потоков с пакетами разной длины, поступающими поочередно).

Еще одним преимуществом двух названных способов является также отсутствие необходимости поддерживать таблицы проверки соответствия пришедшего пакета тому или иному потоку.

Разделение на основе адреса отправителя и получателя

Основой данного способа является создание и поддержка в памяти маршрутизатора таблиц соответствия входящих потоков и предпочитаемых путей *LSP* для них.

Преимуществом данного способа является возможность достаточно гибко распределять входящий трафик между несколькими подходящими путями *LSP*, соблюдая при этом условие пересылки пакетов одного соединения по одному *LSP*, а недостатками - высокие требования к объему памяти граничного маршрутизатора и снижение его быстродействия при поиске соответствующего пакету класса *FEC* в большой таблице. Кроме того, при некоторых условиях (например, при установке между некоторой парой узлов нескольких *TCP*-соединений, пакеты по которым передаются с большой интенсивностью) данный способ не позволяет оптимально распределить нагрузку.

Разделение на основе адреса отправителя и получателя, и номеров портов

Этот способ является модификацией предыдущего, но для идентификации потоков используются не только адреса отправителя и получателя, а еще и номера портов (в том случае, если установлено *TCP*-соединение). С одной стороны, это приводит к еще большим требованиям к объему памяти и снижению быстродействия (вследствие увеличения размеров таблиц), что является существенным недостатком, с другой стороны, появляется возможность более гибко распределить нагрузку. Как и в предыдущем способе, при его применении переупорядочивания пакетов не происходит.

Разделение по значению hash-функции от заголовка пакетов

Данный способ предполагает вычисление значения *hash*-функции, используя в качестве аргумента некоторые поля *IP*-пакета (например, адрес отправителя, получателя и номера портов) [8]. Как и в предыдущем способе, необходимо соз-

дать и постоянно обновлять таблицы в памяти маршрутизатора (в которых содержатся значение *hash*-функций и номера соответствующих предпочитаемых путей *LSP*), но размер их можно выбрать в соответствии с объемом доступной памяти и требуемым быстродействием. Это решение является компромиссом между возможностью гибко распределять потоки между путями *LSP* и требованием к быстродействию граничного маршрутизатора, а также требуемым объемом памяти. Так как в качестве аргументов *hash*-функции выбираются поля заголовка пакета, которые постоянны для одного *TCP*-соединения, переупорядочивания пакетов при использовании этого способа не происходит.

Предложенный способ разделения трафика на граничных узлах

В рассмотренных выше способах разделения потоков для обеспечения поступления пакетов на исходящий узел в правильном порядке необходимо анализировать значение заголовка пакета. Такая процедура выполняется на граничном маршрутизаторе сети *MPLS* для определения класса *FEC* поступившего пакета. Однако при использовании методов разделения потоков пакетов по адресам отправителя и получателя приходится просматривать две таблицы: одна из них устанавливает соответствие между заголовком пакета и классом *FEC*, а другая – между значениями заголовка пакетов и подходящими путями *LSP*. Было бы целесообразно объединить эти две таблицы, но это приводит фактически к определению множества классов *FEC*, вплоть до отдельного класса для каждого *TCP*-потока. Из-за ограниченного объема памяти маршрутизатора, и снижения его производительности при поиске в таблицах с большим количеством записей, этот подход является неприемлемым.

Суть предложенного модифицированного способа состоит в следующем: для хранения записей соответствия FEC и *LSP* используется модифицированная таблица (табл. 1). Кроме стандартных полей (*IP*-адреса отправителя и получателя, адреса портов, а также исходящей метки и исходящего интерфейса, определяющего *LSP*), в таблице присутствуют такие поля:

- значение типа протокола (*TCP*, *UDP*);
- счетчик байт данного потока за некоторый промежуток времени (Δt);
- начальное значение *hash*-функции, которая берется от полей *IP*-пакета, одинаковых для одного *TCP*-потока.

Таблица 1. Модифицированная таблица маршрутизации

Protocol	*dst IP/mask*	*src IP/mask*	*src port*	*dst port*	*hash*		*interface*	*label*	*bytes*
					from	*to*			
TCP	195.245.194/24	*any*	*any*	*any*	0	63	1	2	35354
TCP	195.245.194/24	*any*	*any*	*any*	64	127	2	3	32532
TCP	195.245.194/24	*any*	*any*	*any*	128	255	3	3	35325
UDP	12.2.3.1	80.54.1.20	*any*	*any*	0	255	5	1	14143
UDP	62/8	*any*	*any*	80	0	255	1	5	3000
any	61.1.2.3	212.55.5.1	1024	22	0	255	8	2	32320
any	212.11.7.2	*any*	*any*	*any*	0	255	5	2	1500

Для упрощения вычисления *hash*-функции берется сумма по модулю 2^8, таким образом, максимально возможное количество записей для одного предварительно определенного класса *FEC* равняется 256.

В качестве адреса отправителя и получателя может использоваться как адре-

са подсети, так и отдельные узлы. Это позволяет осуществить агрегирование редко используемых комбинаций адресов, объединив их в одну запись.

Специальный формат данной таблицы позволяет использовать как настройку классов *FEC* вручную администратором сети, так и эффективное автоматическое распределение трафика с помощью специального программного модуля. Программный модуль осуществляет мониторинг загрузки путей, при возрастании интенсивности трафика отдельного класса разделяет более обобщенные записи *FEC* на несколько более определенных и производит агрегирование трафика при уменьшении его интенсивности. Например, запись с адресом получателя 212.40.4/24 и неопределенными адресами портов может быть разделена на две записи: 212.40.4.0/25 и 212.40.4.128/25.

Для упрощения процедуры разделения обобщенной записи на более определенные, а также увеличении производительности маршрутизаторов предлагается использовать *hash*-функцию от значений полей *IP*-заголовка.

Как правило, входной граничный маршрутизатор соединен путями *LSP* с несколькими выходными маршрутизаторами, а не с одним, и пакеты, поступающие на него, не обязательно покидают сеть *MPLS* в одном и том же выходном граничном маршрутизаторе. Поэтому, чтобы поступающие пакеты не были направлены по неправильному пути из-за совпадения значений *hash*-функций у пакетов, принадлежащих разным классам *FEC* (что вполне возможно при наборе всего из 256 значений), некоторые поля таблицы являются обязательными для рассмотрения. Таким полем, прежде всего, является адрес получателя (заметим, что в традиционной *IP*-маршрутизации именно это поле пакета определяет следующий узел для передачи пакета). Поэтому предлагается использовать флаг (принимающий значения 0 или 1) для полей *dst IP*, *src IP*, *dst port* и *src port*. При установке флага в 1 поле не может игнорироваться при просмотре и выборе подходящего *LSP*, и проверка заголовка пакета на соответствие его полей данным значениям обязательна. Если же значения флага равно 0, решение принимается на основании значения *hash*-функции.

Таблица 2. Маршрутизация, без использования диапазонов значений *hash*-функции

Protocol	*dst IP/mask*	*src IP/mask*	*src port*	*dst port*	*hash*	*Interface*	*Label*	*Bytes*
TCP	195.245.194.142	*any*	*any*	*any*	0	1	5	5253255
...	...	...	...	...	...	...	...	...
TCP	195.245.194.142	*any*	*any*	*any*	127	1	5	5543345
TCP	195.245.194.142	*any*	*any*	*any*	128	2	4	6754544
...	...	...	...	...	...	...	...	...
TCP	195.245.194.142	*any*	*any*	*any*	255	2	4	4355994

Поле, содержащее значение *hash*-функции, разделено на две части. Это сделано для того, чтобы уменьшить количество записей в таблице, предоставив возможность использовать диапазоны значений. Например, вместо использования 256 записей для двух диапазонов значений (табл. 2), используется всего 2 записи (табл. 3). Хотя поля *src IP*, *src port*, *dst port* содержат значение *"any"* (указывает на то, что любое значение подходит), при определении *hash*-функции действительные значения данных полей из заголовков пакетов будут учитываться. Соде-

ржимое данных таблиц одинаково определяет соответствие между классом *FEC* и путем *LSP*. Так, для значений *hash*-функции от 0 до 127 используется путь LSP с исходящим интерфейсом 1 и меткой 5, а для значений *hash*-функции от 128 до 255- используется другой *LSP* (интерфейс 2 и метка 4). Диапазоны значений *hash*-функций для одного набора обязательных полей (отмеченных флагом со значением 1) не должны пересекаться.

Таблица 3. Маршрутизация, использующая диапазоны значений *hash*-функции

Protocol	*dst IP/mask*	*src IP/mask*	*src port*	*dst port*	*hash*		*Interface*	*Label*	*Bytes*
					from	*to*			
TCP	195.245.194.142	*any*	*any*	*any*	0	127	1	5	79743644
TCP	195.245.194.142	*any*	*any*	*any*	128	255	2	4	46465456

Предложенный алгоритм

Основываясь на рассмотренном выше модифицированном способе разделения потока входящих пакетов на граничном маршрутизаторе сети *MPLS*, разработаны алгоритмы управления записями таблицы соответствия *FEC* классов и путей *LSP* для распределения нагрузки между путями. На рис. 4, 5 представлены алгоритмы объединения и разделения записей.

```
begin   { начало алгоритма объединения записей }
procedure   { выбор и сравнение записей }
      begin
      if (Ni[label] = Nj[label] AND Ni[interface] = Nj[interface] AND
          Ni[hash] ≠ Nj[hash]) then
            begin
            procedure;  { объединение записей }
            procedure;  { вычисление размера таблицы V }
            end;
      end;
if (V → Vdop) then   { размер таблицы стремится к допустимому }
      begin
      procedure   { выбор и сравнение записей, mask1 - маска подсети }
            begin
            if(Ni[dstIP] = Nj[dstIP] AND Ni[mask1] = Nj[mask1]) then
            begin
            procedure;  { объединение записей }
            procedure;  { вычисление размера таблицы V }
            end;
            end;
      if (V → Vdop) then  { размер таблицы стремится к допустимому }
            begin
            procedure   { выбор и сравнение записей, mask2 - маска подсети }
            begin
            if(Ni[dstIP] ≈ Nj[dstIP] AND Ni[mask2] = Nj[mask2]) then
            procedure;  { объединение записей }
            end;
            end;
      end;
end.   { конец алгоритма объединения записей }
```

Рис. 4. Алгоритм объединения записей

```
begin   { начало алгоритма разделения записей }
procedure   { выбор записей – определение путей к одному выходному LSR}
      begin
      if (Ni[srcIP] = Nj[SrsIP]) then
      LSP[n]:= Nj;   { сохранение путей в массиве LSP }
      end;
procedure   { перебор записей в массиве LSP }
      begin
      if(LSP[n;Wi(b)] = min) then
      i:=LSP[n];   { i - путь с наименьшим значением Wi(b)}
      end;
procedure   { перебор записей в массиве LSP }
      begin
      if(LSP[n;Wj(b)] = max) then
      j:=LSP[n];   {j - путь с наибольшим значением Wj (b)}
      end;
procedure   { перебор записей в массиве LSP }
      begin
      if(LSP[n;proto] = UDP AND (Wi(Bi+ΔB)- Wj(Bj-ΔB))<e)
      then
      j:= i;   { замена пути j на путь i в найденной записи}
      end;
procedure   { перебор записей в массиве LSP }
      begin
      if(LSP[n;proto] = TCP AND (Wi(Bi+ΔB)- Wj(Bj-ΔB))<e)
      then
      j:= i;   { замена пути j на путь i в найденной записи}
      end;
procedure     { перебор записей в массиве LSP }
      begin
      procedure              { вычисление значения ΔB }
      if(Wi(Bi+ΔB/255)-Wj(Bj-ΔB/255)>0 AND
         Wi(Bi+ΔB)-Wj(Bj-ΔB)<0) then
            begin
            procedure;   { вычисление значения k }
            procedure;   { разбиение записей }
            LSP[n;hash from]:=[1..k];   { установка значения hash from }
            LSP[n;hash to]:=[k+1..255];   { установка значения hash to }
            end;
      end;
end.   { конец алгоритма разделения записей }
```

Рис. 5. Алгоритм разделения записей

Здесь $W_i(B)$ – функция стоимости использования пути i, зависящая от интенсивности потоков пакетов; B_i – интенсивность потока пакетов по пути i; ΔB – разность между интенсивностями потоков B_i и B_j.

Моделирование предложенного алгоритма

Для проверки работоспособности предложенного способа, и реализованного на его основе алгоритма, и сравнения его со стандартными реализациями необходимо провести имитационное моделирование.

В качестве системы моделирования выбрана система *OPNET*, которая является промышленным стандартом моделирования сетей.

На рис. 6 изображена сеть, построенная для проведения моделирования в системе *OPNET*. Маршрутизаторы в сети поддерживают технологию многопротокольной коммутации по меткам. Между граничными маршрутизаторами *LER4* и

LER1 вручную установлены пути коммутации по меткам (*LSP*).

Таблица соответствия *FEC* и *LSP* настроена таким образом, что трафик от каждого узла *WS1-WS3* направляется к узлам *WS4-WS7* по отдельным путям *LSP*, которые изображены разными штриховыми линиями.

На рис. 7 представлена зависимость загрузки путей между маршрутизаторами *LER4* и *LER1* от времени при использовании *MPLS* стандартной реализации. На рис. 8 - при использовании *MPLS* с предложенным алгоритмом, работающем на узле *LER4*.

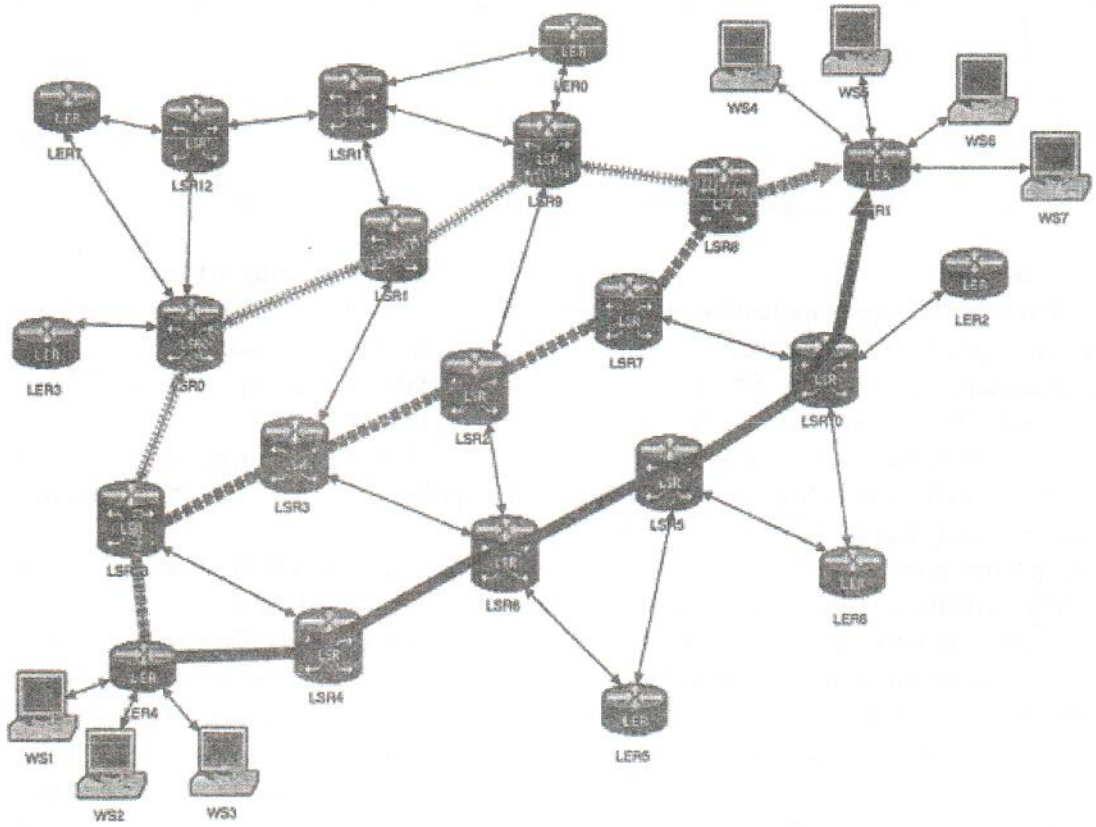

Рис. 6. Структура моделируемой сети *MPLS*

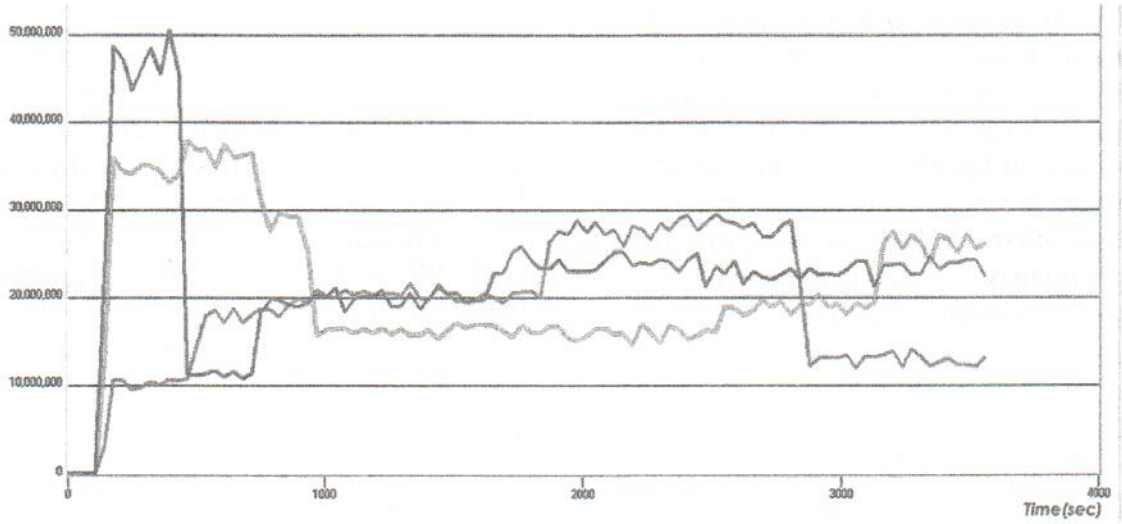

Рис. 7. Стандартная маршрутизация *MPLS*

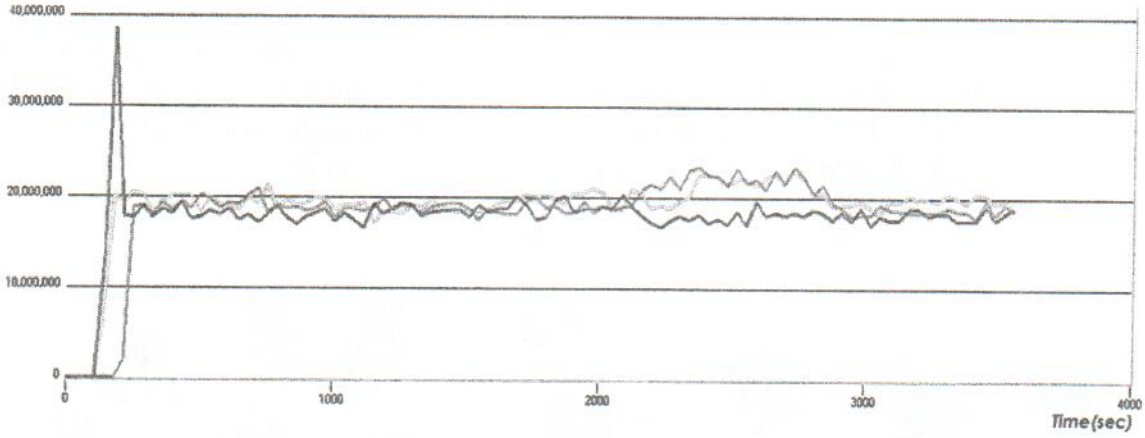

Рис. 8. *MPLS* с модифицированным алгоритмом на граничном маршрутизаторе

Выводы

Результаты моделирования, показывают, что предложенный алгоритм позволяет распределять трафик более эффективно, чем статическая настройка путей *LSP*. По сравнению с традиционной *IP*-маршрутизацией (где трафик передается по одному наилучшему пути), даже статическое распределение потоков позволяет получить оптимальную загрузку каналов.

В среднем при применении разработанного алгоритма неравномерность использования каналов составляет не более 10%, а при значительных изменениях интенсивности потоков за короткое время составляет около 30%. Это свидетельствует о достаточно высокой эффективности данного алгоритма и возможности его применения в сетях с потоками данных, изменяющимися за промежутки времени порядка 10 минут. Вместе с тем, при резких всплесках интенсивности потоков на протяжении времени до 1 минуты загрузка каналов может отличаться примерно в 2 раза. Поэтому данный алгоритм будет эффективным только при использовании в средних и крупных сетях.

Список литературы

1. *Rosen E., Rekhter Y., Tappan D., Fedorkow G., Farinacchi D., Conta A.* MPLS Label Stack Encoding // IETF RFC-3032. – January 2001.

2. *Rosen E., Viswanathan A., Callon R.* Multiprotocol Label Switching Architecture // IETF RFC-3031. – January 2001.

3. *Moy J.* OPSF Version 2 // IETF RFC-2178. – April 1998.

4. *Andersson L., Doolan P., Feldman N., Fredette A., Thomas B.* LDP Specification // IETF RFC-3036, Internet Engineering Task Force. – January 2001.

5. *Davie B., Lawrence J., McCloghrie K., Rekhter Y., Rosen E., Swallow G., Doolan G.* MPLS using LDP and ATM VC Switching // IETF RFC-3035. – January 2001.

6. *Aggarwal R., Papadimitriou D., Yasukawa S.* Extensions to RSVP-TE for Point to Multipoint TE LSPs // IETF – 2004.

7. *Deering S., Hinden S. and R.* Internet Protocol, Version 6 (IPv6) Specification // IETF RFC-2460. – December 1998.

8. *Elwalid A., Jin C., Low S., Widjaja I.* MATE: MPLS Adaptive Traffic Engineering // IEEE INFOCOM – 2003.