Лекция 11
Средства QoS коммуникационного узла. Протоколы сигнализации QoS и службы QoS IP-уровня

11.1. Алгоритмы управления очередями

11.1.1. Общие сведения об очередях к коммуникационным устройствам

Очереди возникают при временных или постоянных перегрузках коммуникационных устройств, возникающих при превышении темпа поступления пакетов над темпом их обработки коммуникационным устройством. В течении этих перегрузок в очередях сохраняются пакеты, которые не успевают обрабатываться коммуникационным устройством.

По причинам перегрузки очереди классифицируются на входные и выходные. Во *входные очереди* помещаются пакеты, которые не успевают обрабатываться процессором коммуникационного устройства (обработка пакета может включать, например, выполнение таких дополнительных функций как фильтрация, выполнение которых может задерживать обработку этого пакета). Такие очереди связываются с входным портом, из которого поступают пакеты. В *выходные очереди* помещаются пакеты, прошедшие обработку процессором, в случае, если выходные порты не успевают отправить их (в один выходной порт, могут, например, направляться пакеты из нескольких входных портов той же пропускной способности). Такие очереди связываются с выходными портам

Введем понятие коэффициента перегрузки коммуникационного устройства $K_п = V_{вх.ср} / V_{отпр.м}$, где $V_{вх.ср}$ - средняя скорость входящего трафика, а $V_{отпр.м}$ - максимальная скорость отправки пакетов выходным портом.

Если $K_п >= 1$ – очередь (ее длина) будет всегда расти и стремиться к бесконечности. Если $K_п < 1$ – очереди могут возникать и существовать продолжительное время. Причиной возникновения очередей при $K_п < 1$ являются пульсации трафика. Вопросами теоретического исследования длин очередей занимается теория массового обслуживания. Однако математические модели этой теории не позволяют адекватно учесть взаимодействий нескольких очередей по всему маршруту сетевого соединения. Некоторые модели, применимые для простейших комбинаций очередей рассмотрены, например, в книге (Уолренд Дж. Телекоммуникационные и компьютерные сети. Вводный курс. М: Постмаркет, 2001, 476 с.).

Отметим, что реализации очереди имеют конечный размер, поэтому могут переполняться. При переполнении очереди, вновь поступающие пакеты выбрасываются.

Перейдём к рассмотрению известных методов обработки очередей.

11.1.2. Очереди FIFO

Очереди FIFO (First In First Out - первым пришёл, первым обслужен) соответствуют привычным из обычной жизни очередям с полностью равноправными участниками (без льготников), обслуживаемых строго в соответствии их постановки в очередь. Этот способ организации очередей применяется в коммуникационных устройствах “по умолчанию”, если только эти коммуникационные устройства не настраиваются для работы в составе службы QoS. Но, поскольку предоставление некоторого наперёд заданного QoS является своеобразной “льготой” очереди этого типа не используются в коммуникационных устройствах, “задействованных” в службе QoS.

11.1.3. Очереди с приоритетами

Этом методе единственная очередь с приоритетами организуется в виде совокупности N внутренних очередей (ниже называемых просто очередями), упорядоченных в соответствии с их приоритетами (совпадающими с номерами очередей) от 0 до N-1 (приоритет 0 является самым высоким). Каждая из этих очередей обслуживается по методу FIFO, но с учётом существования других очередей.

При поступлении пакета в очередь с приоритетами, он заносится в очередь, определяемую приоритетом, указанным в заголовке пакета.

При выборе очередного элемента общей очереди для последующей обработки (процессором или выходным портом) первым делом анализируется наличие пакетов в очереди с приоритетом 0 и, в случае наличия в ней хотя бы одного пакета, первый из них выбирается из этой очереди и направляется на обработку. После завершения этой обработки вновь проверяется эта же очередь и только в случае, когда она пуста анализируется очередь с номером 1. Если она не пуста, первый пакет этой очереди выбирается для обработки. Однако после завершения этой обработки анализируется наличие пакетов не в очереди 1, а в очереди 0. Очередь с приоритетом *i* начинает анализироваться лишь в том случае, если все более приоритетные очереди (с меньшими номерами) пусты. И при завершении обработки первого элемента этой очереди вновь анализируется состояние 0-й очереди.

Очевидно, что при такой дисциплине обслуживания очередей 0-я очередь обладает абсолютным приоритетом: находящиеся в ней пакеты обрабатываются раньше пакетов всех других очередей, факт существования которых практически не заметен пакетам из 0-й очереди. Единственным случаем, когда существование других очередей становится заметным, является случай, в котором в момент поступления некоторого пакета в 0-ю очередь коммуникационное устройство выполняет обработку пакета из какой либо менее приоритетной очереди. В этом случае начало обработки вновь поступившего высокоприоритетного пакета задерживается до завершения обработки текущего пакета. Но для высокоскоростных (например - 100 Гигабитных) портов коммуникационных устройств величина этой задержки является ничтожно маленькой.

Что касается всех остальных (кроме 0-й) очередей, то дать прогноз величинам возможных задержек пакетов в этих очередях совершенно невозможно. Для каждой очереди возможные задержки пакетов в ней зависят от интенсивности информационных потоков, проходящих через все предыдущие очереди. Наличие такой неопределённости не позволяет гарантировать какой-либо определённый уровень QoS для информационных потоков, проходящих через любую из очередей, кроме самой приоритетной. Поэтому этот алгоритм в чистом виде не используется в средствах обеспечения QoS.

11.1.4. Взвешенные настраиваемые очереди

В этом методе, как и в предыдущем единственная взвешенная очередь к коммуникационному устройству организуется в виде совокупности N отдельных очередей, пронумерованных от 0 до N-1. При этом поступающие во взвешенную очередь пакеты заносятся во внутреннюю очередь с номером, указанным в заголовке пакета. А обработка пакетов из совокупности внутренних очередей выполняется в режиме временн'ого мультиплексирования с периодическим предоставлением всей пропускной способности коммуникационного устройства (его процессора или выходного порта) для поочередной обработки групп пакетов из циклически перебираемых внутренних очередей. Различное качество обслуживания пакетов из различных внутренних очередей обеспечивается путём выделения для обработки пакетов из разных очередей разных по продолжительности промежутков времени.

Организация такой обработки выполняется следующим образом. Каждой очереди присваивается положительный вес w_i (weight - вес) так, что сумма всех w_i равна 1. Этот вес определяет ту долю времени, которая должна выделяться для обработки пакетов из очереди с этим весом. Кроме того, выбирается величина промежутка времени T, равная времени полного цикла обработки всех очередей. Значения всех этих параметров могут

быть заданы при конфигурировании взвешенной очереди, которая в силу этого является настраиваемой.

А дальнейшая (после завершения конфигурирования) обработка взвешенной очереди выполняется по следующему алгоритму.

Первой выбирается для обработки 0-я очередь и для неё вычисляется продолжительность обработки пакетов из этой очереди $T_0 = w_0 * T$. Затем из очереди выбирается пакеты *p* и для него вычисляется время его обработки *t(p)*, и перевычисляется $T_0 := T_0 - t(p)$. Если после этого $T_0 >= 0$, то пакет отправляется на обработку, а из 0-й очереди выбирается следующий пакет и с ним выполняются те же действия (но без начальной установки T_0).

Процесс циклически повторяется до тех пор, пока не окажется, что либо, $T_0 < 0$, либо очередь пуста. При наступлении такого условия очередной пакет остаётся 0-й очереди, а не направляется на обработку. А алгоритм переходит к такой же обработки 1-й очереди. И далее циклически обрабатываются очереди с возрастающими номерами до тех пор, пока не завершится обработка очереди *N*-1.

Такой полный цикл обработки взвешенных очередей завершится в промежуток времени, продолжительность которого не превосходит T, поскольку обработка некоторых из взвешенных очередей может завершаться по исчерпании пакетов, а не промежутка времени, выделенного для их обработки. В последнем случае задержки в прохождении пакетов через коммуникационное устройство повышается, но может возрасти джиттер.

После завершения полного цикла обработки очередей процесс повторяется сначала и так вплоть до принудительной остановки этого процесса или отказа коммуникационного устройства.

Рассмотренный алгоритм обеспечивает пропорциональное весам очередей распределение между этими очередями пропускной способности коммуникационного устройства. Схема является почти идеальной в случае, когда не предъявляются слишком жёсткие требования к максимально допустимой величине задержки пакетов. Если эта величина должна быть меньше, чем *T*, то добиться выполнения этого требования можно лишь путём уменьшения значения *T*. А это значение можно уменьшать лишь до определённых пределов так, чтобы в полном цикле обработки очередей для очереди с наименьшим весом мог быть обработан хотя бы один пакет (а лучше - несколько пакетов).

Таким образом, при рассмотренной организации очередей может оказаться невозможным предоставление требуемого QoS для сверхчувствительных к задержкам приложений.

11.1.5. Взвешенное справедливое обслуживание

Очередь со взвешенным справедливым обслуживанием (Weighted Fair Queuing - WFQ) подобна по своей организации обычной взвешенной очереди, но взамен 0-й внутренней взвешенной очереди содержит высокоприоритетную очередь, а остальные N-1 очередей являются взвешенными. При этом по умолчанию всем взвешенным очередям присваиваются равные веса и, в этом случае, обслуживание является в определённом смысле справедливым

Вначале обработки при наличии пакетов в 0-й очереди они обрабатываются до исчерпания этой очереди. Затем начинается цикл обработки взвешенных очередей. Если после обработки некоторого пакета из некоторой взвешенной очереди выясняется, что в 0-й очереди появился хотя бы один пакет циклическая обработка взвешенных очередей временно прерывается для обработки высокоприоритетных пакетов. После того, как такие пакеты закончатся, продолжается прерванная обработка той же взвешенной очереди, что и обрабатывалась до прерывания. При этом используется значение счётчика оставшегося времени обработки очереди, вычисленное к моменту прерывания.

Выполненная модификация “чистого” метода взвешенного обслуживания обеспечивает возможность эффективной обработки пакетов сверхчувствительных приложений (которым должен быть присвоен 0-й класс обслуживания), практически не влияя на эффективность обработки очередей остальных классов, если средняя согласованная интенсивность трафика агрегированного потока сверхчувствительных

приложений составляет незначительную часть (например, не более 1-2 %) пропускной способности коммуникационного устройства.

Рассмотренный алгоритм является наиболее часто применяемым алгоритмом обслуживания очередей реализован в коммуникационном оборудовании большинства производителей. В частности, в оборудовании Cisco реализовано более одной разновидности WFQ, а именно: FWFQ - WFQ, основанное на потоках (*Flow-based*); и CWFQ - WFQ, основанное на классах (*Class-based*). Однако детальное рассмотрение этих разновидностей выходит за рамки настоящего учебника. Отметим лишь, что механизме FWFQ при перегрузке всех взвешенных очередей их веса автоматически усредняются (назначаются равными) и, в этом режиме работы алгоритм иногда называют просто FR (Fair Queuing) - справедливое обслуживание.

11.2. Механизмы профилирования и формирования трафика

В настоящем параграфе рассматриваются несколько механизмов (алгоритмов) профилирования трафика и один механизм формирования трафика.

11.2.1. Простые механизмы профилирования трафика

Механизм Traffic Shaping маршрутизаторов

Хотя буквальным переводом слов “traffic shaping” является “формирование трафика”, в соответствии с классификацией средств кондиционирования трафика, как механизмов QoS уровня узла сети механизм Traffic Shaping маршрутизаторов является механизмом профилирования трафика, совмещённого с предварительной классификацией трафика. Поскольку классификация трафика обычно выполняется лишь на пограничных маршрутизаторах сетей (AS) или некоторых их подсетей, механизм Traffic Shaping должен применяться именно на таких маршрутизаторах.

Напомним, что этот механизм выполняет отбрасывание пакетов агрегированного потока определенного класса (заданного своим ACL - списком контроля доступа) в случае, если скорость поступления данных из этого потока начинает превышать согласованную для этого потока максимальную скорость CIR. Если пакеты начинают отбрасываться, то, естественно, отправители этих пакетов не получают подтверждений об их доставке и приходят к выводу о том, что пакеты потеряны. Тогда в соответствии с принципом обратной связи протокола TCP отправитель уменьшает размер окна TCP (а значит - и количество отправленных без ожидания подтверждения об их доставке) и тем самым снижает темп отправки пакетов. После того, как количество потерянных пакетов опустится до определённых пределов размер окна TCP начнёт повышаться, что позволит увеличить и скорость отправки данных (до очередного превышения CIR).

Используемый в этом механизме принцип профилирования предельно прост: вершины всех пиков в скорости передачи потока, превосходящие CIR, просто “горизонтально срезаются сверху “на высоте” CIR. Рассматриваемые ниже методы профилирования допускают временные превышения согласованной средней скорости передачи данных CBR в некоторых “допустимых” пределах.

Механизм случайного раннего обнаружения (RED).

Метод случайного раннего обнаружения (*Random Early Detection* - RED) был разработан для предотвращения серьёзных перегрузок на магистралях интернета.

В соответствии с этим методом, применяемом для профилирования трафика некоторого магистрального канала, для этого канала вводятся два порога скорости передачи данных через этот канал: нижний порог b_1 (b - *barrier*) “абсолютной допустимости” и верхний порог b_2 “условной допустимости” такой скорости.

Если *текущая средняя скорость V* передачи потока данных через канал не больше нижнего порога b_1, то все пакеты потока безусловно передаются через этот канал. Если же $b_1 < V <= b_2$, то пакеты потока случайно выбрасываются с линейно возрастающей в зависимости от текущего значения V вероятностью $p = (V-p_1)/(p_2-p_1)$. Очевидно, что при $V=p_2$ эта вероятность будет равна 1. И при $V> p_2$ все пакеты безусловно выбрасываются.

Здесь, как и в предыдущем методе, при выбрасывании пакетов начинает работать механизм саморегуляции скорости отправки данных, основанный на принципе обратной связи протокола TCP.

Отметим, что оценка чрезмерности загрузки канала на основе текущей средней скорости допускает кратковременные всплески скорости передачи данных внутри периода усреднения этой скорости. Но превышение средней скоростью нижнего порога скорости сразу же приводит к потерям некоторых случайно выбранных пакетов.

11.2.2. Алгоритм дырявого ведра

Названный алгоритмом дырявого ведра (*Leaky Bucket*) метод профилирования трафика обеспечивает приведение этого трафика к форме с заранее заданными значениями средней скорости этого трафика CBR и величины допустимой его пульсации B_c (*Burst committed*), рассчитываемой по формуле B_c = CBR**T*, где - период усреднения скорости. При этом дополнительно указывается величина предельно допустимой превышенной пульсации B_e (*Burst extended*). Величины параметров CBR, *T* и B_e задаются при конфигурировании режима работы алгоритма.

Алгоритм работает следующим образом.

При обработке пакетов ведется счетчик C их суммарной длины (вначале C=0).

При передаче пакета C := C + $L_п$, где – $L_п$ длина пакета.

Если C <= B_c - пакет пропускается

Если B_c < C <= B_e – пакет пропускается, но помечается (биты DE в поле TOS байта DS)

Если C > B_e – пакет отбрасывается

Счетчик C каждые T секунд уменьшается: на min (C, B_c) – сбрасывается в 0 (при C<= B_c) или уменьшается на B_c. Таким образом алгоритм «помнит» превышение предыдущей пульсации, «оставшееся в наследство» от обработки входного потока на предыдущих интервалах времени T и учитывает это.

Сравнение с дырявым ведром (метафора).

11.2.3. Алгоритм ведра токенов (маркеров)

Метод предназначен для формирования (а не для профилирования) трафика

Цель – устранение неравномерности продвижения пакетов (для потокового трафика, чувствительного к вариациям задержек).

Ведро токенов равномерно заполняется токенами (специальными записями), генерируемым специальным генератором токенов со скоростью *K* байт/сек.

Реализуется счетчиком *N*, равномерно увеличиваемым по таймеру n раз в секунду на K/n.

Входной трафик помещается в буфер *B*.

Выдача очередного пакета длиной *M* из входного буфера в выходной порт осуществляется тогда, когда *M* <= *N.* После этого *N* := *N* - *M*

В результате скорость потока, выдаваемого на выходной интерфейс, не превосходит скорости генерации токенов. Это достигается путем искусственной задержки пакетов в буфере при превышении скорости поступления данных над скоростью генерации токенов.

11.3. Реализации службы QoS на уровне IP

Основные службы QoS реализуются именно на уровне IP, поскольку это самый нижний из уровней, обеспечивающий связность «из конца в конец».

Эти службы делятся на 2 класса

1) Службы поддержки «жесткого» качества обслуживания, обеспечивающие гарантированное обслуживание с требуемым качеством «из конца в конец». Сложно реализуется в масштабах Интернет

Жесткое QoS обеспечивается службой IntServ и протоколом RSVP для *микропотоков* (потоков между конкретными приложениями). Указанные средства были предложены в 1997 г.

2) Службы поддержки «мягкого» качества обслуживания, не дающие конкретных (выраженных числовыми показателями) гарантий качества обслуживания. Вместо этого для приоритетных классов трафика обеспечивается «по возможности высокое QoS».

Мягкое QoS обеспечивается службой DiffServ, предложенной в 1998 г. Обеспечивает приоритетное обслуживание *макропотоков*, соответствующих различным *классам трафика*. Применяется на магистральных каналах.

11.3.1. Служба IntServ с протоколом RSVP

Описание протокола RSVP

RSVP – ReSerVation Protocol

Обеспечивает резервирование ресурсов коммуникационных устройств для предоставления QoS «из конца в конец». Ориентирован на поддержку резервирования для группового вещания (у одного источника может быть несколько получателей (приемников)).

В качестве инициатора резервирования выступает источник. Он информирует все приемники о максимально возможном уровне QoS, предоставляемым этим источником.

Приемники могут запросить меньший уровень скорости трафика (например, если воспроизводящее оборудование видеоконференцсвязи не способно воспользоваться всей разрешающей способностью видеооборудования источника).

Логика работы RSVP

1) Источник направляет по уникальному или групповому адресу получателя специальный пакет Pass Tspec (Traffic Specification), содержащий рекомендуемые параметры для качественного приема трафика: верхние и нижние значения пропускных способностей, макс. задержки и их вариации (jitter).

2) Каждый маршрутизатор, поддерживающий RSVP (прочие маршрутизаторы пропускают это сообщение прозрачно) получивший это сообщение «запоминает направление пути для пакетов RSVP» - запоминает маршрутизатор, от которого пришло сообщение PASS. Тем самым по всему пути от источника к получателю запоминается маршрут для сеанса RSVP. Сеанс идентифицируется IP-адресом и номером порта источника (в IPv6 для идентификации потока используется поле «метка потока»).

3) После получения сообщения Pass приемник принимает решение о требуемом ему уровне QoS (уровень QoS, указанный источником может снижаться) и направляет источнику запрос на резервирование ресурса Resv RSpec (Required Specification)

4) Каждый маршрутизатор по получении Resv

4.1) проверяет, есть ли у маршрутизатора ресурсы (нераспределенные емкости каналов), требуемые для удовлетворения запроса **&**

4.2) при помощи централизованных средств политики проверяет, имеет ли право приемник зарезервировать затребованное количество ресурсов

Если **да**

а) резервирует ресурсы, включает исполнительные механизмы (например – механизмы управления взвешенными очередями) и продвигает пакет по направлению к источнику:

б) Последний маршрутизатор перед источником, получивший и корректно отработавший Resv посылает приемнику подтверждение и передает успешно отработанный Resv источнику.

в) После установления зарезервированного соединения источник начинает посылать всем получателям данные в объеме минимальных для всех получателей Rspec (иначе потребовалась бы отправка различных данных, соответствующих различным Rspec различным индивидуальным (unicast) получателям. Отметим, что выполненное резервирование может быть отменено явно или по тайм-ауту.

иначе: отвечает отправителю Resv, что затребованное количество ресурсов не может быть выделено. По дороге к приёмнику отменяются выполненные резервирования и отключаются включенные механизмы резервирования.

Приёмник, получивший такой ответ, может попробовать запросить меньший объем ресурсов.

О расположении в сети, маршрутизаторов, не понимающих RSVP

1) на скоростных маршрутах между «узкими горлышками», которыми обычно являются пограничные маршрутизаторы сетей, понимающие RSVP

2) интерфес, ведущий к предыдущему RSVP маршрутизатору не может привести к «обходным» путям этого маршрутизатора; через этот интерфейс маршрутизаторы должны быть связаны либо единственным путём, либо гамаком путей (напомним, что гамаком называется направленный граф, имеющий одну входную и одну выходную вершину и не содержащий ориентированных циклов).

Недостатки службы IntServ с протоколом RSVP

1) Эту службу и протокол практически невозможно реализовать в масштабах всей сети Интернет ввиду организационных сложностей создания единой службы политики учета и выделения коммуникационных ресурсов

2) RSVP применим только к микропотокам и не могут быть применены к макропотокам; в частности – к агрегированным макропотокам IP-сетей. А значит средствами RSVP нельзя зарезервировать определённую полосу пропускания для интересующей нас IP-сети.

3) Резервирование коммуникационных ресурсов для соответствующих приложений *выполняется по явному запросу* этих приложений. Это значит, что если приложение нужно использовать совместно с RSVP, то ***в код таких приложений должны быть включены команды выдачи таких запросов***. Код стандартных сетевых приложений не содержит таких запросов к RSVP. Для RSVP-TE (если его применение настраивается для определённых классов (меток) запросы на резервирование выполняются ПО пограничного LSR'a

Краткие сведения о протоколе MPLS RSVP-TE и преодолении в нём большинства недостатков RSVP

С созданием технологии MPLS для сетей MPLS был разработана ориентированная на работу в таких сетях разновидность протокола RSVP, получившая название MPLS RSVP-TE или просто RSVP-TE. Этот протокол был разработан с использованием расширений протокола управления маршрутизацией OSPF, относящегося к протоколам состояния связей. В число этих расширений входят, в частности, введение нового типа LSA (объявления состояния связи) и применение версии CSPF (*Constrained* SPF – ограниченный SPF). Отметим, что существует версия протокола RSVP-TE, построенная с использованием протокола управления маршрутизацией IS-IS, также относящегося к классу протоколов состояния связей и используемому во многих сетях операторов доступа к интернету.

Проанализируем, как в протоколе как RSVP-TE могут быть полностью или частично преодолены отмеченные недостатки протокола RSVP.

1) Сеть MPLS может быть использована достаточно крупным (например, национального масштаба) оператором связи для построения магистральной инфраструктуры своей сети. Такой оператор связи обычно знает текущий состав своих VIP клиентов и свои обязательства перед такими клиентами, которые предоставляют необходимую информацию для выработки политики резервирования требуемых для реализации упомянутых обязательств ресурсов сетевых устройств своей сети MPLS.

2) Поскольку с каждым значением метки в MPLS связывается некоторый агрегированный поток данных, протокол RSVP-TE (в отличие от обычного RSVP) позволяет зарезервировать полосу пропускания для такого агрегированного потока.

3) Для RSVP-TE запросы на резервирование выполняются программным обеспечением пограничного LSR'a MPPLS сети для выделенных им при классификации входящего внешнего трафика потоков внутреннего трафика MPLS сети. При этом к программному обеспечению приложений, трафик которых агрегируется в макропотоки, для которых средствами RSVP-TE выполняется резервирование требуемых сетевых ресурсов, не предъявляется никаких требований по включению запросов к RSVP.

11.3.2. Служба DiffServ

Общее описание службы DiffServ

Служба DiffServ представляет собой более простой, по сравнению с IntServ механизм, ориентированный на приоритетное обслуживание макропотоков, выделяемых при классификации входящего в сеть трафика пограничными маршрутизаторами сети.

В качестве протокола сигнализации служба использует 5 битное поле TOS DS байта заголовка IPv4 или полей «приоритет» и «метка потока» IPv6.

DiffServ не резервирует ресурсы, а обеспечивает лишь *приоритетное обслуживание* макропотоков в соответствии с правилами пошагового поведения (*Per Hop Behavior* – PHB). В IPv4 для организации приоритетного обслуживания используются 5 битов поля TOS: 3 бита – класс (приоритет); 2 бита – пометки о возможном отбрасывании пакетов.

В службе DiffServ маркировкой (классификацией) пакетов занимаются обычно пограничные маршрутизаторы (именно они устанавливают 3-битный класс пакета или поле приоритета в пакете IPv6) сети, находящейся под единым административным управлением.

При выходе из сети и при входе в другую сеть (AS) маркировка меняется. Для обеспечения единой приоритетной политики в глобальной сети должны существовать соглашения о политике маркировки, заключенными между организациями, управляющими различными доменами DiffServ (AS). Тогда при входе в другую сеть старая маркировка не будет отбрасываться, а может каким-то образом транслироваться в новую маркировку (в простейшем случае – маркировка может сохраняться).

Рекомендации по маркировке: наивысший приоритет присваивать трафику, наиболее чувствительному к задержкам.

Достоинства DiffServ

1) Простота (и эффективность) реализации
2) Возможность применения в глобальных сетях
3) Возможность обеспечивать QoS для макропотоков

Недостатки DiffServ

1) Отсутствие гарантий требуемого уровня QoS
2) Невозможность применения к микропотокам

Область применения: главным образом магистрали сетей

Возможности совместного использования DiffServ и MPLS

И DiffServ, и MPLS на своих пограничных маршрутизаторах выполняют классификацию трафика. Для эффективной реализации DiffServ в MPLS сетях можно «наложить» множество классов DiffServ на определённое подмножество классов MPLS и использовать средства технологии TE для организации эффективных маршрутов передачи агрегированных потоков классов DiffServ, повысив тем самым эффективность работы DiffServ.

Общий вывод, относительно совместного применения MPLS со службами IntServ и DiffServ

Применение MPLS при реализации обеих служб QoS позволяет повысить эффективность этих служб и минимизировать присущие некоторым из них (IntServ) недостатки и повысить эффективность других (DiffServ).