Лабораторная работа №1

ИЗУЧЕНИЕ ПРИНЦИПОВ АДРЕСАЦИИ, КОММУТАЦИИ И МАРШРУТИЗАЦИИ В КОМПЬЮТЕРНЫХ СЕТЯХ

Цель работы: ознакомится с принципами адресации, коммутации и маршрутизации в компьютерных сетях.

Краткие сведения из теории

Еще одной новой проблемой, которую нужно учитывать при объединении трех и более компьютеров, является проблема их адресации, точнее адресации их сетевых интерфейсов. Один компьютер может иметь несколько сетевых интерфейсов. Например, для создания полносвязной структуры из N компьютеров необходимо, чтобы у каждого из них имелся $N - 1$ интерфейс.

По количеству адресуемых интерфейсов адреса можно классифицировать следующим образом:

• **уникальный адрес** (unicast) используется для идентификации отдельных интерфейсов;

• **групповой адрес** (multicast) идентифицирует сразу несколько интерфейсов, поэтому данные, помеченные групповым адресом, доставляются каждому из узлов, входящих в группу;

• данные, направленные по **широковещательному адресу** (broadcast), должны быть доставлены всем узлам сети;

• **адрес произвольной рассылки** (anycast), определенный в новой версии протокола IPv6, так же, как и групповой адрес, задает группу адресов, однако данные, посланные по этому адресу, должны быть доставлены не всем адресам данной группы, а любому из них. Адреса могут быть **числовыми** (например, 129.26.255.255 или 81.la.ff.fF) и **символьными** (site.domen.ru, willi-winki).

Символьные адреса (имена) предназначены для запоминания людьми и поэтому обычно несут смысловую нагрузку. Для работы в больших сетях символьное имя может иметь иерархическую структуру, например ftp-arch1.ucl.ac.uk. Этот адрес говорит о том, что данный компьютер поддерживает ftp-архив в сети одного из колледжей Лондонского университета (University College London – UCL) и эта сеть относится к академической ветви (ac) Интернета Великобритании (United Kingdom – UK). При работе в пределах сети Лондонского университета такое длинное символьное имя явно избыточно и вместо него можно пользоваться кратким символьным именем ftp-arch 1. Хотя символьные имена удобны для людей, из-за переменного формата и потенциально большой длины их передача по сети не очень экономична.

Множество всех адресов, которые являются допустимыми в рамках некоторой схемы адресации, называется **адресным пространством.**

Адресное пространство может иметь плоскую (линейную) организацию (рисунок 1) или иерархическую организацию (рисунок 2).

При **плоской организации** множество адресов никак не структурировано. Примером плоского числового адреса является **MAC-адрес,** предназначенный для однозначной идентификации сетевых интерфейсов в локальных сетях. Такой адрес обычно используется только аппаратурой, поэтому его стараются сделать по возможности компактным и записывают в виде двоичного или шестнадцатеричного числа, например 0081005e24a8. При задании MAC-адресов не требуется выполнение ручной работы, так как они обычно встраиваются в аппаратуру компанией-изготовителем, поэтому их называют также аппаратными адресами (hardware address). Использование плоских адресов является жестким решением – при замене аппаратуры, например, сетевого адаптера, изменяется и адрес сетевого интерфейса компьютера.

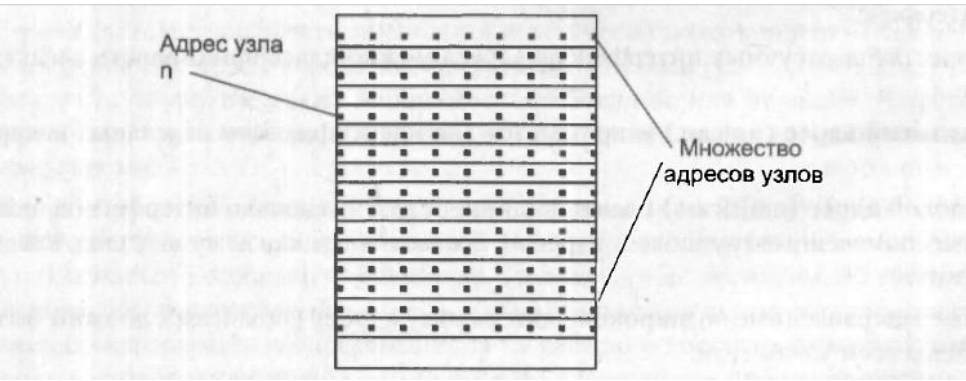

Рисунок 1 – Плоская организация адресного пространства

Рисунок 2 – Иерархическая организация адресного пространства

При иерархической организации адресное пространство структурируется в виде вложенных друг в друга подгрупп, которые, последовательно сужая адресуемую область, в конце концов, определяют отдельный сетевой интерфейс.

В показанной на рис. 2.13 трехуровневой структуре адресного пространства адрес конечного узла задается тремя составляющими:

идентификатором группы (K), в которую входит данный узел, идентификатором подгруппы (L) и, наконец, идентификатором узла (n), однозначно определяющим его в подгруппе. Иерархическая адресация во многих случаях оказывается более рациональной, чем плоская. В больших сетях, состоящих из многих тысяч узлов, использование плоских адресов приводит к большим издержкам – конечным узлам и коммуникационному оборудованию приходится оперировать таблицами адресов, состоящими из тысяч записей. В противоположность этому иерархическая система адресации позволяет при перемещении данных до определенного момента пользоваться только старшей составляющей адреса (например, идентификатором группы K), затем для дальнейшей локализации адресата задействовать следующую по старшинству часть (L) и в конечном счете – младшую часть (n).

Типичными представителями иерархических числовых адресов являются сетевые IP-адреса. В них поддерживается двухуровневая иерархия, адрес делится на старшую часть – номер сети и младшую – номер узла. Такое деление позволяет передавать сообщения между сетями только на основании номера сети, а номер узла требуется уже после доставки сообщения в нужную сеть; точно так же, как название улицы используется почтальоном только после того, как письмо доставлено в нужный город.

На практике обычно применяют сразу несколько схем адресации, так что сетевой интерфейс компьютера может одновременно иметь несколько адресов-имен. Каждый адрес задействуется в той ситуации, когда соответствующий вид адресации наиболее удобен. А для преобразования адресов из одного вида в другой используются специальные вспомогательные протоколы, которые называют **протоколами разрешения адресов.**

Пользователи адресуют компьютеры иерархическими символьными именами, которые автоматически заменяются в сообщениях, передаваемых по сети, иерархическими числовыми адресами. С помощью этих числовых адресов сообщения доставляются из одной сети в другую, а после доставки сообщения в сеть назначения вместо иерархического числового адреса используется плоский аппаратный адрес компьютера. Проблема установления соответствия между адресами различных типов может решаться как централизованными, так и распределенными средствами.

При *централизованном подходе* в сети выделяется один или несколько компьютеров (серверов имен), в которых хранится таблица соответствия имен различных типов, например символьных имен и числовых адресов. Все остальные компьютеры обращаются к серверу имен с запросами, чтобы по символьному имени найти числовой номер необходимого компьютера.

При *распределенном подходе* каждый компьютер сам хранит все назначенные ему адреса разного типа. Тогда компьютер, которому необходимо определить по известному иерархическому числовому адресу некоторого компьютера его плоский аппаратный адрес, посылает в сеть широковещательный запрос. Все компьютеры сети сравнивают

содержащийся в запросе адрес с собственным. Тот компьютер, у которого обнаружилось совпадение, посылает ответ, содержащий искомый аппаратный адрес. Такая схема использована в **протоколе разрешения адресов** (Address Resolution Protocol, ARP) стека TCP/IP.

Достоинство распределенного подхода состоит в том, что он позволяет отказаться от выделения специального компьютера в качестве сервера имен, который, к тому же, часто требует ручного задания таблицы соответствия адресов. Недостатком его является необходимость широковещательных сообщений, перегружающих сеть. Именно поэтому распределенный подход используется в небольших сетях, а централизованный – в больших.

До сих пор мы говорили об адресах сетевых интерфейсов, компьютеров и коммуникационных устройств, однако конечной целью данных, пересылаемых по сети, являются не сетевые интерфейсы или компьютеры, а выполняемые на этих устройствах программы – процессы. Поэтому в адресе назначения наряду с информацией, идентифицирующей интерфейс устройства, должен указываться адрес процесса, которому предназначены посылаемые по сети данные. Очевидно, что достаточно обеспечить уникальность адреса процесса в пределах компьютера. Примером адресов процессов являются *номера портов TCP и UDP,* используемые в стеке TCP/IP.

Задача маршрутизации, в свою очередь, включает в себя две подзадачи:

• определение маршрута;

• оповещение сети о выбранном маршруте.

Определить маршрут означает выбрать последовательность транзитных узлов и их интерфейсов, через которые надо передавать данные, чтобы доставить их адресату. Определение маршрута – сложная задача, особенно когда конфигурация сети такова, что между парой взаимодействующих сетевых интерфейсов существует множество путей. Чаще всего выбор останавливают на одном *оптимальном* по некоторому критерию маршруте. В качестве критериев оптимальности могут выступать, например, номинальная пропускная способность и загруженность каналов связи; задержки, вносимые каналами; количество промежуточных транзитных узлов; надежность каналов и транзитных узлов.

Но даже в том случае, когда между конечными узлами существует только *один* путь, при сложной топологии сети его нахождение может представлять собой нетривиальную задачу.

Маршрут может определяться эмпирически («вручную») администратором сети на основании различных часто не формализуемых соображений. Среди побудительных мотивов выбора пути могут быть: особые требования к сети со стороны различных типов приложений, решение передавать трафик через сеть определенного поставщика услуг, предположения о пиковых нагрузках на некоторые каналы сети, соображения безопасности.

Однако эмпирический подход к определению маршрутов малопригоден для большой сети со сложной топологией. В этом случае используются

автоматические методы определения маршрутов. Для этого конечные узлы и другие устройства сети оснащаются специальными программными средствами, которые организуют взаимный обмен служебными сообщениями, позволяющий каждому узлу составить свое «представление» о сети. Затем на основе собранных данных программными методами определяются рациональные маршруты.

При выборе маршрута часто ограничиваются только информацией о топологии сети. Этот подход иллюстрирует рисунок 3. Для передачи трафика между конечными узлами A и C существует два альтернативных маршрута: *A-1-2-3-C* и *A-1-3-C*. Если мы учитываем только топологию, то выбор очевиден – маршрут *A-1-3-C,* который имеет меньше транзитных узлов.

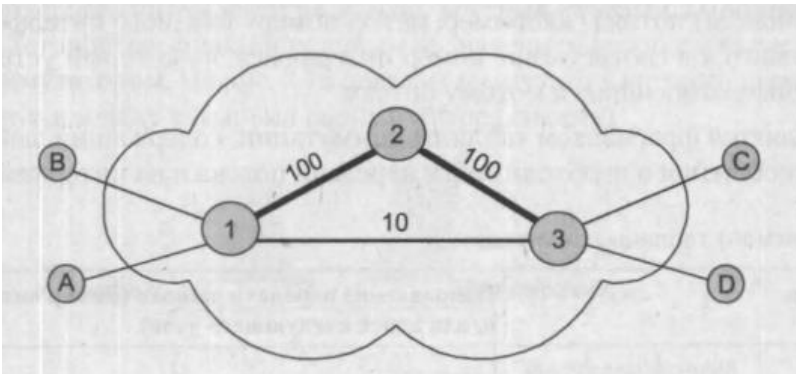

Рисунок 3 – Выбор маршрута

Решение было найдено путем минимизации критерия, в качестве которого в данном примере выступала длина маршрута, измеренная количеством транзитных узлов. Однако, возможно, наш выбор был не самым лучшим. На рисунке показано, что каналы *1-2* и *2-3* обладают пропускной способностью 100 Мбит/с, а канал *1-3* – только 10 Мбит/с. Если мы хотим, чтобы наша информация передавалась по сети с максимально возможной скоростью, то нам следовало бы выбрать маршрут *A-1-2-3-C,* хотя он и проходит через большее количество промежуточных узлов. То есть можно сказать, что маршрут *A-1-2-3-C* в данном случае оказывается «более коротким».

Абстрактный способ измерения степени близости между двумя объектами называется метрикой. Так, для измерения длины маршрута могут быть использованы разные метрики – количество транзитных узлов, как в предыдущем примере, линейная протяженность маршрута и даже его стоимость в денежном выражении. Для построения метрики, учитывающей пропускную способность, часто используют следующий прием: длину каждого канала-участка характеризуют величиной, обратной его пропускной способности. Чтобы оперировать целыми числами, выбирают некоторую константу, заведомо большую, чем пропускные способности каналов в сети. Например, если мы в качестве такой константы выберем 100 Мбит/с, то метрика каждого из канатов *1-2* и *2-3* равна 1, а метрика канала *1-3* составляет 10. Метрика маршрута равна сумме метрик составляющих его

каналов, поэтому часть пути *1-2-3* обладает метрикой 2, а альтернативная часть пути *1-3* — метрикой 10. Мы выбираем более «короткий» путь, то есть путь *A-1-2-3-C*.

Описанные подходы к выбору маршрутов не учитывают текущую степень загруженности каналов трафиком. Используя аналогию с автомобильным трафиком, можно сказать, что мы выбирали маршрут по карте, учитывая количество промежуточных городов и ширину дороги (аналог пропускной способности канала), отдавая предпочтение скоростным магистралям. Но мы не стали слушать радио или телевизионную программу, которая сообщает о текущих заторах на дорогах. Так что наше решение оказывается отнюдь не лучшим, когда по маршруту *A-1-2-3-C* уже передается большое количество потоков, а маршрут *A-1-3-C* практически свободен.

После того как маршрут определен (вручную или автоматически), надо *оповестить* о нем все устройства сети. Сообщение о маршруте должно нести каждому транзитному устройству примерно такую информацию: «каждый раз, когда в устройство поступят данные, относящиеся к потоку *n,* их следует передать для дальнейшего продвижения на интерфейс *F».* Каждое подобное сообщение о маршруте обрабатывается устройством, в результате создается новая запись в **таблице коммутации.** В этой таблице локальному или глобальному признаку (признакам) потока (например, метке, номеру входного интерфейса или адресу назначения) ставится в соответствие номер интерфейса, на который устройство должно передавать данные, относящиеся к этому потоку.

Таблица 1 является фрагментом таблицы коммутации, содержащий запись, сделанную на основании сообщения о необходимости передачи потока *n* на интерфейс *F.*

Таблица 1 – Фрагмент таблицы коммутации

Признаки потока	**Направление передачи данных (номер интерфейса и/или адрес следующего узла)**
...	...
n	F
...	...

Конечно, детальное описание структуры сообщения о маршруте и содержимого таблицыкоммутации зависит от конкретной технологии, однако эти особенности не меняют сущности рассматриваемых процессов.

Передача информации транзитным устройствам о выбранных маршрутах, так же как и определение маршрута, может осуществляться вручную или автоматически. Администратор сети может зафиксировать маршрут, выполнив в ручном режиме конфигурирование устройства, например, жестко скоммутировав на длительное время определенные пары входных и выходных интерфейсов (как работали «телефонные барышни» на первых коммутаторах). Он может также по собственной инициативе внести запись о маршруте в таблицу коммутации.

Однако поскольку топология и состав информационных потоков могут меняться (отказы узлов или появление новых промежуточных узлов, изменение адресов или определение новых потоков), гибкое решение задач определения и задания маршрутов предполагает постоянный анализ состояния сети и обновление маршрутов и таблиц коммутации. В таких случаях задачи прокладки маршрутов, как правило, не могут быть решены без достаточно сложных программных и аппаратных средств.

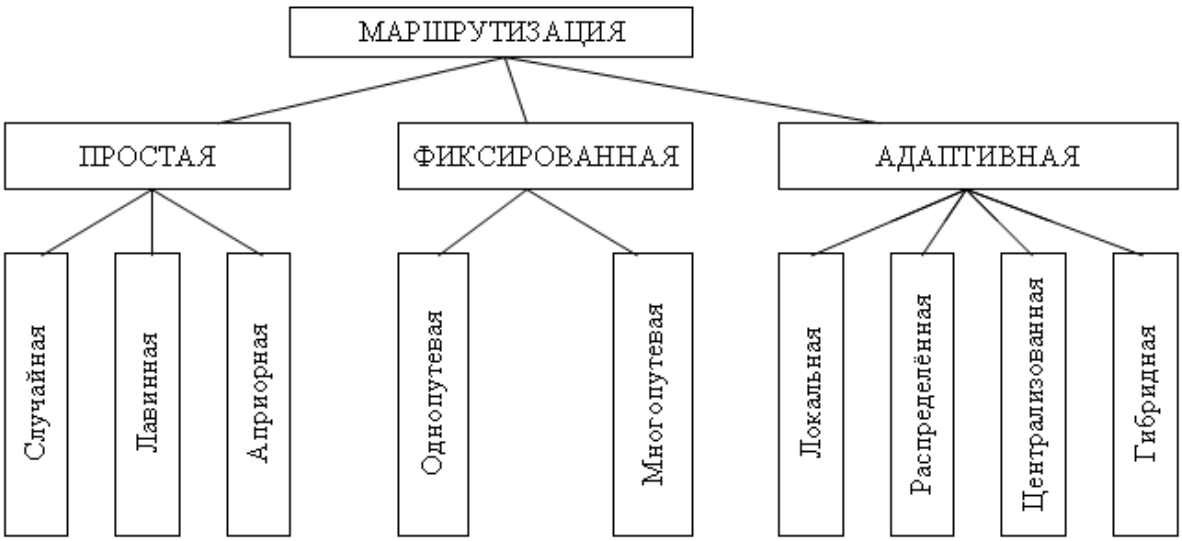

Рисунок 4 – Классификация методов маршрутизации

Простая маршрутизация — способ маршрутизации, не изменяющийся при изменении топологии и состояния СПД. Простая маршрутизация обеспечивается разными алгоритмами, типичными из которых являются алгоритмы случайной и лавинной маршрутизации.

- **Случайная маршрутизация** – передача пакета из узла в любом, случайным образом выбранном направлении, кроме направления, по которому пакет поступил в узел. Пакет, совершая “блуждания” по сети, с конечной вероятностью когда-либо достигает адресата. Очевидно, что случайная маршрутизация неэффективна ни по времени доставки пакетов, ни по использованию пропускной способности сети.
- **Лавинная маршрутизация** – передача пакета из узла во всех направлениях, кроме того, по которому поступил пакет. При этом, если узел связан с n другими узлами СПД, пакет передается в n–1 направлениях, т. е. размножается. Очевидно, что хотя бы одно направление обеспечит доставку пакета за минимальное время, т. е. лавинная маршрутизация гарантирует малое время доставки, однако это достигается за счет резкого ухудшения использования пропускной способности сети из-за загрузки ее большим числом пакетов.
- **Априорная маршрутизация** – передача пакета в направлении, выбираемом на основе анализа потока, проходящего через узел. При этом пакеты, поступая в сеть, снабжаются адресами получателя и источника и счетчиком числа пройденных узлов. Пакет, который пришел в узел со

значением счетчика 1, определяет соседний узел; пакет со значением счетчика 2 определяет узел, находящийся на расстоянии двух шагов, и т. д. Эти данные позволяют установить топологию сети и на ее основе построить таблицу для выбора маршрута. Постоянно анализируя число пройденных узлов, можно изменять таблицу маршрутов, если появился пакет с числом пройденных узлов, меньшим ранее зарегистрированного. Этот способ маршрутизации позволяет узлам приспосабливаться к изменению топологии сети, однако процесс адаптации протекает медленно и неэффективно. Метод изучения пути передачи пакетов используется для построения ряда модификаций алгоритмов простой маршрутизации.

Простая маршрутизация, не обеспечивая направленной передачи пакетов от источников к адресатам, имеет низкую эффективность. Основное ее достоинство — обеспечение устойчивой работы СПД при выходе из строя различных участков сети.

Фиксированная маршрутизация – способ выбора направления передачи по таблице маршрутизации, устанавливающей направление передачи для каждого узла назначения.

Адаптивная маршрутизация – способ выбора направления передачи, учитывающий изменение состояния СПД. При адаптивной маршрутизации узлы СПД принимают решение о выборе маршрутов, реагируя на разного рода данные об изменении топологии и нагрузки.

Ассоциация (объединение) сетей посредством шлюзов

Сети с разными стеками протоколов или с разными системами адресации взаимодействуют через шлюз.

Сетевой **шлюз** (англ. gateway) – аппаратный маршрутизатор или программное обеспечение для сопряжения компьютерных сетей, использующих разные протоколы (например, локальной и глобальной).

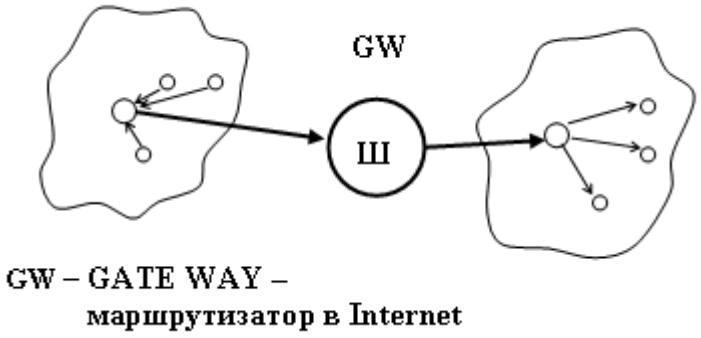

Рисунок 5 – Объединение сетей

Шлюз представляет собой отдельный узел сети, реализуется аппаратными средствами и программными (программа сетевых протоколов + трансляция протоколов). Транслятор протоколов преобразует форматы протоколов одной сети в форматы другой.

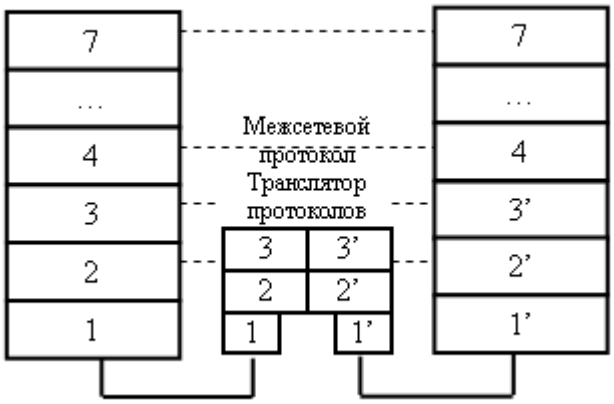

Рисунок 6 – Уровни модели OSI

Недостатком использования шлюзов является только одна точка взаимодействия 2-х сетей, что снижает надежность доставки сообщений из одной сети в другую.

Управление потоком

На всех этапах передачи данных должна осуществляться согласованность скорости поступления сообщений в узлы СПД.

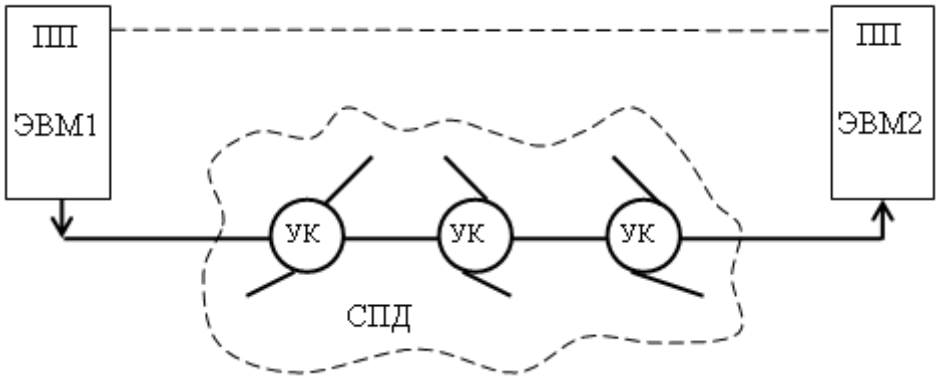

Рисунок 7 – Сеть передачи данных

Скорость передачи сообщений нужно выбирать такой, чтобы ресурсы сети (буферная память и пропускная способность канала) не были перегружены. Если буферная память двух соседних узлов полностью заполнена, то узлы переходят в состояние блокировки и ни принять сообщение, ни отправить не могут.

Перегрузки в сети

С увеличением числа пакетов, передаваемых сетью, возрастает время доставки, производительность при этом сначала возрастает до максимального значения а затем начинает падать. Состояние сети, при котором из-за большого числа передаваемых пакетов резко ухудшаются характеристики сети, называется ***перегрузкой***. При большем, чем max числе

предаваемых пакетов, сеть оказывается заблокированной и интенсивность поступления сообщений на выходе равна нулю.

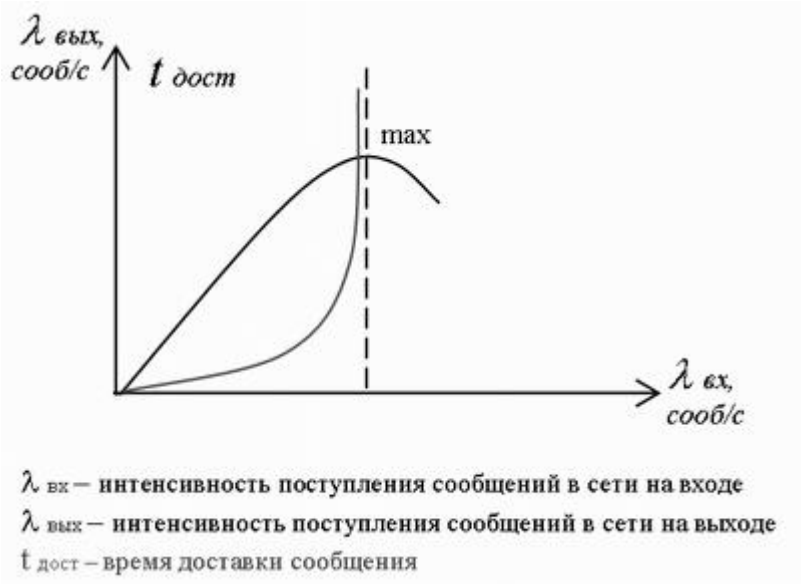

Рисунок 8 – Характеристики загруженности сети

Методы предотвращения блокировки

1. Введение для узлов сети системы разрешений на ввод пакетов в сеть. При этом каждому узлу выделяется ограниченное число разрешений на передачу пакетов в сеть. Если узел вводит пакет в сеть, число разрешений уменьшается на единицу. Когда все разрешения исчерпаны, узел прекращает прием пакетов от ЭВМ-источника. Когда в узел поступает пакет, адресованный ЭВМ, обслуживаемой узлом, число разрешений увеличивается на единицу. Этот механизм исключает возможность переполнения сети пакетами. Поскольку потоки в узлах не сбалансированы, число отправляемых пакетов в общем случае не совпадает с числом принимаемых), в одних узлах может оказаться избыток разрешений, а в других — их дефицит. Поэтому узлы должны передавать избыточные разрешения другим узлам, например, с помощью специальных управляющих пакетов.
2. Сброс сети. Уничтожение заблокированных пакетов.

Задание

1. Используя программное обеспечение «АСМ» собрать сеть (рисунок 9) из одного концентратора и четырех абонентских устройств. Сначала задать все адреса, только потом траффик.

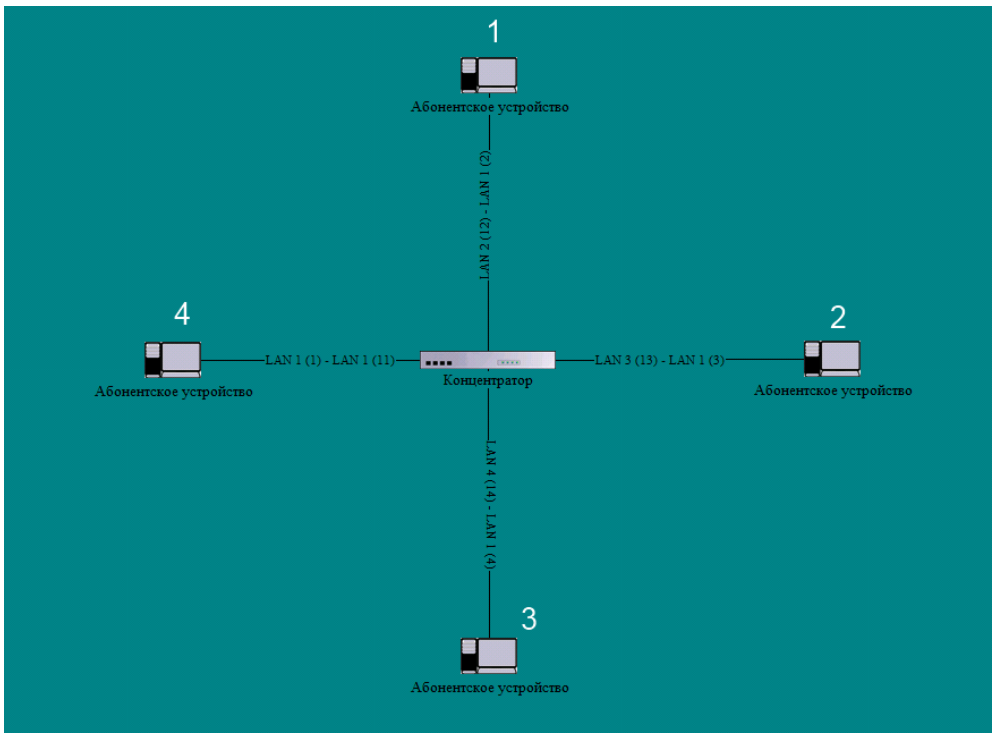

Рисунок 9

Задать трафик между абонентскими устройствами: 1-3, 2-4, 1-4.

Вероятности появления пакетов соответственно: 5+(*N* div 2); 10+(*N* div 2); 15+(*N* div 3), %.

(Целочисленное деление div (от division, деление) возвращает целую часть частного, а дробная часть отбрасывается — 13 div 3 = 4, а не 4,(3). Результат div всегда равен нулю, если делимое меньше делителя.)

Собрать статистику за 60 секунд.

2. Собрать сеть (рисунок 10) из трех концентраторов и шести абонентских устройств.

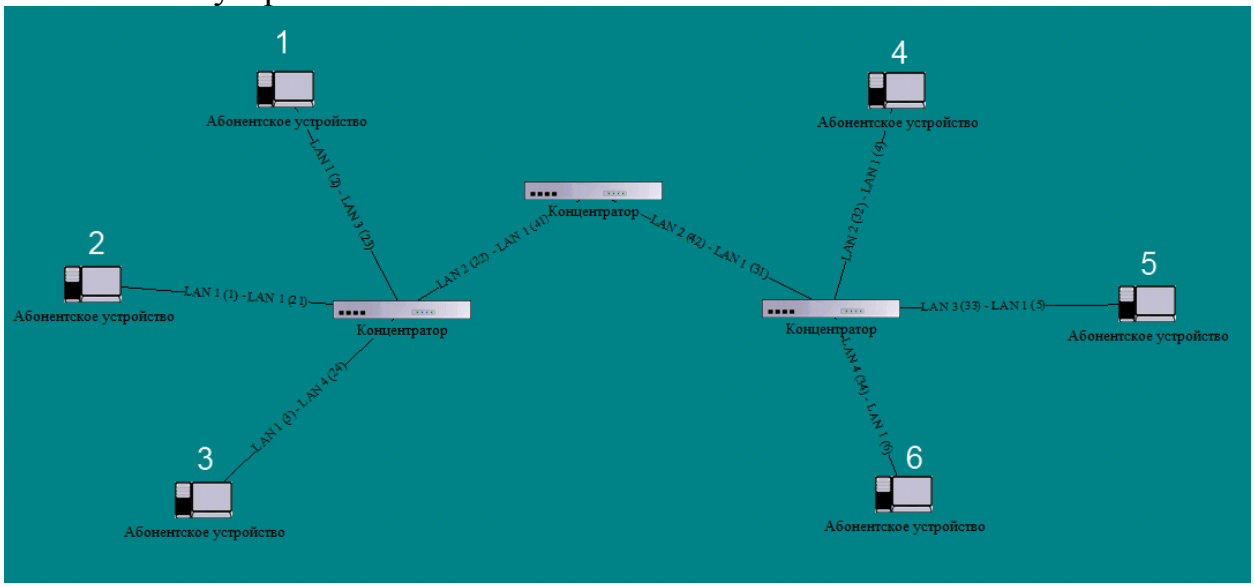

Рисунок 10

Задать трафик между абонентскими устройствами: 1-4, 2-5, 3-6.

Вероятности появления пакетов соответственно: 3+(*N* div 2); 7+(*N* div 2); 12+(*N* div 3), %.

Собрать статистику за 60 секунд.

3. Убрать два абонентских устройства (3 и 6) и соединить концентраторы, образовав кольцо (рисунок 11).

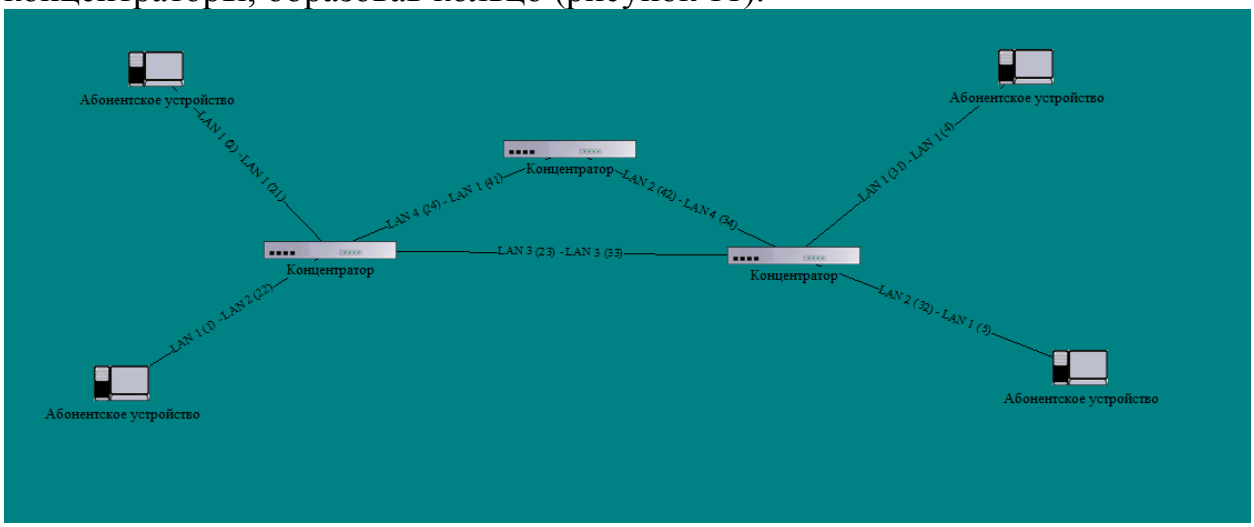

Рисунок 11

Исследовать работу сети.

4. Собрать сеть (рисунок 12) из двух концентраторов, одного моста и шести абонентских устройств.

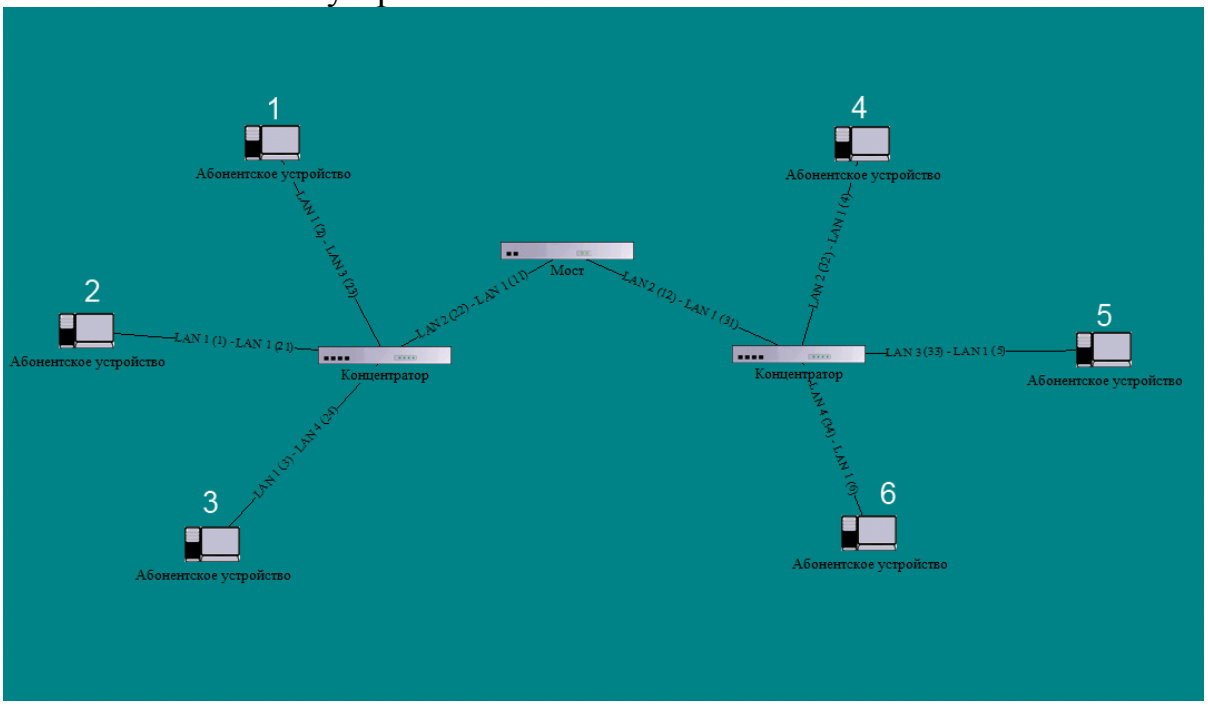

Рисунок 12

Задать трафик между абонентскими устройствами: 1-3, 2-5, 4-6.

Вероятности появления пакетов соответственно: 3+(*N* div 2); 7+(*N* div 2); 12+(*N* div 3), %.

Собрать статистику за 60 секунд.

5. Убрать два абонентских устройства (3 и 6) и соединить концентраторы, образовав кольцо (рисунок 13).

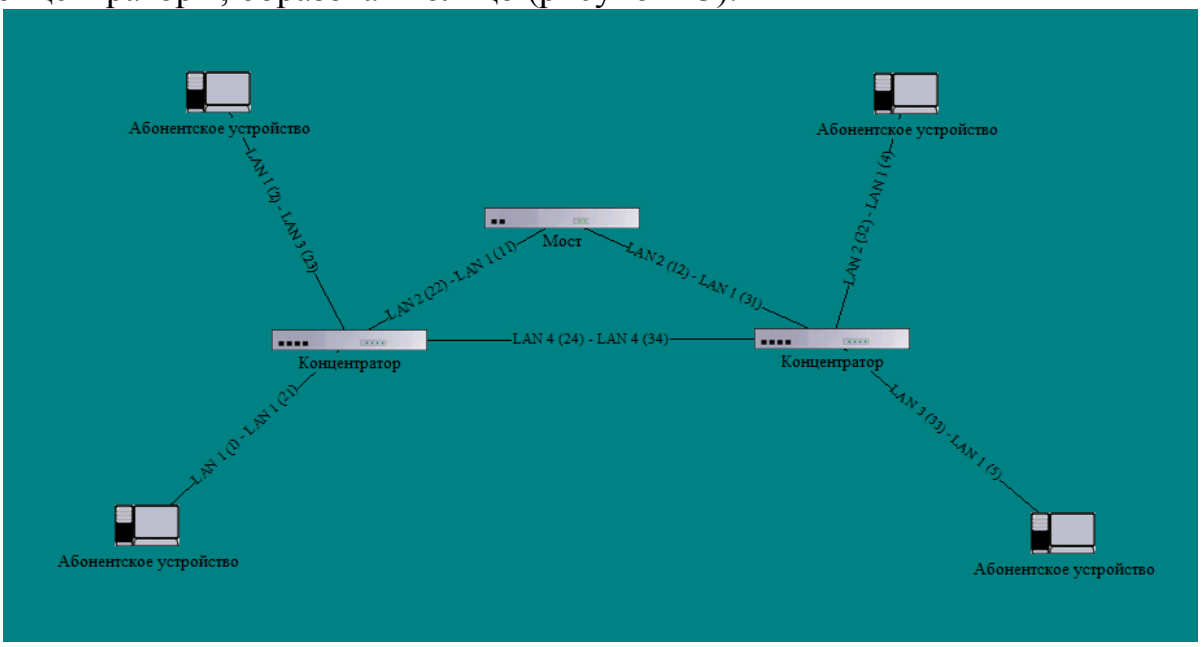

Рисунок 13

Исследовать работу сети.

6. Собрать сеть (рисунок 14) из одного коммутатора и четырех абонентских устройств.

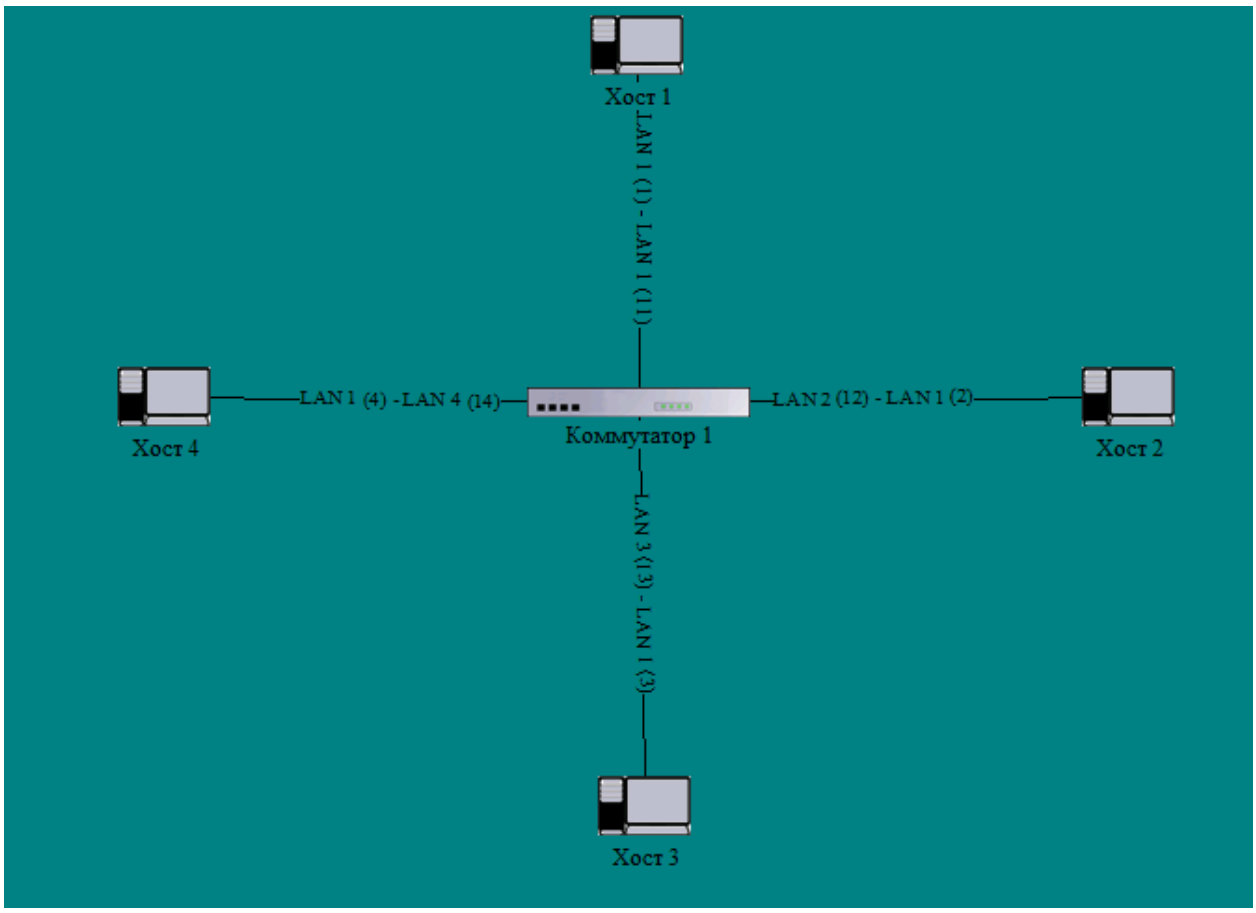

Рисунок 14

Задать трафик между абонентскими устройствами: 1-2, 1-3, 2-4.

Вероятности появления пакетов соответственно: 5+(*N* div 2); 10+(*N* div 2); 10+(*N* div 3), %.

Проверить работу сети (без таблицы маршрутизации).

Задать таблицу маршрутизации.

Собрать статистику на коммутаторе за 100 тактов.

7. Собрать сеть (рисунок 15) из четырех коммутаторов и восьми абонентских устройств.

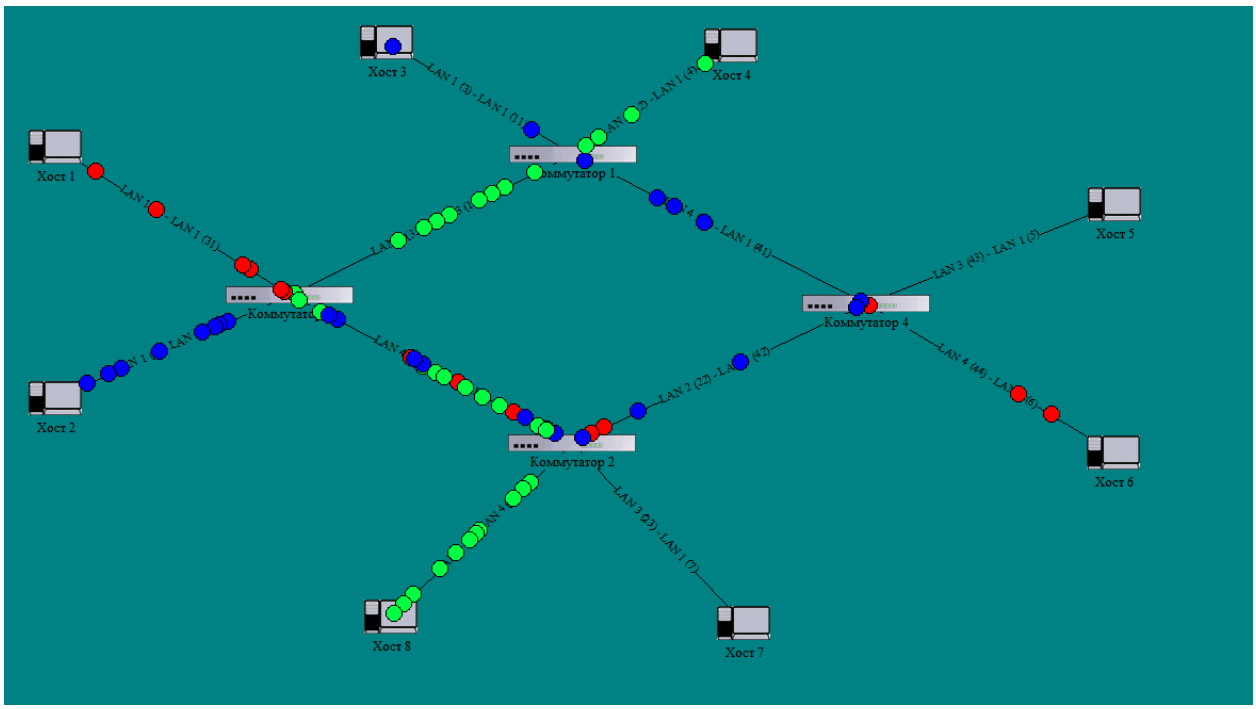

Рисунок 15

Задать трафик между абонентскими устройствами: 1-6, 2-3, 8-4.

Вероятности появления пакетов соответственно: 7+(*N* div 2); 10+(*N* div 2); 10+(*N* div 3), %.

Собрать статистику на коммутаторах за 100 тактов.

8. Закольцевать один из маршрутов (рисунок 16). Проанализировать работу сети.

Проверить изъятие пакетов после окончания его времени жизни.

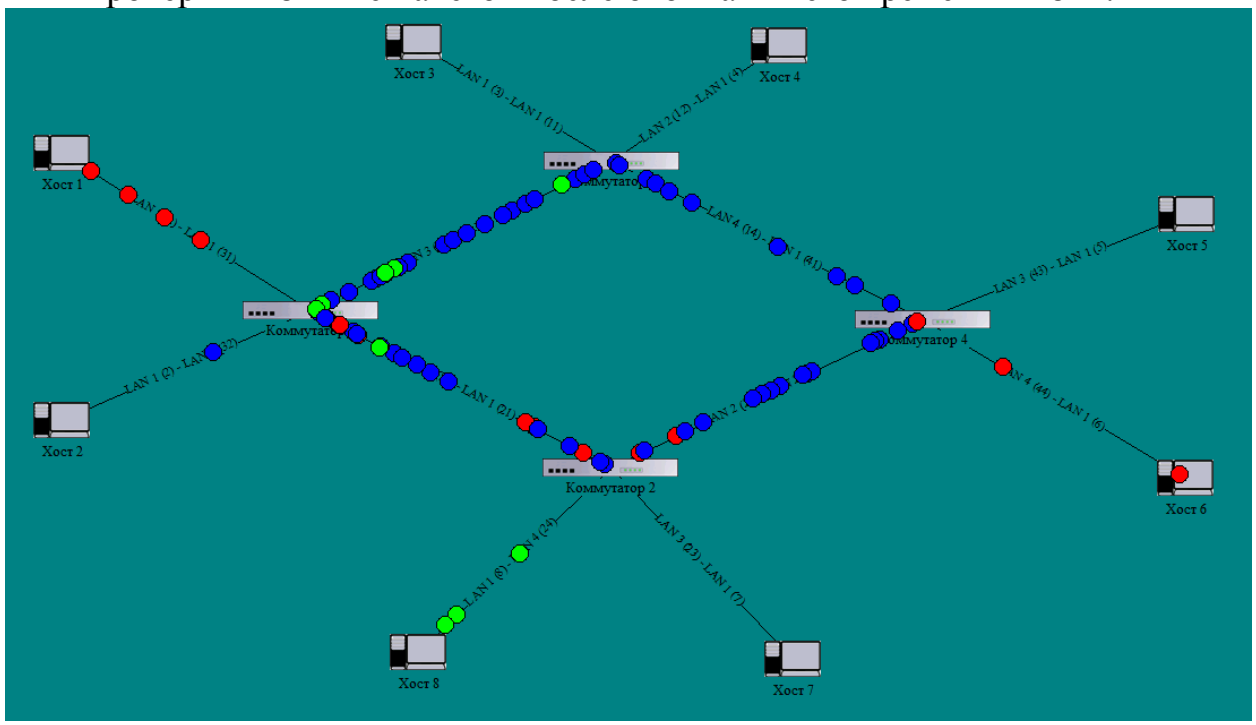

Рисунок 16

9. Удалить четвертый коммутатор (рисунок 17). Проанализировать работу сети.

Восстановить работу сети используя оставшиеся коммутаторы.

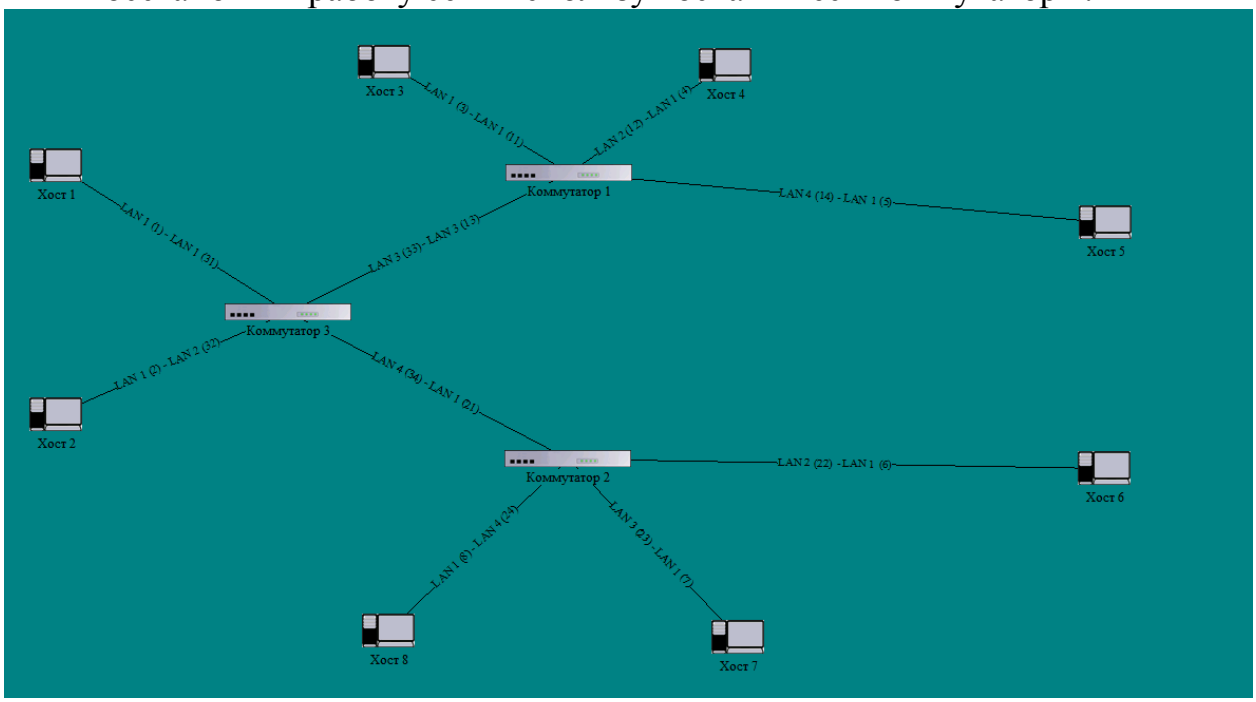

Рисунок 17

Собрать статистику на коммутаторах за 100 тактов.

10. Собрать сеть (рисунок 18) из пяти коммутаторов и восьми абонентских устройств.

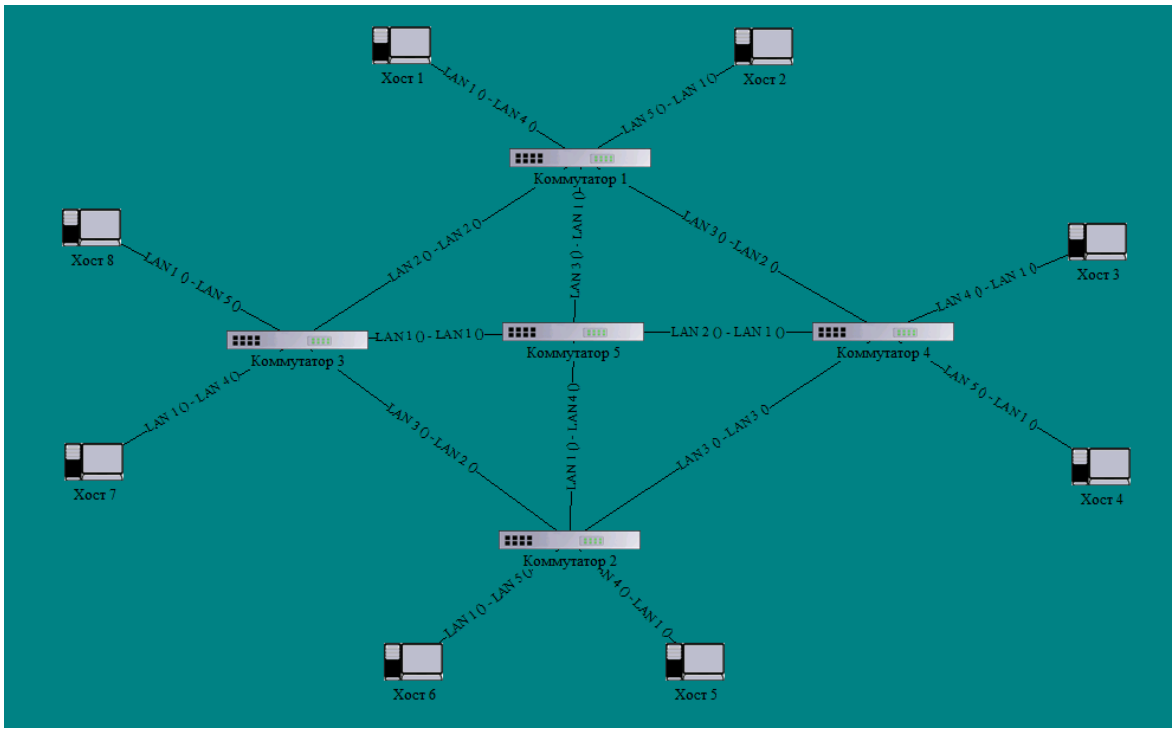

Рисунок 18

Задать трафик между абонентскими устройствами: 1-6, 3-7, 4-1, 7-4.

Вероятности появления пакетов соответственно: 3+(N div 2); 5+(N div 2); 5+(N div 3); 7+(N div 3), %.

Собрать статистику на коммутаторах за 100 тактов.

11. Сделать выводы.