Решение проблем, связанных с фрагментацией IP, а также с MTU, MSS и PMTUD, при помощи протоколов GRE и IPSEC

Содержание

Введение

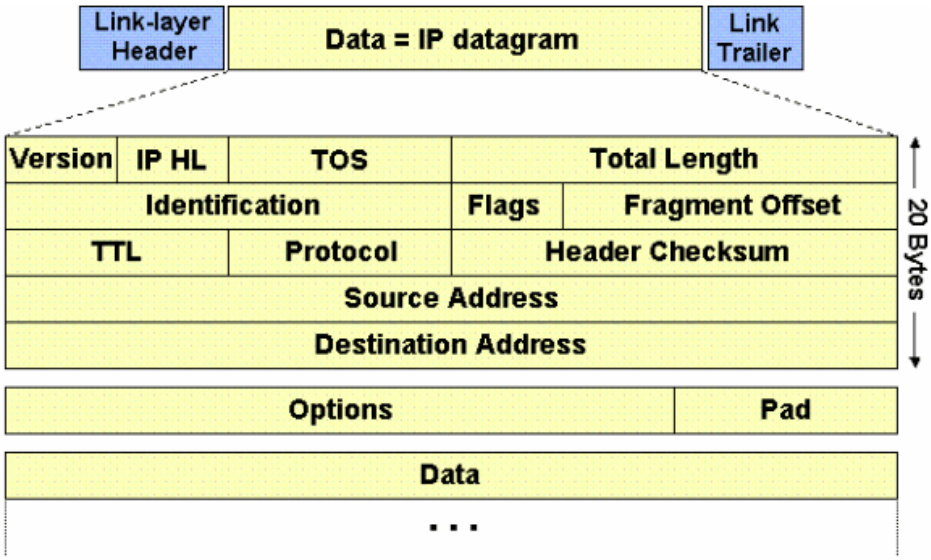
Цель этого документа – описать, как работает IP-фрагментация и алгоритм PMTUD, а также рассмотреть ряд сценариев, включая поведение алгоритма PMTUD с различными комбинациями IP-туннелей. В результате широкого распространения IP-туннелей в Интернете на передний план вышли проблемы IP-фрагментации и PMTUD.

Фрагментация и сборка IP

Семейство протоколов IP было создано для использования в самых различных каналах передачи данных. Несмотря на то что максимальная длинна IP-датаграммы равна 64 Кбайт, в большинстве каналов передачи данных используется меньшее значение. Это значение называется «максимально допустимый размер пакета» (maximum transmission unit – MTU). Величина MTU зависит от типа канала связи. Структура IP учитывает эти различия, позволяя маршрутизаторам фрагментировать IP-датаграммы в соответствии с требованиями канала связи. Приемная станция отвечает за восстановление исходной IP-датаграммы (сборку фрагментов).

Процедура фрагментации заключается в разбивке единой датаграммы на несколько частей, которые позже будут собраны вместе. При IP-фрагментации и сборке используются следующие параметры: источник IP, назначение, идентификация, общая длина и поля смещений фрагментов, а также флаги в заголовке IP – more fragments (еще фрагменты) и don't fragment (не фрагментировать). Более подробно о процессах IP-фрагментации и повторной сборки см. в документе RFC 791 .

На рисунке ниже показана схема IP-заголовка.



Идентификация представляет собой 16-битовое значение, присвоенное отправителем IP-датаграммы для помощи в повторной сборке фрагментов датаграммы.

Поле смещения фрагмента составляет 13 бит и указывает область, к которой принадлежит фрагмент в исходной IP-датаграмме. Это значение является кратным восьми байтам.

Во флаговых полях IP-заголовка три бита используются под управляющие флаги. Здесь необходимо отметить, что флаг DF (don't fragment – не фрагментировать) играет главную роль в алгоритме PMTUD: он сообщает о том, можно ли фрагментировать пакет или нет.

Бит 0 зарезервирован. Его значение всегда равно 0. Бит 1 – это флаг DF (0 = фрагментация разрешена, 1 = фрагментация запрещена). Бит 2 – это флаг MF (0 = последний фрагмент, 1 = еще фрагменты).

Значение	Бит 0 (зарезервирован)	Бит 1 DF	Бит 2 MF
0	0	Разрешено	Последний
1	0	Запрещено	Не последний

На следующем рисунке показан пример фрагментации. При сложении всех величин IP-фрагментов конечное значение будет больше исходной величины IP-датаграммы на 60 байт. Такое увеличение связано с тем, что в ходе фрагментации были созданы три дополнительных IP-заголовка, по одному для каждого фрагмента, который следует после первого фрагмента.

Первый фрагмент имеет нулевое смещение и длину 1500 (сюда входят 20 байт, добавившихся в связи с незначительной модификацией исходного IP-заголовка).

Смещение второго фрагмента составляет 185 (185 x 8 = 1480). Это означает, что блок данных этого фрагмента начинается с 1480 байта исходной IP-датаграммы. Данный фрагмент имеет длину равную 1500 (сюда также входит дополнительный IP-заголовок, созданный для этого фрагмента).

Смещение третьего фрагмента – 370 (370 x 8 = 2960), что означает, что блок данных этого фрагмента начинается с 2960 байта исходной IP-датаграммы. Данный фрагмент имеет длину равную 1500 (сюда также входит дополнительный IP-заголовок, созданный для этого фрагмента).

Четвертый фрагмент имеет смещение 555 (555 x 8 = 4440), что означает, что порция данных этого фрагмента начинается с 4440 байта исходной IP-датаграммы. Данный фрагмент имеет длину равную 700 (сюда также входит дополнительный IP-заголовок, созданный для этого фрагмента).

Определить размер исходной датаграммы можно только тогда, когда получен последний фрагмент.

Значение смещения последнего фрагмента (555) дает смещение, равное 4440 байтам, относительно исходной IP-датаграммы. Если к этом значению добавить байты данных из последнего фрагмента (680 = 700 - 20), то мы получим 5120 байт, это длина исходной IP-датаграммы. В этом случае при добавлении 20 байтов к заголовку IP получается IP-датаграмма исходного размера (4440 + 680 + 20 = 5140).

Original IP Datagram

Sequence	Identifier	Total Length	DF May / Don't	MF Last / More	Fragment Offset
0	345	5140	0	0	0

IP Fragments (Ethernet)

Sequence	Identifier	Total Length	DF May / Don't	MF Last / More	Fragment Offset
0-0	345	1500	0	1	0
0-1	345	1500	0	1	185
0-2	345	1500	0	1	370
0-3	345	700	0	0	555

Проблемы фрагментации IP

Существует ряд проблем, которые делают нежелательным использование IP-фрагментации. При разбивке IP-датаграммы нагрузка на ЦП и память несколько возрастает. Это верно как для отправителя, так и для маршрутизатора, находящегося на пути между отправителем и получателем. Процесс фрагментации фактически состоит из создания заголовков для фрагментов и копирования данных исходной датаграммы в эти фрагменты. Этот относительно эффективный процесс, поскольку информация, необходимая для создания фрагментов, уже имеется в готовом виде.

Однако при сборке фрагментов нагрузка на получателя значительно увеличивается, поскольку получатель должен выделить память для поступающих фрагментов и, после того как все фрагменты получены, объединить их в исходную датаграмму. Сборка фрагментов на узле не представляет никаких трудностей, поскольку узел обладает достаточными ресурсами (время и память) для выполнения этого задания.

Однако сборка фрагментов на маршрутизаторе, первоочередной задачей которого является переадресация пакетов с максимально возможной скоростью, представляется малоэффективной. Маршрутизатор не предназначен для «удержания» пакетов в течение какого-либо периода времени. Кроме того, маршрутизатор, выполняющий сборку фрагментов, должен выделять под этот процесс самый большой буфер обмена (18 Кбайт), поскольку размер исходного IP-пакета остается для него неизвестным до тех пор, пока не получен последний фрагмент.

Другая проблема, связанная с фрагментацией, относится к обработке «выпавших» фрагментов. При выпадении (утрате) одного из фрагментов IP-датаграммы требуется повторная отправка всей датаграммы, которая также должна пройти фрагментирование. Рассмотрим эту проблему на примере сетевой операционной системы NFS (Network File System). По умолчанию размер блока чтения и записи данных в NFS равен 8192 байт. Таким образом, в NFS размер IP/UDP-датаграммы приблизительно составит 8500 байт, включая NFS-, UDP- и IP-заголовки. Отправляющая станция, которая подключена к сети Ethernet (MTU = 1500 байт), должна будет разбить датаграмму размером в 8500 байт на шесть частей (пять фрагментов по 1500 байт и один фрагмент – 1100 байт). Если из-за перегрузки канала данных какой-либо фрагмент будет утерян, то отправитель будет вынужден повторно отправить всю исходную датаграмму, а это означает, что потребуется повторная разбивка этой датаграммы на шесть частей. Если из шести пакетов канал сбрасывает один, то вероятность того, что по этому каналу можно передать какие-либо NFS-данные, очень мала, поскольку по крайней мере один IP-фрагмент будет сбрасываться из каждой исходной 8500-байтововой IP-датаграммы.

У брандмауэров, которые фильтруют пакеты или манипулируют ими, используя для этого данные с 4 по 7 уровень (L4 – L7) модели OSI, могут возникнуть проблемы с корректной обработкой IP-фрагментов. При наличии повреждений в IP-фрагментах брандмауэр может заблокировать непервоначальные фрагменты, поскольку в них не содержится информация, которая соответствует установкам пакетного фильтра. Это означает, что принимающий узел не сможет собрать исходную IP-датаграмму. Если брандмауэр настроен таким образом, что он позволяет неначальным фрагментам, имеющим достаточное количество информации, должным образом проходить через фильтр, тогда создается угроза атаки, использующей неначальные пакеты. Кроме того, некоторые сетевые устройства

(например устройства распределения нагрузки – Content Switch Engines) распределяют пакеты на основе данных с 4-го по 7-й уровень. Соответственно, если пакет разбит на множественные фрагменты, у получающего устройства могут возникнуть проблемы с реализацией политик.

Как избежать использования IP-фрагментации: что такое максимальный размер сегмента протокола TCP и как это работает

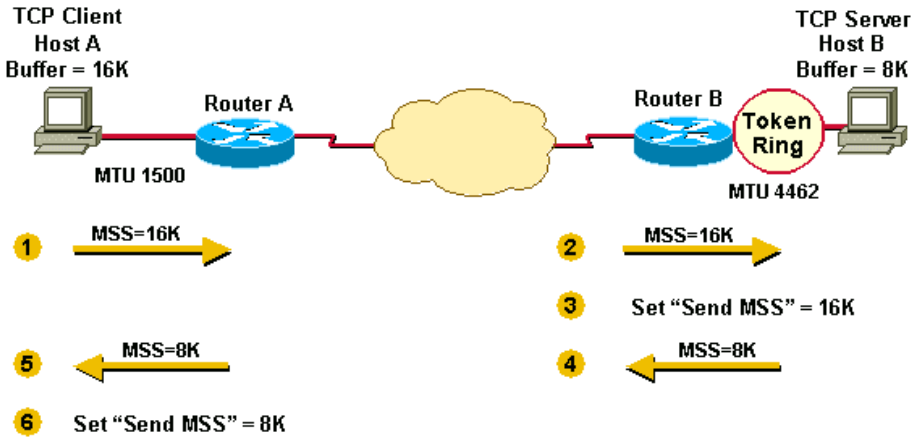
Максимальный размер сегмента TCP (MSS) определяет максимальный объем данных, который узел намеревается принять в одной TCP/IP-датаграмме. Эта TCP/IP-датаграмма может быть фрагментирована на уровне IP. Значение MSS может передаваться в качестве опции TCP-заголовка только в сегментах SYN. Каждая сторона, участвующая в TCP-соединении, передает другой стороне значение MSS. В противоположность расхожему мнению, узлы значение MSS не согласуют. Отправляющий узел должен ограничить размер данных, пересылаемых в одном TCP-сегменте. Это значение не должно превышать значения MSS, которое было сообщено принимающим узлом.

Первоначально MSS определял, насколько большой буфер (больше или равный 65496K) расположен на получающей станции для хранения данных TCP, содержащихся внутри простой IP-датаграммы. MSS был максимально допустимым сегментом (блоком) данных, которые TCP-получатель был согласен принять. Размер данного сегмента TCP мог достигать 64 кбайт (максимальный размер IP-датаграммы) и мог фрагментироваться на уровне IP для последующей передачи через сеть принимающему узлу. Перед передачей полного сегмента TCP на уровень TCP узел-получатель пересобирал IP-датаграмму.

Ниже приводятся несколько примеров, демонстрирующих, каким образом используются значения MSS для того, чтобы ограничить размер TCP-сегментов и, соответственно, размер IP-датаграмм.

Пример 1 иллюстрирует первоначальное применение MSS. Размер буфера узла A составляет 16 кбайт, а размер буфера узла B – 8 кбайт. Узлы отправляют и получают значения MSS и настраивают собственные отправляемые MSS для передачи данных друг другу. Обратите внимание, что узла A и узла B придется фрагментировать IP-датаграммы, размер которых больше размера блока MTU на интерфейсе, но меньше отправленного значения MSS, потому что TCP-стек может передавать данные размеров в 16 кбайт или 8 кбайт вниз на стек IP. В случае узла B пакеты придется фрагментировать дважды: один раз – перед передачей в сеть Token Ring и второй раз – перед передачей в сеть Ethernet.

Сценарий 1



1. Узел A отправляет свое значение MSS (16 кбайт) узлу B.
2. Узел B получает от узла A значение MSS равное 16 кбайт.
3. Узел B устанавливает значение MSS для отправки равное 16 кбайт.
4. Узел B отправляет узлу A значение MSS равное 8 кбайт.
5. Узел A получает от узла B значение MSS равное 8 кбайт.
6. Узел A устанавливает значение MSS для отправки равное 8 кбайт.

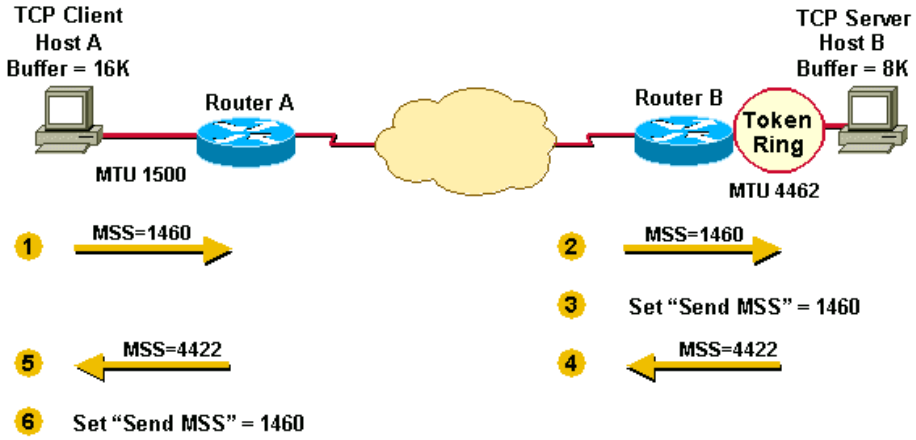
Для того чтобы избежать IP-фрагментации на конечных точках TCP-соединения, значение MSS было установлено равным минимальному размеру буфера и размеру MTU исходящего интерфейса (-40). Числовые значения MSS обычно на 40 байт меньше

числовых значений MTU, поскольку MSS представляет собой обычный размер данных в TCP (сюда не входят 20-байтовые заголовки IP и 20-байтовые заголовки TCP). Значение MSS основано на размерах заголовков, установленных по умолчанию: стек отправителя должен вычесть соответствующие значения для IP- и TCP-заголовков в зависимости от того, какие опции TCP и IP используются.

Способ работы MSS в данный момент заключается в том, что каждый узел сначала сравнивает MTU исходящего интерфейса с собственным буфером и выбирает наименьшее значение для отправки MSS. Затем на узлах сравнивается размер полученного MSS и размер MTU собственного интерфейса, после чего вновь выбирается меньшее из значений.

Сценарий 2 иллюстрирует эти дополнительные действия, выполняемые отправителем с целью избежать фрагментации на локальной и удаленной линиях. Обратите внимание на то, каким образом каждый из узлов учитывает значение MTU исходящего интерфейса (еще до того, как узлы обменяются значениями MSS) и как это позволяет избежать фрагментации.

Сценарий 2



1. Узел A сравнивает свой буфер MSS (16 кбайт) и свой MTU (1500 - 40 = 1460 байт) и для отправки узлу B качестве значения MSS использует наименьшее значение (1460).
2. Узел B получает от узла A значение MSS для отправки (1460) и сравнивает это значение со значением MTU своего исходящего интерфейса MTU - 40 (4422).
3. Узел B устанавливает наименьшее значение (1460) в качестве MSS для отправки IP-датаграмм узлу A.
4. Узел B сравнивает значение своего буфера MSS (8 кбайт) и своего MTU (4462 - 40 = 4422) и в качестве MSS, которое будет отправлено узлу A, использует значение 4422.
5. Узел A получает от узла B значение MSS для отправки (1460) и сравнивает это значение со значением MTU своего исходящего интерфейса MTU -40 (4422).
6. Узел A устанавливает наименьшее значение (1460) в качестве MSS для отправки IP-датаграмм узлу B.

1460 – это значение, выбранное обоими узлами в качестве посылки MSS друг другу. Очень часто на обеих концах TCP-соединения выбирается одно и то же значение MSS для отправки.

В сценарии 2 удалось избежать фрагментации на обеих концах TCP-соединения, поскольку узлы учли значения MTU для обоих исходящих интерфейсов. Однако пакеты могут подвергнуться фрагментации в сети между маршрутизатором A и маршрутизатором B. Это произойдет в том случае, если они попадут в канал, значение MTU которого меньше значения MTU на исходящем интерфейсе какого-либо из узлов.

Что такое PMTUD?

Как описано выше, TCP MSS отвечает за фрагментацию в двух конечных точках подключения TCP. Но он не работает в том случае, если канал с меньшим MTU располагается посередине между этими конечными точками. Алгоритм PMTUD позволяет избежать фрагментации на пути между конечными точками. Он позволяет в динамическом режиме определить наименьшее значение MTU на всем пути от источника пакета до точки назначения.

Примечание: PMTUD поддерживается только в протоколе TCP. Протокол UDP и другие протоколы его не поддерживают. Если на узле включен PMTUD (а чаще всего так оно и бывает), тогда во всех пакетах TCP/IP с этого узла будет установлен бит DF.

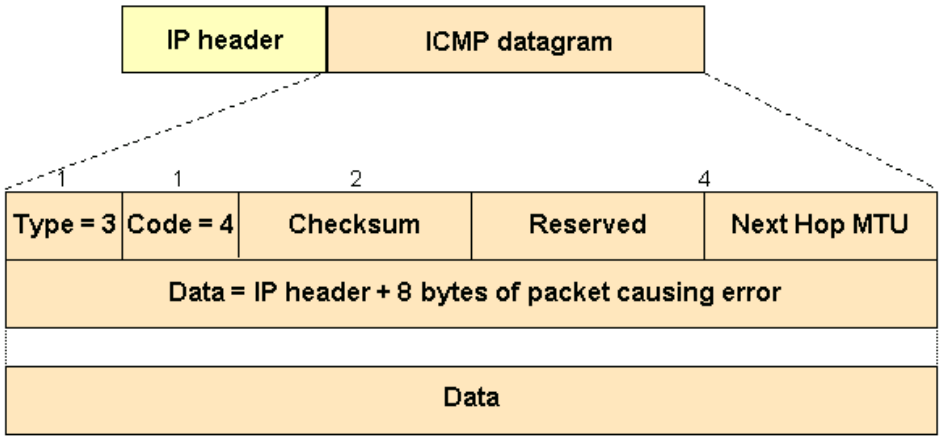
Когда узел отправляет полный MSS-пакет данных с набором битов DF, алгоритм PMTUD уменьшает значение отправки MSS для данного соединения, если он получил сведения о том, что пакету необходима фрагментация. Обычно узел запоминает значение MTU для конечного адреса, создавая в таблице маршрутов запись об узле (/32) с этим значением MTU.

Если маршрутизатор попытается переадресовать IP-датаграмму с установленным битом DF по каналу, значение MTU которого меньше размера передаваемого пакета, то в этом случае маршрутизатор сбросит пакет и отправит источнику этой IP-датаграммы ICMP-сообщение «Конечный адрес не может быть достигнут», в котором будет содержаться информационный код «необходима фрагментация и установлен бит DF» (тип 3, код 4). Получив ICMP-сообщение, исходная станция уменьшает посланный MSS, а когда TCP будет передавать этот сегмент повторно, то станция будет использовать меньший размер сегмента.

Ниже приведен пример ICMP-сообщения «необходима фрагментация и установлен бит DF», которое можно видеть после включения команды **debug ip icmp**:

```
ICMP: dst (10.10.10.10) frag. needed and DF set
unreachable sent to 10.1.1.1
```

На диаграмме ниже показана структура заголовка ICMP «необходима фрагментация и установлен бит DF» сообщения «Назначение недоступно».



В соответствии с документом RFC 1191 , маршрутизатор, отправляющий сообщение «требуется фрагментация и установлен бит DF», должен включать значение MTU для следующего сетевого перехода. Для этого сообщения используются 16 бит низшего разряда в поле дополнительного заголовка ICMP, которое в спецификации по ICMP (RFC 792) помечено как неиспользуемое.

В ранних версиях документа RFC 1191 информация о значении MTU следующего сетевого перехода на указывается. Даже если эта информация и предоставлялась, некоторые узлы ее игнорировали. На этот случай в RFC 1191 также содержится таблица, в которой указаны рекомендуемые значения, которые должны вычитаться из значения MTU при работе PMTUD. Эта таблица используется узлами для того, чтобы быстрее определить необходимое значение для MSS отправки.

Plateau	MTU	Comments	Reference
	65535	Official maximum MTU	RFC 791
	65535	Hyperchannel	RFC 1044
65535			
32000		Just in case	
	17914	16Mb IBM Token Ring	ref. [6]
17914			
	8166	IEEE 802.4	RFC 1042
8166			
	4464	IEEE 802.5 (4Mb max)	RFC 1042
	4352	FDDI (Revised)	RFC 1188
4352 (1%)			
	2048	Wideband Network	RFC 907
	2002	IEEE 802.5 (4Mb recommended)	RFC 1042
2002 (2%)			
	1536	Exp. Ethernet Nets	RFC 895
	1500	Ethernet Networks	RFC 894
	1500	Point-to-Point (default)	RFC 1134
	1492	IEEE 802.3	RFC 1042
1492 (3%)			
	1006	SLIP	RFC 1055
	1006	ARPANET	BBN 1822
1006			
	576	X.25 Networks	RFC 877
	544	DEC IP Portal	ref. [10]
	512	NETBIOS	RFC 1088
	508	IEEE 802/Source-Rt Bridge	RFC 1042
	508	ARCNET	RFC 1051
508 (13%)			
	296	Point-to-Point (low delay)	RFC 1144
296			
68		Official minimum MTU	RFC 791

В связи с динамическим изменением пути между отправителем и получателем, процесс PMTUD выполняется непрерывно для всех пакетов. Каждый раз когда отправитель получает ICMP-сообщение Can't Fragment (фрагментация невозможна), он начинает обновлять маршрутную информацию (в том месте, где у него хранится PMTUD).

Во время PMTUD может произойти следующее:

- Пакет может пройти весь путь до получателя, не подвергаясь фрагментации.

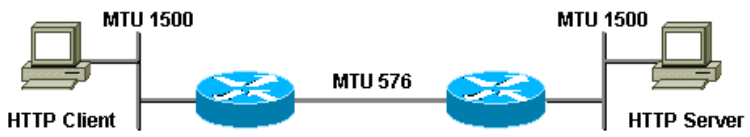
  Примечание: Для того чтобы защитить свой ЦП от DoS-атак, маршрутизатор дозирует количество отправляемых им ICMP-сообщений о недостижимости конечного пункта – 2 сообщения в секунду. Таким образом, если в вашей сети могут быть ситуации, когда маршрутизатор должен отвечать более чем двумя ICMP-сообщениями (код = 3, тип = 4) в секунду (например, для разных узлов), то в этом случае может понадобиться отключить дозирование (ограничение) ICMP-сообщений с помощью команды **no ip icmp rate-limit unreachable [df] interface**.

- Отправитель может получать сообщения ICMP Can't Fragment с любого (или каждого) перехода, расположенного на пути к приемнику.

PMTUD выполняется независимо для обоих направлений потока TCP. Могут быть случаи, когда PMTUD в одном направлении потока запускает одну из конечных станций для понижения отправляемых MSS, а другая конечная станция хранит начальное MSS, так как она никогда не посылала достаточно большую для запуска PMTUD IP-датаграмму.

Хорошим примером такой ситуации является HTTP-соединение, изображенное ниже (сценарий 3). Клиент TCP отправляет малые пакеты, а сервер отправляет большие пакеты. В этом случае процесс PMTUD будет инициирован только посредством больших пакетов, отправляемых сервером (более 576 байт). Клиентские пакеты невелики (меньше 576 байт), и, соответственно, они не запустят PMTUD, потому что они не требуют фрагментации для передачи через канал связи, имеющий MTU равный 576 байтам.

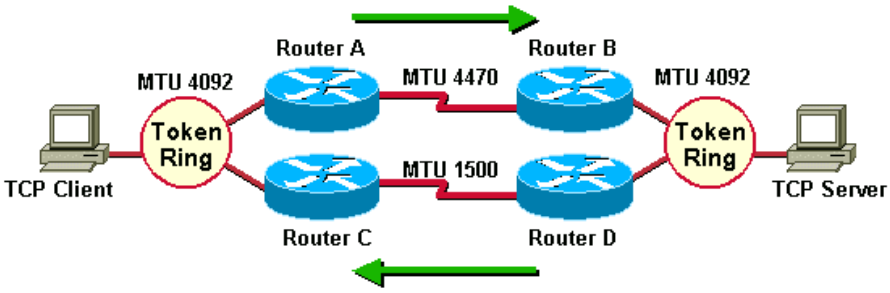
Сценарий 3



Сценарий 4 иллюстрирует пример ассиметричной маршрутизации, при которой один из путей имеет меньшее значение MTU, чем другой. Ассиметричная маршрутизация имеет место тогда, когда для отправки и получения данных между двумя конечными точками, используются два разных пути. В данном сценарии процесс PMTUD инициирует уменьшение значения MSS отправки (send MSS) только в одном направлении потока TCP. Трафик, отправляемый TCP-клиентом на сервер, проходит через маршрутизаторы A и B, в то время как обратный трафик, посылаемый сервером, проходит через маршрутизаторы D и C. При отправке TCP-сервером пакетов клиенту PMTUD инициирует уменьшение значения MSS отправки, поскольку маршрутизатор D должен будет фрагментировать 4092-байтные пакеты перед их оправкой на маршрутизатор C.

С другой стороны, клиент никогда не получит сообщение ICMP Destination Unreachable с кодом «необходима фрагментация и установлен бит DF», поскольку маршрутизатор A не должен фрагментировать пакеты при отправлении их на сервер через маршрутизатор B.

Сценарий 4



Примечание: Команда **ip tcp path-mtu-discovery** используется для активации функции обнаружения пути TCP MTU для соединений TCP, инициированных маршрутизаторами (например, BGP и Telnet).

Проблемы, связанные с PMTUD

Существует три причины нарушения PMTUD, две из которых встречаются очень редко, а третья – часто.

- Маршрутизатор может сбросить пакет и не отправить ICMP-сообщение. (редко)
- Маршрутизатор может сформировать и отправить ICMP-сообщение, которое, однако, будет заблокировано маршрутизатором или брандмауэром на пути между маршрутизатором и получателем. (часто)
- Маршрутизатор может сформировать и отправить ICMP-сообщение, однако получатель его проигнорирует. (редко)

Первая и последняя из этих трех меток обычно не встречаются и появляются в результате ошибки, а средняя описывает распространенную проблему. Пользователи, использующие фильтры ICMP-пакетов, стремятся заблокировать ICMP-сообщения всех типов, а не только отдельные виды этих сообщений. Фильтр пакетов может заблокировать все виды ICMP-сообщений, *кроме* тех, которые содержат сообщение unreachable (достичь невозможно) или time-exceeded (время истекло). Успех или неудача PMTUD зависит от ICMP-сообщений о недоступности, проходящих к отправителю пакета TCP/IP. ICMP-сообщения time-exceeded играют важную роль в других случаях, связанных с IP. Ниже приведен пример фильтра пакетов, который реализован на маршрутизаторе.

```
access-list 101 permit icmp any any unreachable
access-list 101 permit icmp any any time-exceeded
access-list 101 deny icmp any any
access-list 101 permit ip any any
```

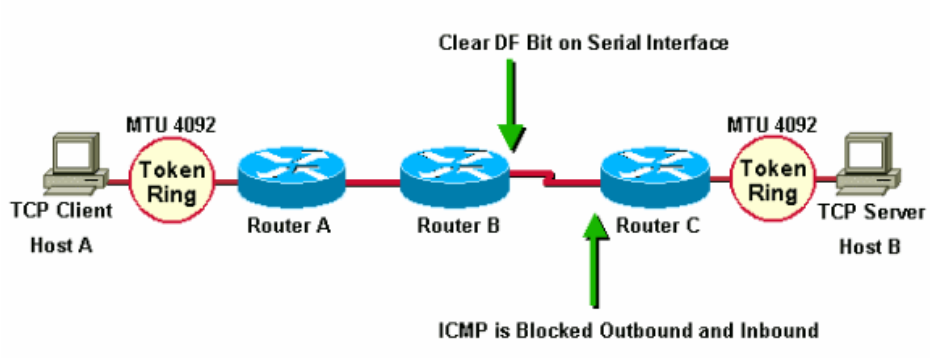
Существуют и другие способы уменьшения проблем полной блокировки ICMP.

- Сбросьте бит DF на маршрутизаторе и разрешите фрагментацию в любом случае (хотя это может быть не слишком удачная идея). Более подробно см. в разделе Проблемы, связанные с IP-фрагментацией.
- Значением MSS опции TCP MSS можно управлять при помощи интерфейсной команды **ip tcp adjust-mss <500-1460>**.

В приводимом ниже сценарии 5 маршрутизатор A и маршрутизатор B находятся в одном административном домене. Маршрутизатор C недоступен и блокирует ICMP, поэтому PMTUD не работает. Чтобы исправить эту ситуацию, на маршрутизаторе B можно сбросить бит DF по обоим направлениями и тем самым разрешить фрагментацию. Это можно выполнить при помощи политики маршрутизации. Синтаксис, используемый для отключения бита DF, имеется в ПО Cisco IOS® начиная с выпуска 12.1(6) и выше.

```
interface serial0
...
ip policy route-map clear-df-bit
route-map clear-df-bit permit 10
match ip address 111
set ip df 0

access-list 111 permit tcp any any
```



Можно также попробовать изменить значение параметра TCP MSS для пакетов SYN, которые проходят сквозь маршрутизатор (доступен в Cisco IOS начиная с версии 12.2(4)T и выше). Это приводит к сокращению значения параметра MSS в пакете TCP SYN, которое становится меньше, чем значение (1460) в команде **ip tcp adjust-mss**. Результатом является то, что TCP-отправитель будет отправлять сегменты, размер которых не превышает это значение. Размер IP-пакета будет на 40 байт больше (1500 байт) значения MSS (1460 байт), чтобы вместить TCP-заголовок (20 байт) и IP-заголовок (20 байт).

Для изменения значения MSS для пакетов TCP SYN можно использовать команду **ip tcp adjust-mss**. Указанный ниже синтаксис позволяет уменьшить значение MSS в TCP-сегментах до 1460 байтов. Данная команда воздействует как на входящий, так и на исходящий трафик интерфейса serial0.

```
int s0
ip tcp adjust-mss 1460
```

С увеличением использования IP-туннелей проблемы, связанные с фрагментацией IP, получили большее распространение. Причина, по которой использование туннелей приводит к большей фрагментации, заключается в том, что в ходе туннельного инкапсулирования размер передаваемого пакета увеличивается. Например, при добавлении протокола IP-туннелирования GRE (Generic Router Encapsulation) размер пакета увеличивается на 24 байта. Это значит, что после такого увеличения может понадобиться фрагментация пакета, поскольку его размер превышает величину исходящего MTU. Ниже мы приведем примеры проблем, которые связаны с туннелированием и IP-фрагментацией.

Распространенные топологии сети, использующие PMTUD

PMTUD необходим в тех случаях, когда промежуточные каналы имеют меньшие MTU по сравнению с MTU конечных каналов. Некоторые общие причины существования каналов с меньшими значениями MTU:

- Конечные узлы, подключенные к сетям Token Ring или FDDI, которые соединены между собой посредством Ethernet. Величина MTU оконечных сетей Token Ring или FDDI больше величины MTU в промежуточных сетях Ethernet.
- Для PPPoE (часто используется с ADSL) требуется 8 байт для заголовка. Это снижает максимальный размер блока данных Ethernet до 1492 (1500 – 8).

Протоколы туннелирования (такие как GRE, IPsec и L2TP) также требуют места для размещения заголовков и трейлеров. Это также приводит к уменьшению фактического размера MTU исходящего интерфейса.

В последующих разделах будет рассмотрена работа PMTUD для случаев, когда протокол туннелирования расположен где-то между двумя оконечными узлами. Из всех трех случаев, что были рассмотрены выше, данный случай является наиболее комплексным: он иллюстрирует все проблемы, которые были рассмотрены в других примерах.

Что такое туннель?

Туннель – это расположенный на маршрутизаторе Cisco логический интерфейс, который предоставляет механизм для инкапсулирования пакетов-пассажиров (passenger packets) в транспортный протокол. Данный механизм предоставляет службы, которые позволяют реализовать схему инкапсуляции «точка – точка». Механизм туннелирования состоит из трех основных компонентов:

- транспортируемый протокол (AppleTalk, Banyan VINES, CLNS, DECnet, IP или IPX)
- транспортирующий протокол (один из протоколов инкапсуляции, приведенных ниже):
    - GRE – многопротокольный транспортирующий (несущий) протокол Cisco. Дополнительные сведения см. в документах RFC 2784 и RFC 1701 .
    - Туннели IP-in-IP – дополнительные сведения см. в документе RFC 2003 .
- транспортный протокол – протокол, используемый для переноса инкапсулированного протокола

Пакеты ниже иллюстрируют концепции IP-туннелирования, где GRE – протокол инкапсуляции, а IP – транспортный протокол. Транспортируемым протоколом также является протокол IP. В данном случае протокол IP является как транспортным, так и транспортируемым протоколом.

Обычный пакет

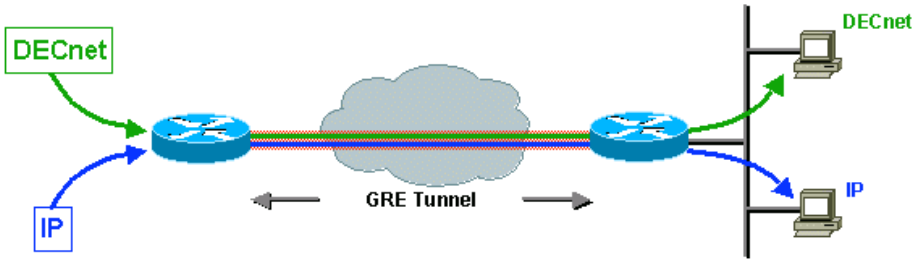
IP	TCP	Telnet

Туннельный пакет

IP	GRE	IP	TCP	Telnet

- IP – транспортный протокол.
- GRE – протокол инкапсуляции.
- IP – транспортируемый протокол.

В следующем примере показана инкапсуляция IP и DECnet как протоколов переноса с GRE в качестве носителя. Это иллюстрирует тот факт, что протокол несущей может инкапсулировать несколько протоколов переноса.



Сетевому администратору может понадобиться использовать туннелирование в том случае, кода имеются «несоприкасающиеся» сети, которые не используют IP и которые разделяются IP-магистралью. Если такие «несоприкасающиеся» сети используют протокол DECnet, то администратор по каким-либо причинам может отказаться соединять эти сети посредством настройки протокола DECnet в магистральной сети. Администратор может не захотеть, чтобы маршрутизация DECnet использовала пропускную способность магистрали по той причине, что это может повлиять на производительность сети IP.

В качестве возможной альтернативы можно осуществить туннелирование DECnet в магистральной IP-сети. В ходе туннелирования пакеты DECnet инкапсулируются в IP и пересылаются по магистрали к конечной точке туннеля, где после удаления инкапсуляции пакеты DECnet маршрутизируются к пункту назначения по сети DECnet.

Механизм инкапсулирования трафика в другой протокол дает следующие преимущества:

- Конечные точки используют свои частные адреса (RFC 1918), а магистраль не поддерживает маршрутизацию этих адресов.
- Возможно организовать виртуальные частные сети (VPN) в сетях WAN или в Интернете.
- Возможно объединение двух несоприкасающихся многопротокольных сетей посредством магистрали, которая использует один протокол.
- Используется шифрование трафика, передаваемого по магистральной сети или Интернету.

Далее во всех примерах IP будет использоваться как транспортируемый протокол (passenger protocol), а также как транспортный протокол.

Замечания, касающиеся туннельных интерфейсов

Далее приведены некоторые соображения по туннелированию.

- Функция быстрой коммутации GRE-туннелей впервые была реализована в программном обеспечении Cisco IOS 11.1, а коммутация CEF (Cisco Express Forwarding) впервые появилась в версии 12.0. Коммутация CEF для многоточечных GRE-туннелей впервые была реализована в версии 12.2(8)T. В более ранних версиях ПО IOS, которые поддерживали только коммутацию процессов, быстродействие функций инкапсуляции и декапсуляции было невысоким.
- При туннелировании пакетов необходимо учитывать вопросы безопасности и топологии. Туннели обладают способностью беспрепятственно миновать списки управления доступом (ACL) и брандмауэры. Если туннель проходит через брандмауэр, то это означает в сущности, что любой транспортируемый протокол беспрепятственно проходит через этот брандмауэр. По этой причине в конечных точках туннеля рекомендуется включить функцию брандмауэра, чтобы в принудительном порядке подвергать транспортируемые протоколы действию установленных политик.
- В связи с увеличением продолжительности задержки в процессе туннелирования могут возникнуть проблемы с транспортными протоколами, для которых предусмотрены ограничения по времени (например DECnet)
- Туннелирование в условиях работы с различными скоростями соединения, например быстрые кольца FDDI в сочетании с медленными (9600 бит/с) телефонными линиями, может вызывать проблемы с переупорядочиванием пакетов. Некоторые транспортируемые протоколы очень плохо работают в сетях со смешанными сетевыми носителями.
- PTP-туннели могут использовать всю пропускную способность физического канала. При использовании протоколов маршрутизации на нескольких PTP-туннелях необходимо помнить, что каждый туннельный интерфейс имеет пропускную способность и что физический интерфейс, через который проходит этот туннель, так же имеет свою пропускную способность. Например, если на 10-мегабитный канал приходится 100 туннелей, то требуется установить пропускную способность туннеля равную 100 кбит/с. Стандартная полоса пропускания для туннеля равна 9 кбит/с.
- Для протоколов маршрутизации предпочтительнее установка туннеля в «реальном» канале, поскольку в противном случае

туннель может обманчиво выглядеть как канал, имеющий только один сегмент ретрансляции и наименьшую стоимость пути, в то время как в действительности он имеет гораздо больше сегментов ретрансляции и является более «дорогим» по сравнению с другим путем. Эту проблему можно отчасти решить, настроив надлежащим образом протокол маршрутизации. Можно рассмотреть возможность использования другого протокола маршрутизации на интерфейсе туннеля – не использовать протокол маршрутизации, работающий на физическом интерфейсе.

- Проблем с рекурсивной маршрутизацией можно избежать путем настройки соответствующих статических маршрутов на получателя туннеля. Рекурсивный маршрут возникает тогда, когда наилучший путь до конечной точки туннеля проходит через сам туннель. Такая ситуация может привести к миганию туннельного интерфейса. При возникновении рекурсивной маршрутизации будет выводиться следующая ошибка.

```
%TUN-RECURDOWN Interface Tunnel 0
temporarily disabled due to recursive routing
```

Маршрутизатор как участник PMTUD в конечной точке туннеля

Когда маршрутизатор является конечной точкой туннеля, он выполняет две различные роли PMTUD.

- Во-первых, он перенаправляет пакеты узла. Для выполнения PMTUD-обработки маршрутизатор должен проверить размер и бит DF исходного пакета и при необходимости выполнить соответствующие действия.
- Вторая роль нужна, после того как маршрутизатор инкапсулирует оригинальный IP-пакет в пакет туннеля. На данном этапе маршрутизатор в основном выполняет роль узла по отношению к PMTUD и туннельному IP-пакету.

Сначала давайте посмотрим, что происходит с PMTUD, когда маршрутизатор выполняет первую задачу (перенаправляет IP-пакеты узла). Эта задача выполняется перед тем, как маршрутизатор инкапсулирует IP-пакет узла в пакет туннеля.

Когда маршрутизатор осуществляет переадресацию пакета узла, он выполняет следующие действия:

- Проверяет, установлен ли бит DF.
- Проверяет, пакет какого размера может быть передан по туннелю.
- Выполняет фрагментацию (в случае если пакет слишком большой и если DF бит не установлен), инкапсулирует фрагменты и отправляет их.
- Или сбрасывает пакет (если он слишком большой и если установлен бит DF) и посылает отправителю ICMP-сообщение.
- Инкапсулирует (если пакет не слишком большой) и отправляет.

В общем случае мы можем выбрать инкапсуляцию с последующей фрагментацией (отправка двух фрагментов инкапсуляции) либо фрагментацию с последующей инкапсуляцией (отправка двух инкапсулированных фрагментов).

Ниже приведены несколько примеров, в которых описан процесс инкапсуляции и фрагментации IP-пакетов, а также два сценария, которые описывают взаимодействие PMTUD с пакетами, которые передаются по сетям, диаграммы которых приведены ниже.

В первом примере показано, что происходит с пакетом, когда маршрутизатор, находящийся на входе в туннель, функционирует как переадресующий маршрутизатор. Необходимо помнить, что для PMTUD-обработки маршрутизатор должен проверить размер пакета и бит DF пакета, содержащего исходные данные, а также выполнить соответствующие действия. В данном примере для туннеля используется инкапсуляция GRE. Ниже можно видеть, что GRE выполняет фрагментацию перед инкапсуляцией. В примерах, которые будут приведены ниже, описываются сценарии, когда фрагментация выполняется после инкапсуляции.

В примере 1 бит DF не установлен (DF = 0), значение IP MTU для GRE-туннеля составляет 1476 байт (1500 - 24).

Пример 1

1. Перенаправляющий маршрутизатор (на туннельном источнике) получает 1500-байтную датаграмму с чистым битом DF (DF = 0) от отправляющего узла. Эта датаграмма включает 20-байтовый заголовок IP и 1480 байт полезных данных TCP.

IP	1480 байт TCP + данные

2. Поскольку после добавления служебных данных (24 байта) GRE размер пакета превысит допустимый предел MTU IP, маршрутизатор разобьет датаграмму на два фрагмента. Один фрагмент – 1476 байтов (20 байтов заголовок IP + 1456 байтов основные данные IP). Второй – 44 байта (20 байтов заголовок IP + 24 байта основные данные IP). Таким образом, после добавления инкапсуляции GRE, размер пакета не превысит значение MTU физического исходящего интерфейса.

IP_0	1456 байт TCP + данные

IP_1	24 байта данные

3. К каждому фрагменту исходной IP-датаграммы переадресующий маршрутизатор добавляет инкапсуляцию GRE: 4-байтовый заголовок GRE и 20-байтовый заголовок IP. Эти две IP-датаграммы имеют размер 1500 и 68 байт и отображаются как отдельные IP-датаграммы, а не как фрагменты.

IP	GRE	IP_0	1456 байт TCP + данные

IP	GRE	IP_1	24 байта данные

4. Маршрутизатор туннеля назначения удаляет инкапсуляцию GRE из каждого фрагмента исходной датаграммы, оставляя два фрагмента IP длиной 1476 и 24 байт. Данные фрагменты будут по отдельности переправлены этим маршрутизатором получающему узлу.

IP_0	1456 байт TCP + данные

IP_1	24 байта данные

5. Получающий узел соберет из этих фрагментов исходную датаграмму.

IP	1480 байт TCP + данные

В сценарии 5 описывается роль перенаправляющего маршрутизатора в контексте топологии сети.

В приведенном ниже примере маршрутизатор также выполняет роль перенапраялющего, но на этот раз бит DF установлен (DF = 1).

Пример 2

1. Перенаправляющий маршрутизатор в начале туннеля получает от передающего узла 1500-байтовую датаграмму с DF = 1.

IP	1480 байт TCP + данные

2. Поскольку бит DF установлен, размер датаграммы (1500 байт) больше, чем IP MTU туннеля GRE (1476), маршрутизатор будет отбрасывать датаграмму и отправлять сообщение ICMP fragmentation needed but DF bit set (Необходима фрагментация ICMP, но бит DF установлен). Из ICMP-сообщения отправитель узнает, что значение MTU – 1476 байт.

IP	ICMP MTU 1476

3. Отправляющий узел получает сообщение ICMP, и при повторной отправке исходных данных он будет использовать 1476-байтную IP-датаграмму.

IP	1456 байт TCP + данные

4. Теперь длина этой IP-датаграммы (1476 байт) по значению равна IP MTU туннеля GRE, поэтому маршрутизатор добавит к IP-датаграмме инкапсуляцию GRE.

IP	GRE	IP	1456 байт TCP + данные

5. Получающий маршрутизатор (в туннеле получателя) удаляет GRE-инкапсуляцию IP-датаграммы и отправляет ее на получающий узел.

IP	1456 байт TCP + данные

Следующий пример показывает, что происходит, когда маршрутизатор выполняет функцию отправляющего узла по отношению к PMTUD и туннельному IP-пакету. Напомним, что эти действия маршрутизатор начинает выполнять после инкапсуляции исходного IP-пакета в туннельный пакет.

Примечание: По умолчанию маршрутизатор не применяет PMTUD к генерируемым им туннельным пакетам GRE. С помощью команды **tunnel path-mtu-discovery** можно включить PMTUD для туннельных пакетов GRE-IP.

Ниже приведен пример, который описывает, что происходит в том случае, когда узел отправляет датаграммы IP, которые имеют размер, не превышающий значения IP MTU на интерфейсе GRE-туннеля. Бит DF в этом случае может быть либо установлен, либо сброшен (1 или 0). Для интерфейса GRE-туннеля не установлена команда **tunnel path-mtu-discovery**, поэтому маршрутизатор не будет применять PMTUD к пакету GRE-IP.

Пример 3

1. Маршрутизатор переадресации на туннельном источнике получает 1476-байтовую датаграмму от отправляющего узла.

IP	1456 байт TCP + данные

2. Этот маршрутизатор формирует IP-датаграмму 1476-байт внутри GRE для получения 1500-байт IP-датаграммы GRE. Бит DF в IP-заголовке GRE будет сброшен (DF = 0). Затем этот маршрутизатор направляет данный пакет в пункт назначения туннеля.

IP	GRE	IP	1456 байт TCP + данные

3. Предположим, что между исходной и конечными точками туннеля имеется маршрутизатор со значением MTU канала равным 1400 байт. Этот маршрутизатор произведет фрагментирование пакета, поскольку бит DF сброшен (DF = 0). Обратите внимание, что в этом примере фрагментации подвергается самый крайний IP. Таким образом заголовок GRE, внутреннего IP и заголовок

TCP будут находиться в первом фрагменте.

IP_0	GRE	IP	1352 байт TCP + данные

IP_1	104 байта данные

4. Маршрутизатор туннеля-получателя должен заново собрать пакет туннеля GRE.

IP	GRE	IP	1456 байт TCP + данные

5. После сборки туннельного пакета GRE маршрутизатор удаляет IP-заголовок GRE и отправляет исходную датаграмму IP по месту назначения.

IP	1456 байт TCP + данные

Следующий пример показывает, что происходит, когда маршрутизатор выполняет функцию отправляющего узла по отношению к PMTUD и туннельному IP-пакету. На этот раз в исходном IP-заголовке установлен бит DF (DF = 1), а команда **tunnel path-mtu-discovery** настроена таким образом, чтобы бит DF копировался из внутреннего IP-заголовка во внешний заголовок (GRE + IP).

Пример 4

1. Маршрутизатор пересылки в начале туннеля получает от передающего узла 1476-байтовую датаграмму с DF = 1.

IP	1456 байт TCP + данные

2. Этот маршрутизатор формирует 1476-байтовую IP-датаграмму внутри GRE для получения 1500-байтовой IP-датаграммы GRE. В заголовке GRE IP будет установлен бит DF (DF = 1), поскольку бит DF был также установлен и в исходной IP-датаграмме. Затем этот маршрутизатор направляет данный пакет в пункт назначения туннеля.

IP	GRE	IP	1456 байт TCP

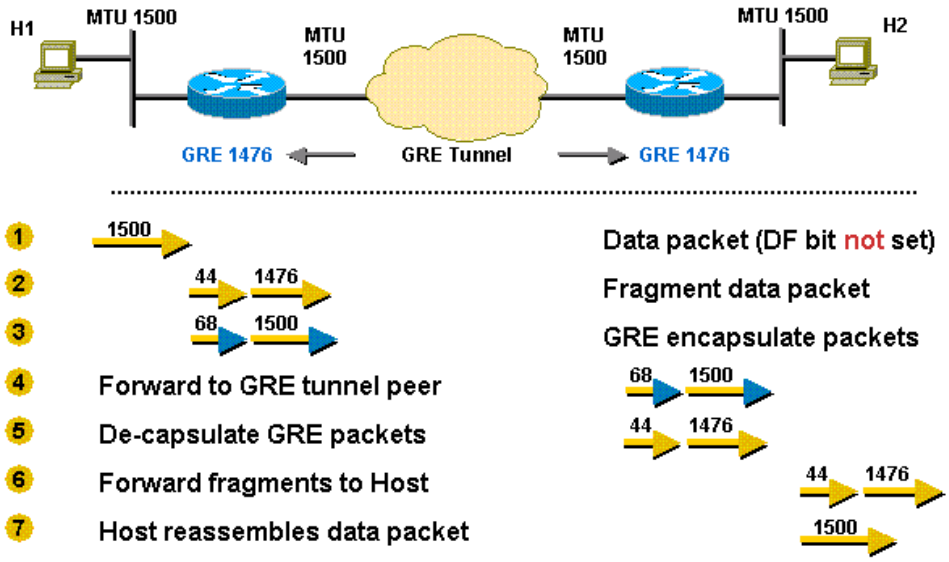
3. Предположим, что между исходной и конечными точками туннеля имеется маршрутизатор со значением MTU канала равным 1400 байт. Этот маршрутизатор не будет фрагментировать пакет, поскольку бит DF установлен (DF = 1). Маршрутизатор должен отбросить этот пакет и отправить маршрутизатору, находящемуся на входе в туннель, ICMP-сообщение об ошибке, поскольку именно этот IP-адрес источника указан в пакете.

IP	ICMP MTU 1400

4. Пересылающий маршрутизатор на туннельном источнике получает это сообщение об ошибках ICMP и снижает MTU IP туннеля GRE до 1376 (1400 - 24). В следующий раз, когда отправляющий узел перенаправит данные в виде IP-пакета величиной в 1476 байт, этот пакет окажется слишком велик, и маршрутизатор пошлет отправителю ICMP-сообщение об ошибке со значением MTU, равным 1376. При повторной передаче данных отправляющий узел сформирует IP-пакет размером в 1376 байт, который сможет пройти по GRE-туннелю и попасть к получающему узлу.

Сценарий 5

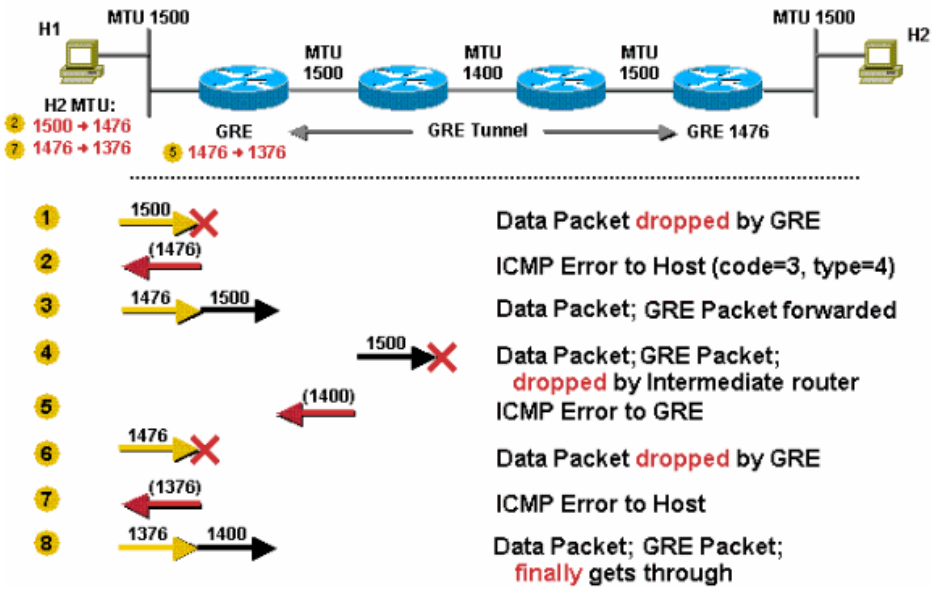
Данный сценарий иллюстрирует фрагментацию GRE. Напомним, что при использовании GRE фрагментация выполняется перед инкапсуляцией, после чего к пакету данных применяется PMTUD, при этом бит DF не копируется при инкапсуляции IP-пакета в GRE. В этом сценарии бит DF сброшен. MTU IP интерфейса туннелирования GRE по умолчанию на 24 байта меньше, чем MTU IP физического интерфейса, таким образом MTU IP интерфейса GRE равняется 1476.



1. Отправитель посылает пакет размером 1500 байт (20 байт заголовок IP + 1480 байт полезные данные TCP).
2. Поскольку значение MTU для туннеля GRE равно 1476, пакет с размером 1500 байт разбивается на два IP-фрагмента размером 1476 и 44 байта с учетом дополнительных 24 байт заголовка GRE.
3. К каждому IP-фрагменту добавляются 24 байта заголовка GRE. Получаются два фрагмента: 1500 байт (1476 + 24 = 1500) и 68 байт (44 + 24).
4. Пакеты GRE + IP, содержащие два IP-фрагмента, направляются на одноранговый маршрутизатор туннеля GRE.
5. Этот маршрутизатор удаляет заголовки GRE из этих двух пакетов.
6. Затем он пересылает два пакета получающему узлу.
7. Данный узел собирает из этих IP-фрагментов исходную IP-датаграмму.

Сценарий 6

Данный сценарий схож со сценарием 5, но в этот раз устанавливается бит DF. В сценарии 6 маршрутизатор при помощи команды **tunnel path-mtu-discovery** настроен на применение PMTUD к туннельным пакетам GRE + IP, а бит DF копируется из исходного заголовка IP в IP-заголовок GRE. Если для пакета GRE + IP маршрутизатор получит ICMP-сообщение об ошибке, тогда он уменьшит значение MTU для IP на интерфейсе туннеля GRE. Снова напомним, что по умолчанию значение MTU IP туннеля GRE на 24 байта меньше чем значение MTU физического интерфейса, поэтому в данном случае MTU IP для GRE составляет 1476 байт. Также отметим, что в состав туннеля GRE входит канал, размер MTU которого равен 1400.



1. Маршрутизатор получает пакет в 1500 байт (заголовок IP: 20 байт + полезные данные TCP: 1480 байт) и отбрасывает его. Это происходит потому, что размер полученного пакета превышает значение IP MTU (1476 байт) на интерфейсе туннеля GRE.

2. Маршрутизатор посылает отправителю ICMP-сообщение об ошибке, в котором указано, что размер MTU следующего сегмента равен 1476 байтам. Узел сохранит полученную информацию (обычно в качестве маршрута узла для пункта назначения) в своей таблице маршрутизации.

3. Передающий узел использует 1476-байтовый размер пакета, когда снова посылает эти данные. Маршрутизатор GRE добавляет инкапсуляцию GRE (24 байта) и передает пакет размером 1500 байт.

4. 1500-байтовый пакет не может пройти через 1400-байтовый канал, поэтому он сбрасывается промежуточным маршрутизатором.

5. Промежуточный маршрутизатор посылает маршрутизатору GRE ICMP-сообщение (код = 3, тип = 4), в котором указано, что величина MTU следующего сегмента составляет 1400 байт. В соответствии с этим маршрутизатор GRE уменьшит размер пакета до 1376 байтов (1400 - 24) и установит значение MTU внутреннего IP на интерфейсе GRE. Это изменение можно увидеть только при использовании команды **debug tunnel command**. Оно не указывается в листинге команды **show ip interface tunnel<#>**.

6. Во время следующей повторной пересылки узлом пакета размером в 1476 байтов маршрутизатор GRE отбрасывает пакет, поскольку он превышает текущий MTU IP (1376) в интерфейсе туннеля GRE.

7. Маршрутизатор GRE отправит маршрутизатору GRE еще одно ICMP-сообщение (код = 3, тип = 4) с MTU равным 1376 байтам, а узел обновит свою текущую информацию запомнив новое значение MTU.

8. Хост посылает данные еще раз, но теперь в меньшем 1376-байтовом пакете, GRE добавит 24-байтовый заголовок инкапсуляции и перешлет его дальше. На этот раз пакет достигнет однорангового маршрутизатора туннеля GRE, где он подвергнется декапсуляции и будет отправлен на узел назначения.

   Примечание: Если в этом сценарии на пересылающем маршрутизаторе не была бы установлена команда **tunnel path-mtu-discovery** и если бы в пакетах, пересылаемых через туннель GRE, был бы установлен бит DF, то узел 1 все равно бы смог отправить узлу 2 пакеты TCP/IP, но при этом пакеты подверглись бы фрагментированию в середине туннеля (канал со значением MTU равным 1400 байт). Кроме того, одноранговый узел туннеля GRE должен был бы собрать эти пакеты перед тем, как выполнить декапсуляцию и последующую переадресацию.

"Чистый" режим туннелирования IPsec

Протокол IPsec обеспечивает конфиденциальность, целостность и подлинность данных, передаваемых по IP-сетям, на основе стандартов. IPsec обеспечивает шифрование данных на сетевом уровне. Применение IPsec увеличивает размер IP-пакета (в режиме туннелирования добавляется как минимум один IP-заголовок). Размер добавляемого заголовка (заголовков) варьируется в зависимости от режима работы IPsec, однако на каждый пакет приходится не более 58 байтов (инкапсуляция зашифрованных данных (Encapsulating Security Payload – ESP) и аутентификация ESP (ESPauth)).

IPsec может работать в двух режимах – в туннельном и в транспортном.

- Туннельный режим является режимом по умолчанию. При работе в туннельном режиме осуществляется защита всего IP-пакета

(пакет шифруется или аутентифицируется, или и то и другое сразу) и пакет помещается между (т.е. инкапсулируется) заголовками и трейлерами протокола IPsec. Затем в пакет вставляется новый IP-заголовок, указывающий конечные точки IPsec (одноранговые узлы), как источник и назначение. Туннельный режим можно использовать с любым одноадресным трафиком. Этот режим должен использоваться, если IPsec защищает трафик от узлов, находящихся позади узлов IPsec. Например, туннельный режим используется в частных виртуальных сетях (VPN), где узлы одной защищенной сети отправляют пакеты узлам другой защищенной сети, используя для этого пару одноранговых узлов IPsec. В сетях VPN туннель IPsec защищает IP-трафик, передаваемый между узлами, с помощью шифрования этого трафика между одноранговыми маршрутизаторами IPsec.

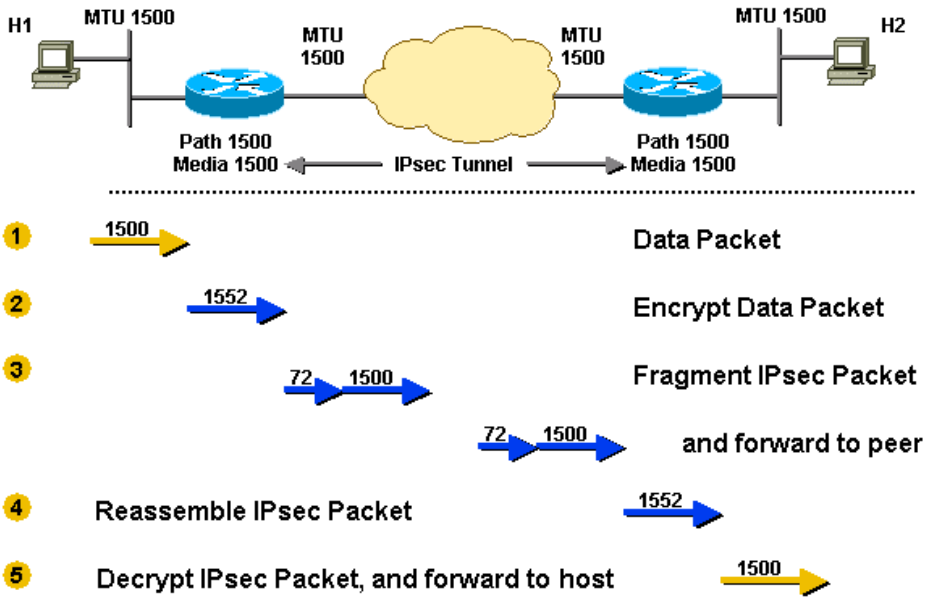
- При работе в транспортном режиме (настраивается с помощью подкоманды **mode transport** в определении преобразования) защита обеспечивается только для полезных данных исходного IP-пакета (пакет зашифровывается, или аутентифицируется, или и то, и другое). Полезная нагрузка размещается (инкапсулируется) между заголовками и трейлерами IPsec. При этом исходные IP-заголовки не затрагиваются. Исключение составляет поле протокола IP, которое изменяется на ESP (50). Значение исходного протокола сохраняется в трейлере IPsec и восстанавливается при дешифровке пакета. Транспортный режим используется только в том случае, когда защищаемый IP-трафик передается между самими узлами IPsec, а IP-адреса источника и назначения совпадают с адресами узлов IPsec. Обычно транспортный режим IPsec используется совместно с туннельным протоколом (например, GRE), т.е. сначала инкапсулируется IP-пакет с данными, а затем для защиты туннельных пакетов GRE используется IPsec.

IPsec всегда применяет PMTUD к пакетам данных, а также к своим собственным пакетам. Для изменения процесса обработки IP-пакета IPsec с помощью PMTUD в IPSec предусмотрены команды конфигурирования. IPsec может сбрасывать, устанавливать или копировать бит DF из IP-заголовка пакета данных в IP-заголовок IPsec. Это называется функцией замещения бита DF (DF Bit Override Functionality).

Примечание: Во время аппаратного шифрования с использованием IPsec желательно избежать фрагментации после того, как была проведена инкапсуляция. В зависимости от аппаратных средств аппаратное шифрование позволяет обеспечить пропускную способность около 50 Мбит/с. Однако при фрагментировании пакета IPsec вы можете потерять от 50 до 90% пропускной способности. Такая потеря происходит потому, что фрагментированные пакеты IPsec проходят процессную коммутацию для повторной сборки, а затем передаются модулю аппаратного шифрования для дешифровки. При аппаратном шифровании пропускная способность может уменьшиться до уровня производительности, который обеспечивается шифрованием, использующим программные средства (2-10 Мбит/с).

Сценарий 7

В данном сценарии описывается процесс фрагментации IPsec. Для данного сценария величина MTU на протяжении всего пути равна 1500 байт, а бит DF сброшен.



1. Маршрутизатор получает 1500-байтовый пакет (20-байтовый IP-заголовок + 1480 байт полезной нагрузки TCP), предназначенный для узла 2.

2. Пакет размером 1500 байт шифруется протоколом IPsec, во время чего добавляется 52 байт служебных данных (заголовок IPsec, трейлер и дополнительный IP-заголовок). Теперь протоколу IPsec предстоит отправить пакет размером 1552 байта. Поскольку исходящий MTU равен 1500 байтам, этот пакет должен быть фрагментирован.

3. Из пакета IPSec создаются два фрагмента. В ходе фрагментации ко второму фрагменту добавляется 20-байтный заголовок IP.

Таким образом получается один фрагмент размером 1500 байт и один фрагмент размером 72 байта.

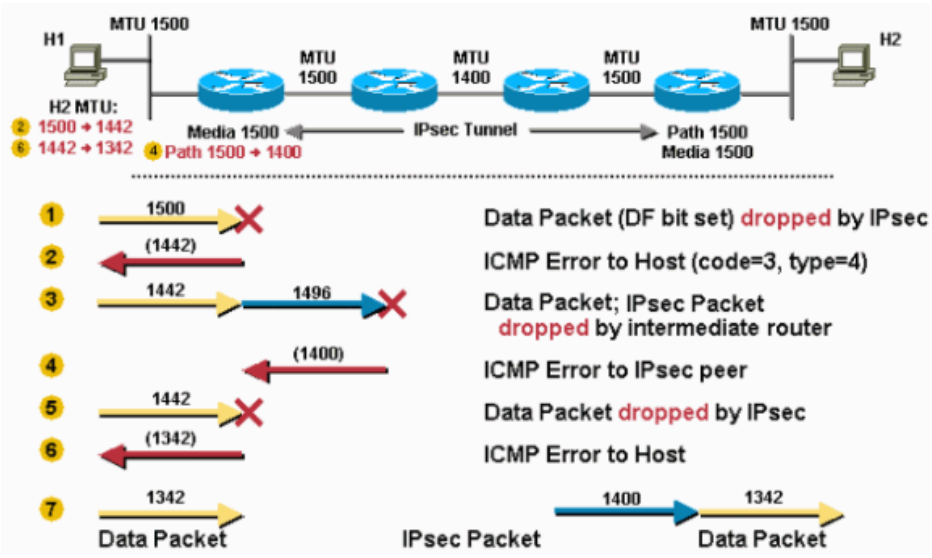
4. Одноранговый маршрутизатор туннеля IPsec получает эти фрагменты, удаляет дополнительный заголовок IP и собирает из IP-фрагментов исходный пакет IPsec. Далее протокол IPSec осуществляет расшифровку пакета.

5. Затем маршрутизатор пересылает оригинальный 1500-байтный пакет на узел 2.

Сценарий 8

Этот сценарий имеет сходство со сценарием 6 за двумя исключениями: в этом сценарии установлен бит DF и на пути между двумя одноранговыми узлами туннеля IPSec имеется канал, MTU которого меньше MTU других каналов. В данном сценарии описывается, каким образом одноранговый маршрутизатор IPSec выполняет обе роли PMTUD (описание см. в разделе Маршрутизатор как PMTUD-участник в конечной точке туннеля.

В этом сценарии описывается, каким образом значение PMTU уменьшается по причине необходимости фрагментации. Напомним, что при шифровании пакета IPsec копирует бит DF из внутреннего заголовка IP во внешний заголовок. Значение media-MTU и значение PMTU хранятся в ассоциации защиты IPsec (Security Association). Media-MTU основано на MTU исходящего интерфейса маршрутизатора, а PMTU основан на минимальном MTU, который имеется на пути между одноранговыми узлами IPsec. Напомним, что IPsec инкапсулирует и шифрует пакет перед его фрагментацией.



1. Маршрутизатор получает пакет размером 1500 байт и отбрасывает его, потому что при добавлении служебных данных IPsec размер пакета становится больше значения PMTU (1500 байт).

2. Маршрутизатор посылает ICMP-сообщения узлу 1 о том, что MTU следующего перехода составляет 1442 (1500 - 58 = 1442). 58 байт – это максимальный объем служебных данных, добавляемых IPsec при использовании IPsec ESP и ESPauth. В действительности объем служебных данных IPsec может быть на 7 байт меньше указанной выше величины. Узел 1 записывает эту информацию в своей таблице маршрутизации (обычно в качестве маршрута узла до адреса назначения, т.е. узла 2).

3. Узел 1 уменьшает свое значение PMTU для узла 2 до 1442 байтов, поэтому узел 1 при повторной передаче данных на узел 2 будет отправлять пакеты меньшего размера (1442 байта). Маршрутизатор получает пакет размером 1442 байт, а протокол IPsec добавит 52 байта шифровальных служебных данных. Таким образом, размер конечного пакета составляет 1496 байт. Поскольку в заголовке этого пакета установлен бит DF, он будет отброшен срединным маршрутизатором, который подключен к каналу с MTU равным 1400 байт.

4. Этот срединный маршрутизатор отправляет источнику IPsec-пакета (первый маршрутизатор) ICMP-сообщение, в котором указано, что MTU следующего сетевого сегмента составляет 1400 байт. Это значение записано в IPsec SA PMTU.

5. При следующей передаче узлом 1 пакета величиной 1442 байт (он не получил подтверждения на этот пакет) IPsec отбросит этот пакет. Маршрутизатор снова отбросит этот пакет, поскольку после добавления служебных данных IPsec размер этого пакета превысит величину PMTU (1400 байт).

6. Маршрутизатор посылает узлу 1 ICMP-сообщение, в котором указано, что в настоящий момент MTU следующего сегмента составляет 1342 байта (1400 - 58 = 1342). Узел 1 снова сохранит эту информацию.

7. При повторной отправке данных узел 1 будет использовать пакет меньшего размера (1342 байта). Данный пакет не потребует фрагментации и без помех пройдет через туннель IPsec к узлу 2.

GRE и IPsec вместе

При использовании IPsec для шифрования GRE-туннелей процессы фрагментации и PMTUD усложняются. Такое совместное использование IPsec и GRE необходимо, потому что IPsec не поддерживает многоадресные IP-пакеты (т.е. в сети VPN, использующей IPsec, невозможно использовать протокол динамической маршрутизации). В то же время, GRE-туннели поддерживают многоадресную рассылку, и поэтому GRE-туннель можно использовать для инкапсуляции многоадресного пакета, принадлежащего протоколу динамической маршрутизации, в одноадресный IP-пакет протокола GRE, который затем можно зашифровать при помощи IPsec. При этом протокол IPsec часто развертывается в режиме транспорта поверх GRE, поскольку одноранговые узлы IPsec и конечные точки туннеля GRE (маршрутизаторы) являются одинаковыми, и режим транспорта обеспечит снижение размера служебной информации IPsec на 20 байт.

Одним из интересных случаев является случай, в котором IP-пакет разбивается на два фрагмента и инкапсулируется в GRE. В этом случае IPsec будет видеть два независимых пакета GRE + IP. Очень часто при использовании настроек по умолчанию один из этих пакетов будет иметь довольно большой размер и после шифрования его понадобиться фрагментировать. Перед дешифровкой одноранговый узел IPsec должен будет собрать этот пакет. Такая двойная фрагментация (один раз перед GRE и один раз после IPsec), выполняемая отправляющим маршрутизатором, ведет к увеличению времени запаздывания и к уменьшению пропускной способности. Кроме того, сборка осуществляется в режиме процессной коммутации, поэтому всякий раз когда это происходит, нагрузка на процессор принимающего маршрутизатора резко увеличивается.

Однако этого можно избежать установив на интерфейсе туннеля GRE значение ip mtu, которое будет достаточно низким, чтобы учесть служебные данные как GRE, так и IPsec (по умолчанию значение ip mtu для интерфейса туннеля GRE устанавливается равным значению MTU физического исходящего интерфейса, за вычетом служебных данных GRE).

Ниже в таблице указаны рекомендуемые значения MTU для каждой комбинации туннель/режим работы при условии, что значение MTU исходящего физического интерфейса равно 1500 байт.

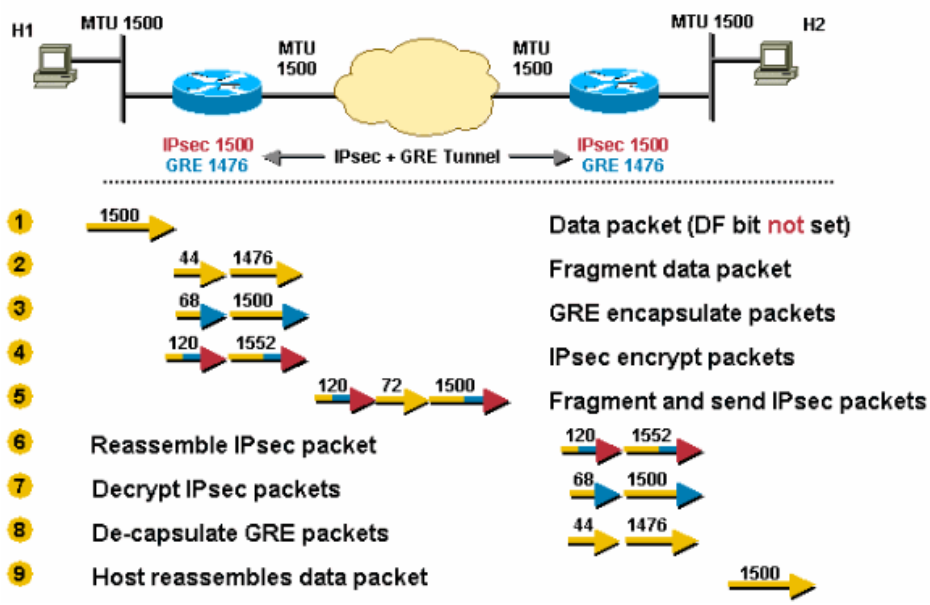
Комбинация туннелей	**Требуемое значение MTU**	**Рекомендуемое значение MTU**
GRE + IPsec (транспортный режим)	1440 байт	1400 байт
GRE + IPsec (туннельный режим)	1420 байт	1400 байт

Примечание: 1400-байтное значение MTU рекомендуется использовать потому, что оно подходит для большинства распространенных комбинаций режимов GRE + IPsec. Кроме того, добавление дополнительных 20 или 40 байт на служебную информацию не приведет к возникновению каких-либо ощутимых проблем. Значительно проще запомнить какое-нибудь одно значение, которое подходит для большинства сценариев.

Сценарий 9

IPsec устанавливается поверх GRE. MTU исходящего физического интерфейса = 1500 байт, IPsec PMTU = 1500 и GRE IP MTU = 1476 (1500 - 24 = 1476). По этой причине пакеты TCP/IP будут фрагментироваться дважды, один раз перед инкапсуляцией в GRE, другой раз после шифрования IPsec. Пакет будет фрагментирован перед GRE-инкапсуляцией, а один из GRE-пакетов снова будет фрагментирован после шифрования IPsec.

Установка значения ip mtu 1440 (транспортный режим IPsec) или ip mtu 1420 (туннельный режим IPsec) для туннеля GRE позволит избежать двойной фрагментации в данном сценарии.



1. Маршрутизатор получает 1500-байтовую датаграмму.

2. Перед инкапсуляцией GRE разобьет 1500-байтовый пакет на две части: 1476 байт (1500 - 24 = 1476) и 44 байта (24 байта данных + 20 байтов IP-заголовок).

3. GRE выполнит инкапсуляцию IP-фрагментов, в результате чего каждый пакет увеличится на 24 байта. Таким образом получаются два пакета GRE + IPsec по 1500 (1476 + 24 = 1500) и 68 (44 + 24) байт каждый.

4. IPsec зашифровывает эти два пакета, добавляя к каждому пакету еще 52 байта (туннельный режим IPsec) служебной информации. Таким образом получаем два пакета размером 1552 байта и 120 байтов.

5. 1552-байтовый пакет IPsec фрагментируется маршрутизатором, так как его величина превышает максимальный размер передаваемого блока данных для исходящих (1500). Пакет размером в 1552 байта разбивается на части, 1500-байтный пакет и 72-байтный пакет (52 байта полезной информации и дополнительно 20 байт – IP-заголовок второго фрагмента). Три пакета – 1500-байтовый, 72-байтовый и 120-байтовый – переадресуются на партнера IPsec + GRE.

6. Принимающий маршрутизатор заново собирает два фрагмента IPsec (1500 байт и 72 байт), чтобы получить исходный пакет IPsec + GRE на 1552 байт. 120-байтный пакет IPsec + GRE не требует никаких операций.

7. IPsec дешифрует 1552-байтный и 120-байтный пакеты IPsec + GRE, в результате чего получаются 1500-байтный и 68-байтный пакеты GRE.

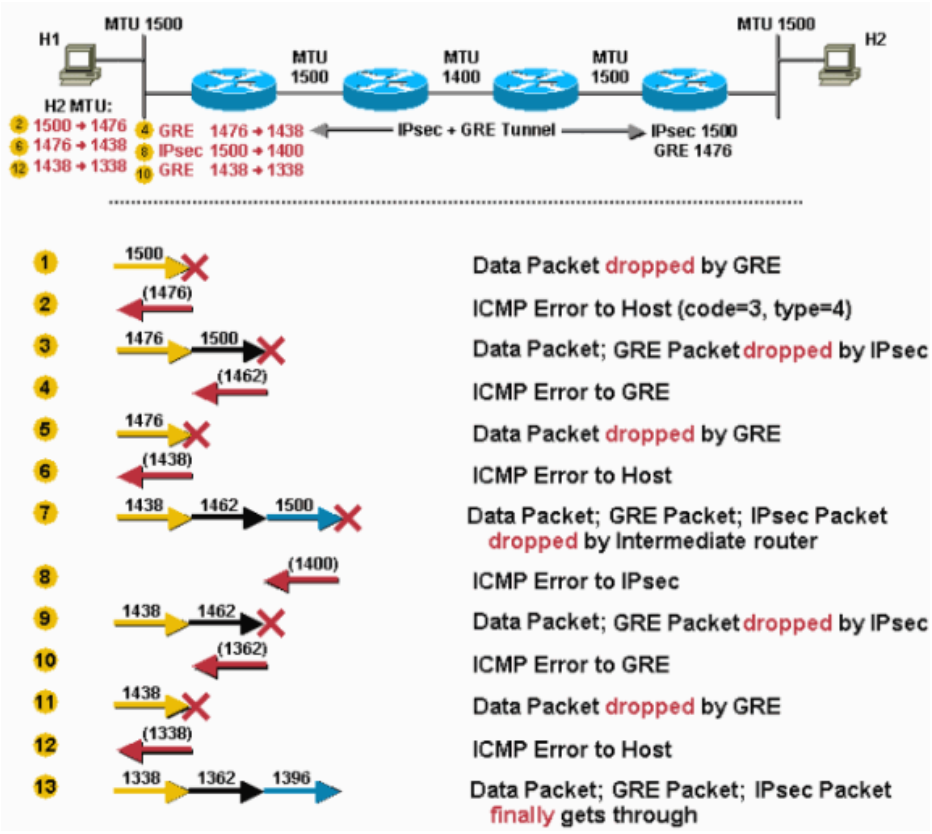
8. GRE декапсулирует 1500-байтный и 68-байтный пакеты GRE, в результате чего получаются 1476-байтный и 44-байтный фрагменты пакетов IP. Эти фрагменты пересылаются на узел назначения.

9. Хост 2 собирает из этих IP-фрагментов исходную IP-датаграмму размером 1500 байт.

Сценарий 10 подобный сценарию 8, за исключением того, что соединение MTU в туннельной части имеет более низкую пропускную способность. Для первого пакета, отправляемого с узла 1 на узел 2, это «пессимистический» сценарий. По завершении последнего этапа этого сценария узел 1 устанавливает для узла 2 корректное значение PMTU, и TCP-соединения между узлом 1 и узлом 2 приходят в норму. TCP-потокам, циркулирующим между узлом 1 и другими узлами (доступными через туннель IPsec + GRE), остается завершить три последних этапа сценария 10.

В данном сценарии на GRE-туннеле настроена команда **tunnel path-mtu-discovery**, а в пакетах TCP/IP, отправляемых узлом 1, установлен бит DF.

Сценарий 10



1. Маршрутизатор получает 1500-байтовую датаграмму. Туннель GRE отбрасывает этот пакет, поскольку из-за установленного бита DF протокол GRE не может выполнить фрагментацию пакета или переадресовать его; кроме того, после добавления служебных данных GRE (24 байта) размер пакета превысит значение ip mtu исходящего интерфейса.

2. Маршрутизатор отправляет ICMP-сообщение узлу 1 и уведомляет, что MTU следующего скачка равняется 1476 (1500 - 24 = 1476).

3. Узел 1 изменяет значение PMTU (1476 байт) для узла 2. Теперь при повторной передаче будет отправлен пакет меньшего размера. GRE выполняет инкапсуляцию пакета, а затем передает 1500-байтный пакет протоколу IPsec. IPsec отбросит этот пакет, поскольку GRE скопировал бит DF (он был установлен) из внутреннего заголовка IP, и после добавления служебной информации IPsec (максимальная величина – 38 байт) размер пакета будет слишком большим для отправки через физический интерфейс.

4. IPsec отправляет ICMP-сообщение для GRE, где указывает, что значение MTU следующего сетевого сегмента составляет 1462 байта (поскольку в результате шифрования и добавления служебных данных IP будут добавлены еще 38 байт). GRE сохраняет значение равное 1438 байтам (1462 - 24) в качестве ip mtu туннельного интерфейса.

   Примечание: Это изменение сохраняется во внутренних данных, в выходных данных команды **show ip interface tunnel<#>** оно не отображается. Изменение видно только при использовании команды **debug tunnel**.

5. В следующий раз когда узел 1 повторно отправит 1476-байтный пакет, GRE отбросит этот пакет.

6. Маршрутизатор отправляет ICMP-пакет на узел 1 с указанием на то, что значение MTU для следующего перехода равно 1438.

7. Узел 1 уменьшает значение PMTU для узла 2 и повторно отправляет пакет размером 1438 байт. На этот раз GRE принимает этот пакет, инкапсулирует его и передает его для шифрования протоколу IPsec. Пакет IPsec пересылается на промежуточный маршрутизатор и отбрасывается, так как MTU исходящего интерфейса равно 1400.

8. Промежуточный маршрутизатор отправляет IPsec сообщение ICMP, где указано, что значение MTU следующего сетевого сегмента составляет 1400 байт. IPSec сохраняет это значение в качестве значения PMTU соответствующей ассоциации защиты (IPsec SA).

9. Когда узел 1 пересылает пакет из 1438 байт, GRE инкапсулирует его и передает IPsec. IPsec отбрасывает пакет, поскольку он установил значение своего собственного PMTU равным 1400 байт.

10. IPsec отправляет GRE ICMP-сообщение об ошибке, где будет указано, что значение MTU следующего сегмента составляет 1362 байта. После чего GRE сохранит во внутренних данных значение равное 1338 байтам.

11. Когда узел 1 повторно отправляет исходный пакет (поскольку он не получил подтверждения), GRE его отбрасывает.

12. Маршрутизатор посылает узлу 1 ICMP-сообщение, где указано, что MTU следующего сегмента составляет 1338 байта (1362 - 24 = 1442). Узел 1 уменьшает значение PMTU для узла 2, и оно становиться равным 1338 байтам.

13. Узел 1 повторно отправляет 1338-байтный пакет, который в конечном итоге доставляется на узел 2.

Дополнительные рекомендации

Настройка команды **tunnel path-mtu-discovery** на туннельном интерфейсе позволит улучшить взаимодействие между GRE и IPsec в случае, если они настроены на одном и том же маршрутизаторе. Напомним, что если команда **tunnel path-mtu-discovery** не была настроена, то бит DF будет всегда сбрасываться в IP-заголовке GRE. Таким образом, пакет GRE IP будет фрагментироваться даже тогда, когда в IP-заголовке инкапсулированных данных установлен бит DF (обычно этот бит запрещает фрагментировать пакет).

Если на интерфейсе туннеля GRE настроена команда **tunnel path-mtu-discovery**, то тогда происходит следующее.

1. GRE копирует бит DF из IP-заголовка данных в IP-заголовок GRE.
2. Если в IP-заголовке GRE установлен бит DF и если после шифрования IPsec размер пакета оказывается слишком большим по сравнению со значением IP MTU физического исходящего интерфейса, тогда IPsec отбрасывает пакет и просит туннель GRE уменьшить величину своего IP MTU.
3. IPsec выполняет PMTUD для своих пакетов, и если значение PMTU IPsec меняется (уменьшается), IPsec не уведомляет об этом GRE. Однако когда придет еще один слишком большой пакет, то повторится пункт 2.
4. Теперь значение IP MTU протокола GRE уменьшилось, поэтому GRE отбрасывает любые IP-пакеты данных, в которых установлен бит DF и которые теперь стали слишком большими, и посылает отправляющему узлу ICMP-сообщение.

Команда **tunnel path-mtu-discovery** помогает интерфейсу GRE динамически установить свой IP MTU (а не статически – по команде **ip mtu**). В действительности рекомендуется использовать обе этих команды. Команда **ip mtu** используется для того, чтобы предоставить место для служебной информации GRE и IPsec относительно IP MTU локального физического исходящего интерфейса. Команда **tunnel path-mtu-discovery** позволит дополнительно уменьшить IP MTU туннеля GRE в случае, если на пути между одноранговыми узлами IPsec существует канал IP MTU с более низким значением IP MTU.

Ниже приводятся еще несколько советов, которыми можно воспользоваться при возникновении проблем с PMTUD в сети, в которой настроены туннели GRE + IPsec.

Самые удачные решения идут в начале списка.

- При неработающем PMTUD проблема обычно связана с тем, что ICMP-сообщения блокируются маршрутизатором или брандмауэром.
- Используйте команду **ip tcp adjust-mss** для туннельных интерфейсов таким образом, чтобы маршрутизатор уменьшил значение TCP MSS в пакете TCP SYN. Это позволит узлам на обоих концах (TCP-отправитель и TCP-получатель) использовать пакеты достаточно маленького размера, избегая тем самым необходимости использовать PMTUD.
- Используйте политику маршрутизации на входном интерфейсе маршрутизатора и настройте карту маршрутов таким образом, чтобы бит DF в IP-заголовке данных сбрасывался до того, как он поступит на интерфейс туннеля GRE. Теперь IP-пакеты данных будут фрагментироваться перед инкапсуляцией GRE.
- Увеличьте значение ip mtu туннеля GRE таким образом, чтобы оно равнялось значению MTU исходящего интерфейса. Теперь IP-пакеты данных будут инкапсулироваться протоколом GRE без предварительной фрагментации. Потом пакет GRE будет зашифровываться при помощи IPsec и затем фрагментироваться для последующей передачи через исходящий физический интерфейс. В этом случае не следует настраивать команду **tunnel path-mtu-discovery** на интерфейсе туннеля GRE. Это может значительно уменьшить пропускную способность, поскольку повторная сборка пакетов IP на одноранговых узлах IPsec выполняется в режиме переключения процессов.

Дополнительные сведения

- **Протоколы маршрутизируемые по IP**
- **Страница поддержки IP-маршрутизации**
- **Страница поддержки IPSec (протокол IP-безопасности)**
- **RFC 1191 Path MTU Discovery (алгоритм определения значений MTU для пути)**

- **RFC 1063 IP MTU Discovery Options (опции алгоритма определения значений MTU для пути)**
- **RFC 791 Internet Protocol (протокол Интернета)**
- **RFC 793 Transmission Control Protocol (протокол управления передачей)**
- **RFC 879 The TCP Maximum Segment Size and Related Topics (максимально допустимый размер сегмента и смежные вопросы)**
- **RFC 1701 Generic Routing Encapsulation (GRE) (общая инкапсуляция маршрутов – GRE)**
- **RFC 1241 A Scheme for an Internet Encapsulation Protocol (схема протокола Интернет – инкапсуляции)**
- **RFC 2003 IP Encapsulation within IP (IP-инкапсуляция в протокол IP)**
- **Техническая поддержка – Cisco Systems**

Дата генерации PDF файла: Jan 05, 2010

http://www.cisco.com/support/RU/customer/content/9/92028/pmtud_ipfrag.shtml