

# 8\_Преобразование сетевых адресов NAT

Преобразование сетевых адресов:

Понятия:

Протоколы транспортного уровня:

ПОРТЫ

UDP

Псевдозаголовок UDP

TCP

Установление TCP-соединения

Процесс завершения TCP-соединения

Состояние TCP -соединение

Механизм скользящего окна

Окно приема

Окно передачи

## 9 Прикладные протоколы TCP/IP

## Преобразование сетевых адресов:

Nat - допускает преобразование внутренних адресов сети в адреса внешней сети.

Позволяет подключить к внешней сети к глобальной сети, не являющиеся уникальными в глобальной сети.

Узлы локальной сети могут совместно использовать 1 или несколько публичных IP-адресов глобальной сети.

Кроме того технология NAT позволяет скрыть структуру локальной сети.

Необходимость в NAT отпадает при использовании адресов IPv6. тогда они уникальные.

**Понятия:**

Внутренний локальный адрес - IP-адрес назначенный узлу во внутренней сети. Как правило он частный.

Внутренний глобальный адрес - публичный IP который представляет 1 или более внутренних локальных IP- адресов для внешней сети.

Внешний локальный адрес, адрес который представляет узел внешней сети для внутренней сети. Берется из адресного пространства внутренней сети.

Внешний глобальный адрес - IP-адрес назначенный узлу во внешней сети.

Каждый пакет имеет внутренний адрес отправителя и внешний локальный адрес получателя, пока пакет находится во внутренней сети. При переходе адрес отправителя заменяется внутренним глобальным адресом, а получателя - внешним глобальным. И наоборот.

(Адрес отправителя - внешний, получателя - внутренним локальным).

NAT используется маршрут который соединяет глобальные сети. Все преобразования отслеживаются в соответствующих NAT-таблицах, что позволяет при получении ответа пакета выполнить обратное преобразование

Преобразование внутренних адресов

Два вида: статический и динамический

Статическое - взаимно однозначное соответствие между внутр и глоб

Полезно, когда узел во внутренней сети должен быть доступен извне по адресу

Динамическое - соответствие между внутренними локальными адресами некоторого множества глобальных адресов

(Внутр глобальны, что это?)

Технология NAT, когда нескольким локальным адресам ставится соответствие глобальный маршрут исполняется на уровне протокола.

Чтобы перевести глобальный в нужный локальный (?)

Передача пакетов имеет маршрут адреса в таблице (?). Старается использовать один и тот же глобальный адрес.

Когда извне: совершает сверку с таблицей используя протокол, глобальный адрес. Адрес порта. Внутренний глобальный в локальный адрес. Посылает его по назначению во внутреннюю сеть.

На рисунке порты после двоеточия.

Перекрытие адресов:

А внутренние адреса могут перекрываться с внешними адресами.

## Протоколы транспортного уровня:

Обеспечивают контроль над передаче данных TCP и UDP - распространенные.

Когда данные будут доставлены на физическом уровне устройства, они образуют кадр. У них есть заголовки.

## ПОРТЫ

Необходимо понять какой прикладной процесс выступает в качестве получателя данных.

Они обладают средствами идентификации прикладных протоколов по номерам портов. Не путать с портами сетевых устройств.

Номера можно разделить на 3 :

Хорошо известные - 0-1023 - базовые системные службы . DHCP - 68

Зарегистрированные - коммерческие приложения 1024-49151, 1433 - Цикл WB server. Могут использоваться и для назначения временных номеров портов.

Динамические

Если прикладной процесс явно не указывает номер порта. Тогда ОС укажет ему временный номер порта из данного диапазона.

Регистрация и выдача портов из первой и второй зоны занимается IANA.

Если при отправке данных указать неиспользуемый номер порта назначения , т.е. номер с которым не связан ни один прикладной процесс, то эти данные будут отвержены , а в ответ будет отправлен ICMP пакет соответствующий о недоступности узла назначения.

(где записан этот порт??)

## UDP

Протокол транспортного уровня . Без установления соединения между отправителем и получателем. Работает в режиме простой передачи данных.

#### ПЛЮСЫ

Эффективен при запрос-ответ, так как не нужно тратить времени на закрытие. Другое преимущество - групповая рассылка.

Их можно отправить на широковещ и ли групповой адрес. Тогда не требуется передавать данные по отдельности на каждое вновь созданное соединение.

#### МИНУСЫ

UDP не обеспечивает порядок получения в процессе передачи данных.

Они могут отличаться.

Связано с тем, что данные отправляются разными путями. И то, что отправлено позже может прийти раньше. Если порядок важен (конечные и начальные данные должны идти по порядку), то необходимо реализовывать порядок самостоятельно.

Разбиение данных на крупные фрагменты и проводит сборку данных из полученных фрагментов UDP не умеет.

1 мб в UDP не поместиться(?)

Данные инкапсулируются в дейтаграмме

# UDP-ДЕЙТАГРАММА

	0								1								2								3							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Порт отправителя																															
32																																

Заголовок - фиксированная длинна - 8 байт

Порт отправителя и Порт получателя -

## Псевдозаголовок UDP

# UDP-ДЕЙТАГРАММА ПСЕВДОЗАГОЛОВОК

	0								1								2								3							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	IP-адрес отправителя																															
32	IP-адрес получателя																															
64	Зарезервировано								Протокол								Длина															

Поле протокола должно принимать значение 17

## TCP

TCP - транспортный протокол

RFC/793

Преимущество: надежная доставка данных. Они гарантированно будут доставлены до получателя.

Путем установления логического соединения между узлами сети - решение этой задачи.

На транспортном уровне.

(вылетел Skype. Что здесь было??)

TCP может следить чтобы данные не были потеряны, продублированы, и пришли к получателю в том же порядке в котором они были отправлены.

Размер сегмента определяется реализацией TCP.

Если данные помещаются в буфер TCP. Из него берутся данные и помещаются в TCP сегмент. В сегмента два буфера: на передачу и принятие.

Нет деления на сообщения.

Передача происходит не сразу. Не факт что n переданных байт будут получены.

Минимальный размер заголовка TCP-сегмента - 20 байт.

/рисунок

Порт отправителя/порт получателя

Порядковый номер -указывает номер байта, который определяет смещение TCP-сегмента в потоке данных. Используется получателем при восстановлении исходных данных из полученных TCP-сегментов.

Номер подтверждения - содержит номер байта ожидаемого в следующем сегменте, используется только с флагом ACK. Это значит смотрит какой сегмент следующий с данными.

Длина заголовка - 4 бита - указывает размер заголовка TCP-сегмента размеров в 4 слова (?). Размер зависит от значений задаваемых полей параметры. Для выравнивания до размер кратного 4 байтам используется дополнение в виде последовательности 0.

Флаги - содержат служебную информацию о сегменте

URG - срочное

ACK - подтверждения принятия сегментом

PSH - с помощью которого отправитель просит получателя передать данные сразу прикладному уровню после получения а не хранить их в буфере.

RST - для отказа от неверного сегмента или от попытки создать соединение

SYN - используется для установления соединения

FIN - для завершения соединения

Размер окна - используется для указания информации о том, какой размер данных узел готов принять в следующем TCP - сегменте.

Контрольная сумма - из содержимого TCP-сегмента вычисляется и псевдозаголовок (аналогичный как и для UDP,но в протоколе 6 записано)

Указатель на срочные данные - указывает на последние данные в сегменте, установить если какие-то данные необходимо отправить без очереди.

Параметры - переменной длины, используется при согласовании параметров при установлении TCP-соединения. При отсутствии данное поле имеет значение 1(?)

Два режима TCP:

активный

от клиента - серверу TCP сегмент с запросом на установление соединения

пассивный

сервер ждет от клиента установления соединения

## Установление TCP-соединения

1. Сервер ожидает TCP сегмент с запросом на установление соединения.
2. На Клиент для установления соединения отправляется SYN-сегмент(номера порто полу/отпр, начальный порядковый номер и размер сегмента)

3. После получения SYN- сегмента сервер отправляет SYN -сегмент с установленными SYN=1, ACK=1

4. Клиент выполняет

1. Если все еще желает получить соединение, отправляет сегмент с ACK=1

Потому что если от клиента нет подтверждения на подтверждения то значит что соединение не будет установлено

2. Если больше не хочет устанавливать соединение отправить RST.

## Процесс завершения TCP-соединения

1. Отправляет FIN-сегмент
2. Подтверждение. Но другая сторона все еще может передавать данные.
3. Завершено только если клиент потом примет FIN, а затем подтвердит это серверу.
4. Если подтверждение так и не придет до в течение MSL - двойное максимальное время жизни сегмента (более 4 минут). Другая сторона заметит, что ей никто не отвечает, то завершение.

*Одновременное завершение - оба послали сегмент на завершение, но еще получили.*

*Важно знать на своей стороне чтобы было завершено.*

## Состояние TCP -соединение

11 состояний.

каждое начинается с фиктивного состояния CLOSED. Соединения еще не существует.

LISTEN - сервер ожидает сегмент

SYN SENT - клиент отправил сегмент

SYN RCVD -сервер получил отправил и получил подтверждение. Если RST - то переход в LISTEN



ESTABLISHMENT - все было ок, тогда работают

FIN WAIT 1 - клиент ждет подтверждения завершения

TIME WAIT - пауза в 2 MSL сегмента

CLOSE WAIT - сегмент ждет завершения

LAST ACK - сервер ждет состояния завершения у клиента

*Обычно клиент разрывает соединение.*

Передача данных в TCP

используется механизм, который гарантирует что каждый отправленный сегмент будет обязательно доставлен. После того, как сегмент был отправлен то начинается RTO. Для подтверждения обратно отправляется TCP сегмент с флагом ACK. Если до этого окончится время RTO, то считается что произошла ошибка и TCP- сегмент отправляется повторно.

Сегмент мог задержаться дольше, чем ждали или мог потеряться с подтверждением .

*Придет сообщение об ошибке в разрыве или о подобном. Максимальное число попыток. Все копии которые придут будут игнорироваться.*

## Механизм скользящего окна

Используется также **механизм скользящего окна**. Который гарантирует, что даже если получены сегменты не в том порядке в котором отправлены, получатель все равно сможет собрать из них данные. Тогда первому байту в сегменте присваивается порядковый номер и передается он в заголовке TCP сегмента. Порядковые номер остальных без очереди.

## Окно приема

Каждый (?) имеет **окно приема** - диапазон порядковых номеров байтов.

Наименьшее значение соответствует левой границе окна приема.

Наибольшее правой границе - порядковый номер последнего байта для которого у TCP сегмента есть место в буфере. Когда принимается сегмент, то (?) отбрасывается (что?).

Если порядковый номер окна приема больше чем следующего сегмента, то значит он прибыл не по порядку. Если совпадает с ожидаемым, то окно приема сдвигается вправо на число байтов принятых в TCP сегменте.

## Окно передачи

Кроме того, у каждого сегмента есть **окно передачи**. В одном расположены отправленные байты, которые не подтверждены, а также те которые еще могут быть отправлены. Сдвигается после подтверждения на число подтвержденных байт.