**Лекция за 04.04.2012, часть 2**

***Структура IP-датаграммы.***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 3 7 15 19 32 | | | | |
| Version | IHL | TOS | Total Length | |
| Fragment ID | | | Flags | Fragment offset |
| TTL | | Protocol | Header CRC16 | |
| Source Address | | | | |
| Destination Address | | | | |
| Options; Padding | | | | |

* Version – версия, (для IPv4 – 4, для IPv6 – 6). Версий < 4 не бывает
* IHL – Internet Header Length, длина заголовка в 32-битных словах (т. е. Макс. 15\*32-битных слов)
* TOS – Type of Service, флаги для определения наилучшей маршрутизации, а именно:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 11 12 13 15 | | | | |
| Precedence | Delay | Throughput | Reliability | Reserved |

* Precedence – приоритет датаграммы (большее значение — меньший приоритет)
* Delay – задержка
* Throughput – пропускная способность
* Reliability – надежность

Для всех трех: 1 — обычная, 0 — повышенная. В современных реализациях все эти параметры рекомендательные

* Total Length – общая длина датаграммы в октетах
* Fragment ID, Flags, Fragment Offset – см. далее
* TTL – Time to Life, сколько ***хопов*** (т. е. прыжков между узлами сети) может сделать датаграмма, прежде чем она будет уничтожена. Нужно чтобы не было зацикливания при маршрутизации. При проходе каждого маршрутизатора значение поля уменьшается на 1, если поле стало равно 0 — датаграмма уничтожается. Раньше первоначальное значение ставилось 16, теперь 32. (Раньше поле должно было уменьшаться на количество секунд, потраченное на обработку датаграммы, но это время достаточно быстро стало меньше 1)
* Protocol – тип протокола (6 для TCP, 17 для UDP)
* Header CRC16 – чек-сумма заголовка
* Source Address, Destination Address – адрес отправителя/получателя
* Options – изначально, чтобы не раздувать размеры служебных пакетов, инкапсулированных в IP, было предложено передавать их данные внутри поля options ip-датаграмм. Также поле может использоваться для передачи служебной информации в сетях ip. Единственное поле переменной длины

|  |  |  |
| --- | --- | --- |
| 0 7 15 | | |
| Op. type | Op. length | Op. data |

* Op. length – длина поля опций в октетах, если не 0. Если 0, то данная опция имеет фиксированную, возможно, ненулевую длину, которая берется из специальной таблицы
* Op. data – данные
* Op. type – типа опции:

|  |  |  |
| --- | --- | --- |
| 0 1 3 7 | | |
| Copy Flag | Op. class | Op. number |

* Copy Flag – см. далее
* Op. class – тип опций: 0b0 (***control***) иди 0b10 (d***ebug and management***)
* Op. number – номер опции, в каждом классе по 32

Если ***IHL*** == 5, то опций нет. Если есть хоть одна опция, то должна быть спец. опция end of options. Т.к. опции входят в заголовок, то контрольная сумма при их изменении пересчитывается.

* Padding – поле-заполнитель, чтобы все было выравнено по 32

**Фрагментация IP-датаграмм**

Для снижения избыточности передачи данных хочется передавать датаграммы максимальной длины, но при этом физические сети имеют ограниченную емкость => они накладывают ограничения на длину датаграмм.

А может так случиться, что по дороге до адресата встретиться еще меньшая емкость. Тогда мы можем либо убить датаграмму, либо разрезать ее на куски:

За это отвечает флаг **DF** (0 – нельзя разрезать, 1 — можно). Fragment ID – **хэш по времени и адресу текущего маршрутизатора**, т. е. он уникален с точностью до коллизии. При разрезании датаграммы хэш будет записан во все куски, у не разрезанной (обычно) хэш равен 0. При разрезании **DF** устанавливается в 0, в fragment offset записывается номер первого октета куска.

Флаг **MF** – есть ли еще фрагменты, 1 у всех, кроме последнего.

Флаг **CF** у опции — если 1, то опция будет скопирована в заголовки всех фрагментов, иначе только в первый.

Так все должно работать, а долгое время работало так:

В стеке 4BSD реализация фрагментации содержала кучу ошибок + не было сказано, кто должен собирать датаграмму (очевидно, что получатель, было не для всех, поэтому собирали, где попало - могли собирать на следующем маршрутизаторе, а через него могли пройти не все элементы датаграммы)

Есть спец. ***буфер ассемблирования*** - как только приходит датаграмма с id != 0, выделяется память под total length, и дальше ждем всех фрагментов. Если через некоторое время не все пришло - все уничтожить (первоначально тайм-аут составлял 5 сек, что очень много) (Одна из типов атак тогда - спам кусками датаграмм) Одна из ошибок реализации: программисты считали, что фрагменты будут приходить по порядку, т.е. датаграмма считалась собранной, если пришел кусок с ***MF*** == 0. Если датаграмма не собралась полностью, то ее уничтожали + многие админы запрещали маршрутизацию фрагментированных датаграмм

Проблемы фрагментации хоть как-то решены только в ipv6.

**TCP датаграммы**

TCP-датаграммы называются **сегментами**.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 3 7 15 32 | | | |
| Source port | | | Destination port |
| Sequence Num | | | |
| Acknowledgment Num | | | |
| Head Length | Reserved | Flags | Window Size |
| Pseudoheader CRC16 | | | Pointer to URGent Data |
| Options | | | Padding |

Flags : UR6|ACK|PSH|RST|SYN|FIN

Поля:

* Первые два поля — номера портов отправителя/получателя (до 65536)
* Следующие два поля (\_ Num) – нужны для поддержания **целостности**, см. след. пару
* Header Length – как IHL в IP (4 бита)
* Flags – см. далее
* Window Size – для управление потоками (см. след. пару)
* Pointer … - см. далее
* Options – как в IP
* Padding – выравнивание
* Pseudoheader CRC16 — контрольная сумма **псевдозаголовка**, он состоит из ip-адресов отправителя и получателя, взятых из инкапсулирующей ip-датаграммы, номера протокола оттуда же, и из длины TCP-сегмента. Наличие этого заголовка нарушает логику инкапсуляции, но он был введен т. к. когда стек TCP/IP создавался, считалось, что в сетях с коммутацией пакетов очень большую опасность представляет вторжение man-in-the-middle, когда злоумышленник, находящийся между отправителем и получателем будет подменять информацию из ip-датаграмм или датаграммы целиком. Поэтому возникла идея, что нужен параметр, который будет смешивать данные из TCP и IP и который можно использовать для проверки целостности.

|  |  |
| --- | --- |
| 0 31 | |
| Source Address | |
| Destination Address | |
| Protocol | TCP Length |

Основные флаги:

* FIN — finalize, инициирует корректное завершение соединения
* SYN — synchronize, для установления соединения
* RST — reset, аварийной разрыв соединения (точнее, этот флаг выставляет сторона, которая не гарантирует целостность соединения, например, система работает от батарейки, а батарейка разряжена)
* PSH — обычно приложение работает так, чтобы заполнить TCP сегмент макс. размера, но для некоторых протоколов этого невозможно добиться, например, для протоколов, интерактивно взаимодействующих с пользователем (удаленный доступ, месседжеры и т. п.) Этот флаг выставляется, чтобы короткие сегменты не трактовались как ошибка.
* ACK — acknowledgment, для подтверждения целостности данных
* URG — urgent, для информирования системы о том, что в сегменте передаются какие-то данные повышенной важности, в urgent pointer выставляется последний октет этих важных данных (0, если их нет). Эти данные извлекаются и передаются в специальный буфер.