

Анализ влияния NAT на контрольные суммы TCP и UDP

Дунайцев Р.А. (СПбГУТ)

roman.dunaytsev@spbgut.ru

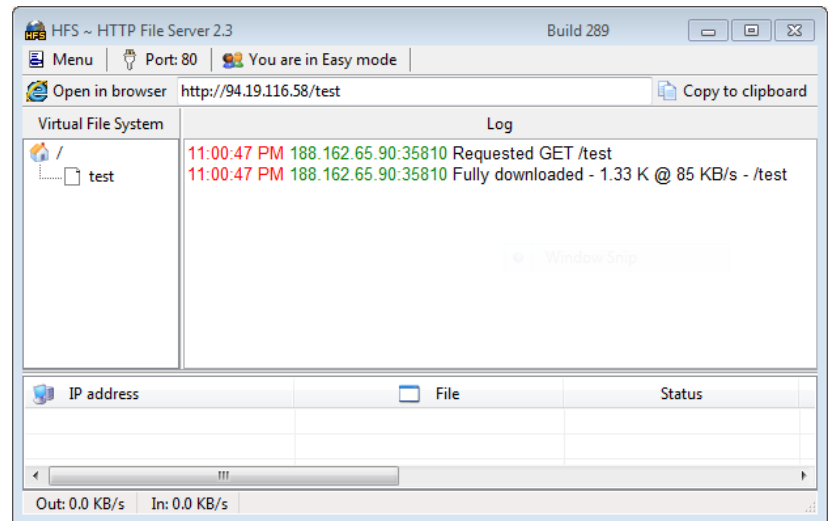
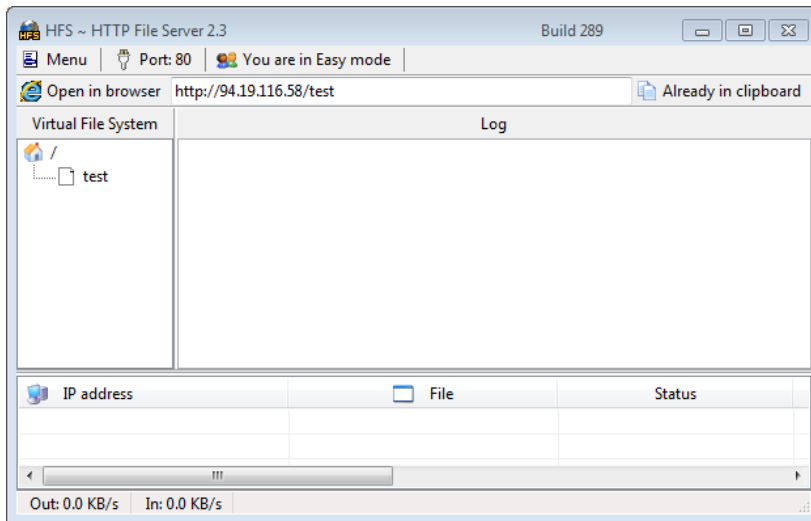
Краткая теоретическая справка

- IP-адрес считается глобально маршрутизируемым, если любой правильно настроенный маршрутизатор в сети Интернет может построить маршрут пакета так, что тот в конечном счете достигнет узла с этим IP-адресом. Для этого любой глобально маршрутизируемый адрес должен быть присвоен только одному узлу. Чтобы обеспечить уникальность глобально маршрутизируемых адресов, организация IANA (Internet Assigned Numbers Authority) и ее филиалы распределяют отдельные блоки IP-адресов между крупными организациями, такими как корпорации, университеты и Интернет-провайдеры, которые затем могут передавать эти адреса своим сотрудникам и клиентам, гарантируя уникальность каждого адреса. Так как IPv4 имеет лишь 32-битное адресное пространство, теоретически существует 4.294.967.296 глобальных IP-адресов. Из-за стремительного роста числа сетевых устройств и особенностей распределения IP-адресов организацией IANA это количество оказалось недостаточным. К счастью, благодаря механизму преобразования сетевых адресов (**Network Address Translation, NAT**) существует возможность подключения к сети Интернет целой подсети, имея единственный глобальный IP-адрес. Чтобы настроить NAT, каждому узлу в этой подсети нужно присвоить локально маршрутизируемый IP-адрес. Обеспечивать уникальность таких адресов не требуется, потому что адреса не являются глобально маршрутизируемыми. В настоящее время NAT используется повсеместно, особенно в домашних сетях и сетях небольших организаций.

Задание на дом

- Экспериментальным путем выяснить, пересчитываются ли контрольные суммы TCP при использовании NAT
 - 1) Найти 2 ПК, один с публичным IP-адресом (например, выдается некоторыми Интернет-провайдерами), а другой с частным IP-адресом
 - 2) Установить между ними TCP-соединение. В качестве сервера можно использовать бесплатную и не требующую установки программу HFS <https://www.rejetto.com/hfs/>
 - 3) С помощью Wireshark собрать пакеты этого TCP-соединения на той и другой стороне (т.е. захват трафика должен выполняться **одновременно как на клиенте, так и на сервере**)
 - 4) Сравнить контрольную сумму **одного и того же** TCP-сегмента, захваченного Wireshark при отправке клиентом и получении сервером. Для сравнения лучше использовать TCP-сегмент с HTTP-запросом GET, как уникальный и легко узнаваемый
 - 5) В качестве альтернативы эксперимент можно провести с UDP
 - 6) **К отчету приложить собранные Wireshark пакеты!**

HFS ~ HTTP File Server



Запрос GET на стороне клиента

The image shows a Wireshark network traffic capture. The main pane displays a list of packets. Packet 4 is highlighted, showing an HTTP GET request to /test. The details pane below shows the structure of this request, including the Host, User-Agent, Accept, and Accept-Language headers. The hex and ASCII panes at the bottom show the raw data of the captured packet.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|----------|--------------|--------------|----------|--------|--|
| 1 | 0.000000 | 10.0.0.10 | 94.19.116.58 | TCP | 66 | 49506 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1360 WS=4 SACK_PERM=1 |
| 2 | 0.153417 | 94.19.116.58 | 10.0.0.10 | TCP | 66 | 80 → 49506 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1360 WS=256 SACK_PERM=1 |
| 3 | 0.153524 | 10.0.0.10 | 94.19.116.58 | TCP | 54 | 49506 → 80 [ACK] Seq=1 Ack=1 Win=66640 Len=0 |
| 4 | 0.153786 | 10.0.0.10 | 94.19.116.58 | HTTP | 344 | GET /test HTTP/1.1 |
| 5 | 0.405124 | 94.19.116.58 | 10.0.0.10 | TCP | 320 | 80 → 49506 [PSH, ACK] Seq=1 Ack=291 Win=66560 Len=266 [TCP segment of a reassembled PDU] |
| 6 | 0.405342 | 94.19.116.58 | 10.0.0.10 | HTTP | 1414 | HTTP/1.1 200 OK |
| 7 | 0.405375 | 10.0.0.10 | 94.19.116.58 | TCP | 54 | 49506 → 80 [ACK] Seq=291 Ack=1627 Win=66640 Len=0 |
| 8 | 5.171183 | 10.0.0.10 | 94.19.116.58 | TCP | 54 | 49506 → 80 [FIN, ACK] Seq=291 Ack=1627 Win=66640 Len=0 |
| 9 | 5.206225 | 94.19.116.58 | 10.0.0.10 | TCP | 54 | 80 → 49506 [ACK] Seq=1627 Ack=292 Win=66560 Len=0 |

Frame 4: 344 bytes on wire (2752 bits), 344 bytes captured (2752 bits) on interface \Device\NPF_{BD9A8DA3-4074-4B91-93C1-98AD26C74AF4}, id 0
Ethernet II, Src: HyundaiIn_f0:1a:40 (00:09:3b:f0:1a:40), Dst: GCTSemif_f0:16:70 (00:0a:3b:f0:16:70)
Internet Protocol Version 4, Src: 10.0.0.10, Dst: 94.19.116.58
Transmission Control Protocol, Src Port: 49506, Dst Port: 80, Seq: 1, Ack: 1, Len: 290
Hypertext Transfer Protocol
GET /test HTTP/1.1\r\nHost: 94.19.116.58\r\nUser-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:28.0) Gecko/20100101 Firefox/28.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nConnection: keep-alive\r\n\r\n[Full request URI: http://94.19.116.58/test]
[HTTP request 1/1]
[Response in frame: 6]

```
0000 00 0a 3b f0 16 70 00 09 3b f0 1a 40 08 00 45 00  ..p...;.@-E-
0010 01 4a 15 1e 40 00 80 06 08 39 0a 00 00 0a 5e 13  .J..@...9...^
0020 74 3a c1 62 00 50 47 64 40 bb 7c da a1 b8 50 18  t:b:PGd@|...P-
0030 41 14 0b 2e 0e 00 47 45 54 20 2f 74 65 73 74 20  A...GE T /test
0040 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20  HTTP/1.1 ..Host:
0050 39 34 2e 31 39 2e 31 31 36 2e 35 38 0d 0a 55 73  94.19.11 6.58..Us
```

Запрос GET на стороне сервера

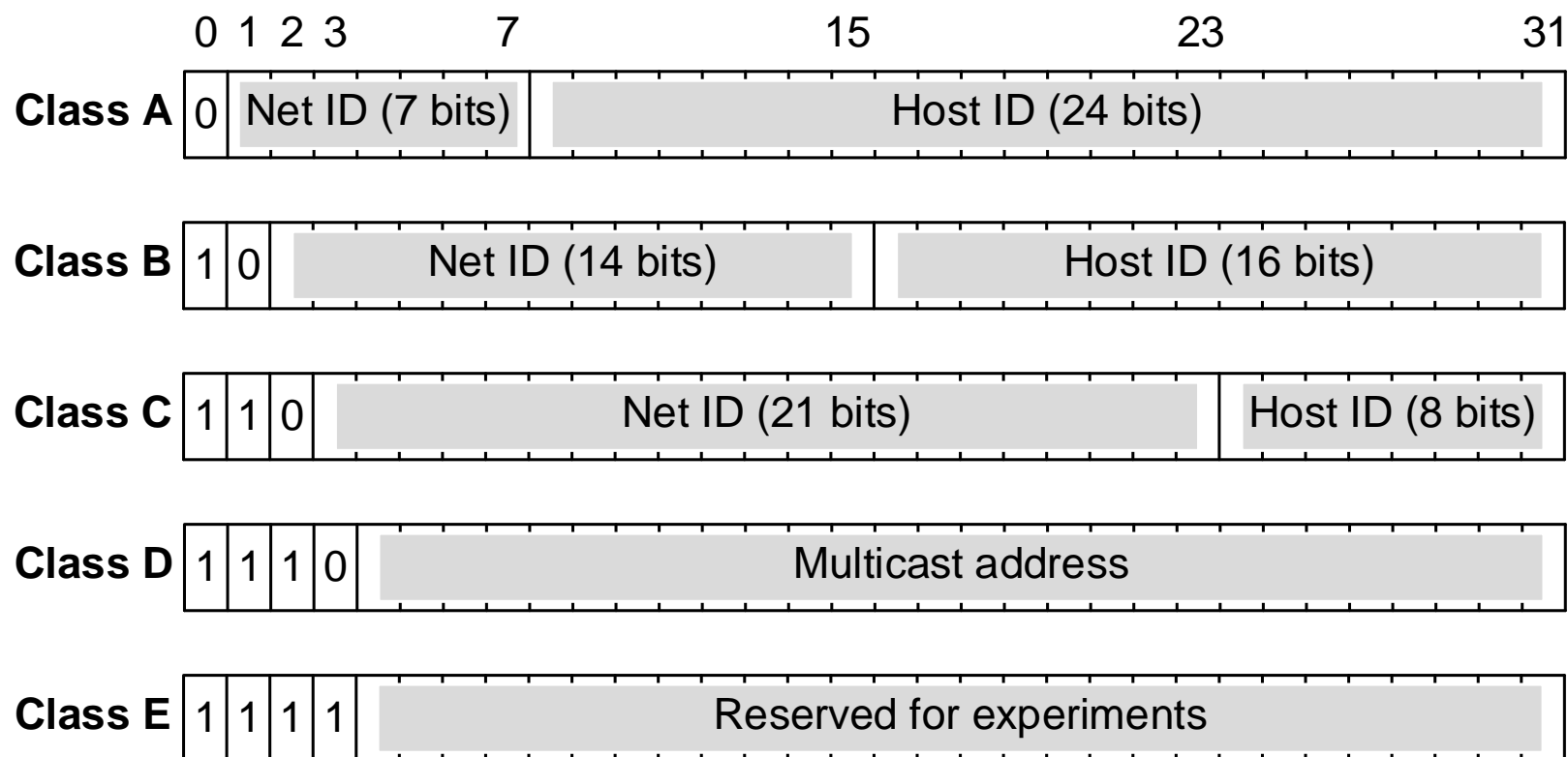
The image shows a Wireshark network traffic capture window titled "server-skyнет-94.18.116.58.pcapng". The main pane displays a list of network packets. Packet 4 is highlighted, showing an HTTP GET request for "/test" from source IP 188.162.65.123 to destination IP 94.19.116.58 on port 80. The details pane below shows the structure of this packet, including Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, and Hypertext Transfer Protocol. The HTTP section shows the request line "GET /test HTTP/1.1\r\n" and various headers like Host, User-Agent, Accept, and Connection. The bottom pane shows the raw packet data in hexadecimal and ASCII.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|----------|----------------|----------------|----------|--------|--|
| 1 | 0.000000 | 188.162.65.123 | 94.19.116.58 | TCP | 66 | 61198 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1360 WS=4 SACK_PERM=1 |
| 2 | 0.000126 | 94.19.116.58 | 188.162.65.123 | TCP | 66 | 80 → 61198 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 3 | 0.058678 | 188.162.65.123 | 94.19.116.58 | TCP | 60 | 61198 → 80 [ACK] Seq=1 Ack=1 Win=66640 Len=0 |
| 4 | 0.152393 | 188.162.65.123 | 94.19.116.58 | HTTP | 344 | GET /test HTTP/1.1 |
| 5 | 0.157074 | 94.19.116.58 | 188.162.65.123 | TCP | 320 | 80 → 61198 [PSH, ACK] Seq=1 Ack=291 Win=66560 Len=266 [TCP segment of a reassembled PDU] |
| 6 | 0.157331 | 94.19.116.58 | 188.162.65.123 | HTTP | 1414 | HTTP/1.1 200 OK |
| 7 | 0.318439 | 188.162.65.123 | 94.19.116.58 | TCP | 60 | 61198 → 80 [ACK] Seq=291 Ack=1627 Win=66640 Len=0 |
| 8 | 5.078351 | 188.162.65.123 | 94.19.116.58 | TCP | 60 | 61198 → 80 [FIN, ACK] Seq=291 Ack=1627 Win=66640 Len=0 |
| 9 | 5.078466 | 94.19.116.58 | 188.162.65.123 | TCP | 54 | 80 → 61198 [ACK] Seq=1627 Ack=292 Win=66560 Len=0 |

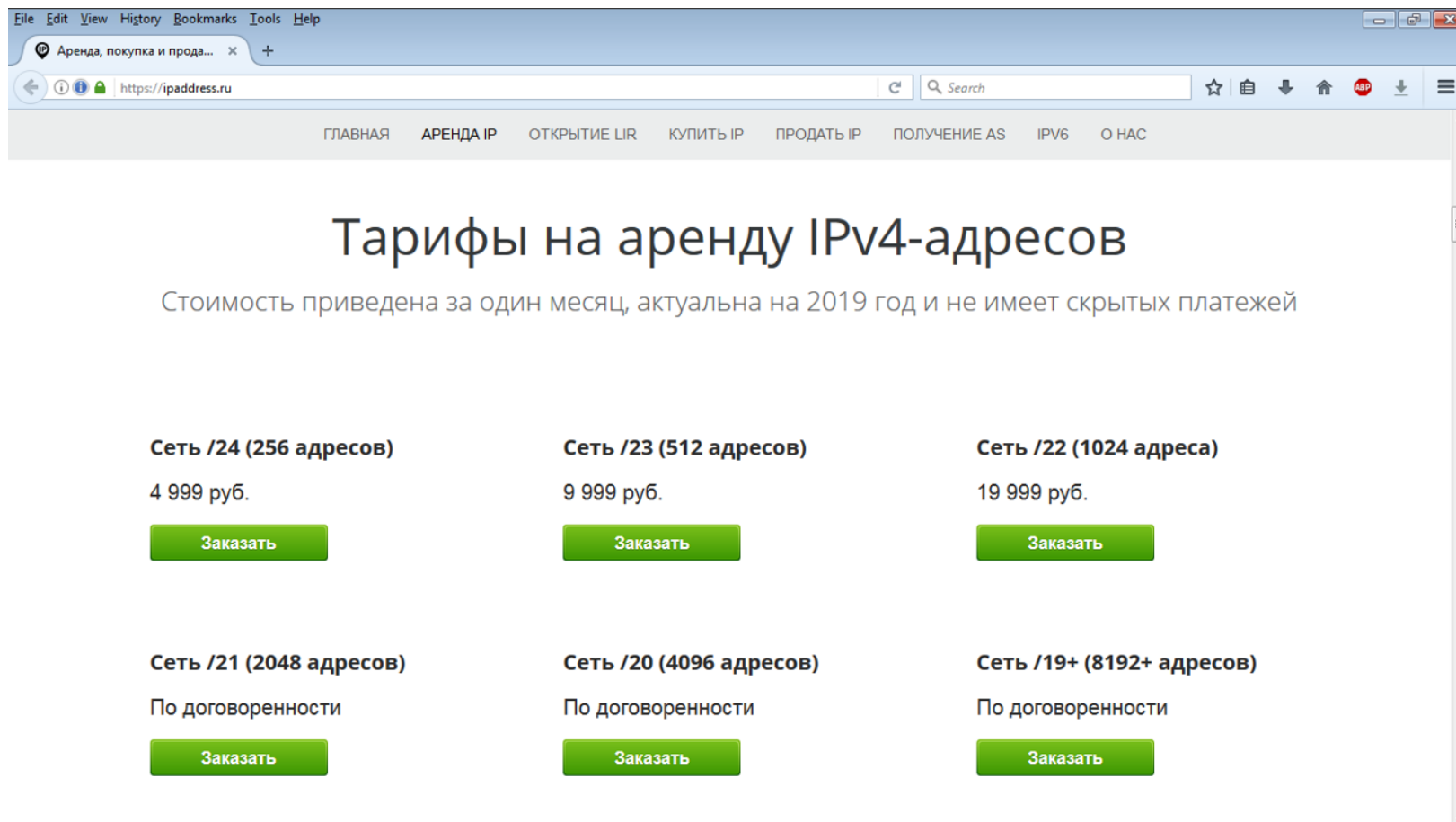
Frame 4: 344 bytes on wire (2752 bits), 344 bytes captured (2752 bits) on interface \Device\NPF_{B772A6FC-D51B-46D9-B1DB-39D3A230A05F}, id 0
Ethernet II, Src: Cisco_23:f5:40 (b4:14:89:23:f5:40), Dst: WistronI_25:bb:5d (f0:de:f1:25:bb:5d)
Internet Protocol Version 4, Src: 188.162.65.123, Dst: 94.19.116.58
Transmission Control Protocol, Src Port: 61198, Dst Port: 80, Seq: 1, Ack: 1, Len: 290
Hypertext Transfer Protocol
GET /test HTTP/1.1\r\nHost: 94.19.116.58\r\nUser-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:28.0) Gecko/20100101 Firefox/28.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nConnection: keep-alive\r\n\r\n[Full request URI: http://94.19.116.58/test]
[HTTP request 1/1]
[Response in frame: 6]

```
0000 f0 de f1 25 bb 5d b4 14 89 23 f5 40 08 00 45 00  ...%...]..#.@..E-
0010 01 4a 15 1e 40 00 73 06 21 25 bc a2 41 7b 5e 13  .J..@.s. !%..A(^
0020 74 3a ef 0e 00 50 8a 62 16 a3 46 29 ad b7 50 18  t:...P.b..F)..P-
0030 41 14 fb 39 00 00 47 45 54 20 2f 74 65 73 74 20  A..9..GE T /test
0040 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20  HTTP/1.1 ..Host:
0050 39 34 2e 31 39 2e 31 31 36 2e 35 38 0d 0a 55 73  94.19.11 6.58 ..Us
```

Классовая адресация



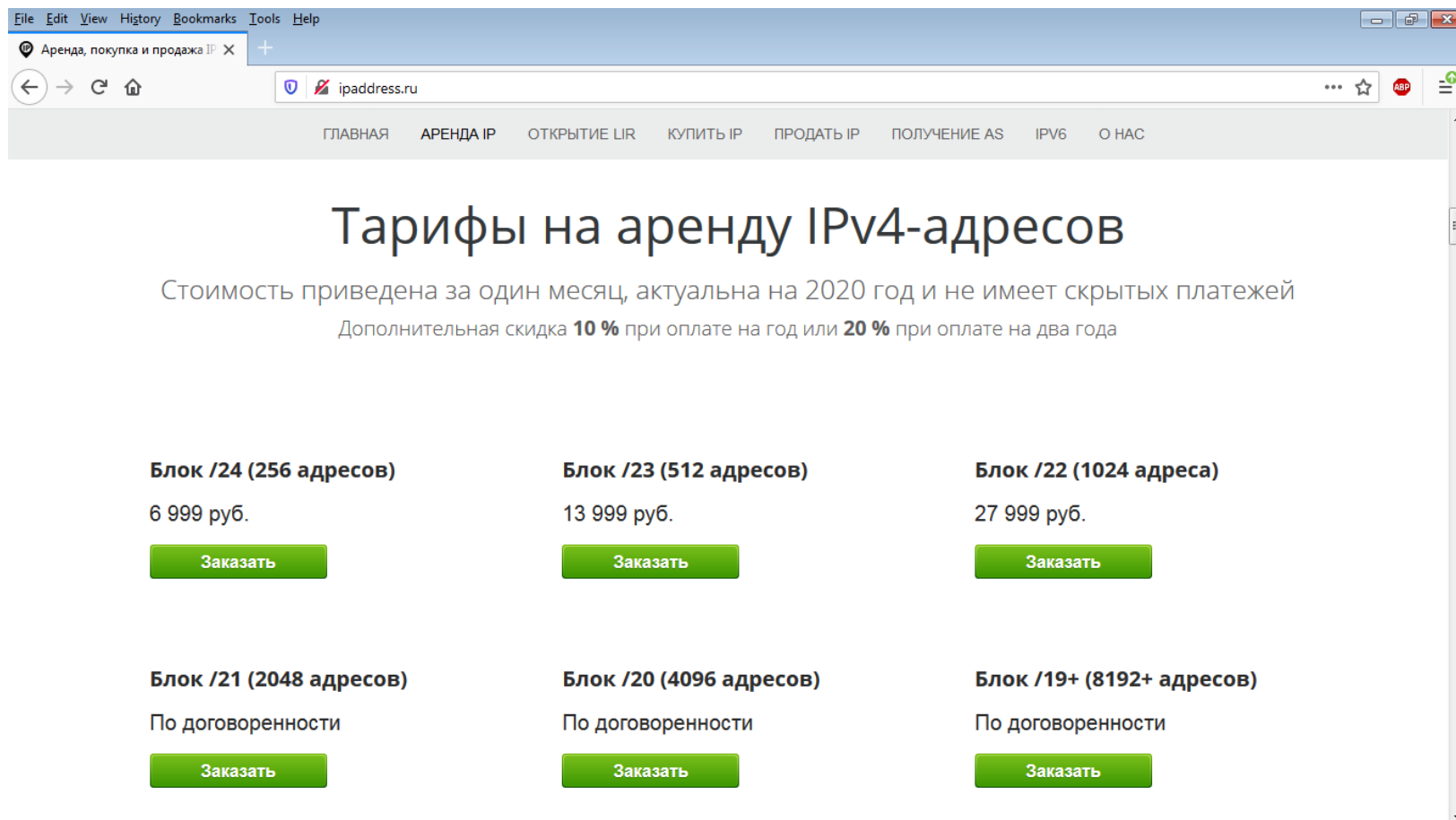
Публичные IP-адреса стоят денег



The screenshot shows a web browser window with the URL <https://ipaddress.ru>. The page title is "Тарифы на аренду IPv4-адресов" (Rates for IPv4 address rental). Below the title, it states: "Стоимость приведена за один месяц, актуальна на 2019 год и не имеет скрытых платежей" (The cost is for one month, current as of 2019, and has no hidden payments). The page displays six rental options in a grid:

| Сеть /24 (256 адресов) | Сеть /23 (512 адресов) | Сеть /22 (1024 адреса) |
|--------------------------|--------------------------|---------------------------|
| 4 999 руб. | 9 999 руб. | 19 999 руб. |
| Заказать | Заказать | Заказать |
| Сеть /21 (2048 адресов) | Сеть /20 (4096 адресов) | Сеть /19+ (8192+ адресов) |
| По договоренности | По договоренности | По договоренности |
| Заказать | Заказать | Заказать |

И стоимость их аренды растет



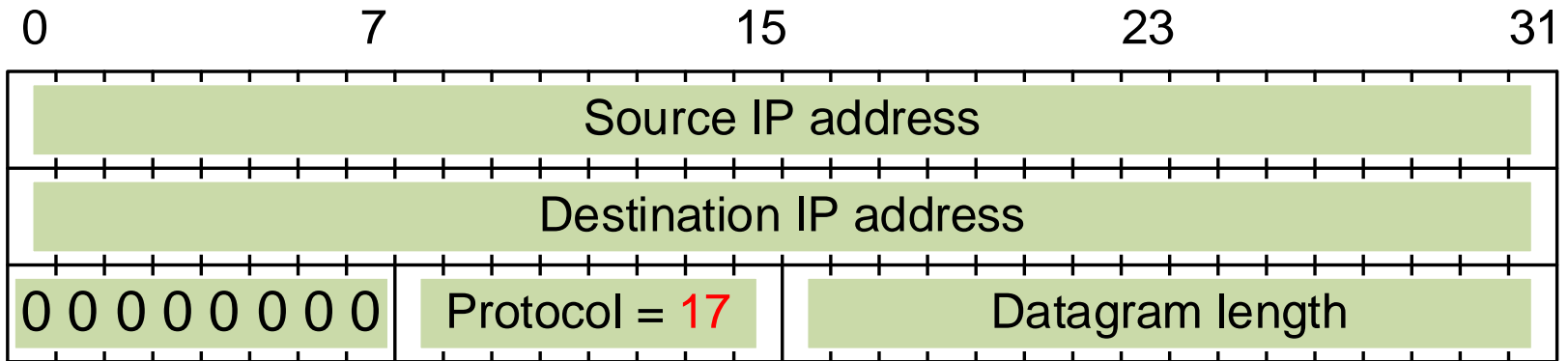
The screenshot shows a web browser window with the URL `ipaddress.ru`. The page title is "Тарифы на аренду IPv4-адресов". Below the title, there is a note: "Стоимость приведена за один месяц, актуальна на 2020 год и не имеет скрытых платежей" and "Дополнительная скидка 10 % при оплате на год или 20 % при оплате на два года". The page lists six different IPv4 address blocks with their respective rental prices and "Заказать" (Order) buttons.

| Блок /24 (256 адресов) | Блок /23 (512 адресов) | Блок /22 (1024 адреса) |
|--------------------------|--------------------------|---------------------------|
| 6 999 руб. | 13 999 руб. | 27 999 руб. |
| Заказать | Заказать | Заказать |
| Блок /21 (2048 адресов) | Блок /20 (4096 адресов) | Блок /19+ (8192+ адресов) |
| По договоренности | По договоренности | По договоренности |
| Заказать | Заказать | Заказать |

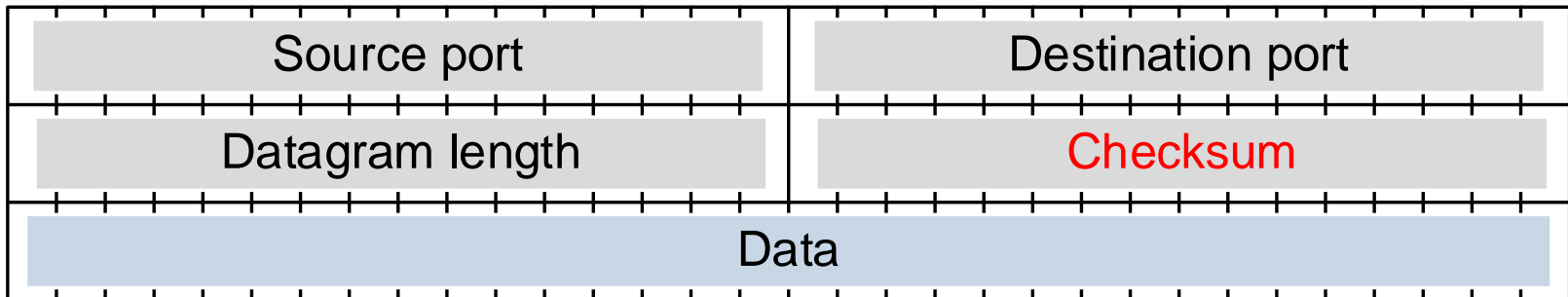
Частные IP-адреса

- 10.0.0.0 – 10.255.255.255 (маска подсети /8)
- 172.16.0.0 – 172.31.255.255 (маска подсети /12)
- 192.168.0.0 – 192.168.255.255 (маска подсети /16)

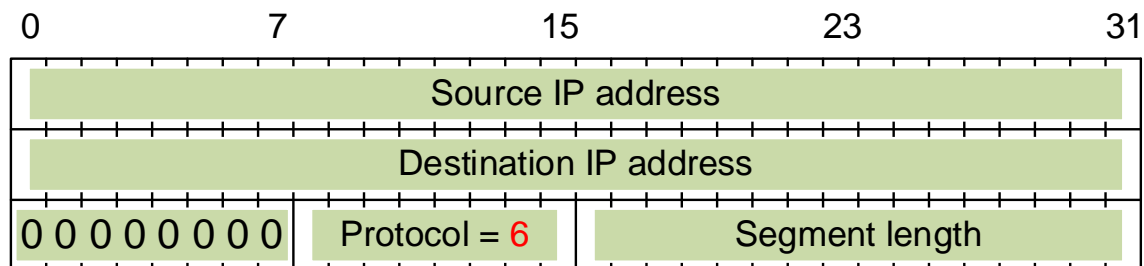
Контрольная сумма UDP



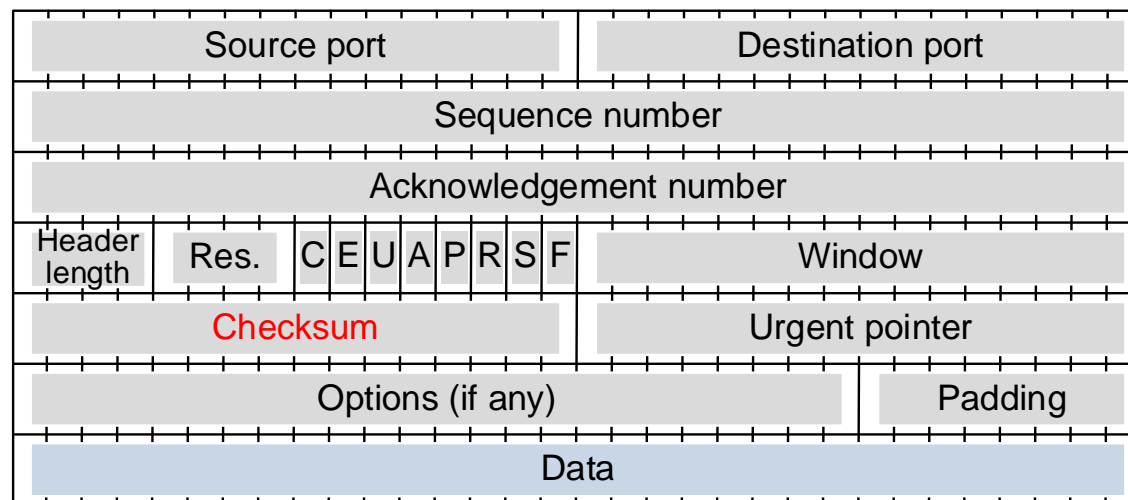
+



Контрольная сумма TCP



+



Пример ТСП-соединения с NAT

| Клиент | NAT | Сервер |
|----------------------|---------------------------|----------------------|
| 10.0.0.10 (49506) | 188.162.65.123 (61198) | 94.19.116.58 (80) |

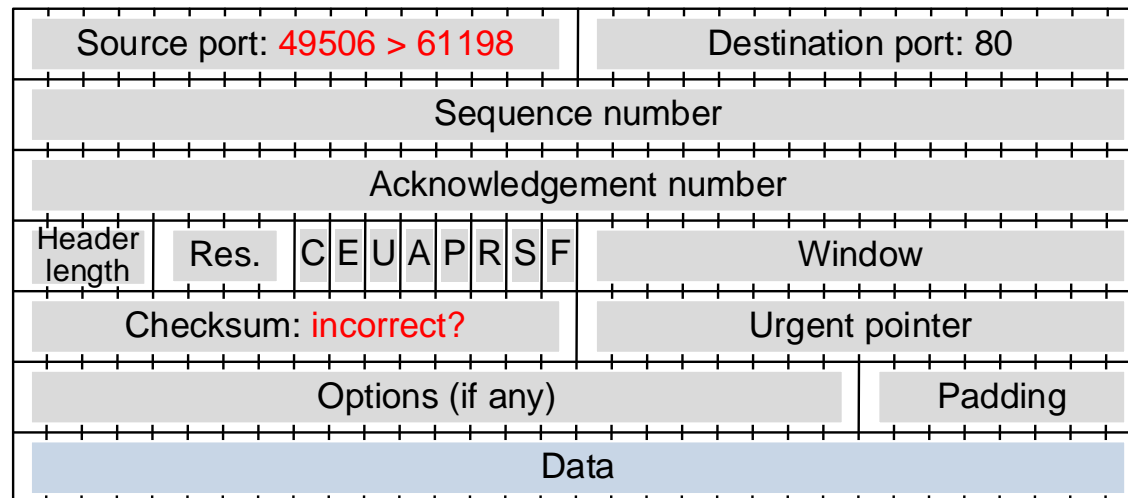
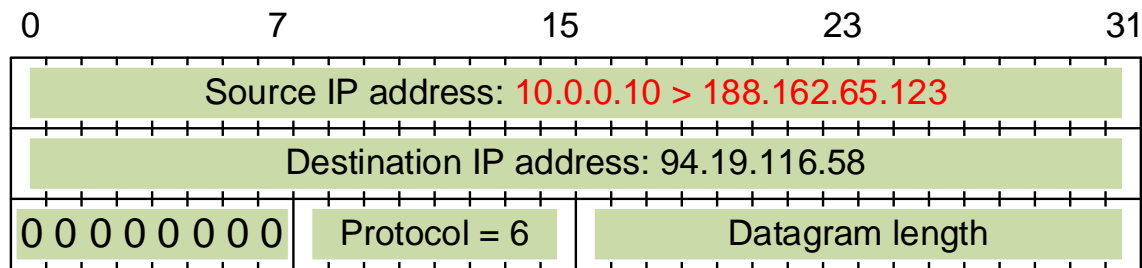
Что видит клиент:

| | | |
|----------------------|---|----------------------|
| 10.0.0.10 (49506) | > | 94.19.116.58 (80) |
|----------------------|---|----------------------|

Что видит сервер:

| | | |
|---------------------------|---|----------------------|
| 188.162.65.123 (61198) | > | 94.19.116.58 (80) |
|---------------------------|---|----------------------|

Как быть с контрольной суммой?



Пересчитывать или отказаться?

- При использовании NAT происходит замена локально маршрутизируемого IP-адреса (т.н. частный/внутренний/серый) на глобально маршрутизируемый (т.н. публичный/внешний/белый)
- Для возможности обратного преобразования часто также меняется и номер порта клиента (**Port Address Translation, PAT**)
- Поскольку исходная контрольная сумма рассчитывалась для прежних значений, подобная замена делает ее неверной и такой TCP-сегмент или UDP-дейтаграмма получателем будет отброшены как ошибочные
- Следовательно, необходимо либо пересчитывать контрольную сумму TCP и UDP всякий раз, когда пакет проходит через NAT туда или обратно, либо отказаться от использования контрольных сумм в TCP и UDP, как это сделано в IPv6 (ее там просто нет)