20. Транспортен слой. Транспортни протоколи TCP и UDP.
21. Управление на потока и задръстванията.

TCP. Управление на потока и задръстванията.

TCP има и механизми за управление на потока (flow control).

Flow control синхронизира скоростите на потока от данни между двете страни (приложенията) в сесията.

Когато източникът е информиран, че определено количество данни в сегментите е получено, тогава може да продължи с изпращане на повече данни за дадената сесия.

TCP. Управление на потока и задръстванията.

Полето Window Size определя количеството данни, което може да бъде предадено, преди да бъде получено потвърждение.

Първоначалният размер на прозореца се определя в началото на сесията чрез three-way handshake.

Механизмът за обратна връзка в TCP нагласява ефективната скорост на предаване на данните към максималния поток, който мрежата и устройството-получател могат да поддържат без загуби и без повторни предавания.

TCP. Управление на потока и задръстванията.

В примера на следващия слайд първоначалният размер на прозореца е 3000 байта.

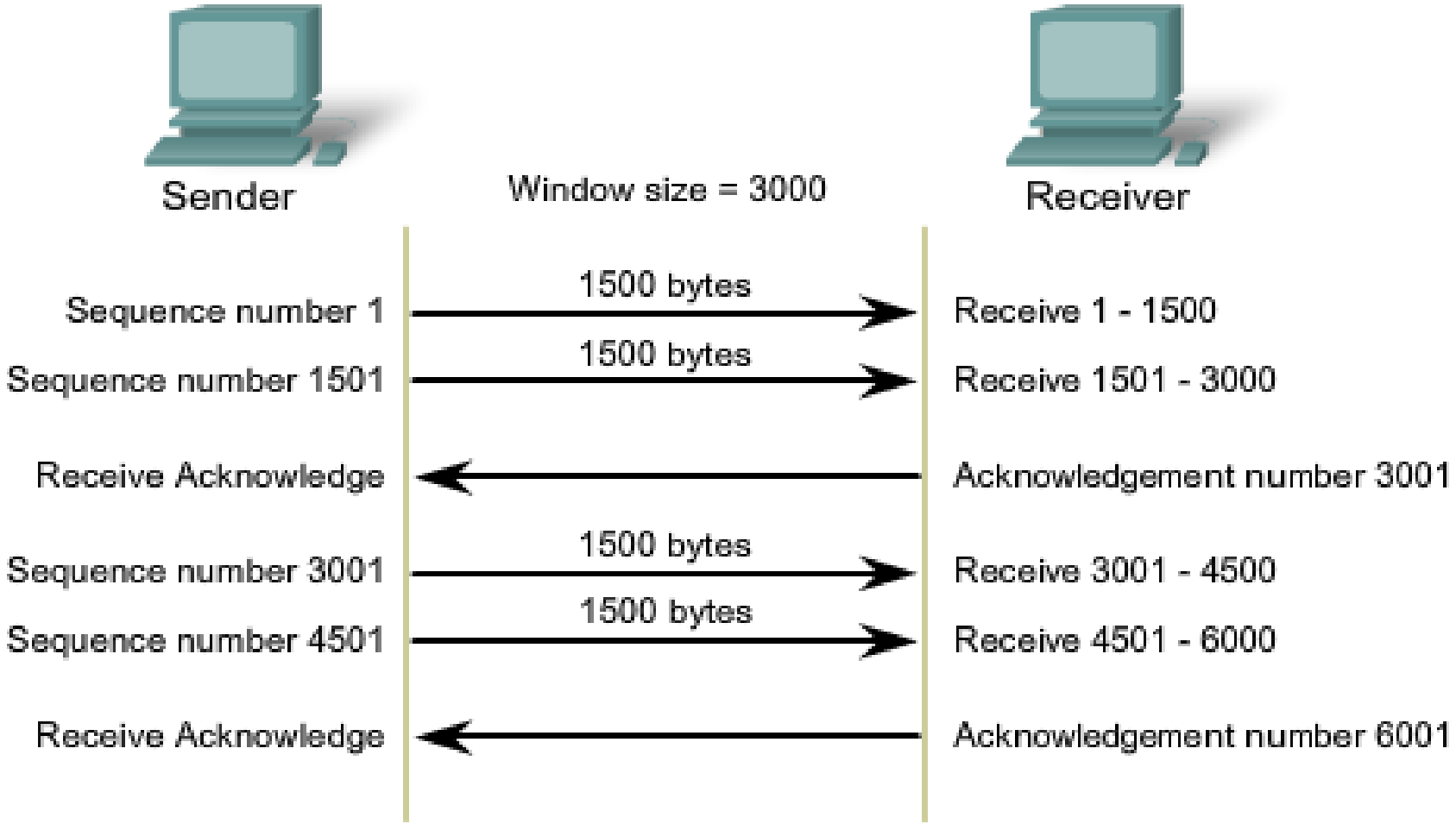
След предаването на тези 3000 байта изпращачът очаква потвърждение, преди да продължи със следващи сегменти. След получаването му може да предаде още 3000 байта.

В периоди, когато мрежата е задръстена или на получателя му липсват ресурси, закъснението може да се увеличи.

И ефективната скорост на предаване на данните за тази сесия да намалее.

Така се справя с проблема недостиг на ресурси.

TCP. Управление на потока и задръстванията.



TCP. Редуциране на загубите.

Друг начин за контролиране на потока от данни е да се определя размера на прозореца динамично.

При недостиг на мрежови ресурси TCP редуцира размера на прозореца, с което намалява и скоростта на предаване.

Хостът-получател в TCP сесията изпраща към подателя стойност на размера на прозореца, която показва броя на байтовете, които е в състояние да получи.

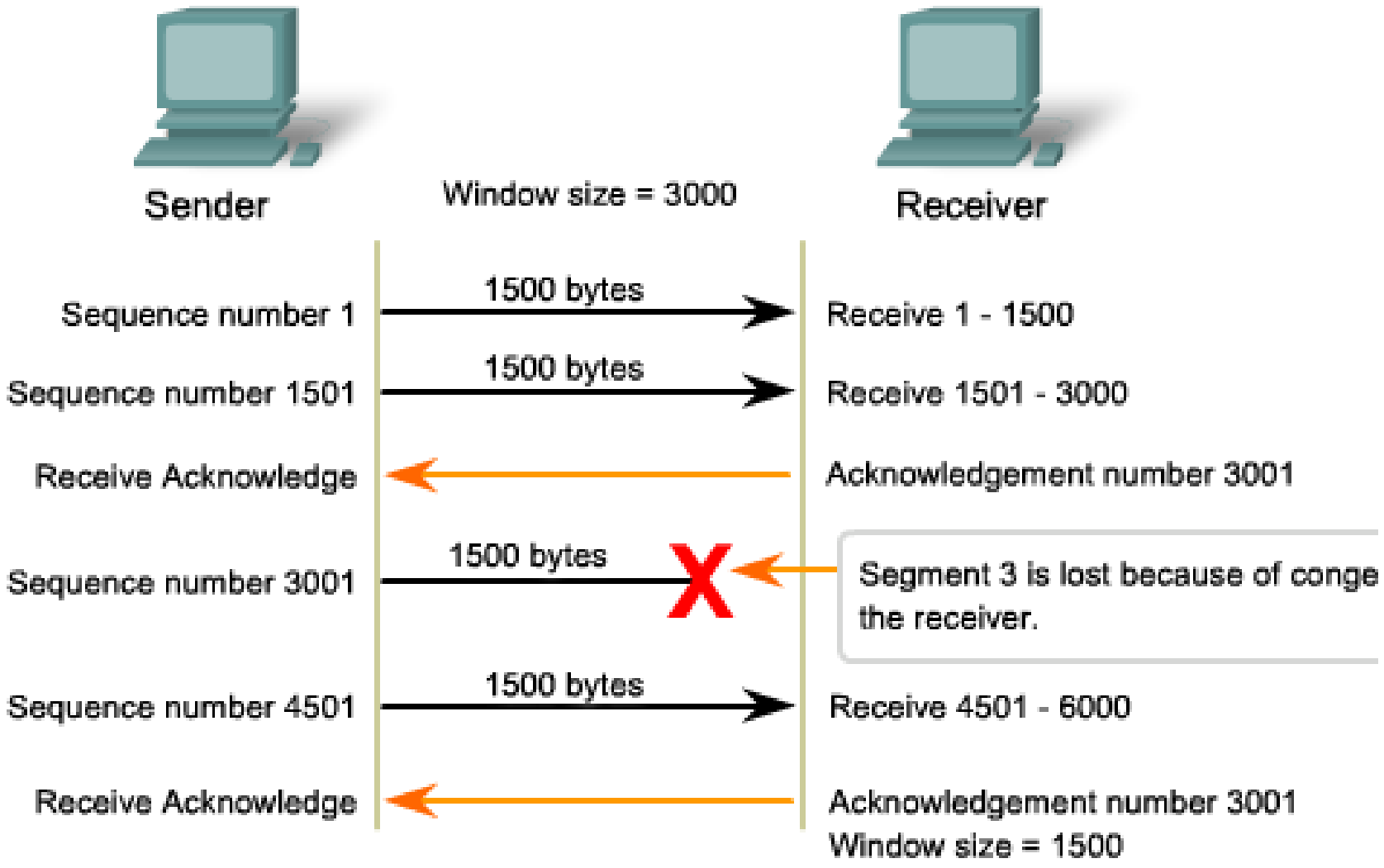
Както се вижда на фигурата в по-следващия слайд, имаме загуба на един сегмент. Получателят променя размера на прозореца от 3000 на 1500.

TCP. Редуциране на загубите.

След периоди без загуби на данни или липса на ресурси получателят ще започне да увеличава прозореца. Това ще вдигне ефективната скорост и ще продължи, докато се появят загуби и трябва да се намали прозореца.

Динамичното увеличаване и намаляване на прозореца е непрекъснат процес в TCP, който определя оптималния размер *във всеки един момент за дадена сесия*.

TCP. Редуциране на загубите.



Window scaling

GNU Linux поддържа подобрения за висока производителност в TCP (RFC 1323).

Window scaling – да използва по-дълги прозорци (> 64K)

($2^{16} = 65536$)

window scale “разширява” опцията TCP window до 32 бита и чрез коефициент на мащабиране пренася 32-битовата стойност в 16-битовото поле Window на TCP заглавието.

Window scaling

```
less
  /proc/sys/net/ipv4/tcp_window_scaling
1
```

В Linux е пуснато по подразбиране, като се увеличават буферите за предаване и приемане.

```
less /proc/sys/net/ipv4/tcp_wmem
4096    16384   4194304
```

(Min default max)

```
less /proc/sys/net/ipv4/tcp_rmem
4096    87380   4194304
```

Размер на прозореца при приемника

Зависи от обема на буферното пространство

TCP процесът от страната на приемника трябва да поддържа:

```
LastByteRcvd-LastByteRead ≤
MaxRcvBuffer
```

за да се избегне препълване на буфера.

Затова се рекламира прозорец:

```
AdvertisedWindow =
MaxRcvBuffer-((NextByteExpected-1) -
LastByteRead)
```

Размер на прозореца при приемника (2)

т.е. количеството свободно пространство в буфера.

Приемникът потвърждава получаването на данните тогава и само тогава, когато са пристигнали всички предишни байтове.

Размерът на прозореца зависи от това колко бързо приложният процес прочита данните.

Ако това става със скоростта на пристигането им (LastByteRead се инкрементира със същата скорост като LastByteRcvd),

`AdvertisedWindow = MaxRcvBuffer)`.

Ако приложната програма изостава в четенето, прозорецът намалява с всеки сегмент и може да достигне 0.

Размер на прозореца при предавателя

TCP процесът от страна на предавателя се придържа към прозореца, който се рекламира от приемника. Т.е.:

```
LastByteSent-LastByteAcked ≤
AdvertisedWindow
```

Предавателят изчислява ефективния прозорец:

```
EffectiveWindow =
AdvertisedWindow-(LastByteSent-LastByt
eAcked)
```

`EffectiveWindow > 0`

Изчисляване на прозореца по високоскоростна линия

Bandwidth [bps] * RTT [s] = TCP window [bits] / 8 = TCP window [Bytes]

В примера:

1,000,000,000 bps * 0.030 s = 30,000,000 bits / 8 = 3,750,000 Bytes

MSS (IPv4)

TCP контролира максималната дължина на сегмента - Maximum Segment Size (**MSS**) за всяка връзка.

За директно свързаните мрежи TCP изчислява MSS въз основа на дължината на MTU на интерфейса:

`MSS = MTU - (IPHL + TCPHL)`*

Напр., Ethernet: MTU = 1500 байта

`MSS = 1500 - (20 + 20) = 1460 байта`

* HL – Header Length

TCP Jumbograms (IPv6)

В TCP header няма поле за дължина, т.е. няма какво да ограничи дължината на TCP пакета (RFC 2675).

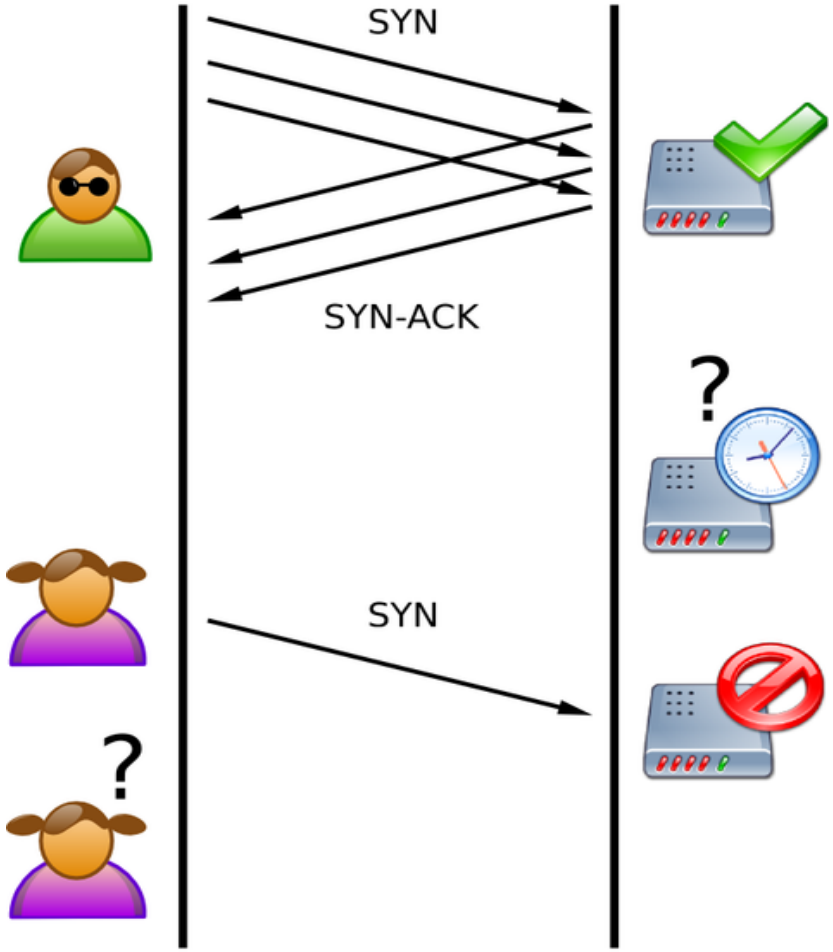
Максималната дължина на сегмента (MSS), която се уговаря в началото на сесията, определя максимална дължина на TCP пакета.

Ако (MTU - 60) >= 65535, MSS = 65535.

При получаване на MSS = 65535, приема се за ∞.

(MSS = Path MTU - 60).

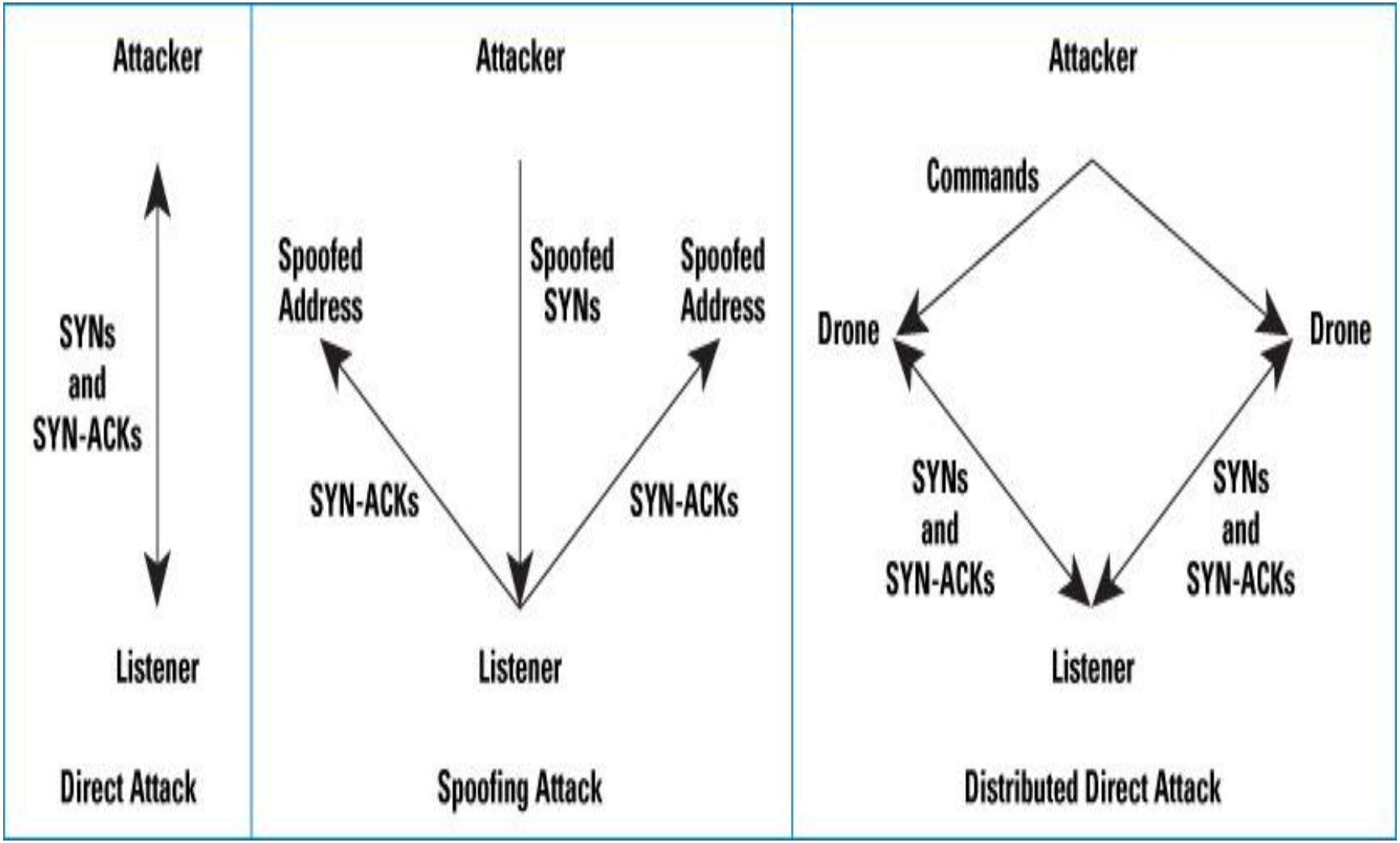
SYN flood атаки



Атакуващият изпраща няколко SYN, но не връща ACK

В резултат: полуотворени конекции. Няма място за легитимния потребител

SYN flood атаки. Варианти.



Защита от SYN flood. Вградена.

В конфигурацията на tcp:

tcp_synack_retries (integer; default: 5)

Максималният брой повторни предавания на SYN/ACK сегмент. **< 255**

tcp_max_syn_backlog

Максималният брой заявки в опашката, които не са получили acknowledgement от клиента. default = 256; = 1024 (RAM >= 128MB); = 128 (RAM <= 32MB).

(вж. /proc/sys/net/ipv4)

Защита. Ръчно.

less /etc/sysconfig/iptables

```
iptables -A INPUT -p tcp --dport 80 --syn -m limit --limit 1/s -j ACCEPT
# не повече от 10 SYN
iptables -I INPUT -p tcp --syn --dport 80 -j DROP -m iplimit --iplimit-above
    10 -j DROP
# проверка "New not syn:"
iptables -A bad_tcp_packets -p tcp --dport 80  !--syn -m state --state NEW
    -j LOG --log-prefix "New not syn:"
iptables  -A bad_tcp_packets -p tcp --dport 80  !--syn -m state --state NEW
    -j DROP
# При условие, че 400 заявки вече са минали, следващите се
# отхвърлят, ако са повече от 300 в секунда:
iptables –A INPUT -d $IP_web -p tcp --dport 80 -m state  --state NEW –m
 limit –limit 300/second -limit-burst 400 -j DROP
# Максимум 10 едновременни съединения към порт 80 от едно IP:
iptables -A INPUT -p tcp --dport 80 -m iplimit --iplimit-above 10 -j DROP
# 20 съединения с мрежа от клас C:
iptables -I INPUT -p tcp --dport 80 -m iplimit --iplimit-above 20 --iplimit-
    mask 24 -j DROP
```

SYN flood. Пример.

За да се определи дали системата е атакувана, може да се използва командата netstat. Многочислените съединения в състояние SYN_RECV свидетелстват за това, че именно в този момент протича атаката.

```
# netstat -n -p TCP
tcp 0 0 10.100.0.200:21 237.177.154.8:25882 SYN_RECV -
tcp 0 0 10.100.0.200:21 236.15.133.204:2577 SYN_RECV -
tcp 0 0 10.100.0.200:21 127.160.6.129:51748 SYN_RECV -
tcp 0 0 10.100.0.200:21 230.220.13.25:47393 SYN_RECV -
tcp 0 0 10.100.0.200:21 227.200.204.182:60427 SYN_RECV -
tcp 0 0 10.100.0.200:21 232.115.18.38:278 SYN_RECV -
tcp 0 0 10.100.0.200:21 229.116.95.96:5122 SYN_RECV -
tcp 0 0 10.100.0.200:21 236.219.139.207:49162 SYN_RECV -
tcp 0 0 10.100.0.200:21 238.100.72.228:37899 SYN_RECV –
```

UDP

UDP (**RFC 768**) е по-опростен транспортен протокол с неустановена връзка.

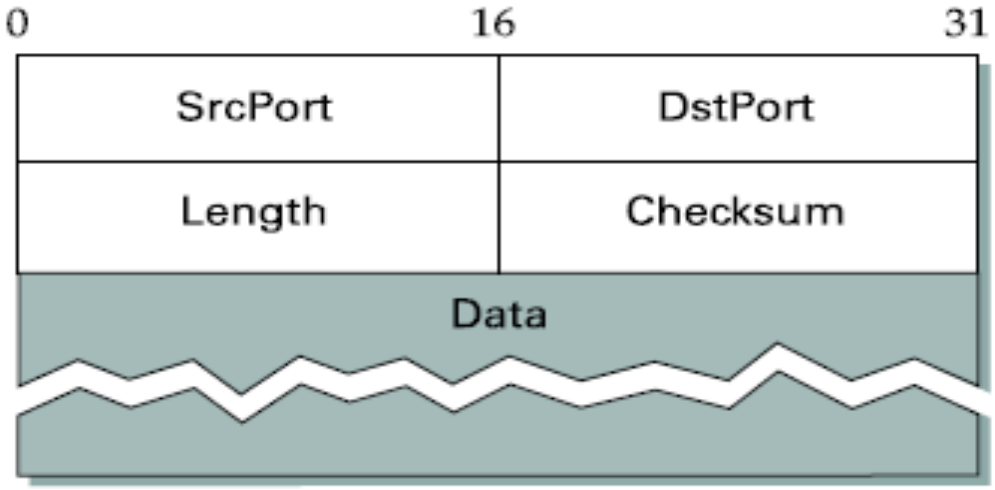
Това не означава, че приложенията, които се базират на UDP, са непременно ненадеждни. Функциите за надеждност се осъществяват от по-горе лежащите протоколи.

Прилага се там, където закъсненията и синхронизацията са критични, а не загуби на пакети:

Онлайн игрите или Vo (Video)IP.

Други приложения като DNS или TFTP ще повторят заявката, ако не получат отговор. И не им трябват гаранциите на TCP.

UDP дейтаграма



UDP единствено гарантира точността на съобщението чрез checksum. (Опция в IPv4 и задължителна в IPv6.):

UDP дейтаграма

- UDP header и UDP data;
- *pseudoheader*: от IP header – No. protocol, source IP и destination IP. Плюс **UDP length**.)

UDP. Възстановяване на дейтаграми.

UDP е connectionless и сесии не се установяват както в TCP. UDP е по-скоро транзакционен протокол. Т.е., ако приложението има данни за предаване, то ги предава.

Много UDP-базирани приложения изпращат малки количества данни, които се побират в единствен пакет.

Но някои изпращат по-големи количества данни, които се разделят на множество пакети.

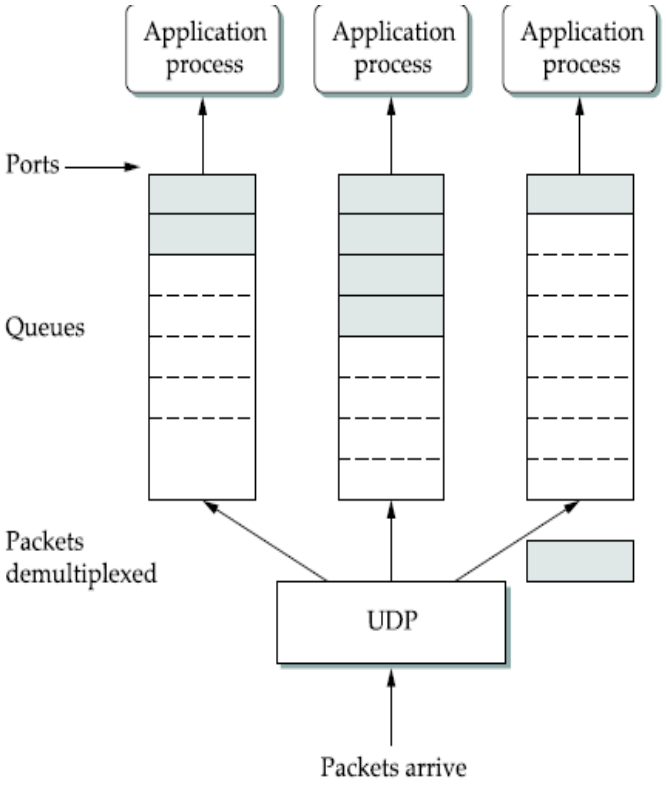
При изпращане на множеството дейтаграми от едно приложение, те могат да поемат различни пътища в мрежата и да пристигнат в разбъркан ред.

UDP не следи последователността на дейтаграмите при приемане като TCP. Ако последователността е от значение, за това се грижи приложната програма.

UDP порт – опашка от съобщения.

Пристигнало съобщение (UDP дейтаграма) се поставя в опашката за съответен порт (приложна програма), ако има място.

Иначе се изхвърля. Няма управление на потока.



UDP. Клиентски процеси.

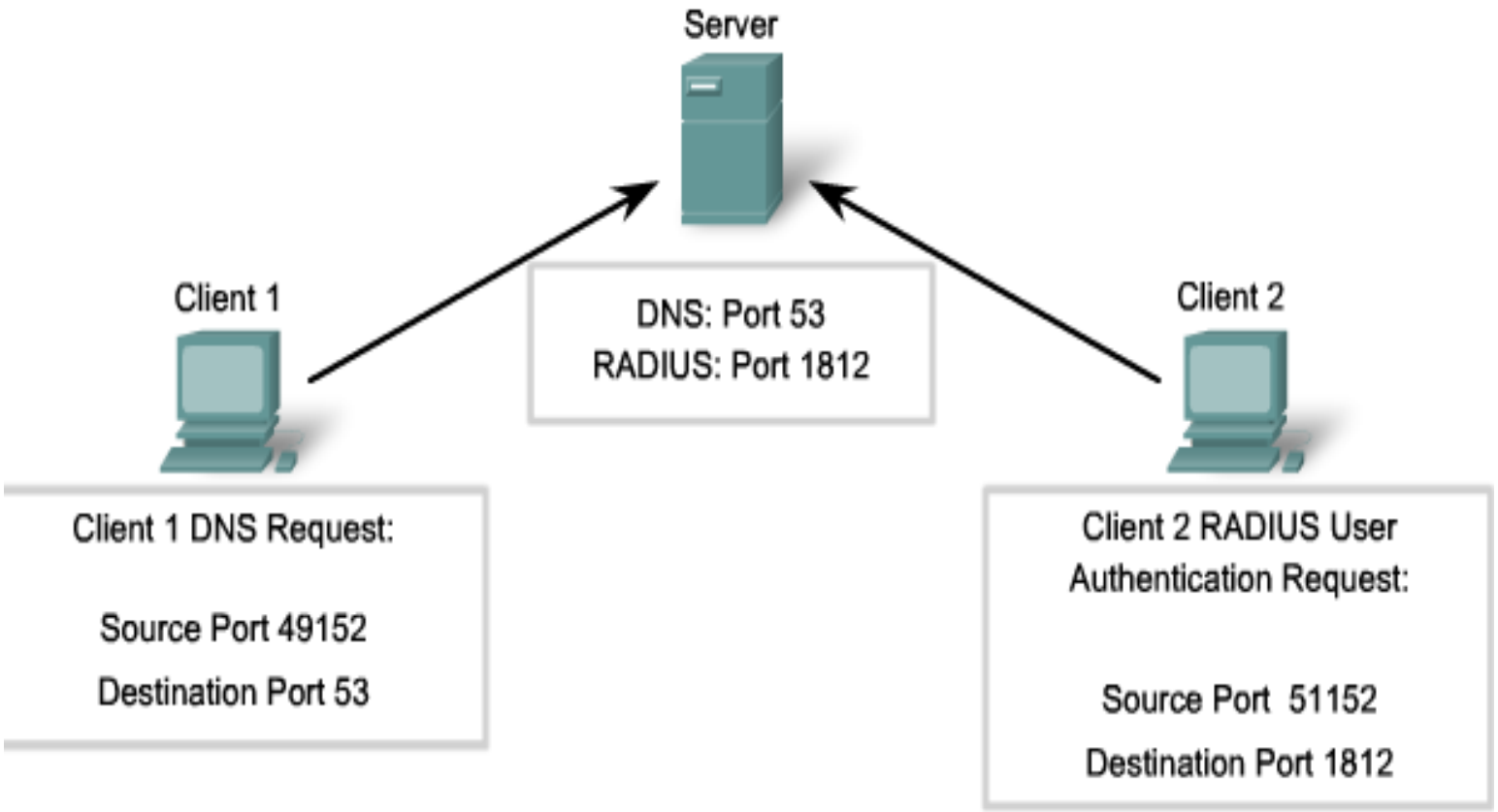
И тук както в TCP клиентското приложение изпраща заявка към сървъра.

Клиентският UDP процес избира на случаен принцип номер на порт от динамичните.

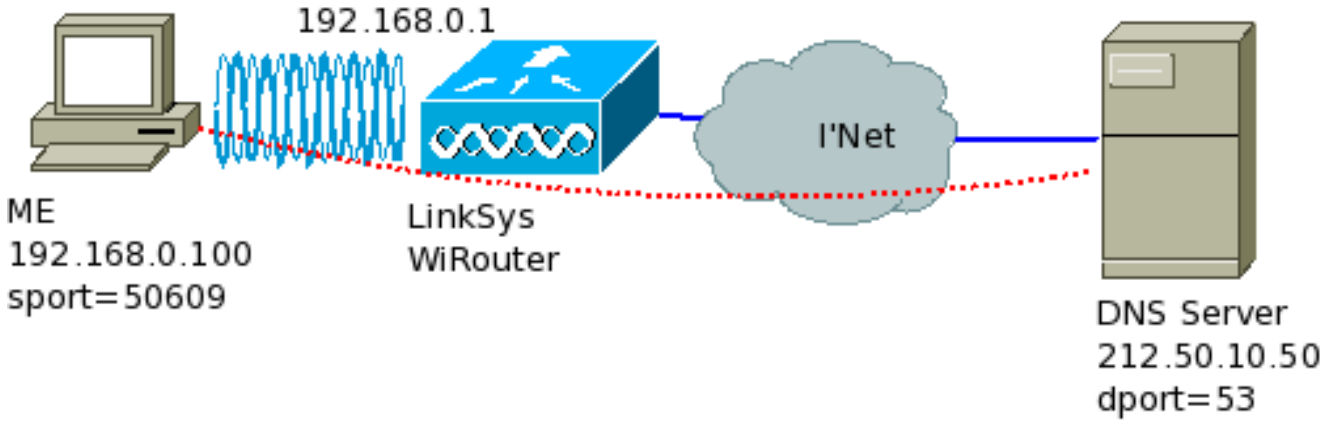
Случайният избор на порт помага за повишаване на сигурността. Номерът няма да е предварително известен на злосторника.

В UDP не се създават сесии. След като данните са готови и портовете са идентифицирани, UDP формира дейтаграма и я подава към мрежовия слой за изпращане по мрежата.

UDP. Клиентски процеси.



UDP съединение по DNS




```
ipv4     2 udp      17 93 src=192.168.0.100
dst=212.50.10.50 sport=50609 dport=53
packets=2 bytes=150 src=212.50.10.50
dst=192.168.0.100 sport=53 dport=50609
packets=2 bytes=210 [ASSURED] mark=0
secmark=0 use=2
```

UDP Jumbograms (IPv6)

16-бит поле Length в UDP header ограничава дължината на UDP пакета (UDP header + данни) <= 65535 октета.

За да се справим с това ограничение (RFC 2675):

UDP пакети > 65 535 октета се изпращат като UDP Length = 0, получателят извлича фактическата дължина на UDP пакета от IPv6 payload length.

UDP-Lite

Lightweight User Datagram Protocol (**RFC 3828**), или UDPLite, е подобен на UDP, но може да обслужва приложения в силно ненадеждна мрежова среда, където целта е данните да бъдат доставени, даже и повредени.

Ако тази характеристика не се използва, UDP-Lite семантично е идентичен с UDP.

UDP-Lite приложения

Приложения, които печелят от това, че данните им се доставят, даже и повредени:

- кодери за глас и видео (напр., AMR [RFC-3267], Internet Low Bit Rate Codec [ILBRC], и H.263+ [ITU-H.263], H.264 [ITU-H.264; H.264] както и MPEG-4).

Тези кодери се справят по-добре с грешки в полето за данни отколкото със загуби на цели пакети.

Защита на канално ниво

Да има силна проверка на интегритета на данните (напр., CRC-32).

Радио технологии (като 3GPP) поддържат такова поведение на линиите.

Ако е ясно коя е чувствителната част от пакета, възможно е физическата линия да осигури по-добра защита именно на тези чувствителни байтове (напр., Forward Error Correction).

Ролята на IP и транспортния слой

IP не е проблем за тези приложения, защото в IP хедъра **няма checksum**, който да покрива полето за данни.

В IPv4 [RFC-791] UDP checksum покрива или целия пакет, или нищо.

В IPv6 [RFC-2460] UDP checksum е задължителна, защото IPv6 хедъра няма header checksum.

Какво защитава UDP-Lite

За описаните по-горе приложения транспортният протокол трябва да защитава важната информация като хедъри и да игнорира грешки, с които се справя приложната програма.

Приложната програма, която изпраща данните, определя кои байтове трябва да се верифицират със checksum.

Частичен checksum

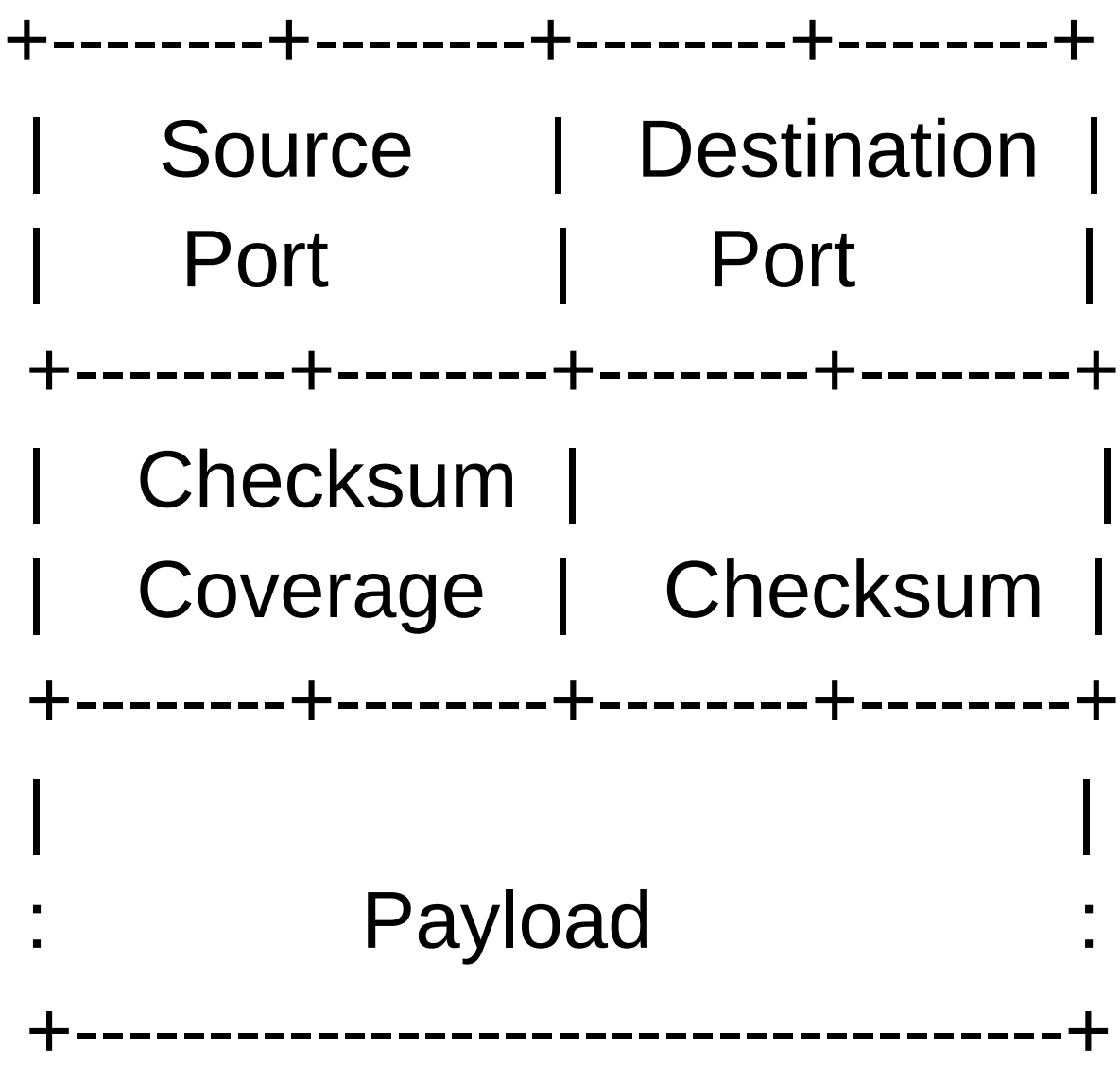
UDP-Lite прилага като опция checksum с частично покритие.

Пакетът се разделя на чувствителна част (която се checksum-ва) и нечувствителна част.

Грешките в нечувствителната част няма да предизвикат отхвърляне на пакета.

Ако checksum покрива целия пакет, UDP-Lite е семантично идентичен с UDP.

UDP-Lite header



Checksum Coverage

Това поле показва броя на байтовете от началото на UDP-Lite хедъра, които се checksum-ват. (Хедърът задължително се checksum-ва).

Checksum Coverage = 0 означава, че целият UDP-Lite пакет се checksum-ва. (Checksum Coverage) = 0 или >= 8. В противен случай (1 - 7) пакетът се изхвърля.