20. Транспортни протоколи TCP и UDP.

**TCP и UDP** - Двата основни протокола на транспортния слой в TCP/IP модела са Transmission Control Protocol (TCP) и User Datagram Protocol (UDP). И двата управляват комуникациите между приложения, работещи на компютри в Мрежата и са от типа “край до край” (end to end). Разликите между двта са във функциите, които реализират. UDP е по-опростен, осигурява ненадеждно обслужване с неустановена връзка (connectionless), дефиниран в RFC 768. Предимство е ниското закъснение. Протоколните единици в UDP се наричат дейтаграми, за които подобно на IP пакетите се полагат максимални усилия за доставяне - "best effort". Типични приложения на UDP са: Domain Name System (DNS), Video Streaming, Voice (Video) over IP (VoIP), мониторинг и управление на мрежите (SNMP), опростен пренос на двоични файлове (Trivial FTP). TCP е протокол, който предоставя надеждно oбслужване с установена връзка (connection-oriented), дефиниран в RFC 793. TCP внася допълнително закъснение заради функциите по надеждност, спазване на реда на подаване на единиците с данни (сегменти) и управление на потока. Всеки TCP сегмент има 20 байта служебна информация, с които опакова приложните данни, докато при UDP дейтаграмите те са само 8 байта. Типични приложения на TCP са: Web браузъри и сървъри, E-mail, сигурен обмен на файлове (FTP).

 **Адресиране с портове.Идентифициране на “разговорите”-** TCP и UDP базираните услуги следят комуникациите между различни приложения в Мрежата. За да разпознаят сегментите и дейтаграмите на всяко приложение, и TCP сегментите, и UDP дейтаграмите имат полета в заглавната част, които уникално идентифицират тези приложения - номерата на портовете. По-конкретно, порта-източник и порта-местоназначение: source port и destination port. source port е номера на комуникацията, свързана с приложението – инициатор на “разговора” (сесия). destination port е номера на комуникацията, свързана с приложението – дестинация, работещо върху отдалечения хост.

**destination port** - Номерата на портове се присвояват по различни начини, в зависимост от това дали съобщението е заявка или отговор. Процесите на сървър са със статични номера, а клиентите динамично избират номер на порт при всяка сесия. Клиент изпраща заявка до сървър: destination port е номер на порт, присвоен на процеса (демона) на услугата, стратирана на отдалечения хост. Клиентският софтуер трябва да знае номера на този порт. Той се конфигурира по подразбиране или ръчно.Например, web браузър прави заявка към web сървър. Използва TCP и порт 80, ако не е дефинирано нещо друго.TCP port 80 е по подразбиране присвоен на web сървърите.Повечето известни приложения имат номера на портове по подразбиране.

**source port** - source port в заглавието на сегмента/дейтаграмата, съдържащ заявката на клиента, е произволно генериран от номера на портове, по-големи от 1023. source port играе ролята на обратен адрес за приложението, заявяващо услугата. Комбинацията от номера на порт на транспортния слой и IP адреса на мрежовия (IP:port No) идентифицира конкретен процес, работещ на даден хост. Тази комбинация се нарича сокет (socket). Двойката (IP:port No) на източник и местоназначение идентифицира конкретна сесия между два хоста. Например, HTTP заявка за web страница, изпратена към web сървър (port 80) с IPv4 адрес 192.168.1.20 е насочена към сокет 192.168.1.20:80. Ако web браузъра е на хост с IP: 192.168.100.48 и динамично му е присвоен порт 49152, сокет, където трябва да бъде “свалена” web страницата ще е 192.168.100.48:49152.

**Видове портове** - Различните видове номера: *Добре известни* - Well Known Ports (0 - 1023). Резервирани за популярни услуги и приложения – HTTP, POP3/SMTP, Telnet и др; *Регистрирани* - Registered Ports (1024 - 49151). Тези номера се присвояват на потребиъелски процеси и приложения.Ако не се използват в момента, могат да се използват динамично от клиентските процеси.

**Динамични или частни портове** (49152 - 65535). Известни и като Ephemeral (краткотрайни) портове. Обикновено се присвояват динамично на клиентски приложения, иницииращи сесия. Услугите не използват такива портове (с изключение на peer-to-peer file sharing програми).Номера, използвани и в TCP, и в UDP. За постигане на по-висока производителност някои приложения в даден момент се опират на TCP, в друг - на UDP. Например, ниското закъснение, внасяно от UDP, позволява на DNS да обслужва бързо много клиентски заявки. Но понякога изпращането на заявената информация може да изисква надеждността на TCP. Т.е DNS използва известния порт 53 и с двата протокола.

**Сегментиране и възстановяване** - TCP и UDP се справят със сегментирането по различен начин. В TCP в заглавната част на всеки сегмент се съдържа пореден номер (sequence number). Благодарение на него в Транспортния слой на дестинацията се възстановява реда, по който сегментите са били предадени. Услугите в UDP също следят сесиите между приложенията, но не и реда, по който се предава информацията, и поддържането на връзката. В UDP заглавието липсва последователен номер (sequence number). UDP е с по-опростена структура и внася по-малко излишна за потребителя служебна информация (overhead) от TCP. Затова осигурява по-бърз пренос на данните. Но приложенията, които се базират на UDP, трябва да се съобразяват с факта, че е възможно данните в различен ред от този, по който са били предадени.

**Transmission Control Protocol-Transmission Control Protocol (TCP**) осигурява надеждно, подредено доставяне на потока от байтове от програма на един компютър до програма на друг компютър. TCP следи за размера на съобщенията,скоростта на обмен и натоварването на мрежовия трафик.

**Приложимост на TCP** - TCP се ползва от най-популярните Internet приложения - WWW, E-mail, FTP, SSH, Telnet и някои групови медийни приложения (streaming media). TCP е оптимизиран за доставяне на съобщения без грешка в съдържанието, но не и за гарантирано време на доставяне. TCP понякога внася големи закъснения (в порядъка на секунди) заради подреждането на пристигналите пакети и повторни предавания. Затова не е подходящ за приложения в реално време като Voice over IP (VoIP) и Video over IP. За тях се препоръчва Real-time Transport Protocol (RTP), който работи върху User Datagram Protocol (UDP).

**TCP и IP** - Между “подател” и “получател” в Интернет се разменят съобщения. Докато IP се грижи за действителното доставяне на данните, TCP следи отделните единици от данни - сегменти, на които е разбито съобщението с цел ефективна маршрутизация. Например, HTML файл се праща от Web сървър. TCP софтуерът на този сървър разделя последователността от байтове във файла на сегменти, които препраща поотделно към IP софтуера – мрежовия слой. Мрежовият слой опакова всеки TCP сегмент в IP пакет. Всички IP пакети, които произхождат от горния HTML файл са с един и същ адрес на получател, но те могат да минат по различни пътища през Мрежата. Клиентската програма на компютъра-получател, по-точно TCP софтуера (транспортния слой), възстановява оригиналното подреждане на сегментите, като гарантира, че са получени без грешка, и ги препраща към приложната програма.

**Структура на TCP сегмента** - TCP сегментът се състои от две части: заглавие (header) и данни (data). Заглавната част на TCP сегмента се състои от 11 полета, от които 10 са задължителни. 11-то е "options". Source port (16 бита) – номер на изпращащ порт Destination port (16 бита) – номер на получаващ порт Sequence number (32 бита) – двуяка роля: ако флаг SYN е вдигнат, това е първоначалният номер. Последователният номер на първия байт ще бъде именно този sequence number + 1; ако флаг SYN не е вдигнат, това е sequence number на първия байт с данни. Полето Acknowledgement number е номерът на първия байт данни, който се очаква да се получи със следващия сегмент, изпратен от другия край на TCP връзката. Например, при успешно получаване на сегмент с размер на полето данни 500 байта и пореден номер на началния му байт n, към източника на този сегмент се изпраща TCP сегмент, в който потвърждението е с номер n+501. Полето TCP header length е 4-битово и определя дължината на заглавната част на TCP сегмента в 32-битови думи. То е задължително, тъй като полето за опции е с променлива дължина. Фактически с това поле се определя началото на полето Data в рамките на TCP сегмента. Заглавната част на TCP сегмента съдържа и 6 еднобитови флага: URG – валиден е указателят за спешни данни (Urgent pointer).Установяването на този флаг означава, че трябва да се преустанови обработката на получените данни, докато не се обработят байтовете, към които сочи указателят за спешни данни; ACK – валиден е номерът на потвърждение, записан в полето Acknowledgement number на заглавната част; PSH – при активирането на този флаг, програмните модули управляващи транспортния слой на източника и на приемника трябва да изпратят незабавно наличните данни колкото е възможно по-бързо към техния получател, т.е. източникът не изчаква да се съберат данните за образуване на пълен сегмент с избрания размер и съответно получателят не чака запълването на приемния буфер; RST – сегмент, в който е установен този флаг, служи за прекратяване на TCP връзката. Използва се в случаите, когато връзката е нарушена (например, поради повреда в хоста) или когато се отхвърля невалиден сегмент или се отказва опит за установяване на връзка; SYN – сегмент с установен флаг SYN се използва при установяване на TCP връзка и за изпращане на началния номер, от който ще бъдат номерирани байтовете на изходящия информационен поток; FIN – сегмент, в който е установен този флаг, означава, че изпращачът прекратява предаването на данни. Поради двупосочния характер на информационния обмен това не означава, че TCP връзката е прекратена. Полето Window size определя темпа на информационния обмен от гледна точка на получателя на информационния поток. Стойността на прозореца указва на отсрещната страна колко байта могат да бъдат изпратени и съответно приети без препълване на входия буфер след последния потвърден номер на байт. При получаване на данни, размерът на прозореца намалява. Ако той стане равен на 0, изпращачът трябва да престане да предава данни. След като данните се обработят, получателят увеличава размера на своя прозорец, което означава, че е готов да получава нови данни. Полето Urgent pointer се използва да укаже позицията на първия байт на спешните данни спрямо началото на полето данни. Полето Checksum се изчислява върху целия TCP сегмент. При неговото изчисляване участват и някои полета от заглавната част на IP дейтаграмата, в която е опакован сегмента. Полето Options на заглавната част на TCP сегмента е предназначено да предостави допълнителни възможности за управление на обмена, които не се осигуряват от останалите полета на заглавието. Най-важната възможност е указване на максимална дължина на сегмента. Всеки хост указва своята максимална дължина на сегмента и за осъществяване на обмена се приема по-малката от двете. Ако максималната дължина на сегмента не се договори се приема по подразбиране, че нейната стойност е 556 байта, което е допустимо за всички интернет хостове.

**Установяване и терминиране на TCP съединение** - Когато два хоста комуникират по TCP, първо трябва да се оформи съединение, преди да започне обмена на данни. След приключване на комуникацията сесиите се затварят - терминират. Всичките тези процедури реализират функциите по надеждност в TCP. За установяване на връзка хостовете изпълняват трипосочна процедура – “ръкостискане” (three-way handshake). Контролните битове в TCP заглавието регистрират прогреса и състоянието на връзката. Three-way handshake прави: Установява, че отсрещното устройство съществува в мрежата; Уверява се, че отсрещното устройство има активна услуга и приема заявки на отдалечения порт, който инициатора смята да използва за сесията; Информира отсрещното устройство, че клиент източник има намерение да установи сесия с този номер на порт.

**3 стъпки на TCP съединение** - При TCP съединенията хостът-клиент инициира сесията към сървъра. За да си изясним как работи “three-way handshake”, нека видим какви стойности на параметри си разменят двете старани. Трите стъпки в установяване на TCP съединение са: 1. Клиентът-инициатор изпраща сегмент SYN, съдържащ начална стойност на последователността SEQ, и представляващ заявка за начало на сесия. 2. В отговор сървърът изпраща сегмент, съдържащ стойност за потвърждение ACK (SEQ + 1). Освен това - собствената си синххронизираща стойност на последователност SYN SEQ, която е по-голяма от получения и потвърден номер ACK – следващия очакван байт. 3. Клиентът-инициатор отговаря със стойност ACK, равна на (получена SEQ + 1). С това съединението е установено.

**TCP. Потвърждение и прозорци.** - Стойностите SEQ и ACK в заглавието на сегмента съвместно служат за потвърждение на получените байтове с данни. SEQ е относителния брой байтове, които са предадени в дадената сесия + 1 (номера на първия байт с данни в дадения сегмент). TCP използва числото ACK в сегментите, които се изпращат обратно на източника, за да покаже кой е следващия байт,който приемника очаква да получи в настоящата сесия. Това се нарича очакваното потвърждение. Източникът на сесията е информиран, че дестинацията е получила всички байтове в този поток от данни с изключение на байта под номер, равен на номера, съдържащ се в потвърждението.Очаква се хоста-инициатор да изпрати сегмент със SEQ = (ACK).Всъщност всяко съединение представлява две еднопосочни сесии. Стойностите SEQ и ACK се разменят и в двете посоки.

**TCP. Повторно предаване -** Колкото и добре да е проектирана дадена мрежа, не може без загуби на данни. TCP осигурява методи за управление на тези загуби на сегменти. Един от тях е механизмът за повторно предаване на сегменти с непотвърдени данни. TCP услугата в хоста-приемник потвърждава само данни, състоящи се от непрекъсната последователност от данни. Ако липсват един или повече сегменти, потвърждават се само данните в сегментите, които попълват плътно потока. Например, получени са сегменти със SEQ = 1500 - 3000 и 3400 – 3500. Тогава ACK = 3001. Това е така, защото има сегменти със SEQ = 3001 – 3399, които не са получени. TCP в хоста-източник не е получил потвърждение след предварително зададено време. Тогава той ще се върне към последния ACK, който е получен, и ще предаде повторно данните от тази точка нататък. Процесът на повторно предаване не е дефиниран в RFC, зависи от конкретната реализация на TCP.В типичната TCP реализация хост предава сегмент, поставя копие от него в опашка за препредаване и стартира таймер. След получаване на потвърждение сегментът се изтрива от опашката.Ако след нулиране на таймера не се получи потвърждение,сегментът се предава отново.Днес хостовете могат да поддържат и опцията “Селективно потвърждение”. В такъв случай дестинацията би потвърдила байтове в непоследователни сегменти, без да е необходимо препредаване на липсващи данни.

**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 услуги в сесията. Когато източникът е информиран, че оределено количество данни в сегментите е получено, тогава може да продължи с изпращане на повече данни за дадената сесия.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. Редуциране на загубите** - Друг начин за контролиране на потока от данни е да определяме 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. 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.

**UDP - UDP** е по-опростен транспортен протокол с неустановена връзка. Това не означава, че приложенията, които се базират на UDP, са непременно ненадеждни. Функциите по надеждност се осъществяват другаде.Основни приложни протоколи, които “стъпват” на UDP са: Domain Name System (DNS); Simple Network Management Protocol (SNMP); Dynamic Host Configuration Protocol (DHCP); Routing Information Protocol (RIP); Trivial File Transfer Protocol (TFTP); онлайн игри. Онлайн игрите или VoIP могат да понесат някои загуби на данни. Но не и закъсненията, които внася TCP. Други приложения като DNS или TFTP ще повторят заявката, ако не получат отговор. И не им трябват гаранциите на TCP.

**UDP. Възстановяване на Дейтаграми** - UDP е connectionless и сесии не се установяват като 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 един сегмент - дейтаграма.Но някои изпраща по-големи количества, които се разделят на множество сегменти.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 тя е от значение, за това се грижи приложната програма.

**UDP. Клиентски процеси** - И тук като в TCP клиентското приложение изпраща заявка към сървъра. Клиентският UDP процес избира на случаен принцип номер на порт от свободните. 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 и я подава към мрежовия слой за изпращане по мрежата.