5.Канално ниво - кадри, предаване, грешки, номерация, прозорци.

**Основни функции**

Каналното ниво има **три основни функции**:

- да осигури подходящ интерфейс на по-горното мрежово ниво,

- да открива грешки по време на предаването и

- да управлява информационния обмен.

Данните за каналното ниво представляват последователност от **кадри** (frame).

Каналите са три вида - **симплексни**, **полудуплексни** и **дуплексни**. Дуплексните канали позволяват едновременно предаване в двете посоки. Полудуплексните канали позволяват предаване и в двете посоки, но в даден момент може да се предава само в една посока. Симплексните канали позволяват предаване само в една посока.

Най-общата услуга - прехвърляне на данни (надеждно или best effort, но изчистено от грешки) между мрежовото ниво на източника и мрежовото ниво на получателя (всъщност самото предаване се извършва от физическото ниво, но това остава невидимо за мрежовото ниво).

Основните варианти на тази услуга са:

- непотвърдено неуставено обслужване (Unacknowledged connectionless service),

- потвърдено неустановено обслужване (Acknowledged connectionless service) и

- потвърдено установено обслужване (Acknowledged connection-oriented service).

**Непотвърденото неустановено** обслужване източникът изпраща независими кадри към получателя, без получателя да ги потвърждава. Няма установяване на връзка между двете машини. Ако един кадър се загуби поради шум в линията, каналното ниво не прави опит да възстанови този кадър. Това обслужване е подходящо при канали с много малка честота на грешките, което позволява функциите по възстановяване

на загубената информация да се поемат от по-горни нива в йерархията.

**Такова обслужване се реализира в повечето LAN**. То също се използва когато навременното получаване на кадрите е по-важно от тяхната достоверност видео, глас в реално време. При **потвърденото и неустановено обслужване** отново не се установява връзка между източника и получателя, но получаването на всеки кадър се потвърждава самостоятелно от получателя. Това дава възможност за повторно изпращане на непотвърдените кадри.

Потвърждаването на получената информация е функция на транспортното ниво, но там то се

отнася до последователности от сегменти. **Потвърждаването на каналното ниво има смисъл при ненадеждна комуникационна среда**, каквато е безжичната, тъй като повторно ще се предават само непотвърдените кадри.

**Основни услуги. Connection Oriented**

Потвърденото и установено обслужване има три фази. Първата фаза се установява връзка и се заделят необходимите ресурси (локални буфери, броячи и т.н.). Втората фаза се изпращат кадрите. Третата фаза се освобождават ангажираните ресурси. Гарантира се не само успешното предаване на кадъра, но и последователността в която се предават кадрите.

**Управление на потока (Flow Control)**

Друг проблем, който е свързан с управлението на обмена на канално ниво е източникът да изпраща кадри по-бързо, отколкото те могат да бъдат приети от получателя. За целта се въвеждат механизми за управление на потока от кадри, който осигурява обратна информация на източника за темпа на предаване. Обикновено механизмите по управление на обмена се изпълняват в транспортния слой над цели масиви от данни, обхващащи последователност от кадри.

**Формиране на кадри**

Каналното ниво взима пакетите, които му се подават от мрежовото ниво и ги опакова в кадри. Всеки кадър се състои от заглавна част (header), поле за данни (data или payload), което съдържа мрежовия пакет и опашка (trailer). Дължината на кадъра обикновено е ограничена отгоре. Физическото ниво възприема информацията от каналното ниво като поток от битове, без да се интересува от нейната структура.

Получателят идентифицира в потока от битове кадрите и въз основа на служебната информация в тях ги контролира за грешки. За целта опашката на кадъра съдържа контролна сума (обикновено 2 байта), която се изчислява върху останалата част от кадъра преди той да бъде предаден. Когато кадърът пристигне при получателя, контролната сума се преизчислява и ако тя е различна от предадената контролна сума, то получателят отхвърля кадъра и евентуално изпраща съобщение за грешка към източника.

Разделянето на потока от битове на кадри не е тривиална задача. Един начин е **между всеки два кадъра да се въведе времеви интервал**. Този подход е твърде несигурен, тъй като времевите интервали могат да се променят по време на предаването. Понастоящем основно се използват три метода. При първия метод е **броене на отделните символи**. В заглавието на кадъра се указва броя на символите в целия кадър. Основният проблем на този метод е, че броят на символите може да бъде сгрешен по време на предаването, при което получателят ще загуби синхронизация и няма да може да определи началото на следващия кадър. Дори при неправилна контролна сума, получателят не знае началото на следващия кадър, въпреки че разбира, че текущият кадър е сгрешен. Повторно предаване не помага. Затова не се използва.

**Втори метод** в началото и края на кадъра се вмъкват специални служебни символи - **STX** (start of text) за начало на кадър и **ЕТX** (end of text) за край на кадър, които маркират границите на кадъра. Техниката е известна като **вмъкване на символи** (**byte stuffing**, **character stuffing**). Възможно е служебните символи да се срещат като битови последователности в оригиналните данни. За решение на този проблем се въвежда друг служебен символ **ESC** (escape), който се вмъква преди всяко срещане на служебен символ (STX, ETX, ESC) в данните. Например, ако потокът, предаван от мрежовия слой на източника е A STX ESC B, той ще се преобразува в A ESC STX ESC ESC B.

Каналното ниво на получателя ще премахне символите ESC (като при два последователни ESC, единият се запазва), преди да предаде данните на мрежовото ниво на получателя. При по-новите протоколи се използва един и същ символ за маркиране на началото и края на кадъра - флаг. Недостатъкът на този метод е, че той се обвързва с 8-битови символи, кодирани в ASCII.

С развитието на мрежите стана възможно кадрите да съдържат произволно цяло число битове. За такива кадри се използва третия метод, при който началото и края на всеки кадър се маркира с битовата последователност 01111110, наречена **флагов байт**. За да се предотврати погрешното определяне на граница на кадър, ако тази последователност от битове се срещне в данните на кадъра, след всеки 5 единици в данните източникът добавя по една нула. Техниката се нарича **вмъкване на битове** (**bit stuffing**). Каналното ниво на получателя премахва нулата след всеки 5 единици в данните, преди да ги подаде на мрежовото ниво. За постигане на допълнителна сигурност при много протоколи броенето на символи се комбинира с някой от другите два метода.

**Прооцедури за надеждна работа на канала**

Функциите на каналния слой се реализират в адаптер, предимно хардуерно: специализирани интегрални схеми за управление (ASIC) и програмен код, “прогорен” в EEPROM или записан във Flash памети (firmware). В адаптера е реализиран буфер, в който се записват кадрите, докато изчакват да бъдат предадени нататък. Кадърът преседява в буфера, докато не не се увери, че отсрещната страна го е получила. Да приемем, че **източник *A*** изпраща кадър към ***B***, но той изобщо не стига до там. Пет причини за това:

1) Адаптер *А* дефектен, не излъчва правилен сигнал;

2) “Счупен” канал – жица, роми и т.н.;

3) *B* не съществува;

4) *B* няма свободен буфер;

5) Кадърът постъпва в буфера на *B*.

***A*** може да получи отговор единствено при 5). При изпращане на кадъра ***A*** включва брояч на време- таймер. Чака отговор до определено време –timeout. timeout трябва да е по-голямо от времето за предаване на кадъра, обработката му в приемника и получаване на потвърждение. Ако кадърът не се потвърди в рамките на това време, то ***A*** предава кадърът отново. Възможно е ***A*** да изпрати кадър към ***B***, този кадър да се получи в ***B***, но потвърждението да се изгуби.

При всички положения ***A*** изпраща наново кадъра, ***B*** ще получи същия кадър и ще го изпрати към мрежовото ниво, което ще доведе до недопустимо дублиране на данните. За целта с всеки кадър се свързва пореден номер. В случая е достатъчно номерът да е един бит (0 или 1). Във всеки един момент *B* очаква кадър с определен номер. Ако *B* получи кадър с друг номер, този кадър е дубликат и се отхвърля. Ако *B* получи кадър с очаквания номер, кадърът се приема и очакваният номер на кадър се инвертира (ако е бил 0 става 1, ако е бил 1 става 0). От своя страна *A* номерира алтернативно кадрите, които изпраща към *B*. Естествено, ако даден кадър бъде изпратен отново неговият номер не се променя.

**Откриване на грешки в кадрите**

За целта е измислен **CRC** (*cyclic redundancy check* —*проверка на цикличния остатък*) Алгоритъм за проверка за грешки при предаване и съхранение на данни чрез използване на контролна сума (*контролно число*,  *CRC сума*). Устройството-източник изчислява CRC-сумата на данните, които следва да бъдат проверявани и я изпраща или записва със самите данни. Устройството-получател извършва същото изчисление след прочитане на данните и контролната сума, и установява тяхната автентичност чрез сравнение на записаната CRC сума и новоизчислената CRC сума.

**Видове CRC-та**

CRC-16-CCITT = *x*16 + *x*12 + *x*5 + 1 (Bluetooth, XMODEM, HDLC, PPP)

CRC-32-IEEE 802.3 = *x*32 + *x*26 + *x*23 + *x*22 + *x*16 + *x*12 + *x*11 + *x*10 + *x*8 + *x*7 + *x*5 + *x*4 + *x*2 + *x* + 1

CRC-64-ISO = *x*64 + *x*4 + *x*3 + *x* + 1 (HDLC — ISO 3309)

Тези кодове са разработени отдавна, оптимизирани и вкарани в хардуерни схеми. CRC се изчислява в движение и се слага в края на кадъра като FCS (Frame Control Sum)

**Протоколи с прозорци** (Sliding Window Protocols)

Ще разгледаме два протокола, които спадат към класа на **протоколите с прозорци**. Те са по-ефективни от протокола спри и чакай, тъй като позволяват изпращане на повече от един кадър, преди да се чака за потвърждение. При тези протоколи всеки кадър се номерира с число от 0 до някакъв максимум, обикновено от вида 2*n* - 1, така че номерът да се вмества точно в *n* бита. Протоколът „спри и чакай“ (stop-and-wait sliding window protocol) има n = 1, ограничавайки номера на кадъра до 0 и 1. Във всеки един момент предавателят поддържа множество от поредни номера на кадри, които са готови за изпращане - тези кадри попадат в  **прозореца на предавателя**. От друга страна получателят поддържа **прозорец на получателя**, в който се буферират получените кадри. Не е задължително двата прозореца да имат един и същ размер.

Предавател и приемник:-

Последователен номер nt и nr; (nt следващият пакет за предаване. nr предстоящият за приемане пакет);

- Размер на прозорец: wt и wr (**> 0**).

Приемникът може да следи и още неполучения най-голям номер **ns** :

= nr, ако wr=1

> nr, ако wr>1.

(Всички пакети преди nr са получени, нито един след ns не е получен. По средата, между nr и ns, са получени някои пакети. Когато приемник получи пакет, обновява променливите си и предава потвърждение (acknowledgment) с новото nr. Предавателят следи най-големия номер на потвърден кадър **na**. Всички кадри с пореден номер **< na** са получени. Не е сигурен за **na ≤ nr ≤ ns**. Последователните номера трябва да следват правилото: **na ≤ nr ≤ ns ≤ nt ≤ na + wt**

Sliding Window Protocols. **go back n**

При наличие на сгрешен или изгубен кадър, предавателят ще продължи да предава кадри, преди да разбере че има проблем. Какво да прави получателят с успешно получените кадри след сгрешен или изгубен кадър? Едната стратегия (**go back n**) е тези кадри да се отхвърлят. Тя съответства на прозорец на получателя с размер 1. **wt>1**, но **wr=1**. Приемникът приема единствено и само следващия в последователността. При загуба на кадър при предаване, следващите се изхвърлят (**Discard**), докато се предаде повторно липсващия. Минималната

загуба е един round trip time (1 \* RTT).

**Selective Repeat**

Необходим е по-мощен приемник, който да приема кадри с последователни номера **> nr** (текущо) и да ги съхранява, докато се попълни празнината от загубените.Предимство е, че не е необходимо да се изхвърлят следващите правилно приети кадри за 1 \* RTT, преди предавателят да е разбрал, че е необходимо повторно предаване.Предпочитан за линии с ниска надеждност.Прозорец **wr** трябва да е по-голям от броя на загубените кадри в поносими граници. Обикновено: **wr=2**