

Канален слой: Контрол на грешките

Канален слой:

Контрол на грешките

Справяне с:

- **Изгубени кадри**
 - Кадри недостигнали до получателя
 - Установени с помощта на поредни номера
 - Възстановени чрез повторно предаване (за трафик чувствителен към загуби) или интерполация/екстраполация (за трафик чувствителен към закъснения)
- **Повредени кадри**
 - Разпознаваме кадър пристига, но някои битове (в неизвестни позиции) са сгрешени.
 - 2 подхода:
 - Бракуване на кадъра, т.е. данните, пренасяни от кадъра, не се изпращат към горния слой (за трафик чувствителен към загуби)
 - Призоваване на кадъра, като се разчита на горните слоеве да се справят с грешките.
 - Укриване/маскиране на грешки (*error concealment*)
 - Заглавната част на кадъра трябва да е без грешки
 - Грешките в данните се коригират в по-горните слоеве с помощта на използване на излишък в тях (*redundancy*) или с помощта на интерполация/екстраполация на другите им стойности
 - За трафик чувствителен към загуби (глас, видео, мултимедия)
 - Намалява изискванията към надеждността и потреблението на енергия, но с цената на по-големи изчислителни усилия

Контрол на грешките: Видове

- **Закъсняващ контрол, използващ обратна връзка** (*Backward Error Control, BEC*)
 - Всеки кадър съдържа само достатъчно излишна информация, позволяваща на получателя да открие дали има грешки, но не и тяхното местоположение.
 - Използва се повторно предаване на сгрешения кадър
 - За двупосочни канали
 - Използва се в каналния, мрежовия и транспортния слой
 - По-малко излишък / по-евтин метод
- **Изпреварващ контрол, без обратна връзка** (*Forward Error Control, FEC*)
 - Всеки кадър съдържа повече излишна информация, така че получателят може да открие кои точно от получените битове са сгрешени.
 - Корекция на грешките се извършва чрез инвертиране на битовете, за които е установено, че са сгрешени.
 - За еднопосочни канали (напр. носители на информация като CD, DVD) или канали с голямо време за разпространение на сигнала (сателитни, космически)
 - Използва се във физическия слой при предаване по шумени (напр. безжични) канали или в по-горните слоеве за мултимедия трафик, предаван в реално време.
 - Повече излишък / по-скъп метод

Закъсняващ контрол, използващ обратна връзка (BEC)

- Базиран на алгоритми за контрол на грешките и схеми за контрол на повторното предаване
 - Известни с общото наименование автоматична заявка за повторно предаване (*Automatic Repeat reQuest, ARQ*)
 - Целта на ARQ е да превърне ненадеждния канал в надежден
- Използва кодиране в канала (*channel coding*), за постигане на надеждно откриване на грешки.
 - Примерни кодове: проверка по четност (*parity check*), цикличен код (*CRC*), контролна сума (*checksum*).

ARQ: Схеми за потвърждение

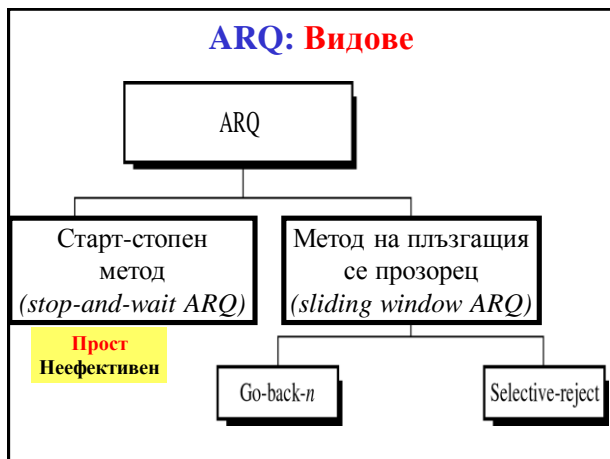


ARQ: PAR

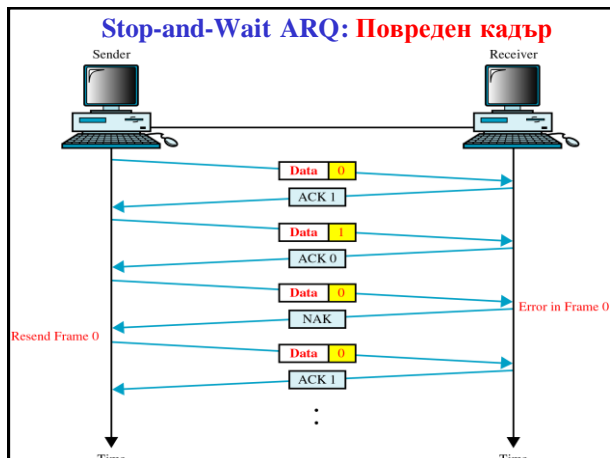
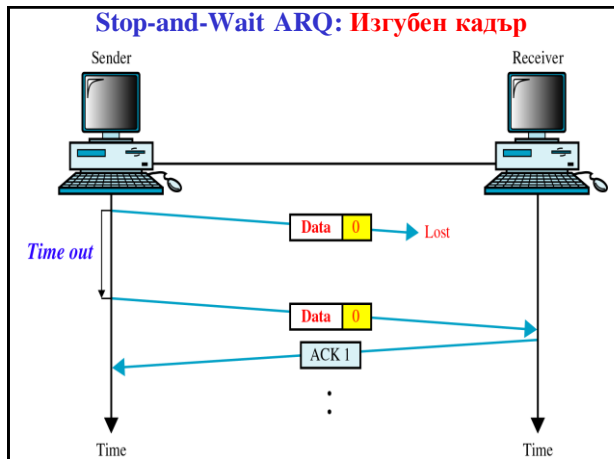
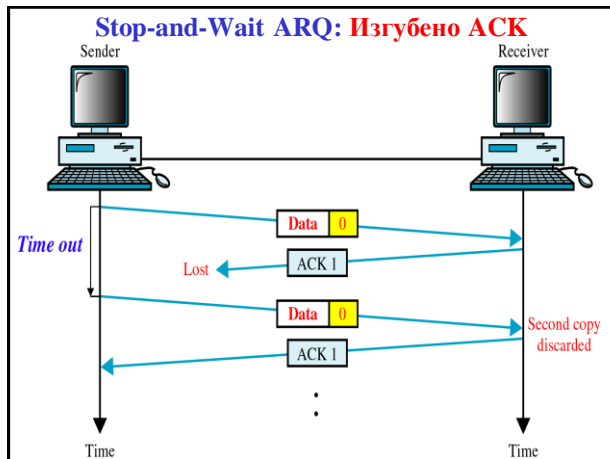
- Получателят връща положителна квитанция/потвърждение (*ACK*) за успешно получените (без грешки) кадри
- Подателят предава повторно кадрите, които не са били потвърдени в рамките на определен период от време (*timeout*)
- Квитанцията има същия номер като:
 - Следващия кадър, очакван от получателя
 - По-често използван метод
 - или последния успешно получен кадър

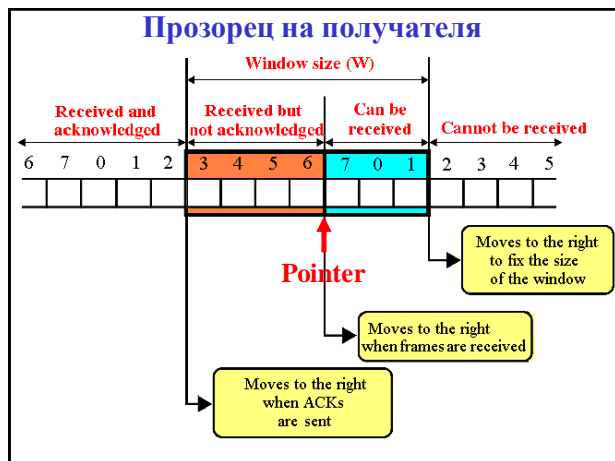
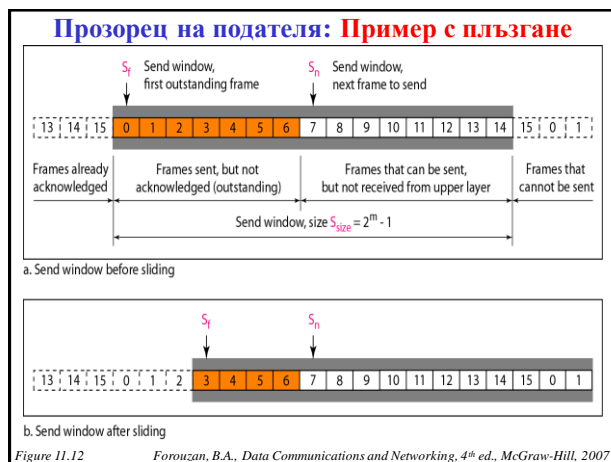
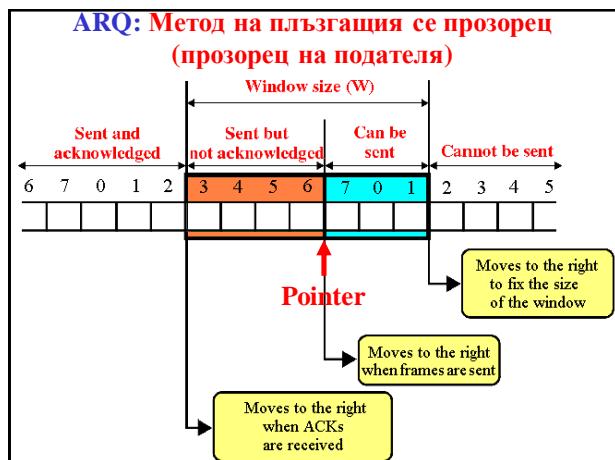
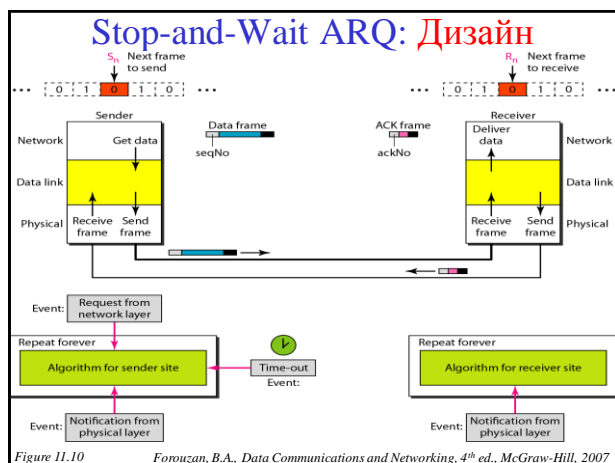
ARQ: NAK

- Получателят връща отрицателна квитанция (*NAK*) за кадър, в който е била открита грешка.
- Подателят предава повторно кадъра
- Видове реализации**
 - Подателят предава повторно всички кадри, започвайки от този, за който е била получена отрицателна квитанция.
 - Връщане N позиции назад (*Go-Back-N*)**
 - Подателят предава повторно само кадъра, за който е била получена отрицателна квитанция.
 - Избирателно отхвърляне (*Selective-Reject*)**
- Различни подходи**
 - Selective-Reject*** игнорира кадрите получени не по ред и изпраща *NAK* само за повредените кадъра
 - Go-Back-N*** изпраща *NAK* и за кадрите получени не по ред

ARQ: Видове**ARQ: Stop-and-Wait**

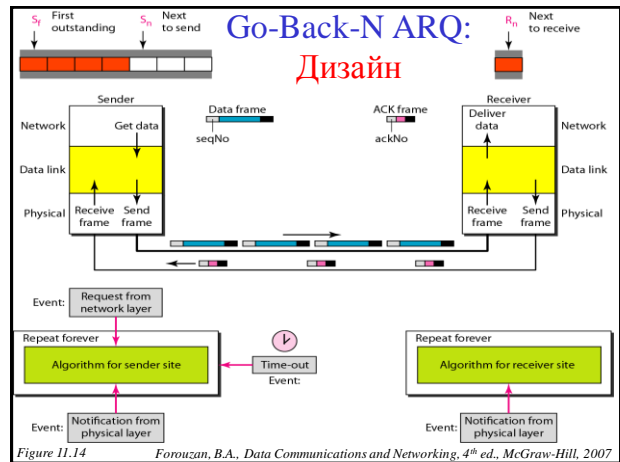
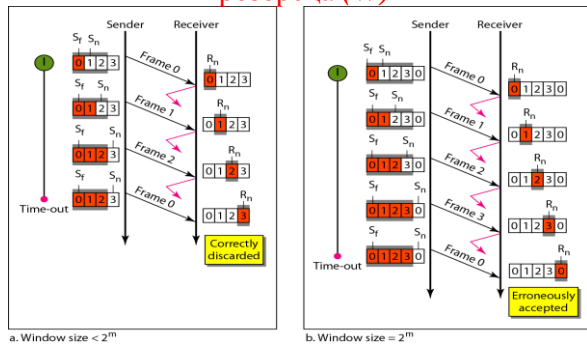
- Подател**
 - Предава един кадър и стартира таймер
 - Спира и изчаква потвърждение преди изпращането на следващия кадър
 - Ако получи *NAK* преди изтичане на времето (или не получи *ACK* или *NAK* до изтичане на времето), предава повторно кадъра.
 - Повторно предаваните копия имат същия номер като оригиналния кадър, което позволява на получателя да открива дубликатите (т.е. копията на един и същ кадър).
 - Ако получи *ACK* преди изтичане на времето, предава следващия кадър, номериран по *mod 2* (т.е. употребяват се редуващи номера: 0, 1, 0, 1, ...)
- Получател**
 - Получава кадър и го проверява за грешки
 - Ако няма грешки, отговора с *ACK*.
 - Ако има грешки, отговора с *NAK* (или мълчание).

Stop-and-Wait ARQ: Повреден кадър**Stop-and-Wait ARQ: Изгубен кадър****Stop-and-Wait ARQ: Изгубено ACK**



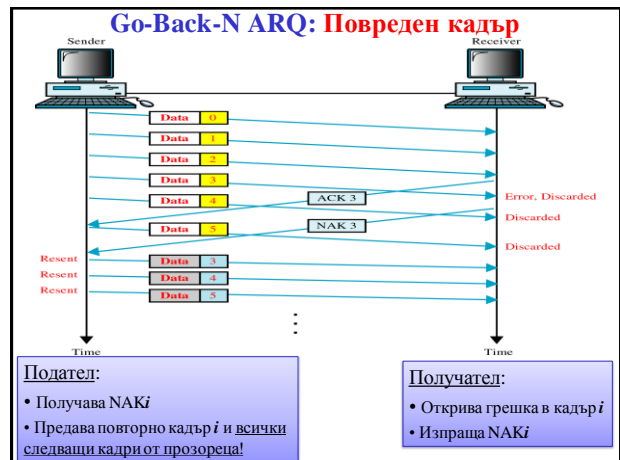
ARQ: Go-Back-N

- Базиран на метода на плъзгащия се прозорец
 - Прозорец (с размер W), използван за контролиране броя на кадрите, които могат да се изпратят по канала без да се чака потвърждение.
- Подател
 - Може да изпрати няколко кадъра (един след друг)
 - Използва буфер с размер, побиращ до $W = 2^m - 1$ кадъра.
 - Стартира отделен таймер за всеки изпратен кадър
- Получател
 - Приема кадрите **само поред!**
 - Т.е. отхвърля кадрите, които са получени правилно, но не поред.
 - Използва буфер само за 1 кадър
 - Ако пристигат кадър с очаквания пореден номер, го проверява за грешки:
 - Ако няма грешки, изпраща потвърждение ACK.
 - Ако има грешки, отговаря с NAK (или не отговаря изобщо).
 - Отхвърля съответния кадър и всички следващи кадри, докато не получи пак този кадър без грешка.
 - Подателът трябва да се върне обратно и да изпрати отново този кадър, и **всички следващи кадри от прозореца**, независимо дали вече са били изпратени или не!
- Ако полученият кадър не е с очаквания поредния номер, го отхвърля (заедно с всички следващи кадри до получаването на този същия кадър) и отговаря с NAK (или с мълчание).

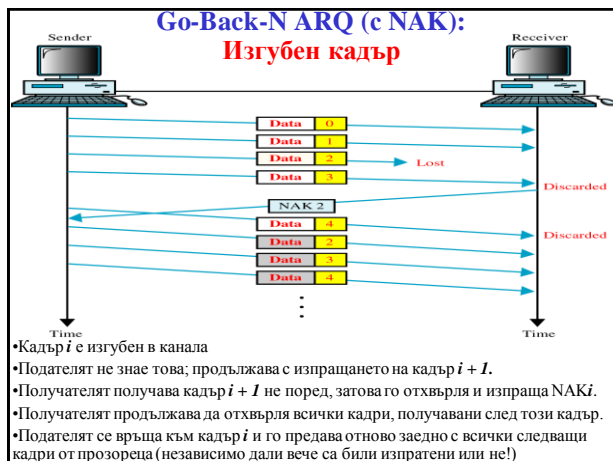
Go-Back-N ARQ: Максимален размер на прозореца (W)

Размерът на прозореца на **подателя** трябва да бъде **по-малък от 2^m** , размерът на прозореца на **получателя** е винаги **1**.

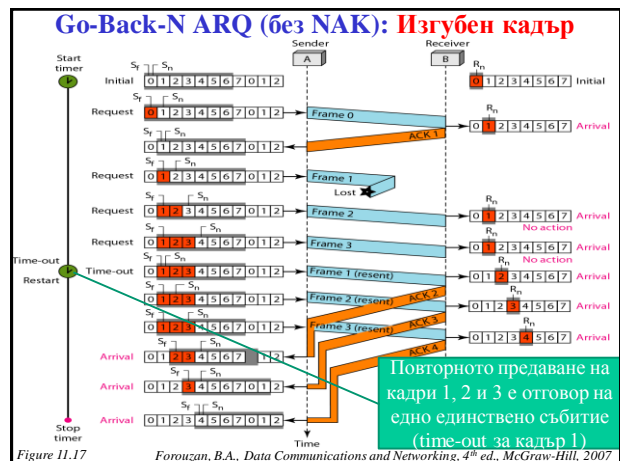
Go-Back-N ARQ: Повреден кадър

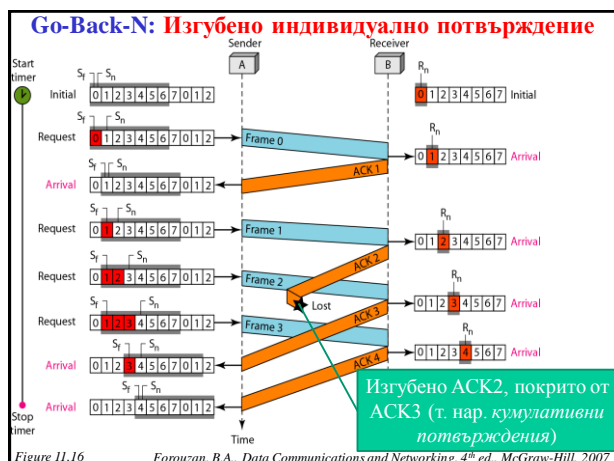
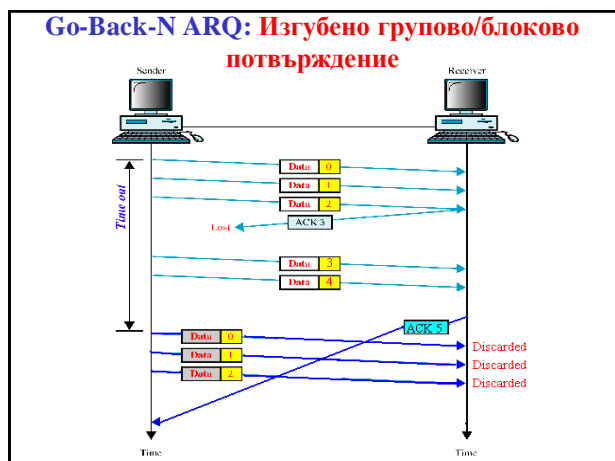


Go-Back-N ARQ (с NAK): Изгубен кадър



Go-Back-N ARQ (без NAK): Изгубен кадър





Go-Back-N ARQ: Предимства и недостатъци

- **Предимства**
 - Малък буфер за получателя
 - Проста логика за подателя
- **Недостатъци**
 - Увеличен брой повторни предавания
 - Вероятно повторно предаване на кадри, получени вече веднъж правилно (но не поред)
 - Излишен разход на енергия
 - Сериозен проблем за мобилни устройства, захранвани с батерии.

Go-Back-N ARQ: Ефективност

$$Eff = \frac{mW}{h + m + a + 2C\tau}$$

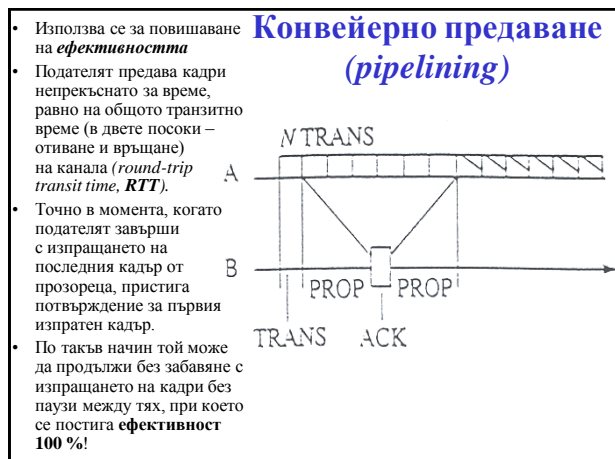
Без грешки
и повторни
предавания

$$Eff = \frac{mP_s}{h + m} [(1/P_s - 1)N + 1]$$

С грешки
(голям прозорец W;
N повторени PDU)

$$Eff = \frac{mWP_s}{h + m + a + 2C\tau} [(1/P_s - 1)W + 1]$$

С грешки
и повторни
предавания
(малък прозорец W)



Piggybacking

- Всеки кадър има 2 номериращи полета:
 - Пореден номер на самия кадър
 - Пореден номер на ACK/NAK в обратната посока
- Ако даден възел има за изпращане данни и ACK/NAK, той ги изпраща заедно в един кадър, а не в 2 отделни кадъра.
 - Спестяват се комуник. ресурси
- Ако даден възел има данни за изпращане, но няма нови ACK/NAK, той повтаря последното изпратено ACK/NAK.
 - Когато другия комуникиращ възел получи дублиращо ACK/NAK, той просто го игнорира.
- Ако възел има ново ACK/NAK за изпращане, но не и нови данни, той изпраща ACK/NAK контролен кадър.



ARQ: Selective-Repeat (Selective-Repeat)

- Подобен на Go-back-N, обаче:
 - Подател
 - Предава отново само изгубените кадри, или приетите с грешки от получателя, или кадрите, на които им се е изгубила квитанцията!
 - Получател
 - Приема кадри **не поред**; стига да са без грешки (буферира ги, докато липсващите кадри не пристигнат).
 - Има буфер за $W = 2^{m-1}$ кадри
 - Избирателно изисква повторно предаване само за сгренените кадри
- **Предимства**
 - Минимизира броя на повторните предавания
 - Не се влияе значително от закъснения, вследствие на разпространението на сигнала.
 - Подходящ за сателитни канали с големи закъснения на сигнала
 - Превъзхожда Go-Back-N
- **Недостатъци**
 - Получателят трябва да поддържа достатъчно голям буфер
 - По-сложна логика за подателя

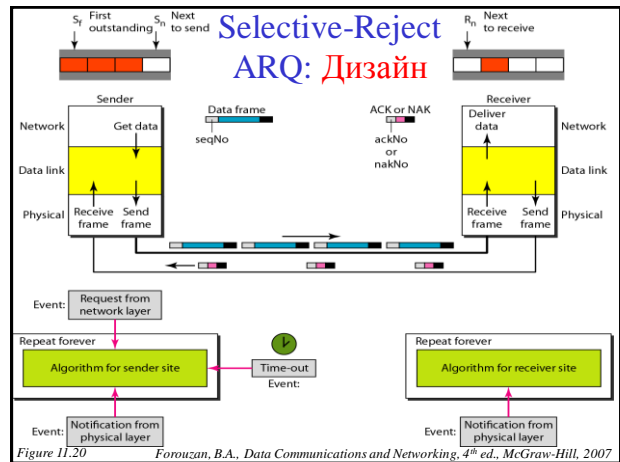
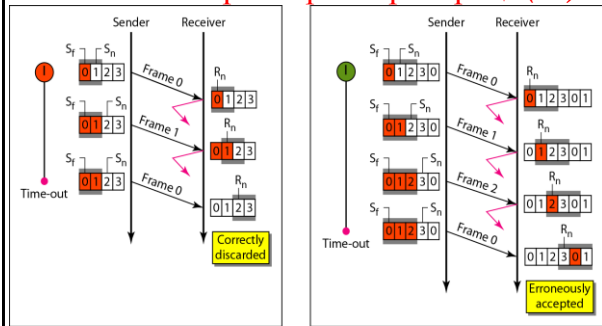
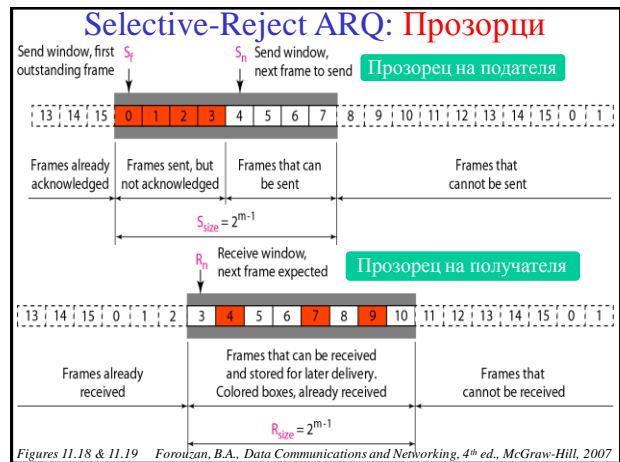


Figure 11.20 Forouzan, B.A., Data Communications and Networking, 4th ed., McGraw-Hill, 2007

Selective-Repeat ARQ:**Максимален размер на прозореца (W)**

Размерът на прозореца на подателя и получателя може да бъде най-много половината от 2^m .



Figures 11.18 & 11.19 Forouzan, B.A., Data Communications and Networking, 4th ed., McGraw-Hill, 2007

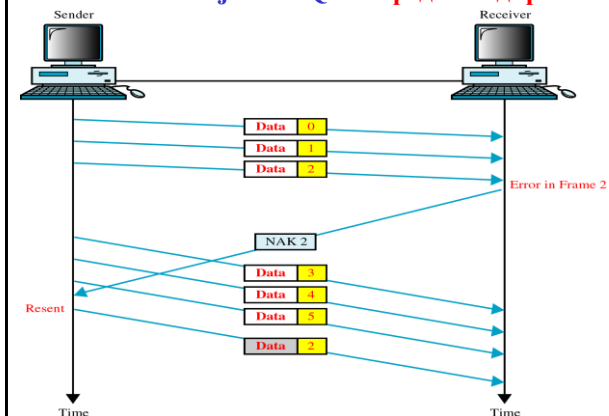
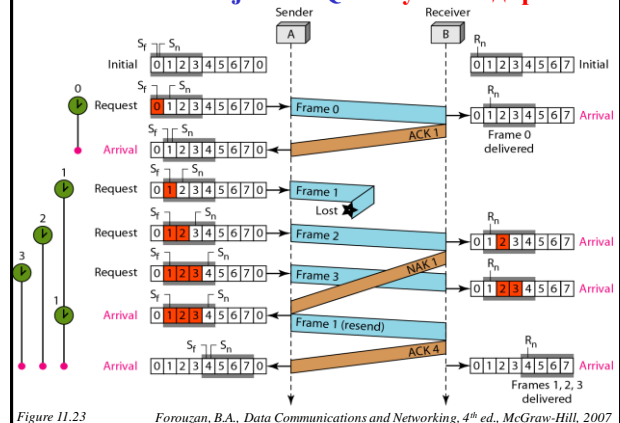
Selective-Repeat ARQ: Повреден кадър**Selective-Repeat ARQ: Изгубен кадър**

Figure 11.23 Forouzan, B.A., Data Communications and Networking, 4th ed., McGraw-Hill, 2007

Selective-Reject ARQ: Ефективност

$$Eff = \frac{mW}{h+m+a+2C\tau}$$

Без грешки
и повторни
предавания

$$Eff = \frac{mP_s}{h+m}$$

С грешки и повторни
предавания
(голям прозорец W)

$$Eff = \frac{mWP_s}{h+m+a+2C\tau}$$

С грешки и повторни
предавания
(малък прозорец W)

Хибридни ARQ (Hybrid ARQ, H-ARQ)

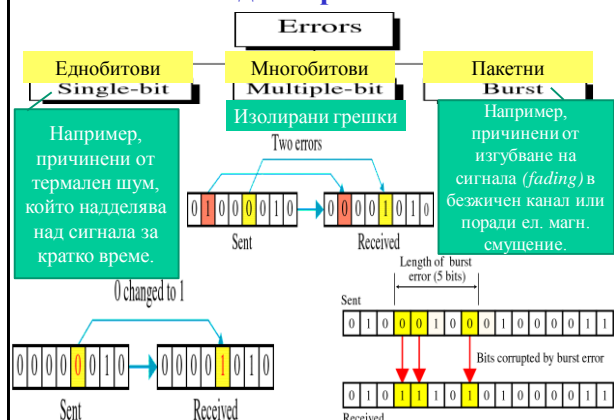
- Съчетават изпреварващ контрол (FEC) с ARQ за постигане на по-ефективно използване на канала и по-висока производителност
- Ако е необходимо повторно предаване, получателят запазва кадъра и по-късно го комбинира с неговите повторно предадени копия за възстановяването му без грешки.
 - Дори ако повторно предадените копия са повредени, тяхната комбинация може да доведе до безгрешен вариант.
- Работи по-добре от оригиналния ARQ в условия, предразполагащи към лошо качество на сигнала, но за сметка на значително по-ниска производителност в условия, позволяващи добро качество на сигнала.
- Използва в стандартите HSDPA и HSUPA, като осигурява високоскоростно предаване на данни в UMTS клетъчни мрежи, както и в стандарта IEEE 802.16-2005 за мобилен широколентов безжичен достъп (WiMAX).

H-ARQ: Схеми за реализация

- **Комбинирано преследване (chase combining, CC)**
 - Подателят предава повторно всеки път едно и също копие на кадъра (кодирано със слаб/олекотен FEC)
 - Вариация: FEC се прилага само за повторно предаваните копия; оригиналният кадър се предава без FEC.
 - Получателят (декодерът) съчетава различните копия на кадъра, претеглени според техния SNR
 - Например, с помощта на *мажоритарен вот (bit по bit)*.
- **Нарастващ излишък (incremental redundancy, IR)**
 - Подателят изпраща копия на кадъра с различен излишък, в съответствие с показателя за качество на канала (feedback channel quality indicator, *CQI*).
 - Представя се по-добре от CC, но с цената на повишена сложност.
 - По-голям буфер в приемника
 - По-сложна схема

Кодиране в канала (шумоустойчиво кодиране)

- Видове грешки и кодове
- Принципи за откриване и коригиране на грешки
- Примерни кодове
 - За откриване на грешки
 - За коригиране на грешки

Видове грешки**Видове кодове**

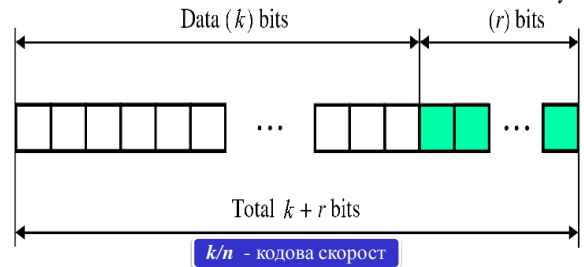
- За **изолирани** грешки
 - Кодове за откриване на грешки (проверка по четност, ...)
 - Кодове за коригиране на грешки (кодове на Хеминг, конволюционни кодове, ...)
- За **пакетни** грешки
 - Кодове за откриване на грешки (циклически кодове, ...)
 - Кодове за коригиране на грешки (код на Рийд-Соломон, ...)
- **Техника за декорелиране на пакетни грешки (interleaving)**
 - Позволява използването на код за изолирани грешки за борба с пакетни грешки

Кодове, използвани по-нататък в лекцията

- Блокови кодове
 - Информационната поредица е разделена на блокове
 - Контролни битове се добавят към всеки блок
- Системни кодове
 - Информационните битове се предават директно, без промяна.
- Линейни кодове
 - Контролните битове са линейна функция на информационните битове, например XOR.

Кодове за откриване/коригиране на грешки:

Кодова дума/комбинация Redundancy



В блоковите кодове, информационните битове са разделени на блокове, състоящи се от k бита. След това към всеки блок се добавят r контролни бита (чрез използване на някаква математическа функция). Получената n -битова комбинация ($n = k + r$) се нарича **кодова дума**.

Блокови (n, k) кодове: Процес на откриване/коригиране на грешки

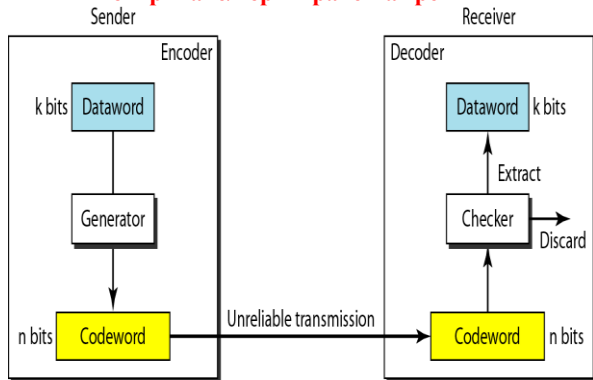
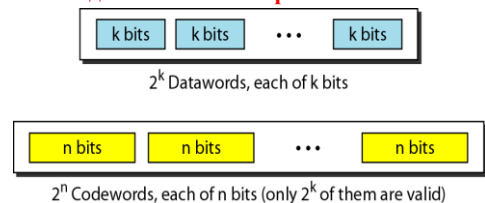


Figure 10.6 Forouzan, B.A., Data Communications and Networking, 4th ed., McGraw-Hill, 2007

Блокови кодове: Разширяване на пространството чрез добавяне на контролни битове



Примерен $(3,2)$ код:

- Валидни код. думи: 000, 011, 101, 110
- Невалидни код. думи: 001, 010, 100, 111

Datawords	Codewords
00	000
01	011
10	101
11	110

Figure 10.5

Forouzan, B.A., Data Communications and Networking, 4th ed., McGraw-Hill, 2007

Разстояние на Хеминг

Разстояние на Хеминг (d)

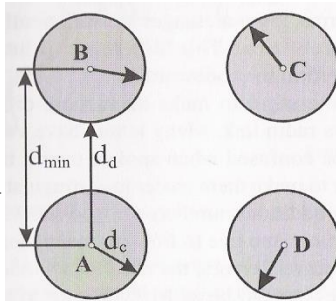
— брой на битовете, по които 2 кодови думи се различават една от друга

- Например, разстоянието на Хеминг $d(000, 011)$ е 2, защото:

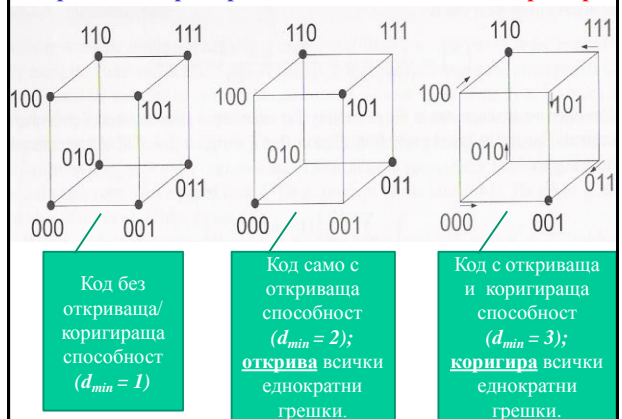
$$000 \oplus 011 \text{ is } 011 \text{ (two 1s)}$$

• **Кодово разстояние (d_{min})** — минималното от всички разстояния на Хеминг за даден код

- Индикатор за откриващата/коригиращата способност на кода



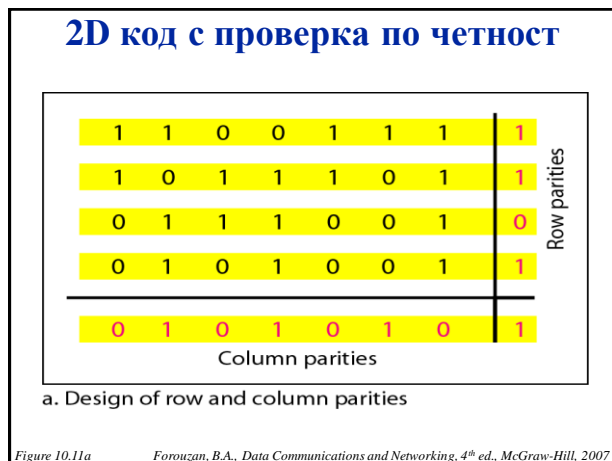
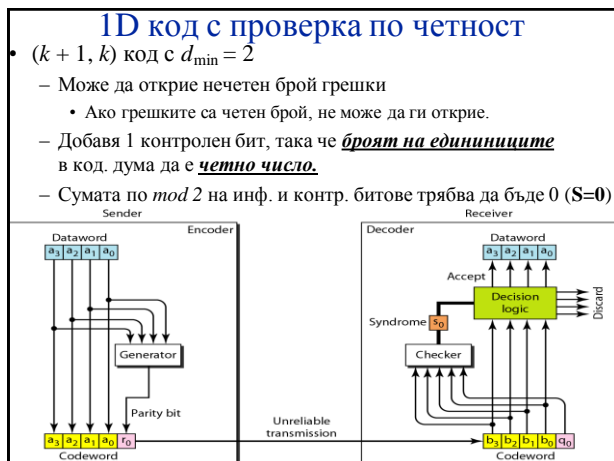
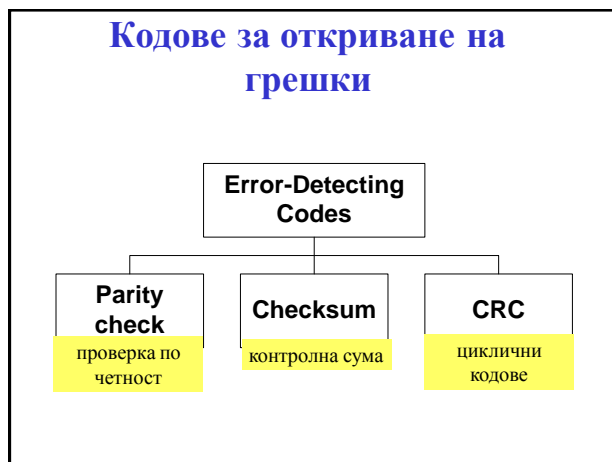
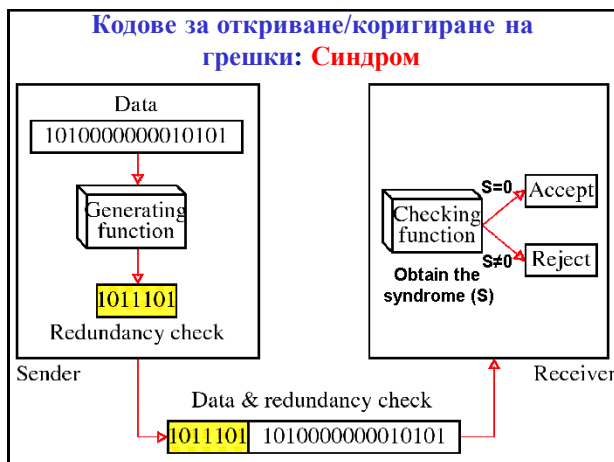
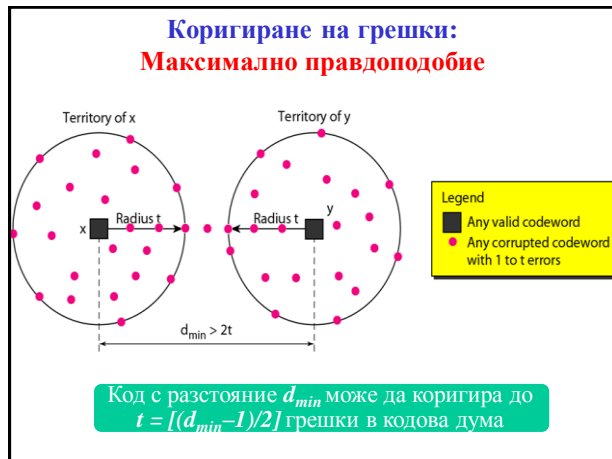
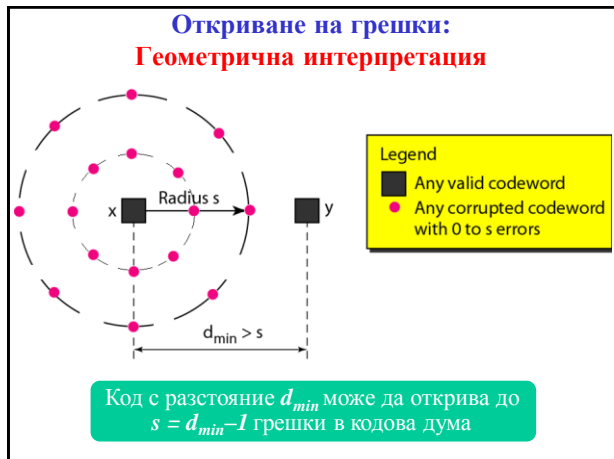
Откриваща/коригираща способност на код: Примери

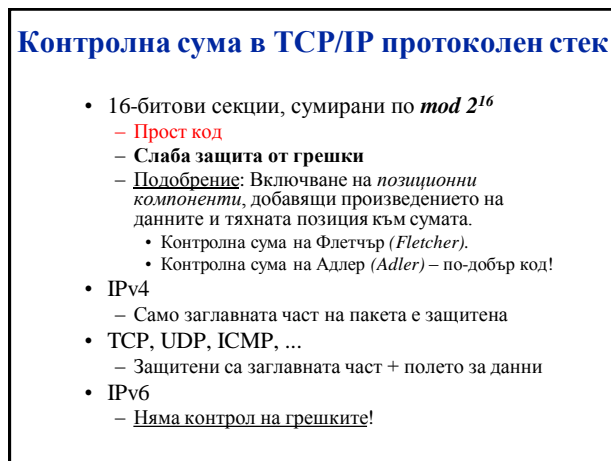
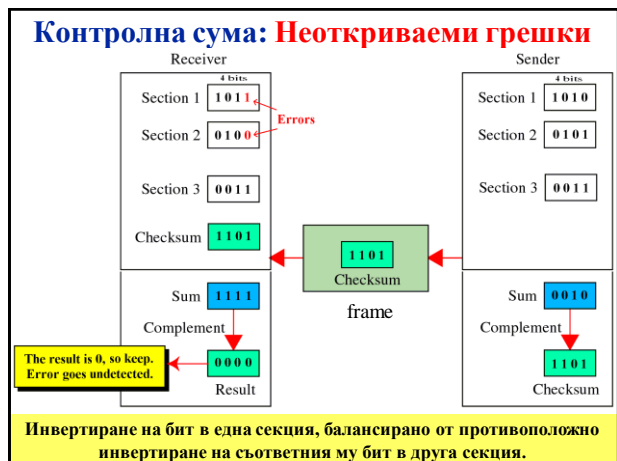
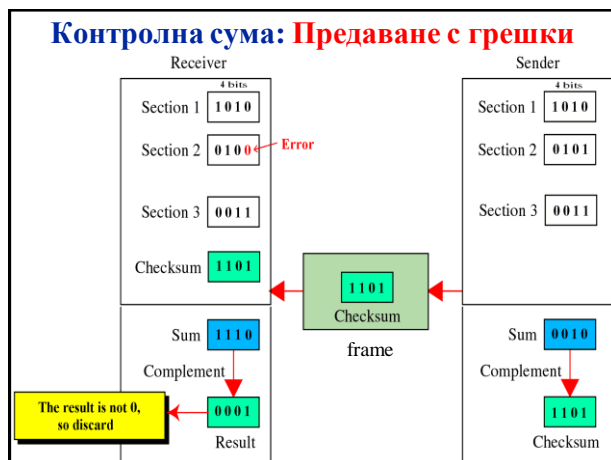
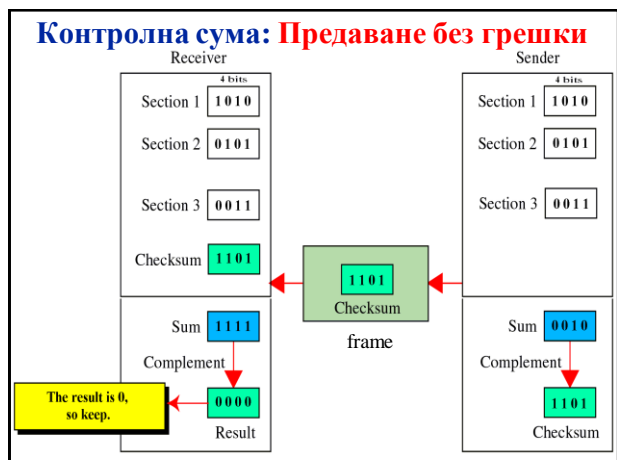
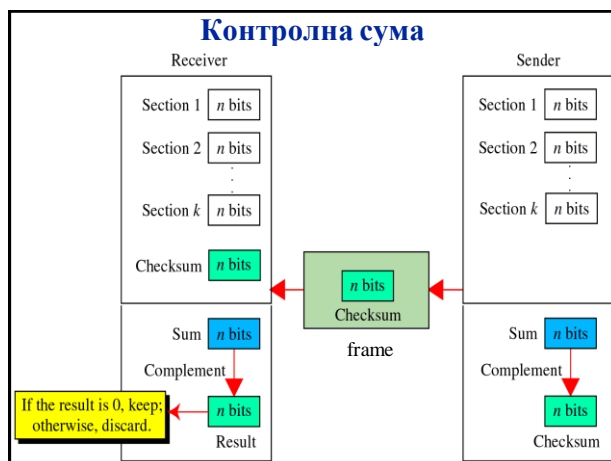


Код без откриваща/коригираща способност ($d_{min} = 1$)

Код само с откриваща способност ($d_{min} = 2$); открива всички еднократни грешки.

Код с откриваща и коригираща способност ($d_{min} = 3$); коригира всички еднократни грешки.



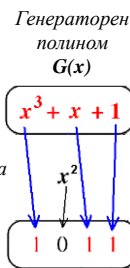


Циклични кодове (Cyclic Redundancy Check, CRC)

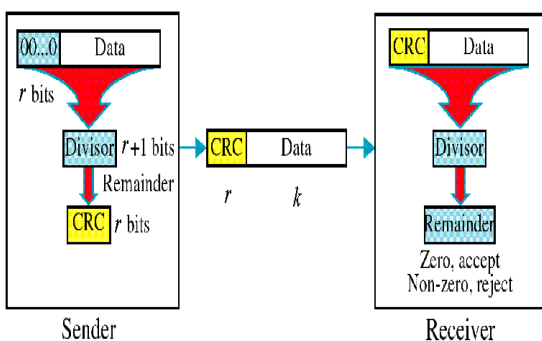
- Много добре дефинирана структура
- Прости за реализация
- Всяко *циклично преместване* на битове в кодова дума води до друга кодова дума на същия код
- Генерират се от **генераторен полином (generator)**
 - Всички валидни кодови думи на даден код се делят без остатък на генераторния полином
- Лесно е да се провери дали дадена кодова дума е получена с грешки
 - Ако остатъкът от това деление е различен от 0
- Кодовите думи се представят също като полиноми

CRC (n, k) код

- Може да се запише и като **CRC-r**
 - n : дължина на код. дума
 - k : брой на инф. битове
 - $r = n - k$: брой на контр. битове
- Подател**
 - Генерира r контролни бита
 - r е степента на генераторния полином $G(x)$
 - За целта добавя r нули след инф. битове (блока с данни, който трябва да се кодира).
 - Разделя резултата на $G(x)$ и определя *остатък*
 - Остатъкът съдържа контрол. битове
 - Добавя *остатък* след инф. битове
 - Т.е. **данни | остатък = кодова дума**
 - Изпраща $n = k + r$ бита
 - Валидна код. дума, която се дели без остатък на $G(x)$
- Получател**
 - Разделя получената код. дума на $G(x)$
 - Ако *остатък* = 0 \Rightarrow код. дума **вероятно** не съдържа грешки
 - Ако *остатък* $\neq 0 \Rightarrow$ код. дума **гарантирано** е с грешки



CRC-r код: Кодиране и декодиране



CRC (7, 4) код: Кодер и декодер

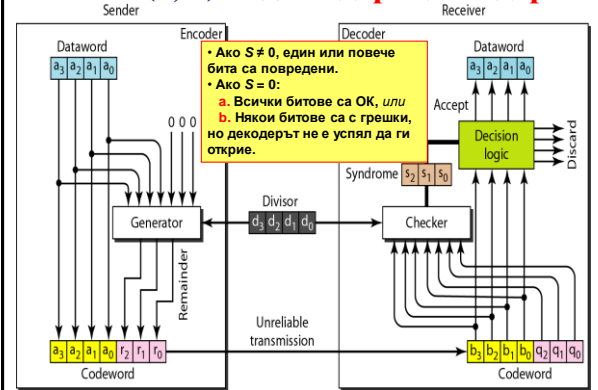
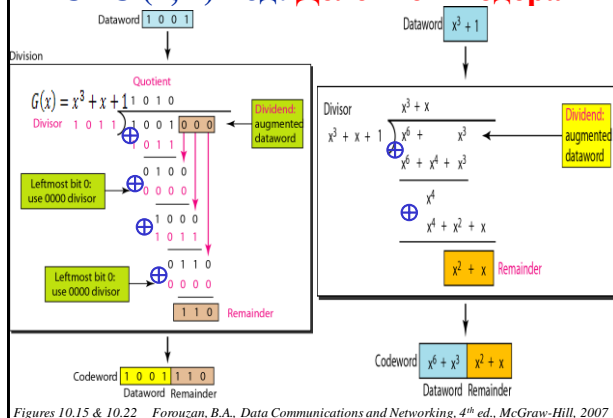


Figure 10.14

Forouzan, B.A., Data Communications and Networking, 4th ed., McGraw-Hill, 2007

CRC (7, 4) код: Деление в кодера



Figures 10.15 & 10.22 Forouzan, B.A., Data Communications and Networking, 4th ed., McGraw-Hill, 2007

CRC (7, 4) код: Деление в декодера

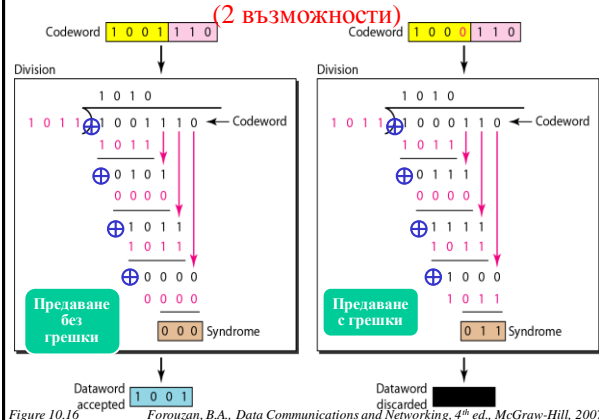


Figure 10.16

Forouzan, B.A., Data Communications and Networking, 4th ed., McGraw-Hill, 2007

CRC- r код: Откриваща способностМоже да открие:

- Всички еднократни грешки
 - Ако генераторният полином има повече от един член и коефициентът пред свободния му член (x^0) е 1
- Всички двукратни грешки
 - Ако генераторният полином не дели $(x^t + 1)$; $0 < t < n-1$
- Всички грешки с нечетна кратност
 - Ако генераторният полином се дели на $(x+1)$
- Всички пакетни грешки с дължина $\leq r$**
- Повечето пакетни грешки с дължина $> r$
 - Вероятността за неоткрита грешка е 2^{-r}

Циклични кодове: Примери от практиката

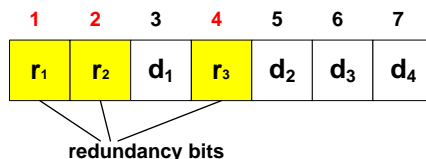
Name	Polynomial	Application
CRC-8	$x^8 + x^2 + x + 1$	ATM header WiMax header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATMAAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32 ($d_{min} = 4$)	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs PPP ATM AAL5

Table 10.7

Forouzan, B.A., Data Communications and Networking, 4th ed., McGraw-Hill, 2007

Кодове на Хеминг

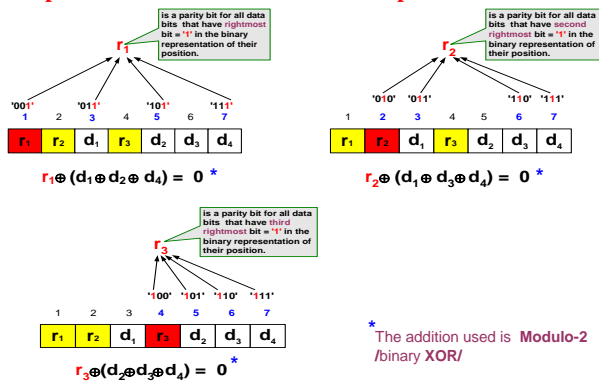
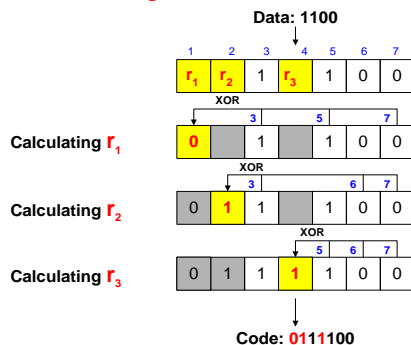
- С кодово разстояние 3
 - Коригират еднократни грешки (напр. в данни, запазени в памет)
 - Откриват двукратни грешки
- Битовите в кодовата дума са номерирани отляво надясно, започвайки с 1
 - Контролните битове** са с номера, представляващи степен на 2 (т.е. 1, 2, 4, ...)
 - Информационните битове** използват останалите номера
- Стойността на контролен бит r_i се определя чрез събиране по mod2 (XOR) на съответните информационни битове, чийто номер (в бинарно представяне) съдържа 1 в позиция i .
- При декодирането всеки контролен бит се използва за проверка (по четност) на съответните (формирани го) информационни битове и на самия него.
- Резултатът от всяка такава проверка формира един (съответен) бит на синдрома S .
 - Ако $S = 0$, приетата кодова дума се счита за валидна.
 - Ако $S \neq 0$, то кодовата дума е невалидна и със сигурност съдържа грешка/и.
- В случай на еднократна грешка (т.е. сгрешен един бит) синдромът S (превърнат в десетичен вид) показва точно номера на сгрешения бит (и декодерът трябва само да го инвертира, за да коригира грешката).

Код на Хеминг (7, 4):**Позиции и номера на контролните битове**

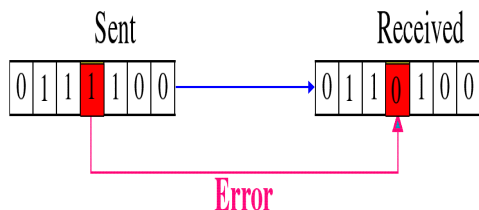
How many redundancy bits we need?

(n,k) code rule: $n+1 \leq 2^r$

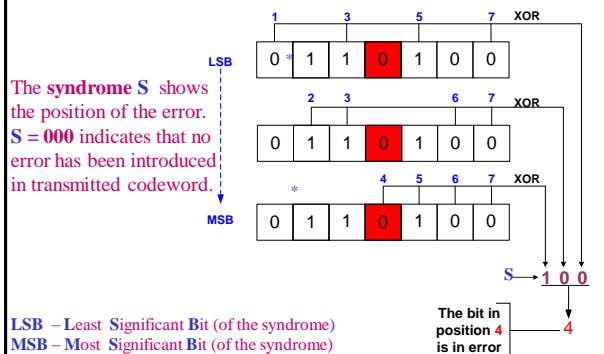
where k = no. of data bits and r = no. of redundancy bits

How to set the redundancy bits amongst the data bits? $2^0 = 1$ - position of r_1 $2^1 = 2$ - position of r_2 $2^2 = 4$ - position of r_3 **Код на Хеминг (7, 4):****Определяне стойността на контролните битове****Код на Хеминг (7, 4):****Пример за определяне стойността на контролните битове**

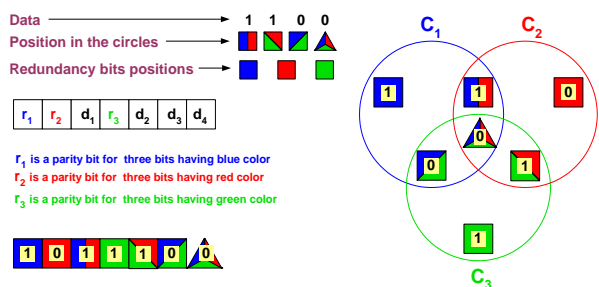
Код на Хеминг (7,4): Еднократна грешка



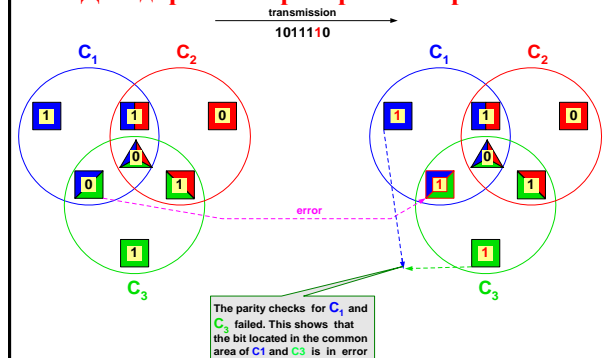
Код на Хеминг (7, 4): Коригиране на еднократна грешка



Код на Хеминг (7, 4): Геометрична интерпретация – Кодиране

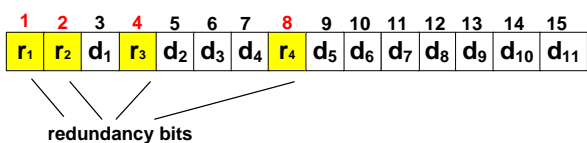


Код на Хеминг (7, 4): Геометрична интерпретация – Декодиране с коригиране на грешката



Код на Хеминг (15, 11):

Позиции и номера на контролните битове



How many redundancy bits we need?

(n,k) code rule: $n+1 \leq 2^r$

where k = no. of data bits and r = no. of redundancy bits

How to set the redundancy bits amongst the data bits?

$2^0 = 1$ - position of r_1

$2^1 = 2$ - position of r_2

$2^2 = 4$ - position of r_3

$2^3 = 8$ - position of r_4