Länkhantering
(feldetektering, felhantering, flödeskontroll)

Maria Kihl

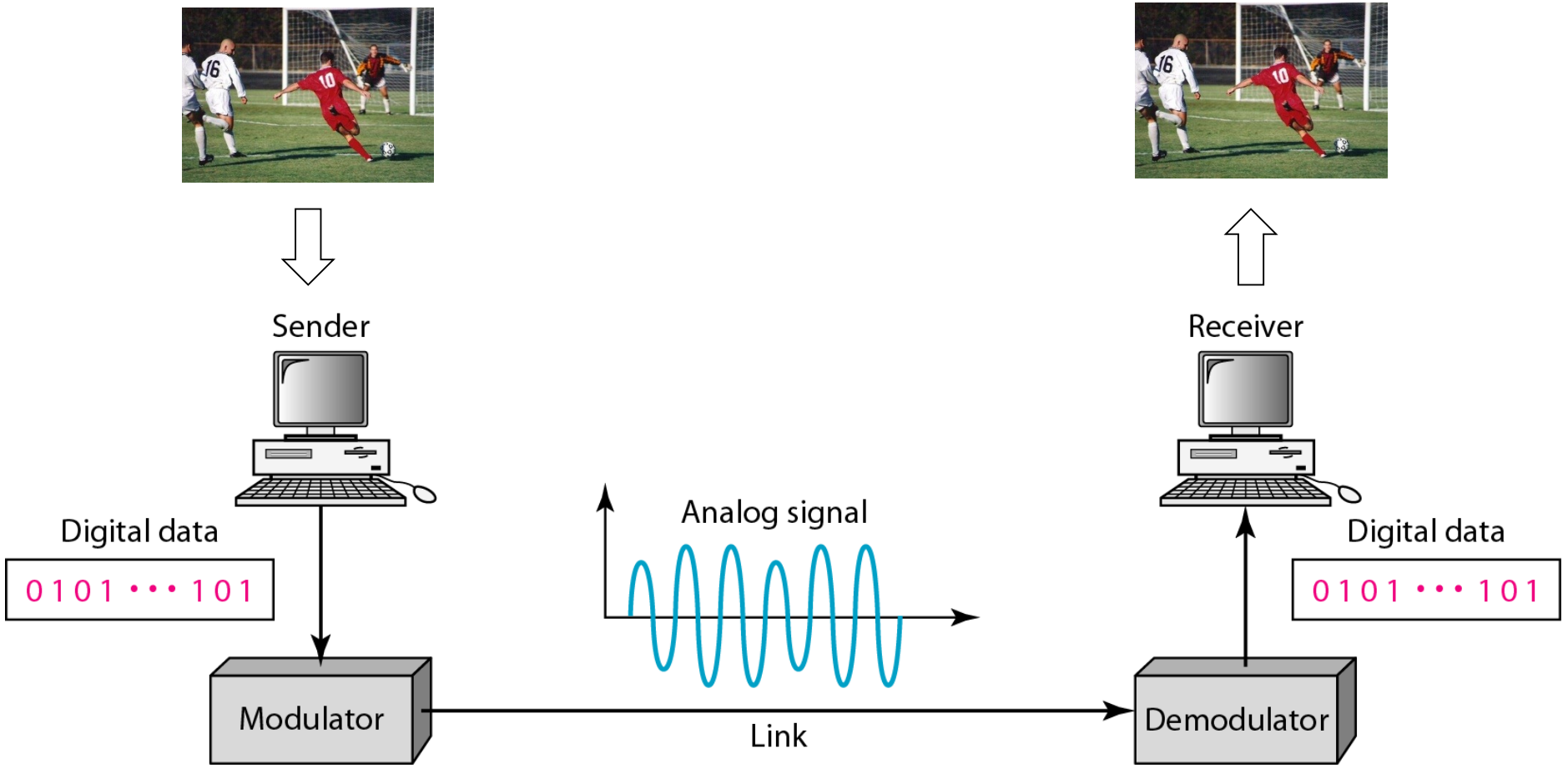
LUND
UNIVERSITY

Läsanvisningar

Kihl & Andersson: 4.1-4.3, 4.5

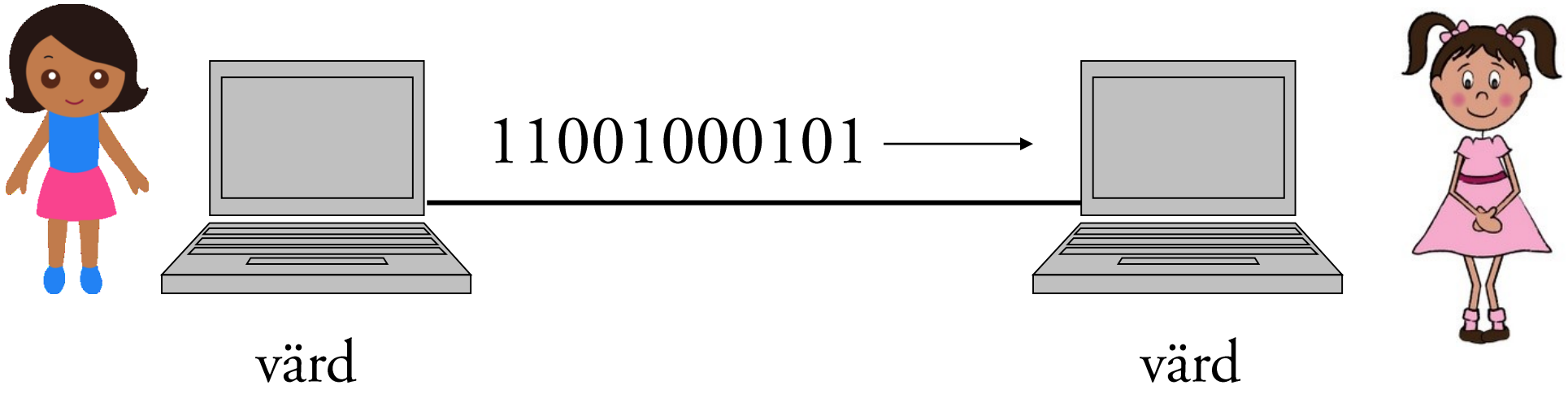
Stallings: 6.1-6.5, 7.1-7.2, (7.3)

Repetition



(eller digital signal)

Att skicka data över en länk



Om en sändare bara skickar en bitström över länken skulle mottagaren ha väldigt svårt för att tolka datan. Det krävs regler för hur datatransmissionen ska gå till.

Datapaket

Grundläggande principen är att all data som ska skickas läggs i datapaket.

I varje datapaket finns en header som innehåller kontrollinformation för det protokoll som används.

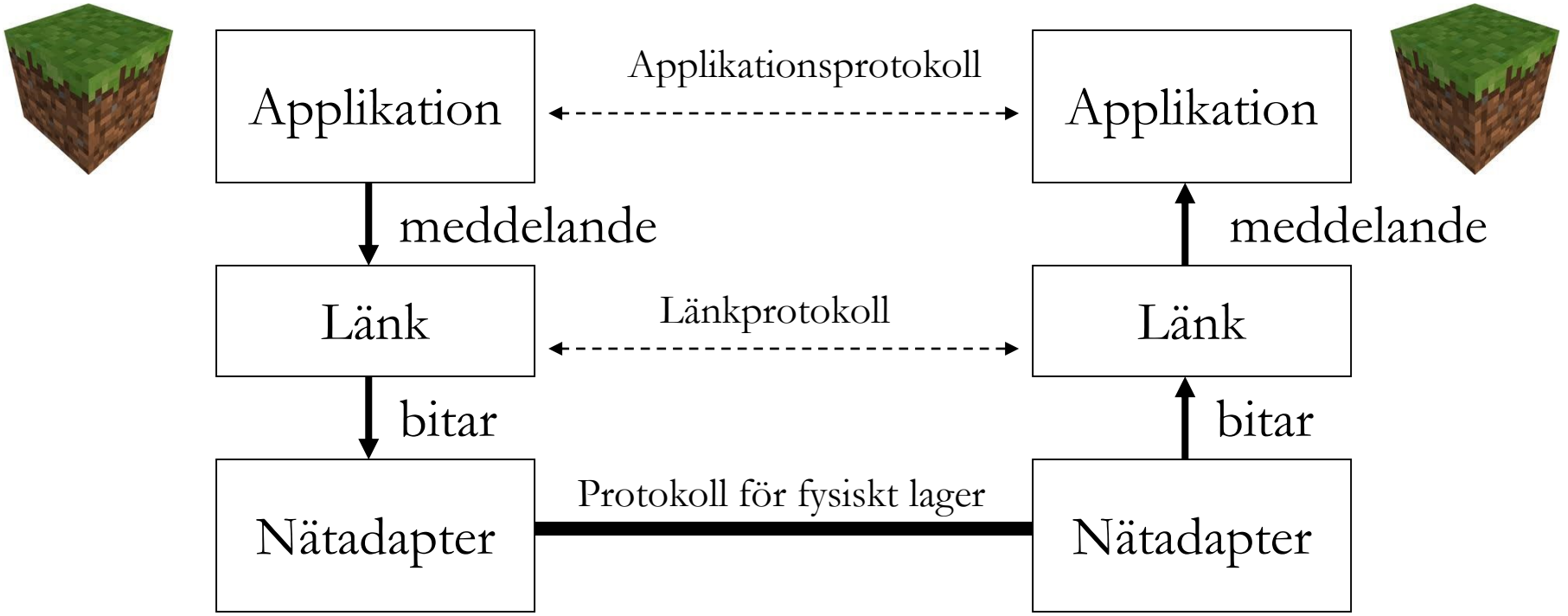
Att skicka data över en länk

Applikationen ska inte behöva veta hur länken fungerar. Det behövs ett annat protokoll som tar hand om kommunikationen över en fysisk länk.

110111 001111 100111 010011

Länkprotokoll

Sändare och mottagare använder ett länkprotokoll för kommunikationen över en länk.



Länkprotokollets uppgifter

Länkprotokollet har tre grundläggande uppgifter:

1. Framing (inramning)
2. Feldetektering (detektera bitfel)
3. Felhantering (finns inte i alla länkprotokoll)
4. Flödeskontroll

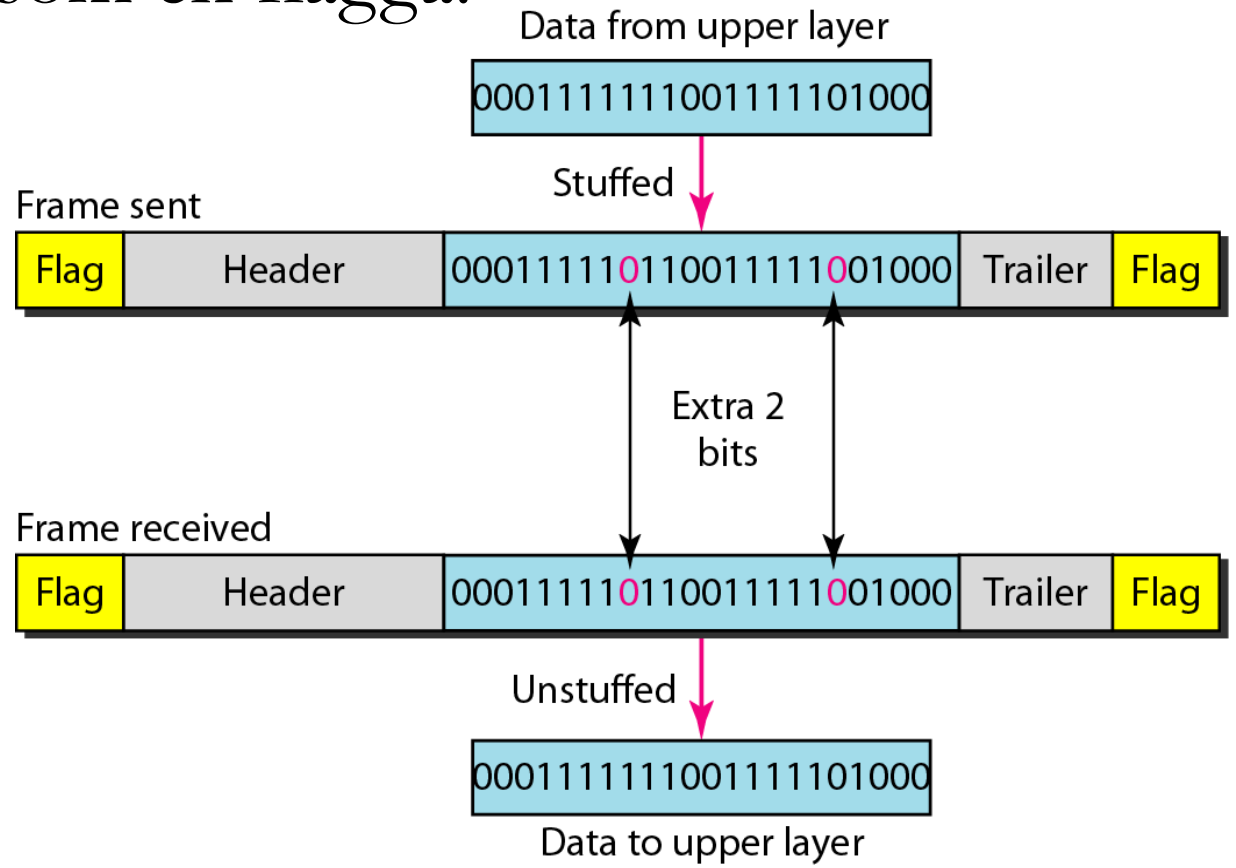
1. Framing (Inramning)

- På länken skickas en bitström. Länkprotokollet måste kunna översätta bitarna så att protokollet kan tolka data i varje paket.
- Därför används flaggor som "ramar in" varje paket.
- En flagga är ett specifikt bitmönster.
  Exempel: 01111110
- Ett datapaket på länklagret brukar kallas för ram.

Flagga	Kontroll	Data	Kontroll	Flagga

Bitutfyllnad (Bitstuffing)

Bitutfyllnad (bitstuffing) används så att data inte kan tolkas som en flagga.

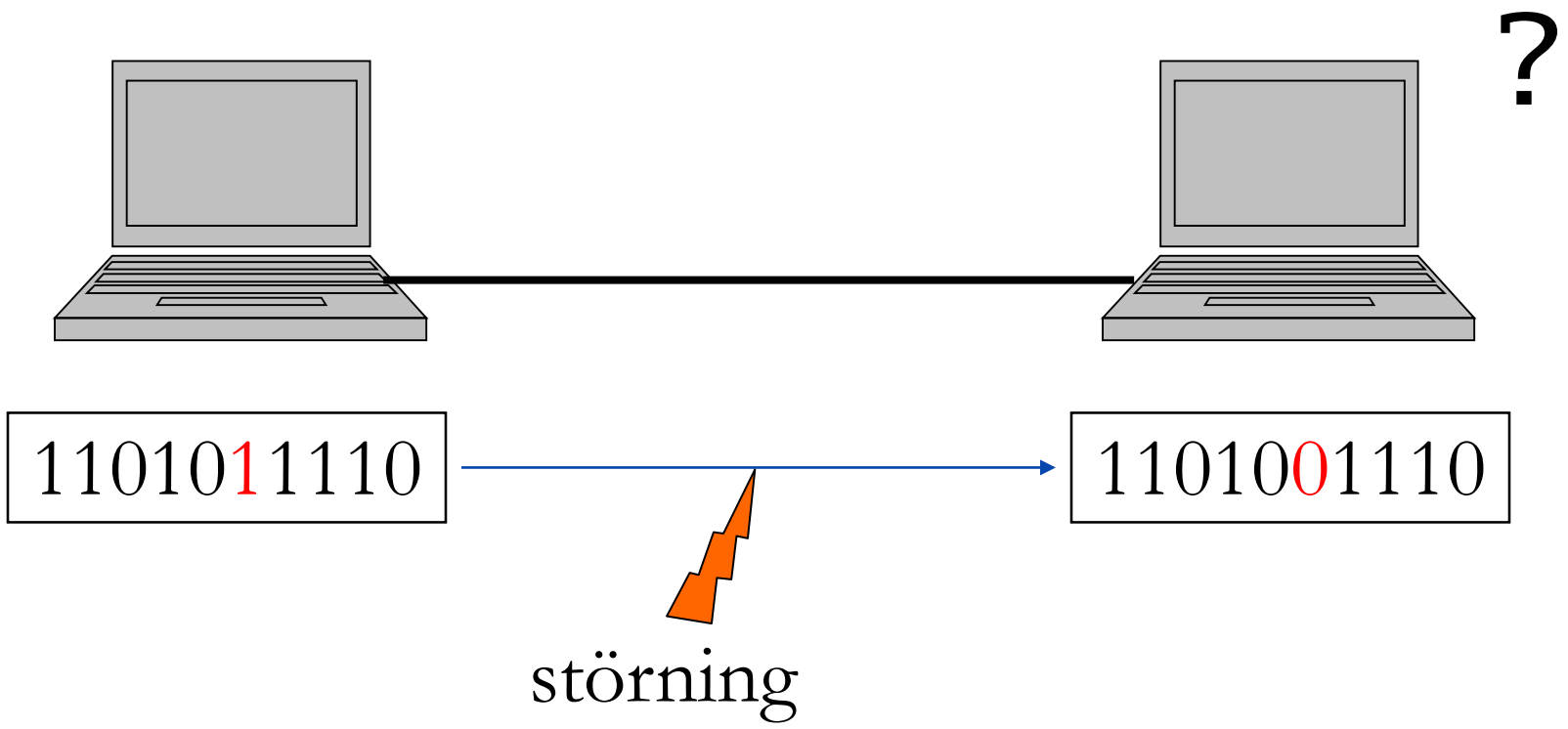


Tentaexempel: Bitstuffing

Följande bitström har tagits emot. Flaggan 01111110 används. Identifiera eventuella flaggor och bitstuffade 0:or.

000101111100001011111101111100011111000001

2. Feldetektering

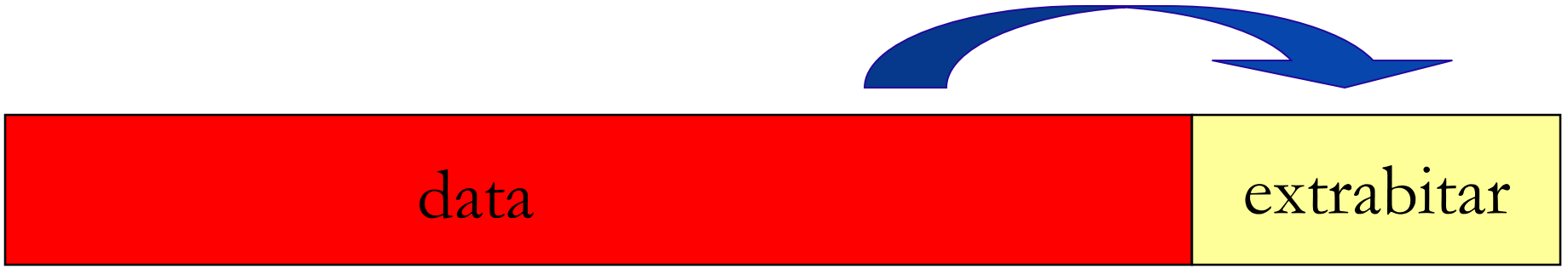


Mottagarens länkprotokoll måste kunna detektera ett bitfel!

Att hitta bitfel

Det är viktigt att mottagaren kan hitta de bitfel som uppstår.

Sändaren lägger till en eller flera redundanta bitar vars värde beror på innehållet i meddelandet.



Tre metoder för feldetektering (block coding)

- Paritetsbit (Simple Parity-Check Code)
- Cyklisk Redundanscheck (CRC)
- Kontrollsumma (Checksum)

Paritetsbit

Sändaren lägger till en bit i slutet av meddelandet.

Jämn paritet = jämnt antal ettor i hela meddelandet.

Ojämn paritet = ojämnt antal ettor i hela meddelandet.

Exempel på jämn paritet:

10011100	+	0	=	100111000

Med hjälp av paritetsbit kan man hitta ett udda antal bitfel i ett meddelande.

Tentaexempel: Paritetsbit

Följande meddelande har tagits emot. Protokollet använder en paritetsbit (jämn paritet). Har meddelandet tagits emot korrekt?

000101111100001

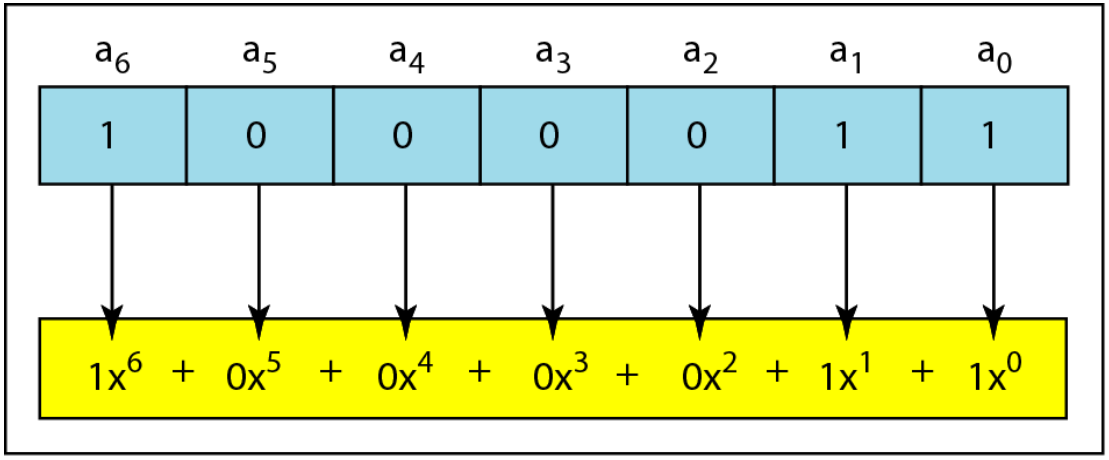
Cyklisk redundanscheck (CRC)

I CRC, använder sändare och mottagare ett förutbestämt generatortal (divisor) för att beräkna de extra bitarna.

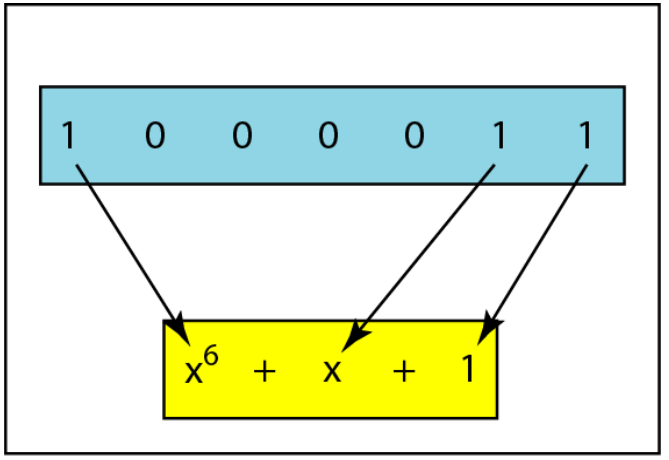
Matten bakom valet av generatortal ingår inte i den här kursen (men läs gärna extramaterial på kursens hemsida).

Polynomrepresentation

Ett block med *k* bitar (dataword) representeras av ett polynom, *M(x)* (*D(x)* i Forouzan och Stallings).



a. Binary pattern and polynomial



b. Short form

Cyklisk Redundanscheck (CRC)

Låt bitarna i paketet representeras av ett polynom.

Exempel:

$$\boxed{10011010} \quad = \quad x^7+x^4+x^3+x = M(x)$$

Använd ett generatorpolynom av grad k.

Exempel: $C(x) = x^3+x^2+1$ (k=3)

(P(x) i Stallings)

Modulo-2 aritmetik

CRC-beräkningarna görs med modulo-2 aritmetik. Detta är binär addition utan överskjutande bitar (carries).

Tex. $10_2 + 10_2 = 00_2$

(kort sagt: $1+1 = 0$)

CRC hos sändaren

Hitta ett polynom, *R(x)*, så att

$$M(x) \cdot x^k + R(x) = C(x) \cdot f(x)$$

(*f(x)* är ett ointressant polynom)

$M(x) \cdot x^k + R(x)$ ska vara jämnt delbart med C(x)

Skicka iväg bitarna som representeras av

$$M(x) \cdot x^k + R(x)$$

CRC-exempel

Dataword: 1001

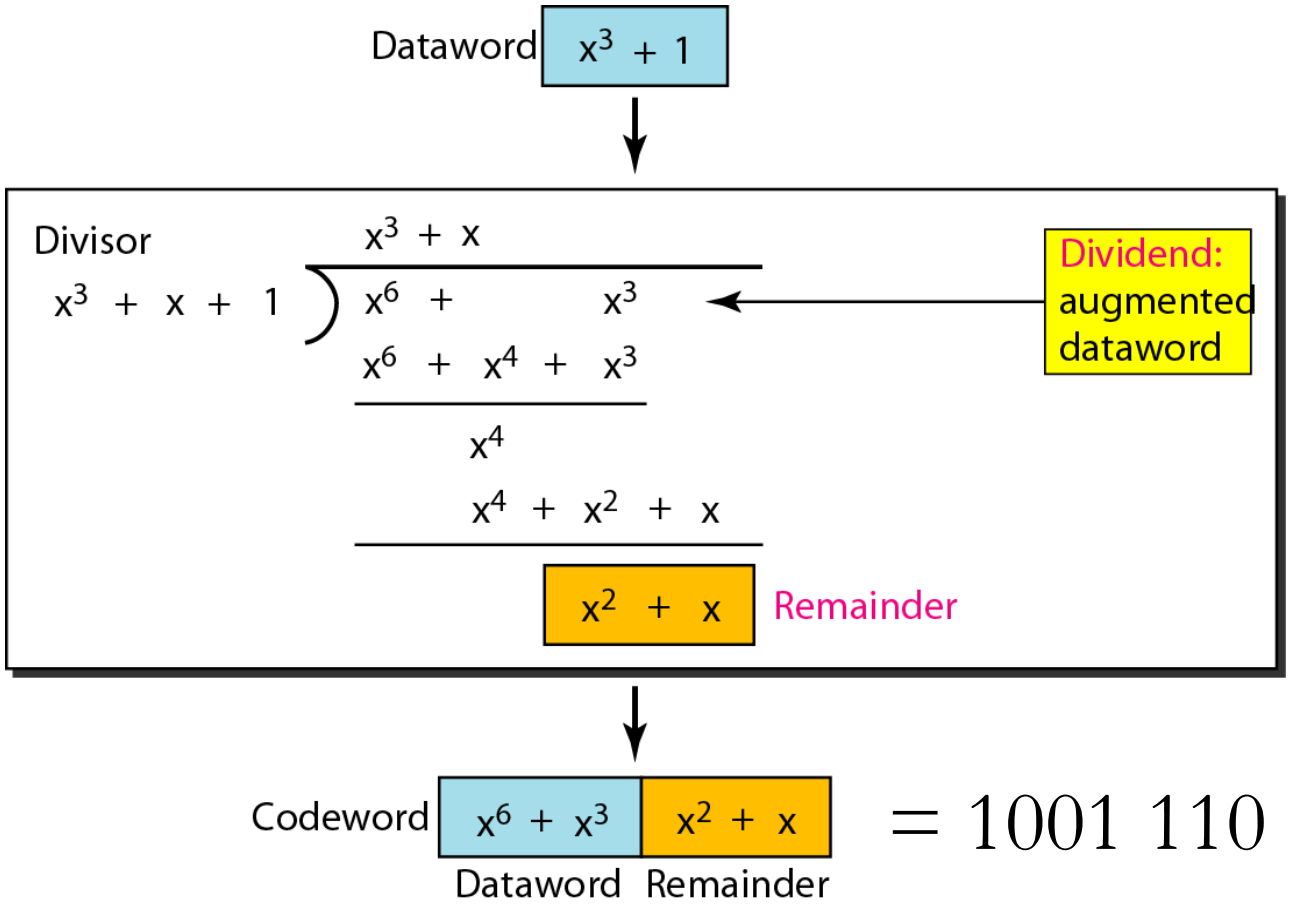
Polynomrepresentation: $x^3 + 1$

Generatortal (divisor): 1011

Generatorpolynom: $x^3 + x + 1$

Dataword multiplicerat med x^3: $x^6 + x^3$

CRC-exempel forts.



CRC hos mottagaren

De mottagna bitarna (codeword) är $P(x)+E(x)$ där $E(x)$ är en representation av bitfelen som inträffat under transmissionen.

Mottagaren beräknar följande:

$$\frac{\text{Mottaget codeword}}{g(x)} = \frac{P(x)}{g(x)} + \frac{E(x)}{g(x)}$$

Om resultatet blir 0 så kan de mottagna bitarna antagas korrekta.

Några standardiserade generatorpolynom

Name	*Polynomial*	*Used in*
CRC-8	$x^8 + x^2 + x + 1$ **100000111**	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$ **11000110101**	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$ **1000100000100001**	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ **10000010011000001000111011011011 1**	LANs

Tentaexempel: CRC

Beräkna en 3-bitars CRC för sekvensen 101101 med generatortalet 1011.

Kontrollsumma (Checksum)

Kontrollsumma används av olika Internetprotokoll, dock inte på länklagret.

Grundprincipen är att dela upp data i segment bestående av *n* bitar. Sedan adderas motsvarande binära tal och summan används som kontrollsumma.

Exempel på kontrollsumma (sändaren)

Meddelande: 10101001 10111001

8-bitars kontrollsumma:

10101001 + 10111001 = 1 01100010

Addera överskjutande bitar (one's complement):
00000001 + 01100010 = 01100011

Komplementet på kontrollsumman skickas för att hjälpa mottagaren:
10101001 10111001 10011100

	10101001	
	10111001	
1	01100010	Delsumma
	1	Överskjutande bit
	01100011	Summa
	10011100	Komplementet

Exempel på kontrollsumma (mottagaren)

Mottagaren adderar alla segment med one's complement.

Om det inte har inträffat några bitfel ska resultatet bli bara ettor (som sedan inverteras till nollor):

$$
\begin{array}{r}
10101001 \\
10111001 \\
10011100 \\
\hline
1\ 11111110 \\
\hline
11111111
\end{array}
$$

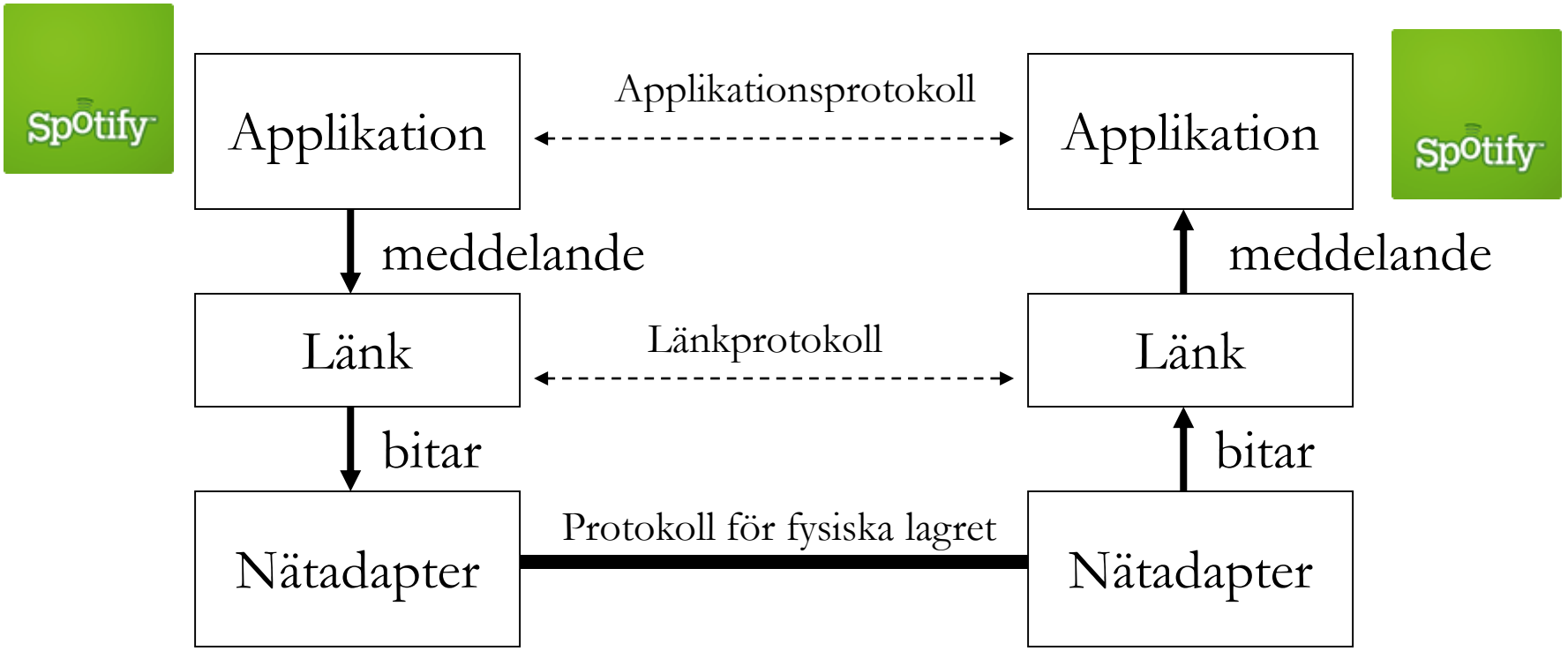
Tentaexempel: Kontrollsumma

Beräkna en 4-bitars kontrollsumma för bitsekvensen

1100 0010 1111

3. Felhantering

Länkprotokollet ska se till att bitfel hanteras så att mottagaren får korrekt data.



Felhantering (error control)

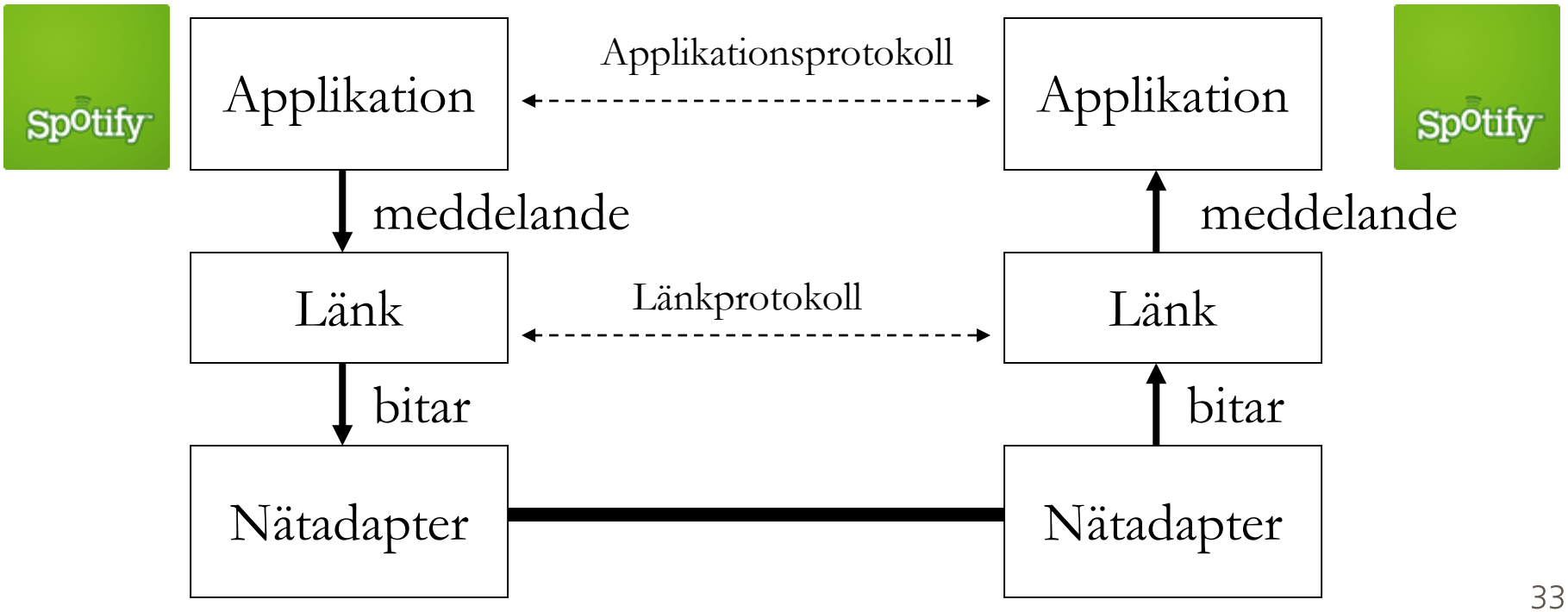
När en korrupt ram detekteras så måste felet korrigeras

Två grundläggande principer:

- Forward Error Correction (FEC)
  - Data kodas på ett sådant sätt att bitfel kan korrigeras.
  - FEC ingår inte i denna kurs.
- Omsändning av data
  - Felaktiga ramar skickas om tills dess att sändningen är felfri.

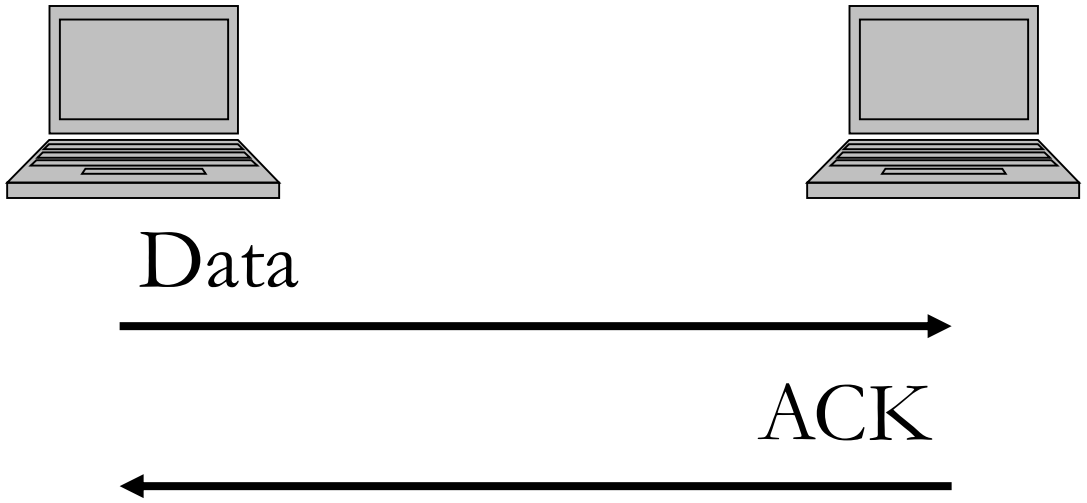
4. Flödeskontroll

Sändaren måste skicka ramar i en takt som mottagaren klarar av att hantera.



Felhantering och flödeskontroll

Den grundläggande principen i felhantering via omsändningar samt flödeskontroll är att mottagaren skickar acknowledgments (ACK) för alla korrekt mottagna paket/ramar.



Automatic Repeat Request (ARQ)

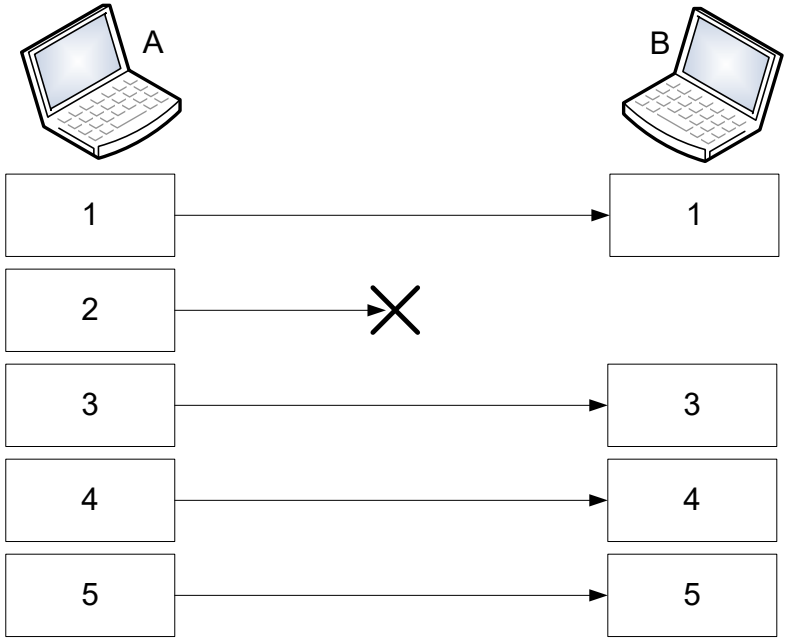
Tre grundläggande metoder:

- Stop-and-wait ARQ
- Go-back-N ARQ
- Selective Repeat ARQ

Stallings *har ändrat i Go-back-N och lagt in ”REJ” meddelanden. Dessutom beskriver Stallings Selective-Reject istället för Selective-Repeat.*

Sekvensnummer

Sekvensnummer används för att kunna detektera vilka paket som är korrekt mottagna och vilka som behöver sändas om.



(I fortsättningen antas att ett bitfel genererar en paketförlust)

Stop-and-wait ARQ

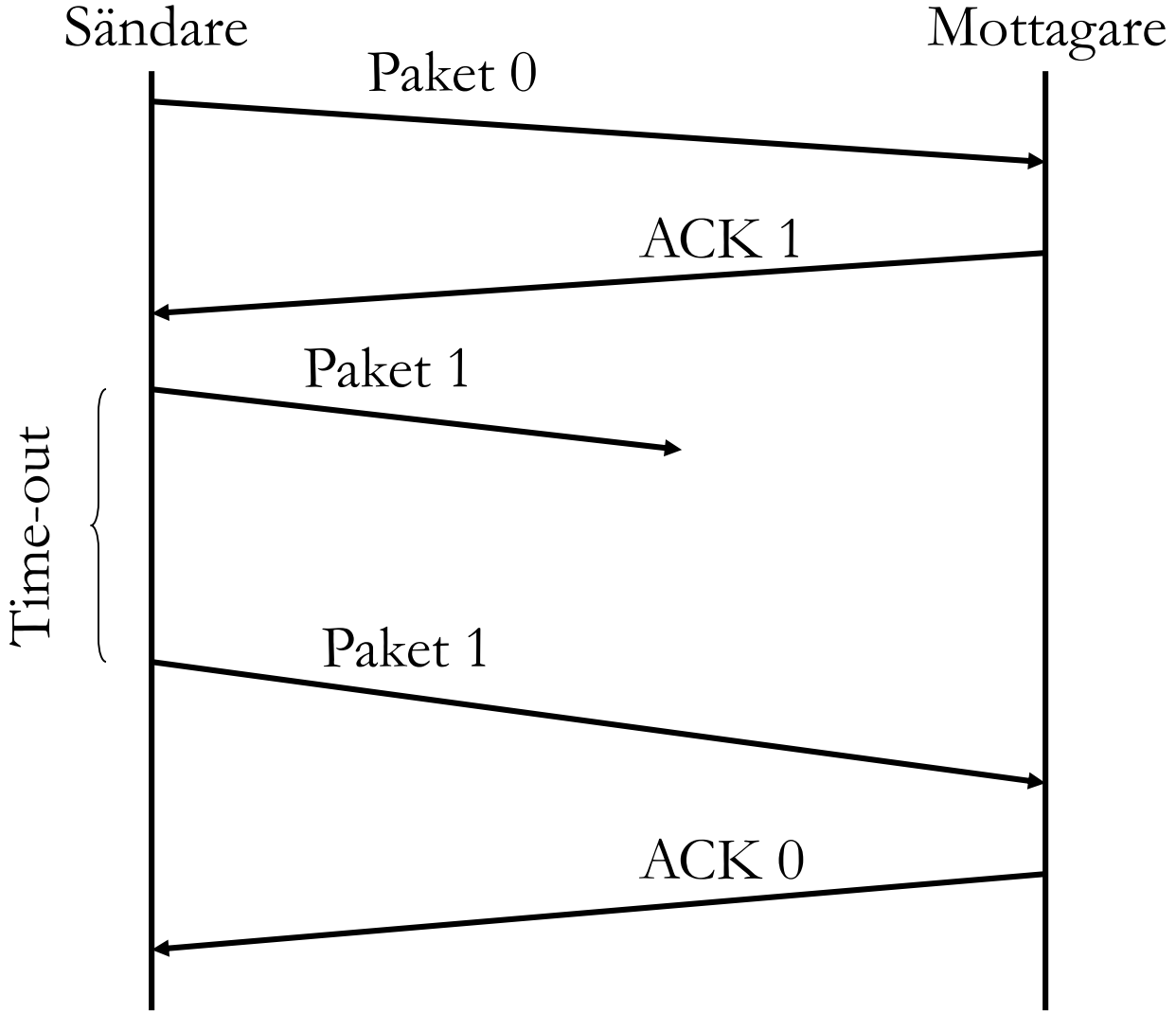
Mottagaren skickar en ACK för varje paket.

Sändaren skickar nästa paket när den fått en ACK för det senast skickade paketet.

Sändaren använder en time-out för varje skickat paket. Om inget ACK har kommit innan time-out så skickas paketet igen.

Paketen är identifierade med sekvensnummer som alternerar mellan 0 och 1.

Exempel: Stop-and-Wait ARQ



Tentaexempel: Stop-and-wait ARQ

En dator skickar 5 paket (DATA 1-5) till en annan dator. DATA 3 försvinner (men alla andra paket kommer fram korrekt).

Hur många paket (DATA och ACK) skickas mellan de två datorerna om de använder Stop-and-wait ARQ?

Go-back-n ARQ

Sändaren kan skicka flera paket åt gången.

Antalet paket som kan skickas bestäms med hjälp av ett så kallat sändfönster (sliding window).

Mottagaren skickar ACK för de paket som tas emot i rätt ordning (mottagaren skickar ”ACK x+1” där x är det paket som tagits emot).

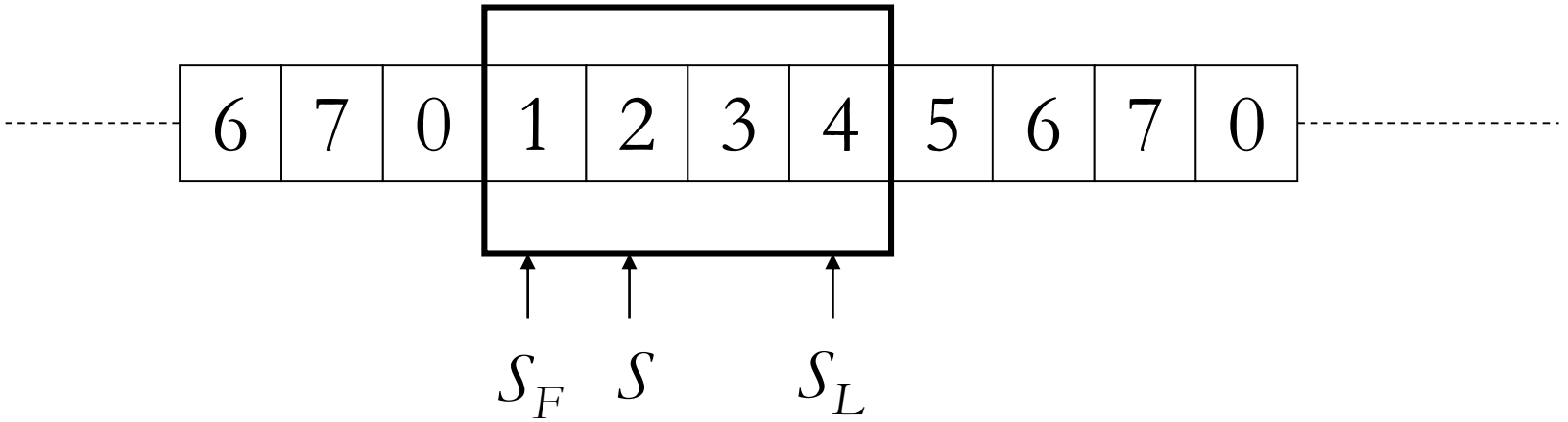
Nya paket kan skickas i samma takt som mottagaren skickar ACK.

Mottagaren brukar spara paket som kommit fram korrekt, även om paketet innan saknas.

Sändfönster

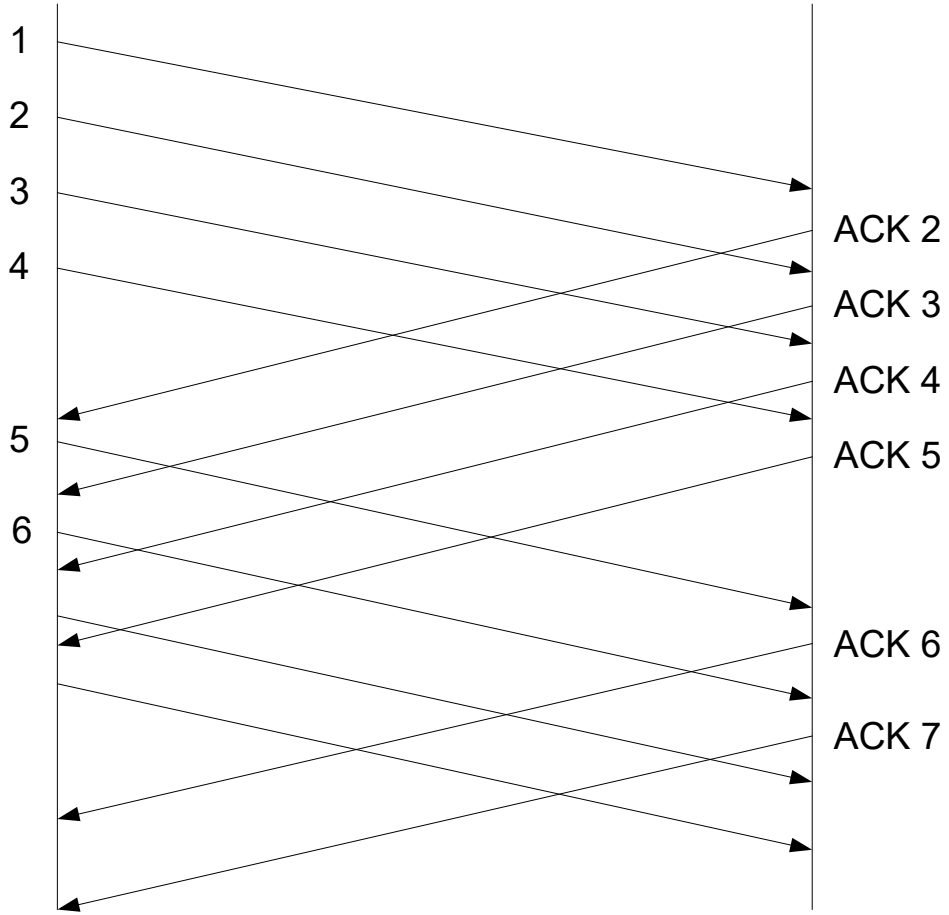
Sändfönstret innehåller de sekvensnummer som är involverade i sändningen just nu.

Sändfönsterstorleken används som flödeskontroll och anger hur många paket som sändaren kan skicka utan att få ACK.



Exempel : Go-back-N ARQ (inga bitfel)

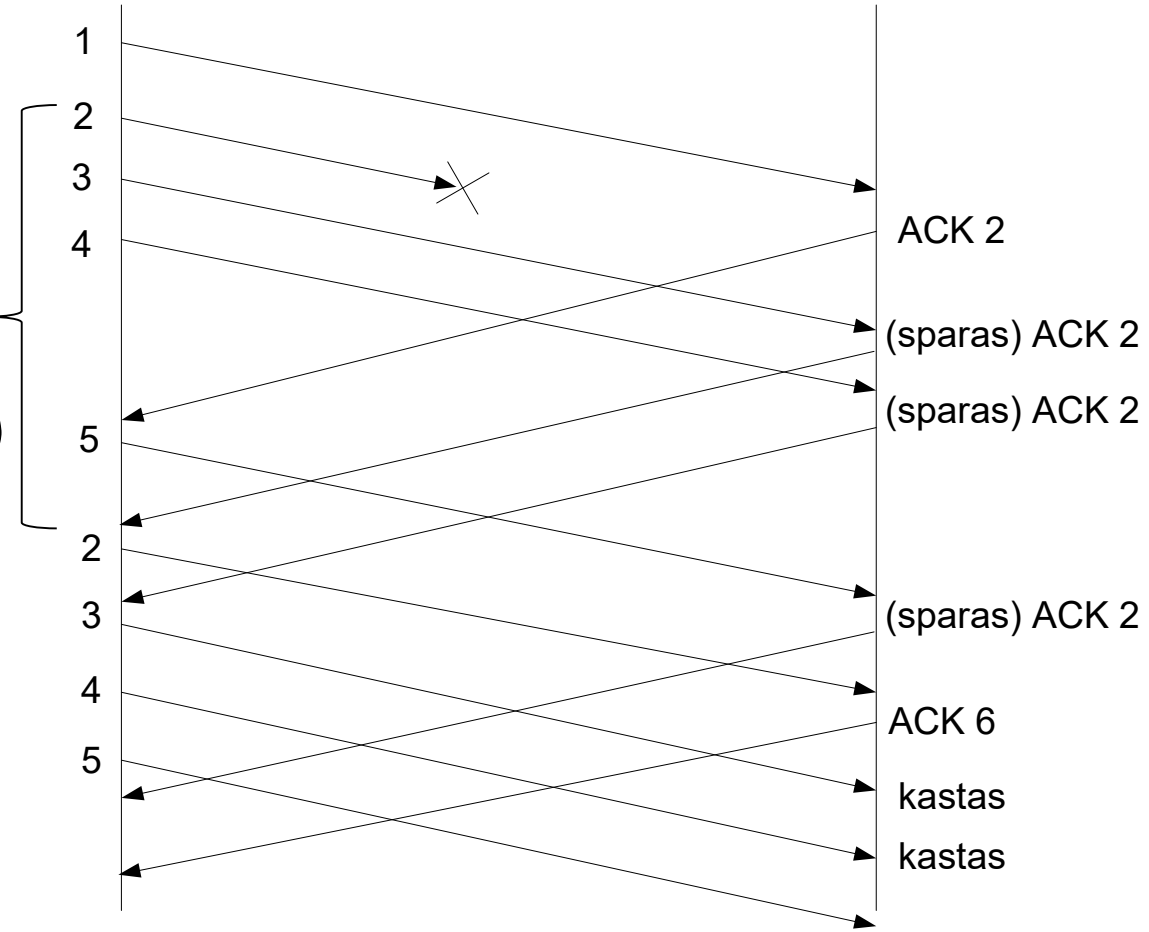
Sändfönsterstorlek = 4



Exempel : Go-back-N ARQ (paketförlust)

Sändfönsterstorlek = 4

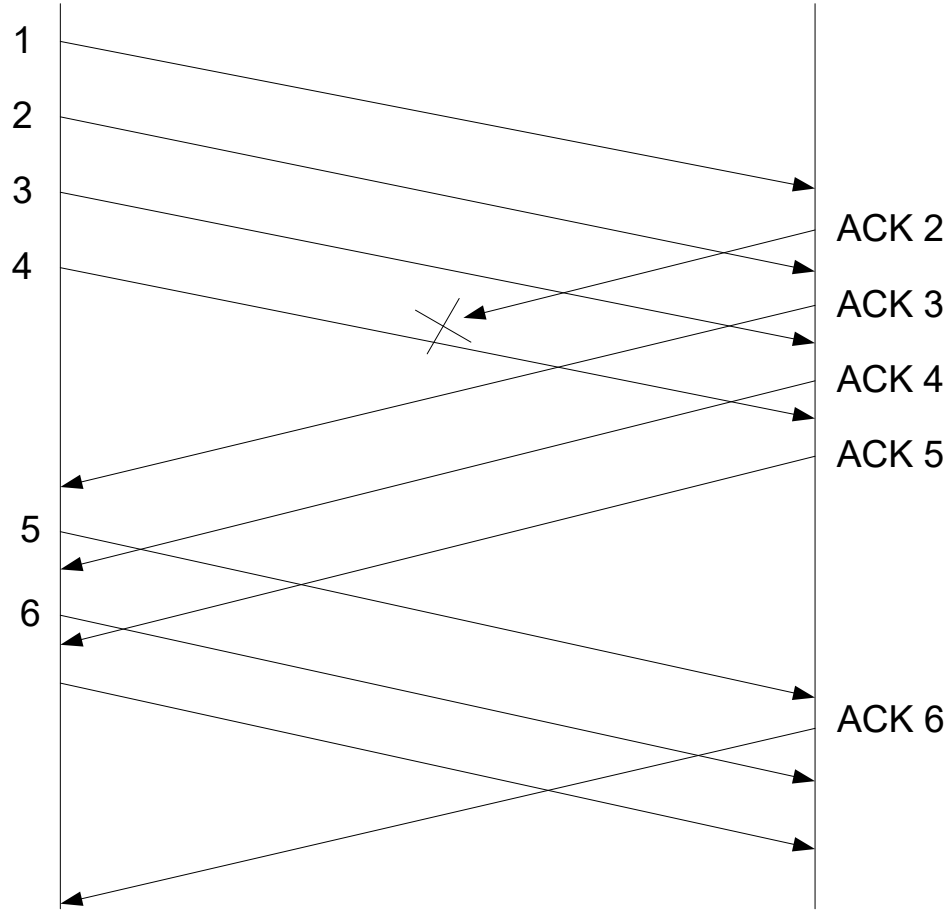
Time-out
(alternativ:
Duplicerat ACK)



Exempel : Go-back-N ARQ (förlorat ACK)

Sändfönsterstorlek = 4

Ett förlorat ACK kan upptäckas om nästa ACK kommer fram korrekt



Notering

Go-back-N ARQ finns i olika varianter beroende på vilket protokoll som det används i.

Det viktiga är att förstå den grundläggande principen för Go-back-N ARQ och att kunna förklara vad som händer när man skickar paket och något paket försvinner.

På kursens hemsida finns länkar till bra beskrivningar av Go-back-N ARQ.

Tentaexempel: Go-back-N ARQ

En dator skickar 5 paket (DATA 1-5) till en annan dator. DATA 3 försvinner (men alla andra paket kommer fram korrekt).

Hur många paket (DATA och ACK) skickas mellan de två datorerna om de använder Go-back-N ARQ med fönsterstorlek 3?

Selective repeat ARQ

Fungerar som Go-back-N så länge som det inte är några paketförluster.

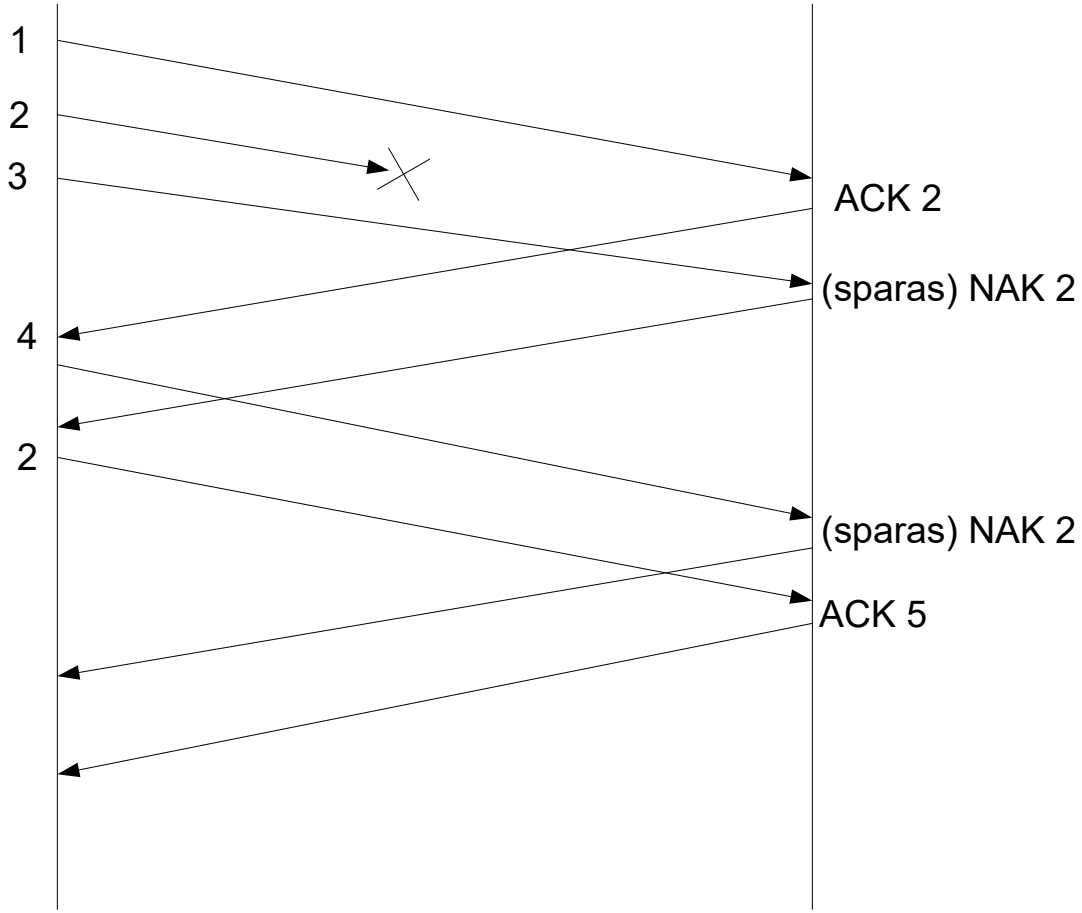
Grundprincip:

Mottagaren kan skicka en negativ acknowledgment (NAK) om den detekterar att ett paket är förlorat.

Enbart de förlorade paketen omsänds.

Exempel: Selective repeat ARQ

Sändfönsterstorlek = 3

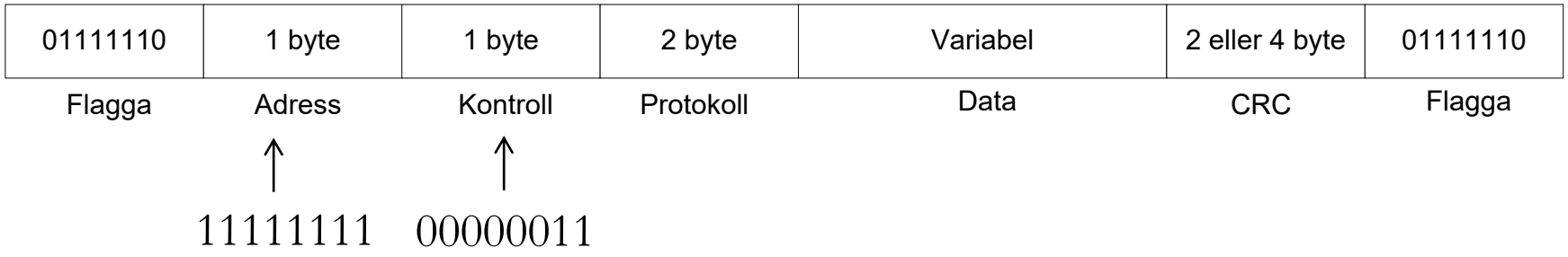


Point-to-Point Protocol (PPP)

Point-to-point protocol (PPP) är ett av de klassiska länkprotokollen. Användes i bland annat bredbandsaccess och mobila nät.

PPP är ett byte-orienterat (character-orienterat) protokoll, dvs all data i ramen räknas i bytes.

Ramformat:



Byte stuffing

Protokollet måste se till att bitmönstret för flaggan inte finns i data.

Eftersom PPP är ett byte-orienterat protokoll, används byte stuffing.

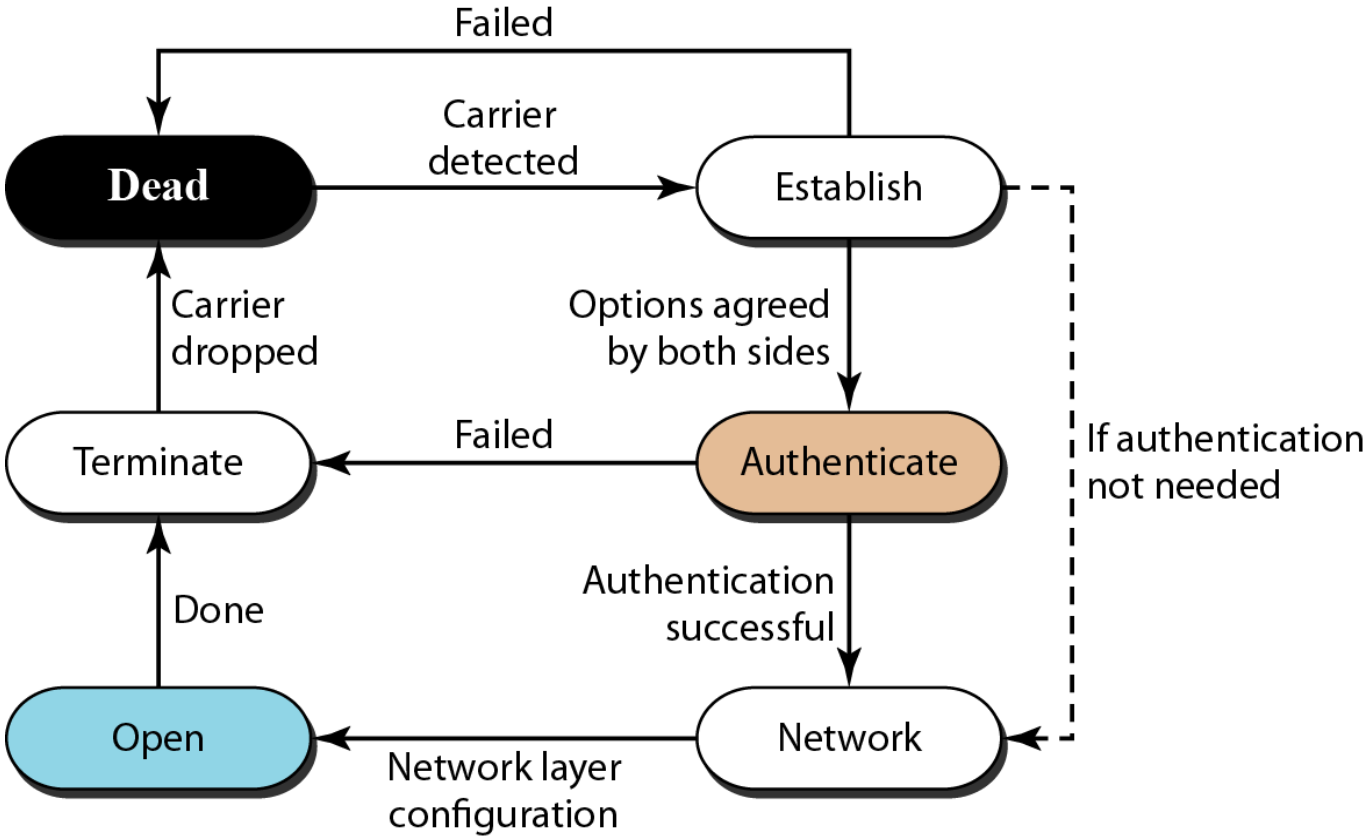
Varje gång bitmönstret för flaggan återfinns i data lägger man till en byte 01111101 **före** så att användaren vet att nästa byte *inte* är en flagga.

Feldetektering

- PPP använder en 2 eller 4-byte CRC för feldetektering.
- Men, protokollet innehåller ingen felhantering eller flödeskontroll.

Protokollfaser

En PPP förbindelse går igenom följande faser:

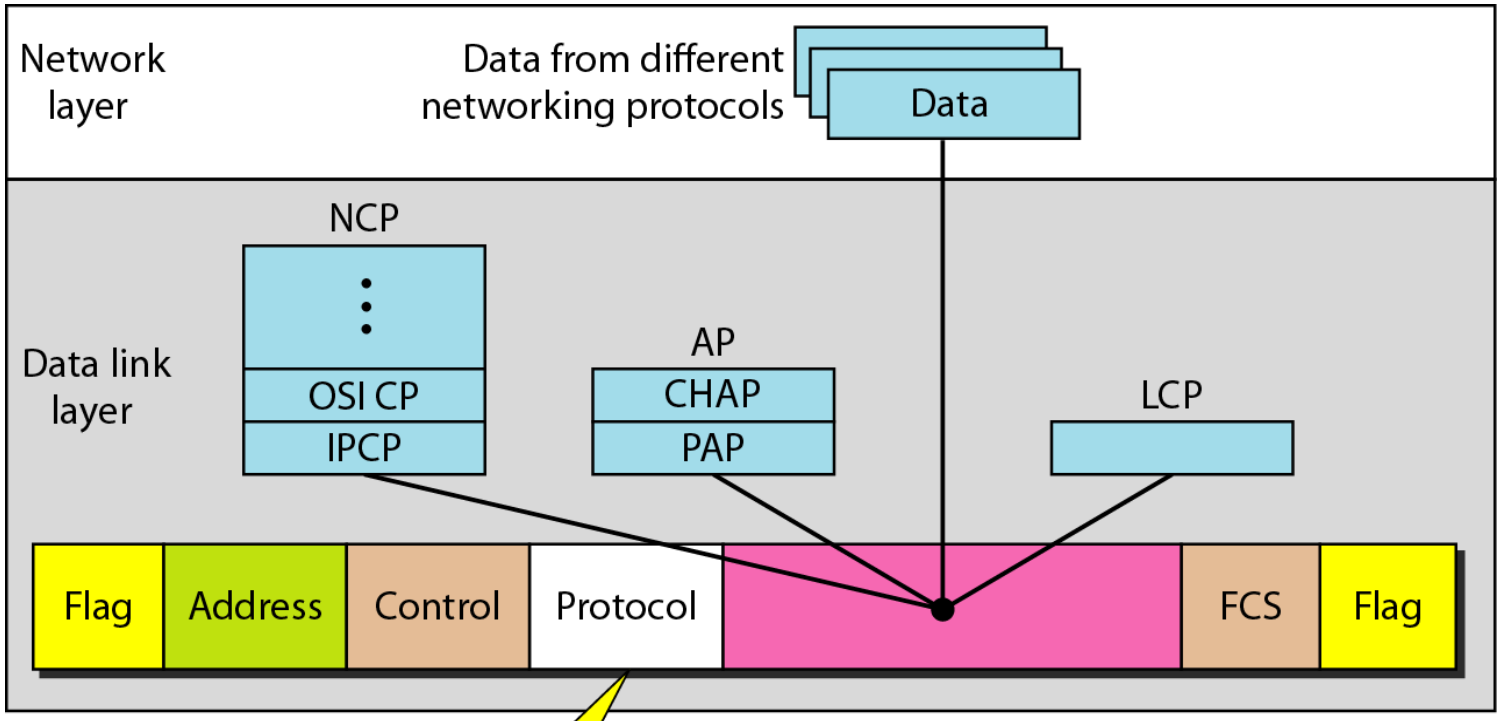


Hjälp-protokoll

PPP använder flera andra protokoll för att hantera förbindelsen:

- Link Control Protocol (LCP) är ansvarigt för att etablera, upprätthålla, konfigurera, och avsluta förbindelsen.
- Password Authentication Protocol (PAP) och Challenge Handshake Authentication Protocol (CHAP) används för autentisering. Autentisering innebär att sändare och mottagare kan validera varandras identitet.
- Ett Network Control Protocol (NCP) konfigurerar förbindelsen för ett visst nätprotokoll. Ett exempel är Internet Protocol Control Protocol (IPCP) för Internet.

De andra protokollen skickas i PPP-ramar



Inför PPP-laboration

- Läs online-manual med förberedelser!
  http://www.eit.lth.se/ppplab/overview.htm
- Om du behöver läsa mer om PPP:
  - http://en.wikipedia.org/wiki/Point-to-point_protocol
  - http://www.tcpipguide.com/free/t_PointtoPointProtocolPPP.htm