Рачунарске мреже

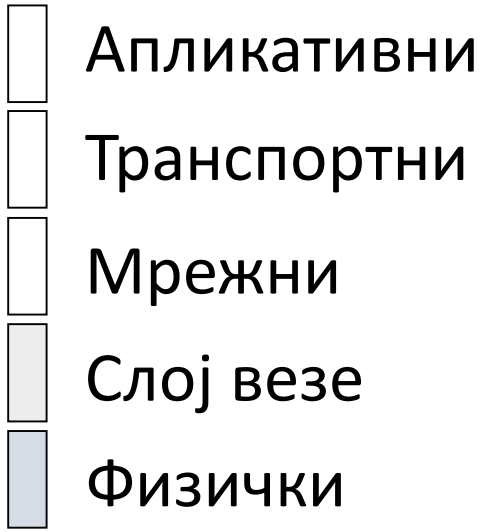
Александар Картељ

kartelj@matf.bg.ac.rs

Слој везе

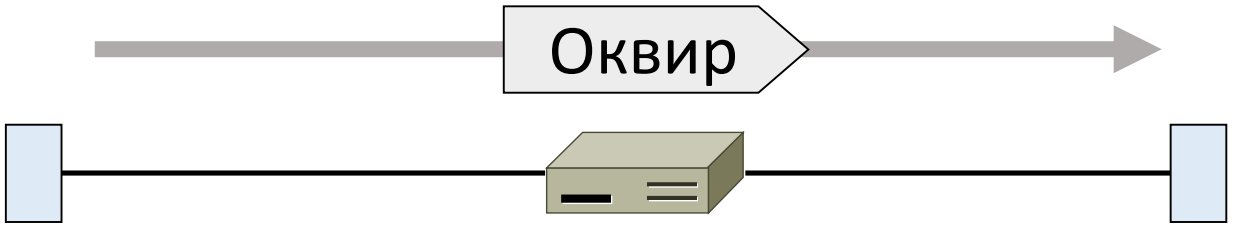
Општи појмови

Где смо тренутно...

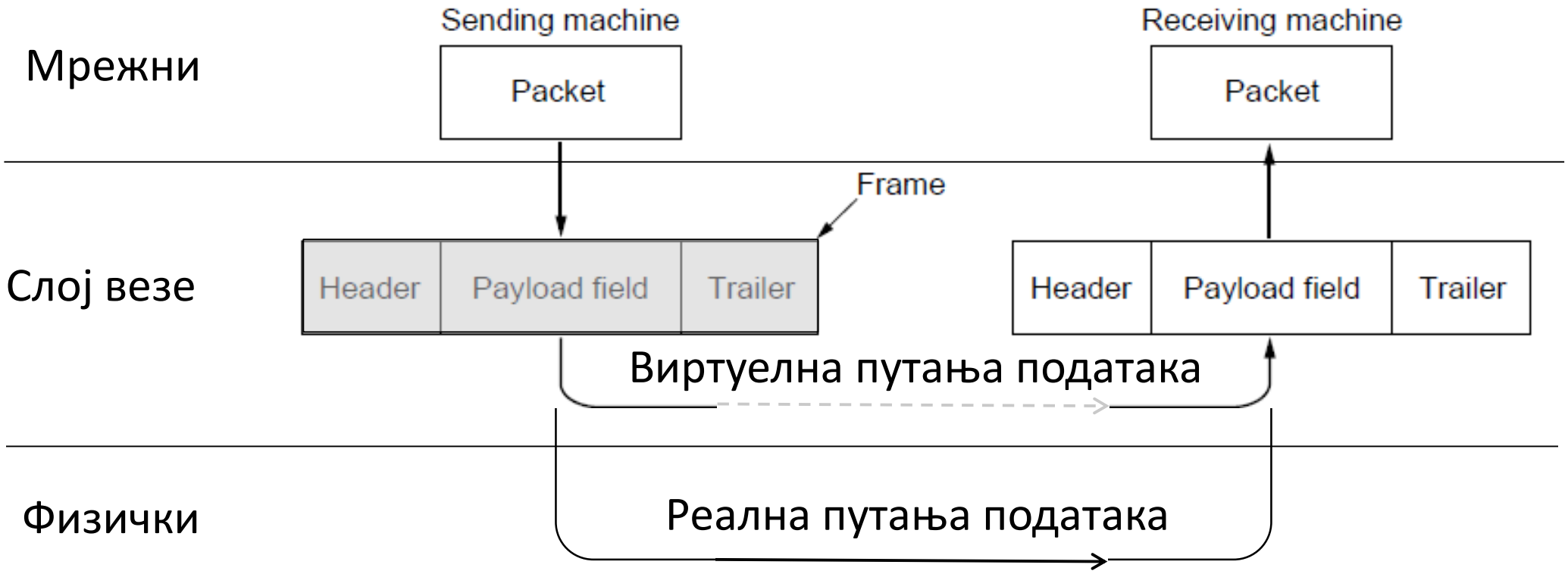


Одговорност слоја везе

- Пренос оквира путем једног или више повезаних комуникационих канала
  - Оквири су фиксиране величине
  - Наслања се на физички слој



Расподела активности по слојевима



Теме

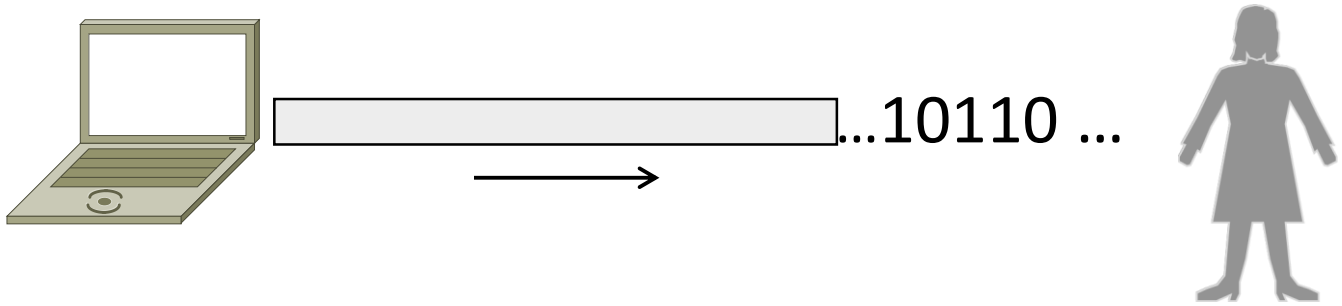
1. Уоквиривање
   - Заглавље, порука и завршетак оквира
2. Детекција и корекција грешака
3. Ретрансмисија и контрола тока
   - Поновно слање у случају губитка
   - Разрешавање дупликата
   - Усаглашавање брзина пошиљаоца и примаоца
   - Основни протоколи слоја везе

Слој везе

Уоквиривање

Тема

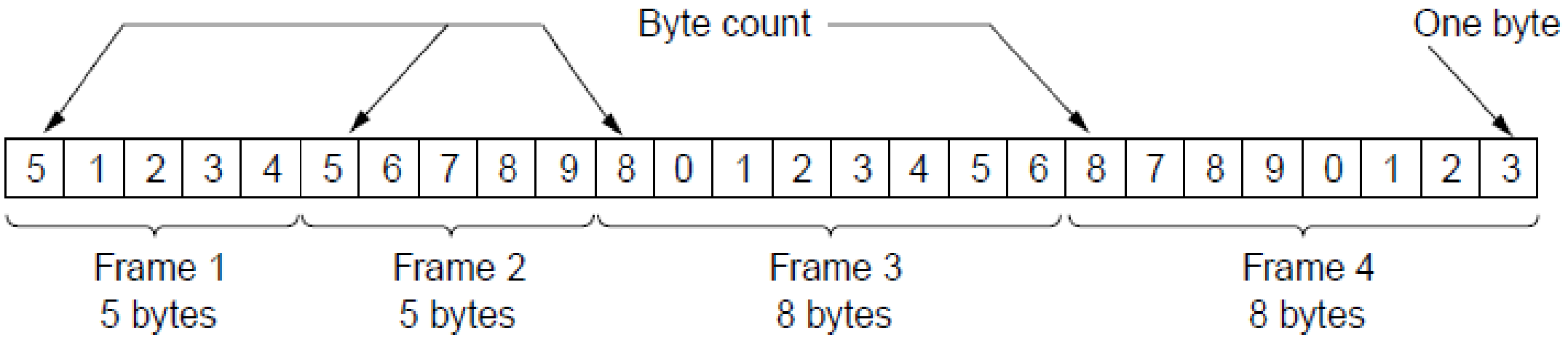
- Физички ниво нам даје ток података.
  Како га даље интерпретирамо као низ оквира?



Методе уоквиривања

- Стандардни приступи:
  - Бројање бајтова (само мотивациони)
  - Уметање бајтова
  - Уметање битова
- У теорији, уоквиривање је потпуно невидљиво физичком слоју
- Ипак, у пракси понекад физички слој помаже у идентификацији оквира

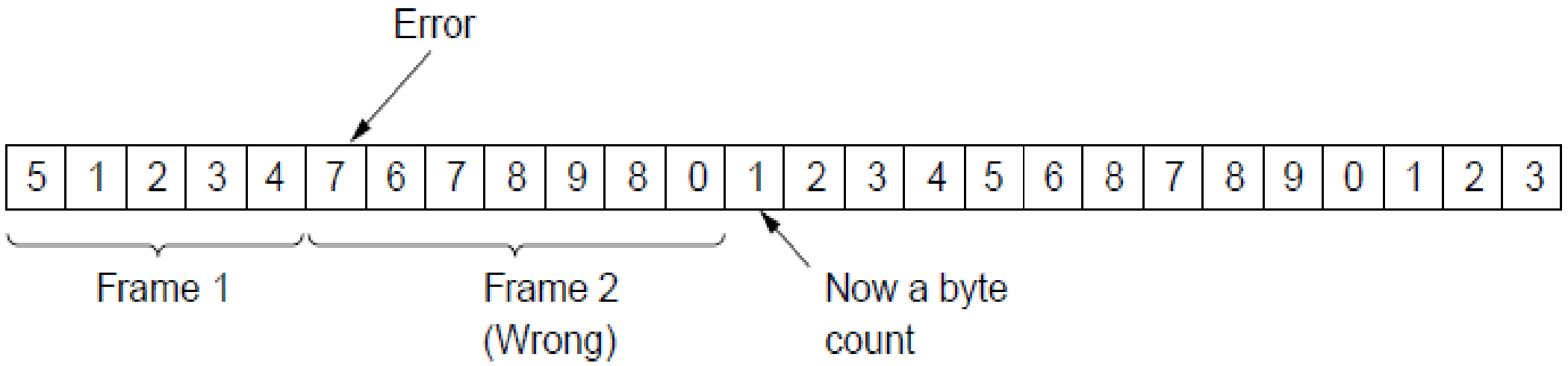
Бројање бајтова



- Започнимо сваки оквир са пољем о његовој дужини
- Колико добро ово ради?

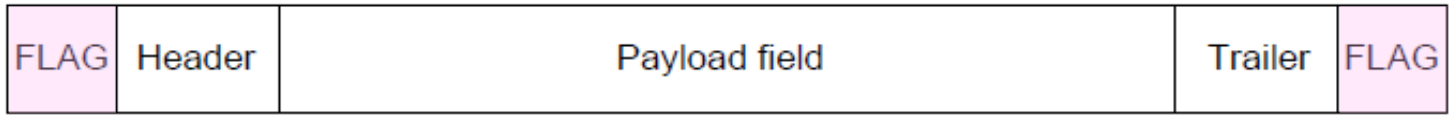
Бројање бајтова (2)

- Тешко за усаглашавање у случају грешке
  - Како да пронађемо почетак следећег оквира у случају грешке?



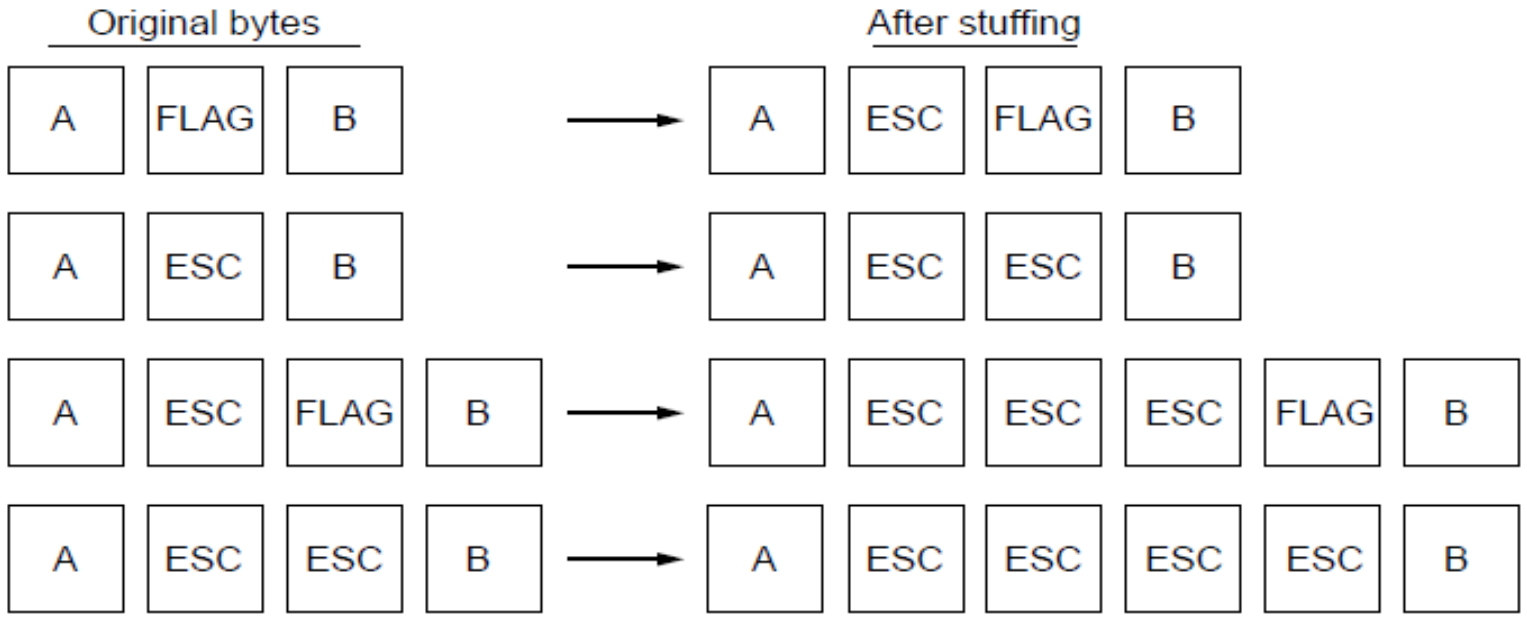
Уметање бајтова

- Боља идеја:
  - Индикаторска ознака у виду бајта који означава почетак/крај оквира
  - Шта ако се тај индикатор налази у подацима?
  - Шта ако се у подацима налази и знак ESC?



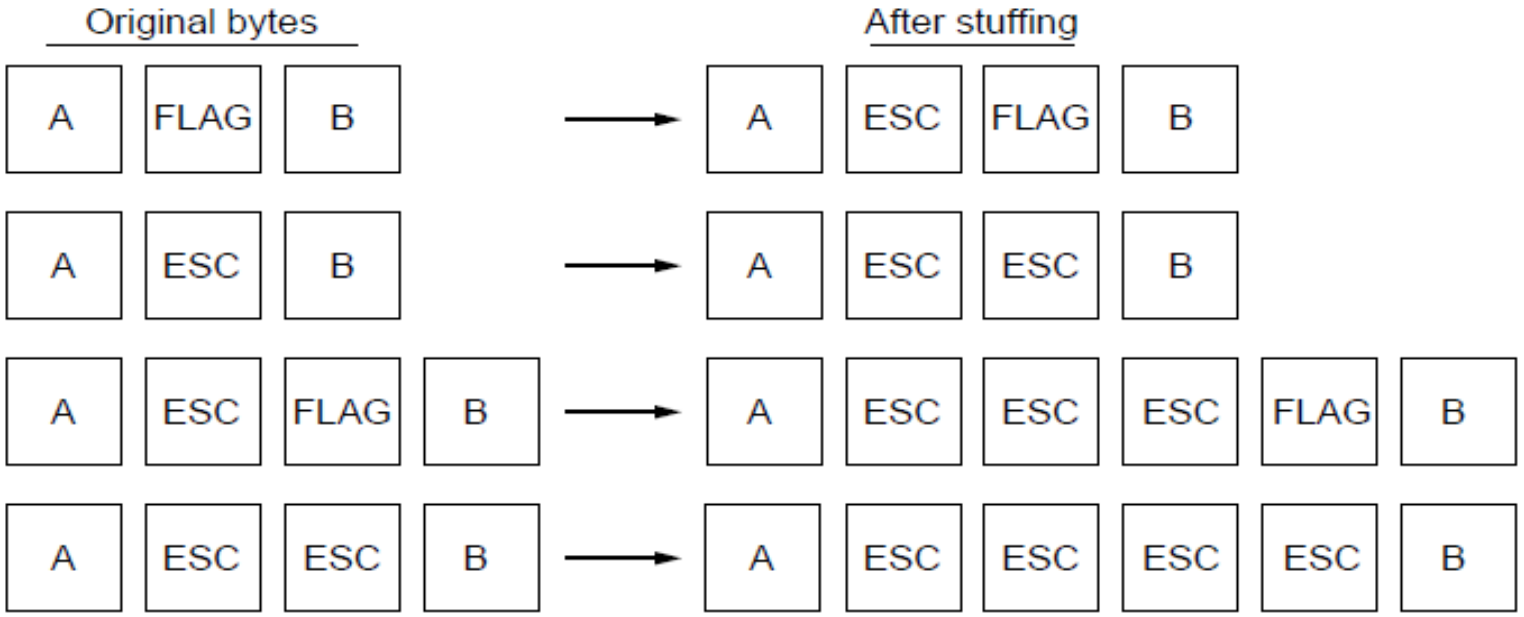
Уметање бајтова (2)

- Правила за промену знакова унутар података:
  - Заменити сваки „индикатор“ са „ESC индикатор“
  - Заменити сваки „ESC“ из података са „ESC ESC“



Уметање бајтова (3)

- Друга страна (пријемник) увек скида први ESC, а следећи не дира
- Након тога, сваки индикатор без непосредног ESC испред је стварно индикатор, а онај коме остане

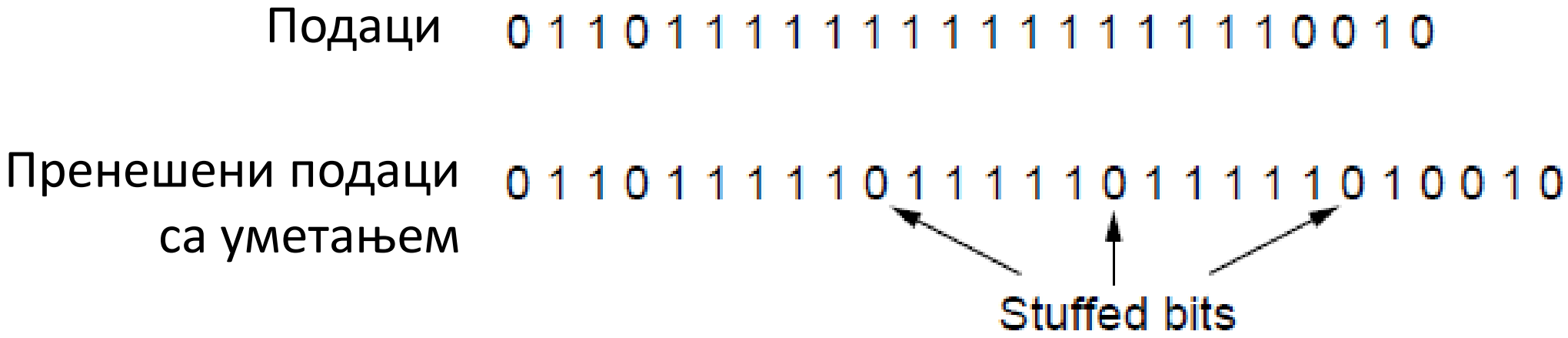


Уметање битова

- Уметање се може радити и на нивоу битова
  - Нека је индикатор шест узастопних јединица
  - При слању, након сваких 5 јединица података уметнути нулу (како би се разликовало од индикатора)
  - При прихватању података, нула после сваких 5 јединица се брише

Уметање битова (2)

- Која је разлика у односу на уметање бајтова?



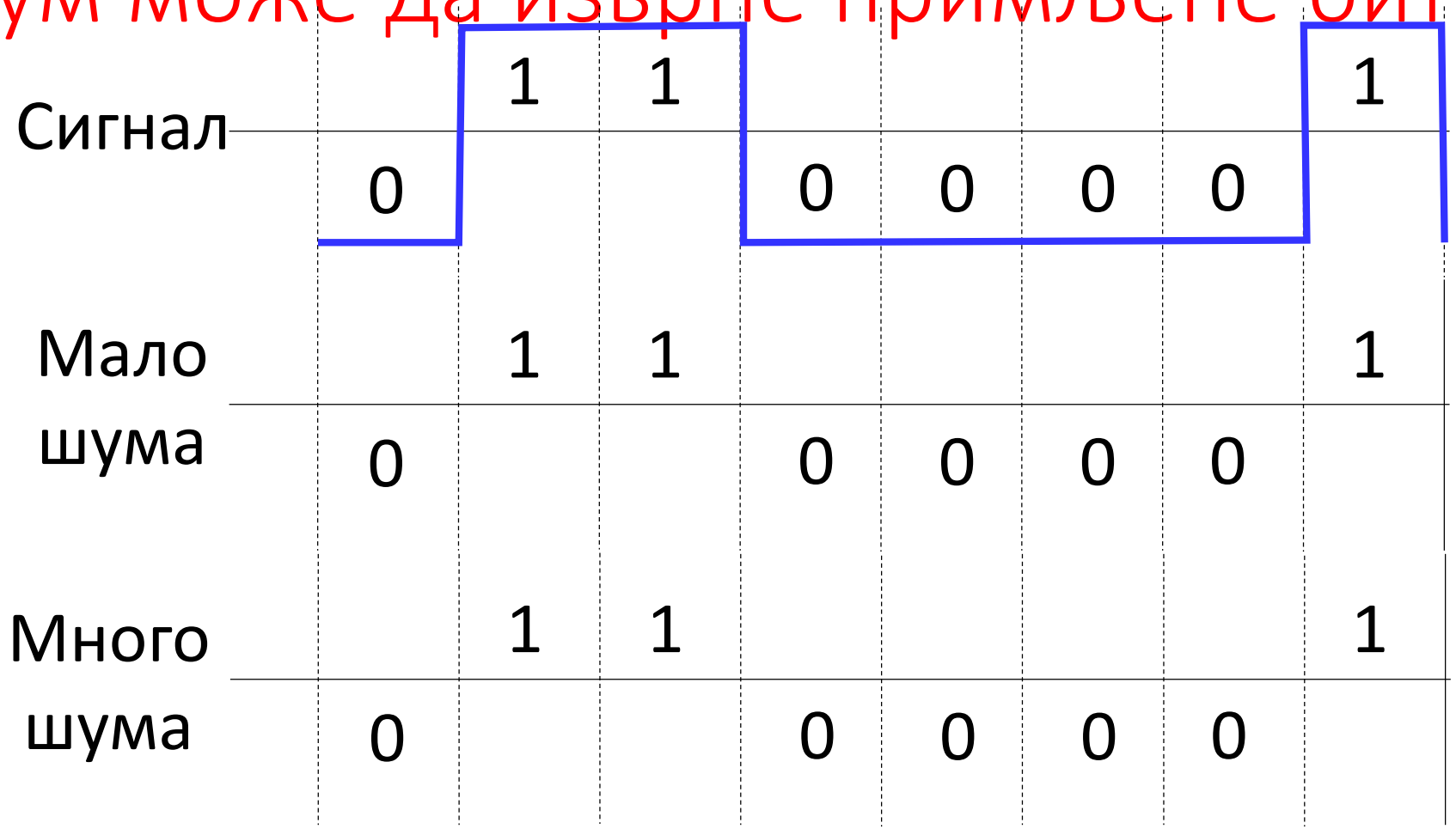
Слој везе

Кодирање грешака

Тема

- Долази до грешке на неким битовима због шума. Шта се може радити у том случају?
  - Детекција грешака
  - Корекција грешака
  - Ретрансмисија иначе

Проблем –
шум може да изврне примљене битове



Приступ – додавање редудантности

- Детекција грешака
  - Додавање контролних битова
- Кодови за корекцију грешака
  - Додавање још битова које омогућавају корекцију

Мотивишући пример

- Наивни приступ:
  - Послати две копије за сваку поруку. Ако су различите, онда је грешка!

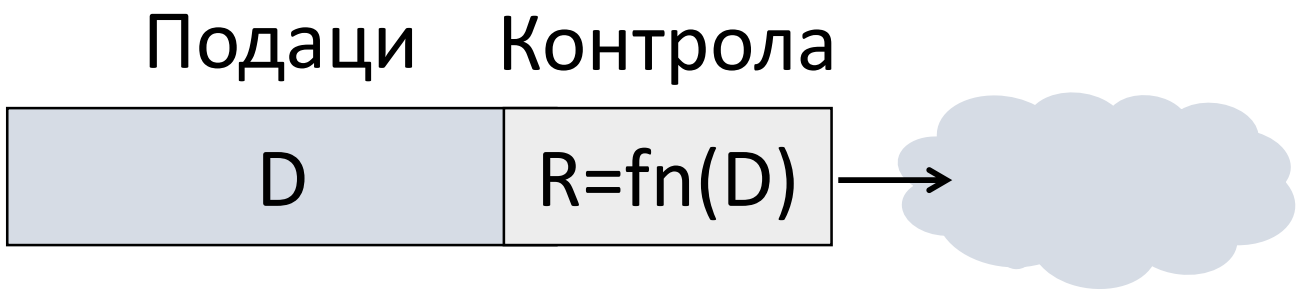
- Колико је добар овај приступ?
  - Колико грешака може да детектује/исправи?

Мотивишући пример (2)

- Циљ је разрешити што више грешака са што мање редудантности
  - Примењена математике, теорија бројева, ...
  - Али не могу да разрешавају све грешке
  - Фокус је на ненамерним грешкама (насталим као последица шума)

Кодови за грешке

- Кодна реч се састоји од D битова података и R контролних битова



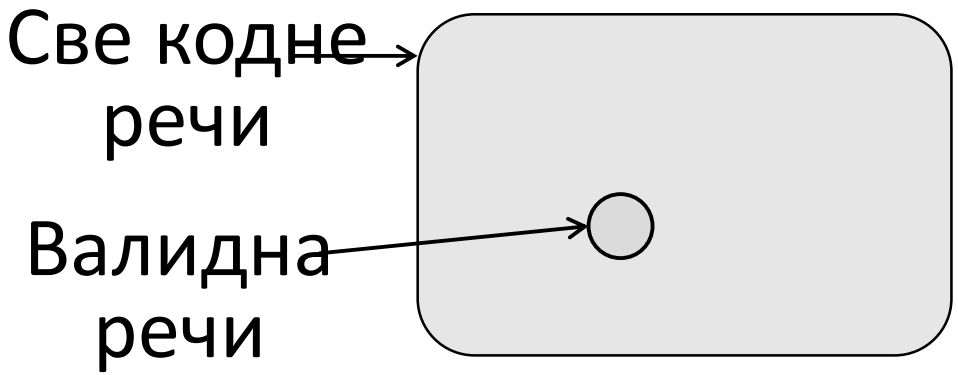
- Пошиљалац:
  - Израчуна R контролних битова као функцију од D и потом пошаље свих D+R битова

Кодови за грешке (2)

- Прималац:
  - Прихвата D+R са потенцијалним грешкама на било ком од битова
  - Рачуна R’ контролних битова на основу D по истом алгоритму као пошиљалац; грешка ако R није исто као R’

Интуиција иза кодова за грешке

- Скуп валидних кодних речи је доста мањи од скупа свих могућих



- Случајно одабрана реч има малу шансу да буде и валидна → мала шанса да ће се десити грешка и да ће након тога реч и даље бити валидна

Хамингово растојање

- Хамингово растојање је минималан број инверзија битова потребних да се од једне речи добије нека друга реч

- Хамингово растојање кода минимално Хамингово растојање између свих парова валидних кодних речи
- Обично су у делу за податке могуће све комбинације
- Док је скуп контролних битова ограничен

Хамингово растојање (2)

- Детекција грешака:
  - Да би се поуздано открило до d грешака, Хамингово растојање кода мора бити најмање d+1
  - Тада је немогуће да d једнобитних грешака промене валидну кодну реч у неку другу валидну кодну реч

Хамингово растојање (3)

- Корекција грешака:
  - За код са Хаминговим растојањем 2d+1, највише d грешака се увек може исправити до најближе исправне валидне речи
- Пример кода са валидним скупом:
  - 0000000000, 0000011111, 1111100000 и 1111111111
  - Може се исправити било која двобитна грешка, нпр. за 0000000111?
  - Ако очекујемо да се може јавити и тробитна грешка, онда 0000000111 може бити и кодна реч 0000000000?

Слој везе

Детекција грешака

Тема

- Само откривање да ли грешка постоји?
  - Провера парности
  - Контролни збирови
  - Цикличне провере редундансе CRC

- Детекција омогућава само индиректну поправку, јер захтева ретрансмисију (поновно слање)

Провера парности

- За D битова података, додамо 1 контролни бит који представља суму битова података
  - Сума је без преноса, односно сума по модулу 2

Провера парности (2)

- Колико добро ова техника ради?
  - Колико је Хамингово растојање кода? (односно колики је минималан број инверзија да би од једне валидне речи добили исто валидну)
  - Колико грешака може да детектује/исправи?
- Шта је са већим (рафалним) грешкама, која је вероватноћа да ћемо њу открити?
  - Ове грешке се јављају са високом учесталости на узастопним позицијама

Контролни збирови

- Идеја: сумирање података по колонама за N-битне речи
  - Често се користи у TCP/IP/UDP

1500 бајтова	16 битова

- Боља детекција него код провере парности, посебно у случају рафалних грешака

Интернет контролни збир

- Рачуна се у аритметици непотпуног комплемента
  - Две репрезентације нуле
  - Једну од нула можемо користити да означимо да контролни збир не постоји
- Контролни збир има 16 битова и представља непотпуни комплемент суме речи по колонама

Интернет контролни збир (2)

Слање:

1. Сложити податке као 16 битне речи једну испод друге
2. Постављамо нуле на позиције контролног збира
3. Евентуални пренос са позиције највише тежине се преноси на почетак (овако се сабира у непотпуном комплементу)
4. Негирамо добијену суму и тиме добијамо контролни збир

```
  0001
  f203
  f4f5
  f6f7
+(0000)
------
 2ddf0

  ddf0
+    2
------
  ddf2

  220d
```

Интернет контролни збир (3)

Прихватање:

1. Сложити примљене податке као 16 битне речи једну испод друге (укључујући и реч за контролни збир)
2. Ако постоји пренос са позиције највише тежине, додамо га на најнижи бит
3. Негирамо резултат и проверимо да ли је 0; ако јесте, онда је све у ред

```
  0001
  f203
  f4f5
  f6f7
+ 220d
------
 2fffd
   ↓
  fffd
+    2
------
  ffff
   ↓
  0000
```

Цикличка провера редундансе (CRC)

- Још боља детекција
  - Секвенца битова се сматра полиномом чији су коефицијенти само нуле и јединице
  - За датих n битова података, генеришемо k контролних битова тако n+k битова буде број дељив са неким унапред одабраним полиномом односно бројем C
  - Број C се назива генератор и знају га и пошиљалац и прималац

CRC (2)

- Пример:
  - 10011010 је полином $x^7 + x^4 + x^3 + x^1$
  - Даље радимо са бинарним вредностима и аритметиком по модулу 2
    (игноришемо потенцијалне преносе са највише позиције)

CRC (3)

- Слање података:

1. Проширити n битова података са k нула
2. Поделити број са одабраним бројем C
3. Задржимо само остатак, количних нас не занима
4. Додамо остатак на проширени број

- Примање података:

1. Поделити поруку са бројем C и проверити да ли је остатак при дељењу једнак нули

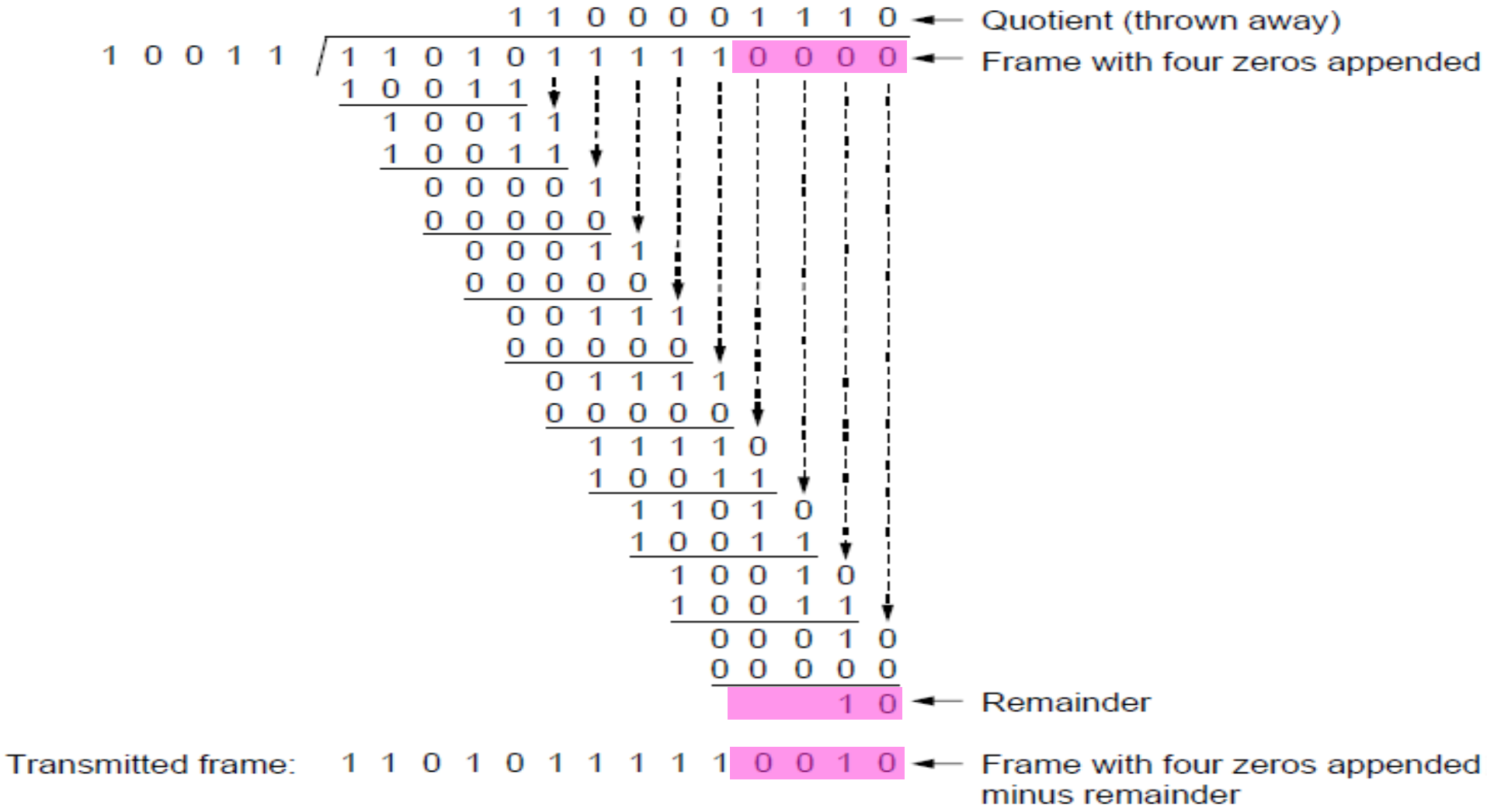
CRC (4)

Подаци:
1101011111

Делилац:
$C(x)=x^4+x^1+1$
C = 10011
k = 4

$$10011 \overline{)\,1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1}$$

CRC (5)


1 1 0 0 0 0 1 1 1 0
Quotient (thrown away)
1 0 0 1 1 | 1 1 0 1 0 1 1 1 1 1 0 0 0 0
Frame with four zeros appended
1 0 0 1 1
1 0 0 1 1
0 0 0 0 1
0 0 0 0 0
0 0 0 1 1
0 0 0 0 0
0 0 1 1 1
0 0 0 0 0
0 1 1 1 1
0 0 0 0 0
1 1 1 1 0
1 0 0 1 1
1 1 0 1 0
1 0 0 1 1
1 0 0 1 0
1 0 0 1 1
0 0 0 1 0
0 0 0 0 0
1 0
Remainder
Transmitted frame: 1 1 0 1 0 1 1 1 1 1 0 0 1 0
Frame with four zeros appended minus remainder

Детекција грешака у пракси

- Од генератора зависи и квалитет детекције (теорија бројева и емпиријска анализа, ...)
  - Стандардни CRC-32 користи 1 0000 0100 1100 0001 0001 1101 1011 0111
- CRC се често користе у:
  - Ethernet, 802.11, ADSL, Кабловска, …
- Контролни збир, иако лошији, користи се на вишим слојевима
  - IP, TCP, UDP …
- Провера парности
  - Слабо се користи

Слој везе

Корекција грешака (у припреми)

Тема

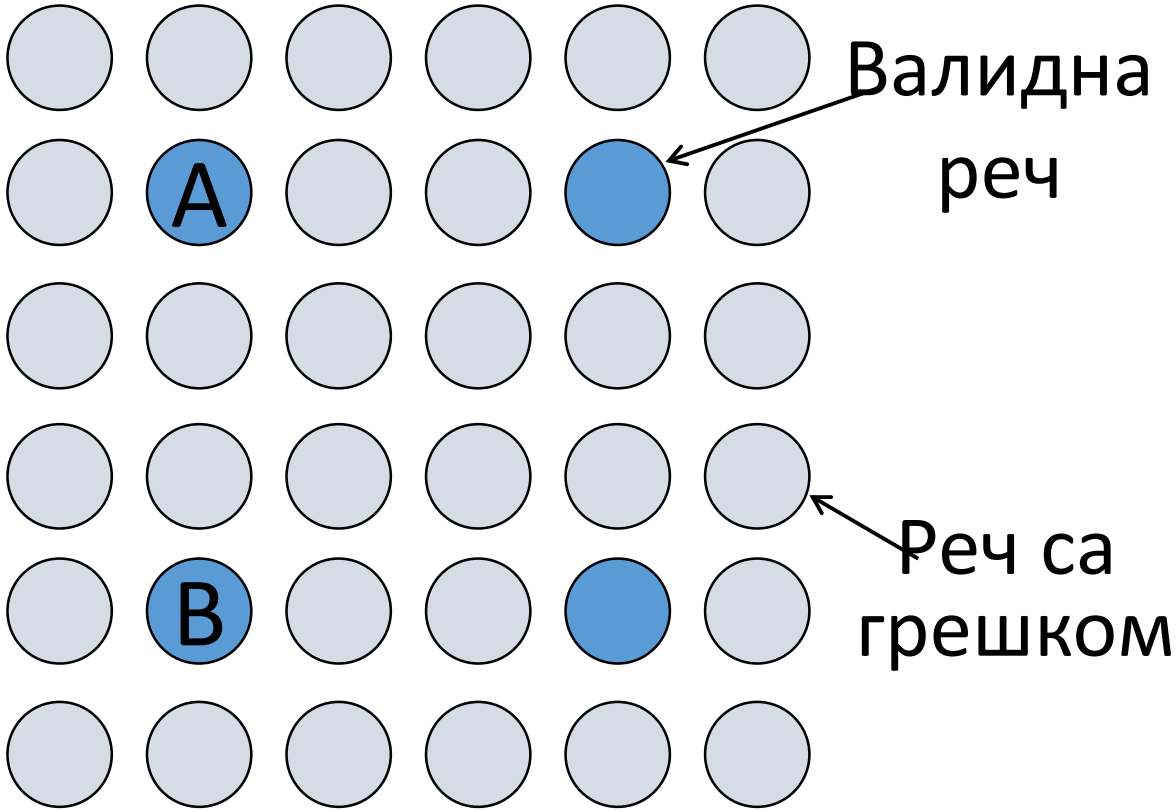
- Могуће је контруисати кодове који су у стању и да поправе грешку:
  - Хамингови кодови за корекцију
  - Остали кодови (нећемо обрадити)

- Шта ће нам корекција, ако можемо да детектујемо грешку и онда урадимо поновно слање (ретрансмисију)?

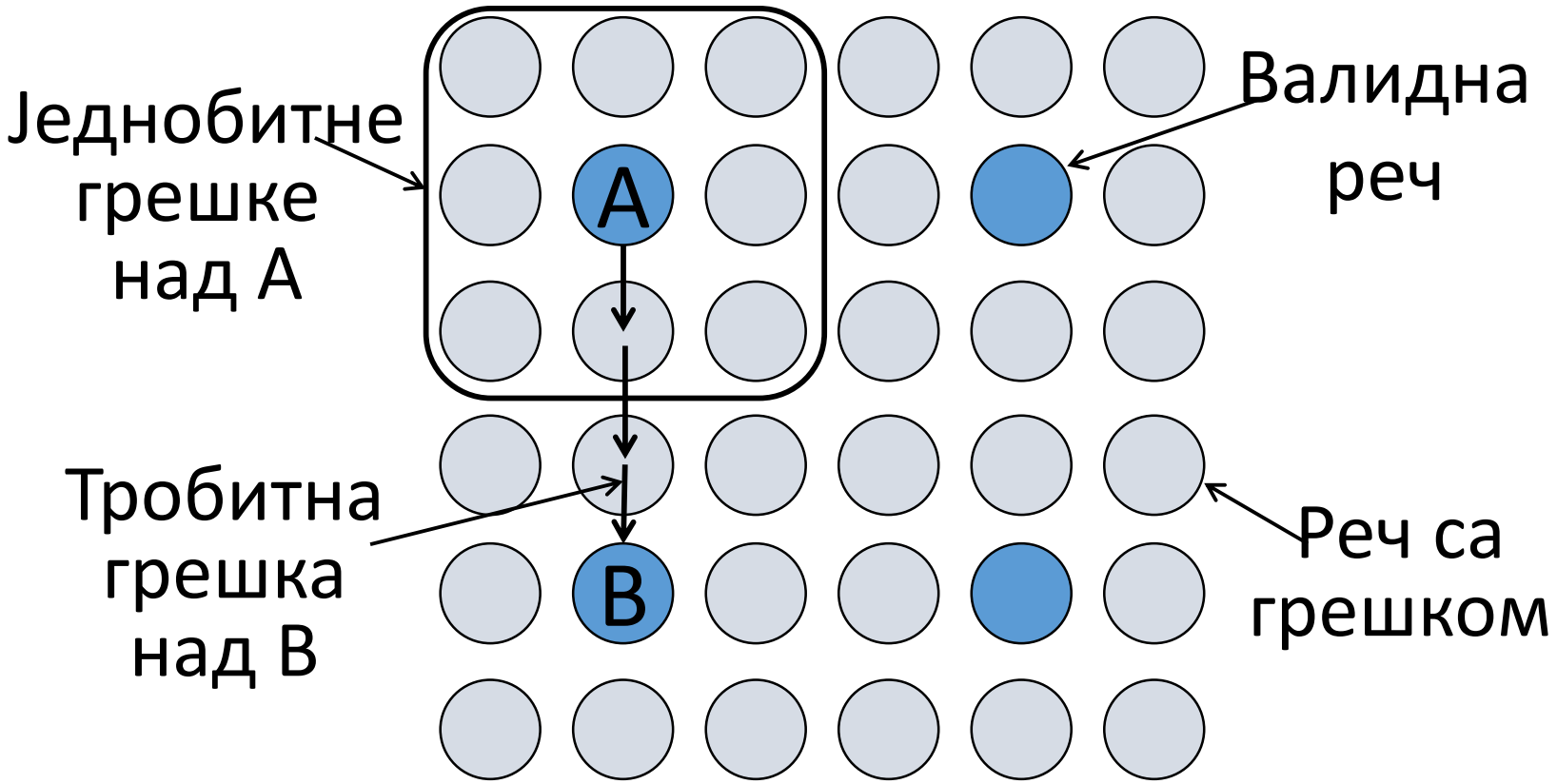
Интуиција иза кодова за корекцију грешака

- Ако имамо Хамингов код са растојањем најмање 3 (HD ≥3)
  - Потребно нам је ≥3 битова да пређемо из једне валидне речи у другу
  - Једнобитне грешке ће, дакле, бити ближе некој јединственој валидној речи
- Под претпоставком да се може десити само једнобитна грешка, можемо да исправимо реч тако што ћемо је мапирати у најближу валидну
  - Дакле, приступ ради ако HD ≥ 2d + 1, где је d максимални очекивани број једнобитних грешака

Интуиција (2)



Интуиција (3)


Једнобитне грешке над А
Валидна реч
А
Тробитна грешка над В
Реч са грешком
В

Хамингов код за корекцију

- Овај код има HD=3
  - Он користи n битова података и k контролних
  - Веза између број n и k је: $n = 2^k – k – 1$, нпр., n=4, k=3
  - Идеја је да се стави контролни бит на позиције које су степен двојке: 1, 2, 4, ...
  - Контролни бит на позицији (i) се даље рачуна као бит парности за битове чије позиције у бинарној репрезентацији имају 1 на (i)-том месту

Хамингов код за корекцију (2)

- Пример: подаци=0101
- Потребна су 3 контролна бита (позиције 1, 2 и 4)
- 7-битни код
  - Бит 1 покрива позиције 1, 3, 5, 7 (1=000**1**, 3=001**1**,…)
  - Бит 2 покрива позиције 2, 3, 6, 7 (2=00**1**0, 3=00**1**1, …)
  - Бит 4 покрива позиције 4, 5, 6, 7 (4=0**1**00, 5=0**1**01, …)

Хамингов код за корекцију (3)

- Пример: подаци=0101
- Потребна су 3 контролна бита (позиције 1, 2 и 4)
- 7-битни код
  - Бит 1 покрива позиције 1, 3, 5, 7 (1=000**1**, 3=001**1**,…)
  - Бит 2 покрива позиције 2, 3, 6, 7 (2=00**1**0, 3=00**1**1, …)
  - Бит 4 покрива позиције 4, 5, 6, 7 (4=0**1**00, 5=0**1**01, …)

0	1	0	0	1	0	1
1	2	3	4	5	6	7

p_1= 0+1+1 = 0, p_2= 0+0+1 = 1, p_4= 1+0+1 = 0

Хамингов код за корекцију (4)

Декодирање:

- Након прихватања, прималац ради исти процес
  - поново рачуна контролне битове као функцију битова података
- Након тога сложи контролне битове као бинарни број (ово се зове синдром)
- Тај број ће открити позицију грешке
- Корекција подразумева само комплементирање бита на тој позицији

Хамингов код за корекцију (5)

$\longrightarrow$ 0 1 0 0 1 0 1

1 2 3 4 5 6 7

p_1=

p_2=

p_4=

синдром=

податак=

Хамингов код за корекцију (6)

$\longrightarrow$ 0 1 0 0 1 0 1

1 2 3 4 5 6 7

p_1= 0+0+1+1 = 0, p_2= 1+0+0+1 = 0,

p_4= 0+1+0+1 = 0

синдром = 000, нема грешке

податак= 0 1 0 1

Хамингов код за корекцију (7)

$\longrightarrow$ 0 1 0 0 1 1 1

1 2 3 4 5 6 7

p_1=

p_2=

p_4=

синдром=

податак=

Хамингов код за корекцију (8)

$\longrightarrow$ 0 1 0 0 1 1 1

1 2 3 4 5 6 7

p_1= 0+0+1+1 = 0, p_2= 1+0+1+1 = 1,

p_4= 0+1+1+1 = 1

синдром= 1 1 0, комплементирати бит 6

податак= 0 1 0 1 (коректно након комплементирања!)

Кодови у пракси

- У пракси се Хамингови кодови ретко користе, а у употреби су најчешће:
  - Конволуциони кодови
  - Метода парности за малу густину (LDPC – low density parity check)
  - Рид-Соломонови кодови
- Сви наведени кодови су ван домена курса!

Детекција или корекција?

- Корекција грешака:
  - Обично када су грешке очекиване
  - Или када нема времена за ретрансмисију
  - Најчешће се ради на физичком слоју
- Детекција грешака:
  - Ефикаснија када грешке нису очекиване, тј. када су ретке
  - А када се десе, онда нема везе и ако су велике
  - Обично се користи у слоју везе у комбинацији са ретрансмисијом оквира
- Дистрибуција грешке игра битну улогу:
  - Нпр. Ако је вероватноћа грешке 0.001, то може да значи:
    - 1-битна грешка на сваких 1000 битова
    - 100-битна (рафална) грешка на сваких 100000 битова
  - Када је боља детекција, а када је боља корекција ако је нпр. величина оквира 100 битова?

Слој везе

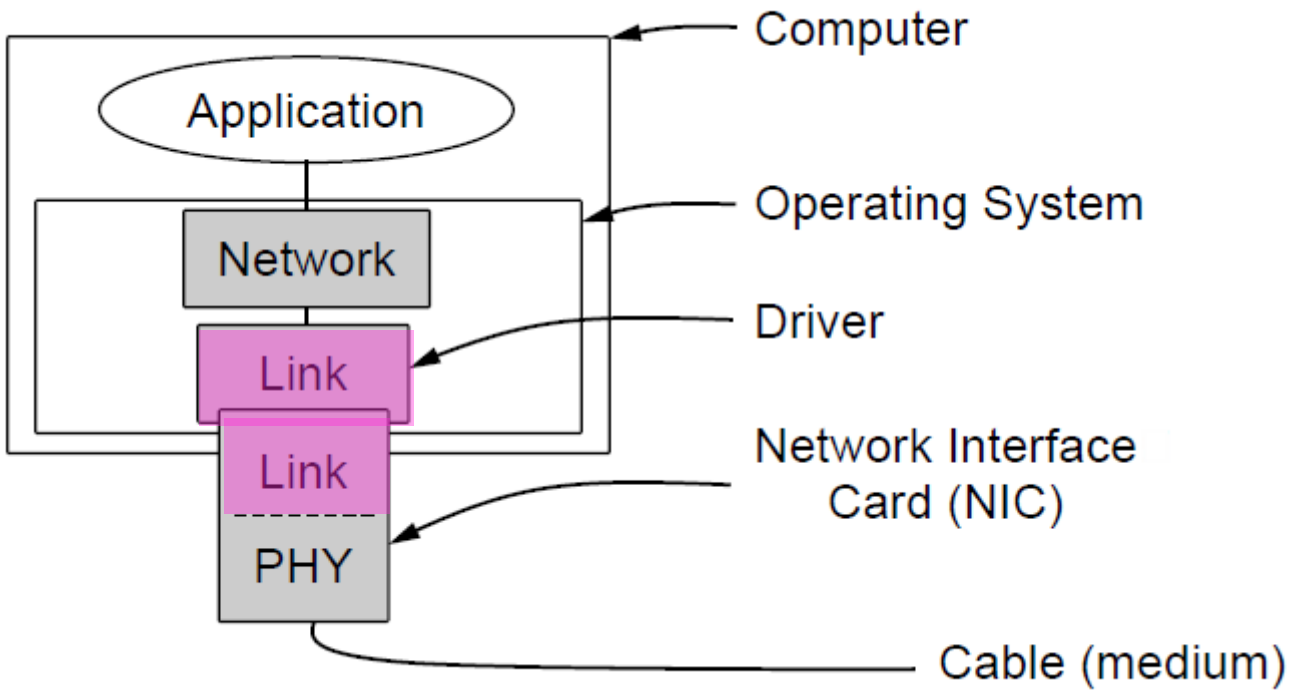
Ретрансмисија и контрола тока

Типови сервиса

- Сервис без успоставе везе и без потврде пријема
  - Оквир се шаље независно и без ретрансмисије у случају грешке
  - Пример је Етернет
- Сервис без успоставе везе са потврдом пријема
  - Ради се ретрансмисија ако се јави потреба
  - Пример је WiFi
- Сервис са успоставом везе и са потврдом пријема
  - Успостава везе омогућава да подаци теку истим редом којим су и послати
  - Ретко се користи

Окружење у слоју везе (1)

- Овај слој је обично реализован делом на мрежној картици, а другим делом на нивоу оперативног система



Окружење у слоју везе (2)

- Преглед неких функција за интеракцију слоја везе са слојем испод и изнад:

Група	**Функција**	**Опис**
Мрежни слој	from_network_layer(&packet) to_network_layer(&packet)	Узима пакет из мрежног слоја Прослеђује пакет мрежном слоју
Физички слој	from_physical_layer(&frame) to_physical_layer(&frame)	Прихвата оквир из физичког слоја Прослеђује оквир ка физичком слоју
Догађаји & тајмери	wait_for_event(&event) start_timer(seq_nr) stop_timer(seq_nr) start_ack_timer() stop_ack_timer()	Чека на пакет/оквир/истек тајмера Покреће тајмер Прекида тајмер Покреће тајмер аз оквир потврде ACK Зауставља тајмер за оквир потврде ACK

Основни протоколи слоја везе

- Утопијски једносмерни протокол
- „Стани и чекај“ протокол за канал без грешака
- „Стани и чекај“ протокол за канал са грешкама

Утопијски једносмерни протокол

- Оптимистички протокол (Етернет заправо ово ради!)
  - Не предвиђа појаву грешке
  - Прималац је брз као и пошиљалац
  - Пренос података је једносмеран

```
void sender1(void)
{
 frame s;
 packet buffer;

 while (true) {
      from_network_layer(&buffer);
      s.info = buffer;
      to_physical_layer(&s);
 }
}
```

Пошиљалац шаље оквире стално

```
void receiver1(void)
{
 frame r;
 event_type event;

 while (true) {
      wait_for_event(&event);
      from_physical_layer(&r);
      to_network_layer(&r.info);
 }
}
```

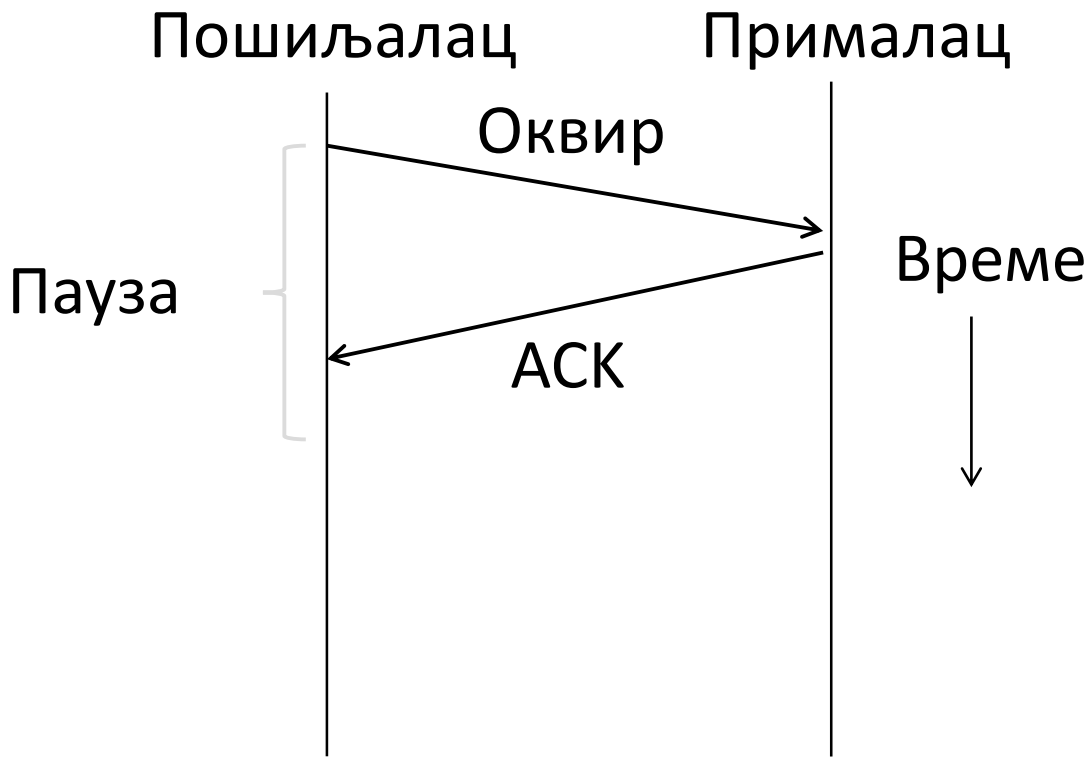
Прималац је у стању да их све прихвати

Контрола тока

- Шта ако пошиљалац и прималац немају исте брзине слања односно примања?
  - Овде је потребна некаква контрола тока

Протокол са контролом тока

- Различите брзине пошиљаоца и примаоца
- Савршен канал (нема грешака нити изгубљених оквира)



Протокол „стани и чекај“ – за савршен канал

- Овај протокол гарантује усаглашеност у брзини комуникације
  - Прималац шаље празан оквир (ack) када је спреман да настави
  - Шаље се оквир по оквир (неефикасно)

```
void sender2(void)
{
  frame s;
  packet buffer;
  event_type event;

  while (true) {
       from_network_layer(&buffer);
       s.info = buffer;
       to_physical_layer(&s);
       wait_for_event(&event);
  }
}
```

Пошиљалац чека на ack
након слања оквира

```
void receiver2(void)
{
  frame r, s;
  event_type event;
  while (true) {
       wait_for_event(&event);
       from_physical_layer(&r);
       to_network_layer(&r.info);
       to_physical_layer(&s);
  }
}
```

Прималац шаље ack
након што прихвати оквир

Несавршен канал

- Шта ако канал није савршен, тј. ако се могу појавити грешке или изгубљени оквири?
  - Детекција и ретрансмисија ARQ (Automatic Repeat reQuest)
  - Корекција грешака - обрађено, ово се више тиче физичког слоја

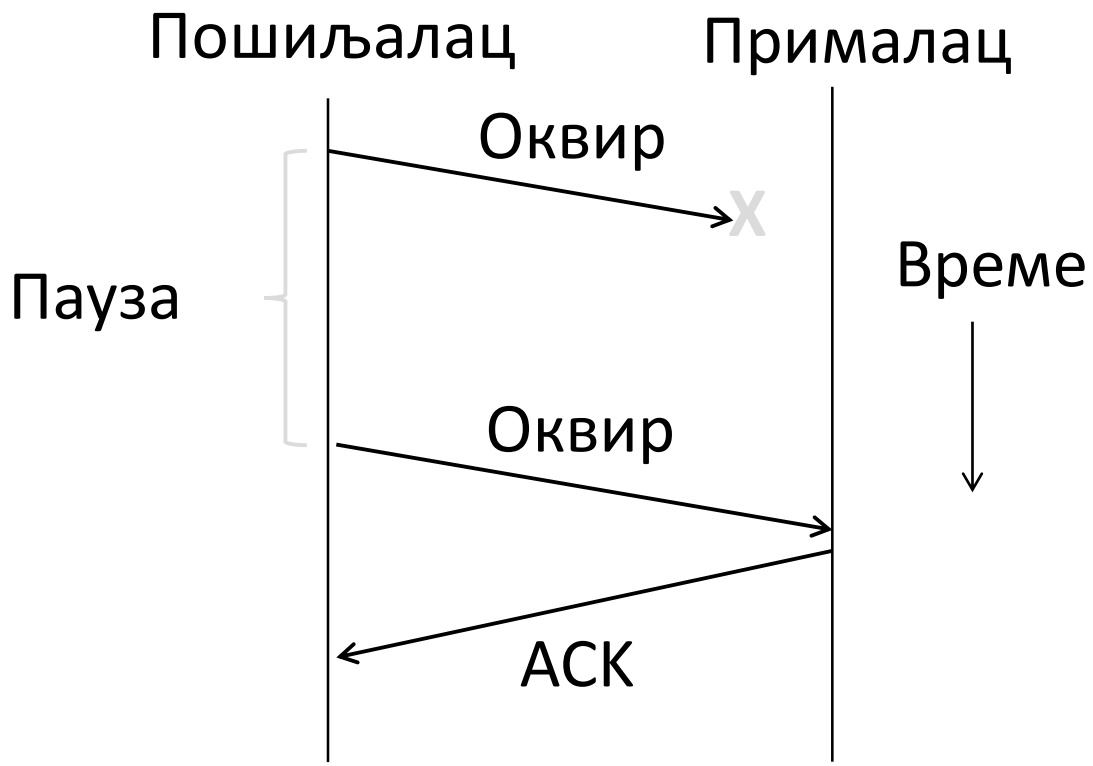
ARQ

- ARQ се обично користи када су грешке уобичајне и када се морају исправити
  - Нпр. WiFi, и TCP (касније)

- Основна идеја:
  - Прималац шаље потврду о пријему исправног оквира ACK
    - ACK је такође оквир
  - Пошиљалац аутоматски шаље поново након временске паузе (тајмаута), осим ако у међувремену пристигне ACK

ARQ кроз несавршен канал

- Сценарио са губитком и ретрансмисијом



Кључна питања у вези ARQ?

- Две битне одлуке:
  - Колика треба да буде пауза (време на тајмеру)?
  - Како да се избегне интерпретирање дуплираних оквира као нових?

- Основни циљ је коректност
  - исправни оквири, без грешака и дупликата
- Али желимо и да комуникација буде ефикасна

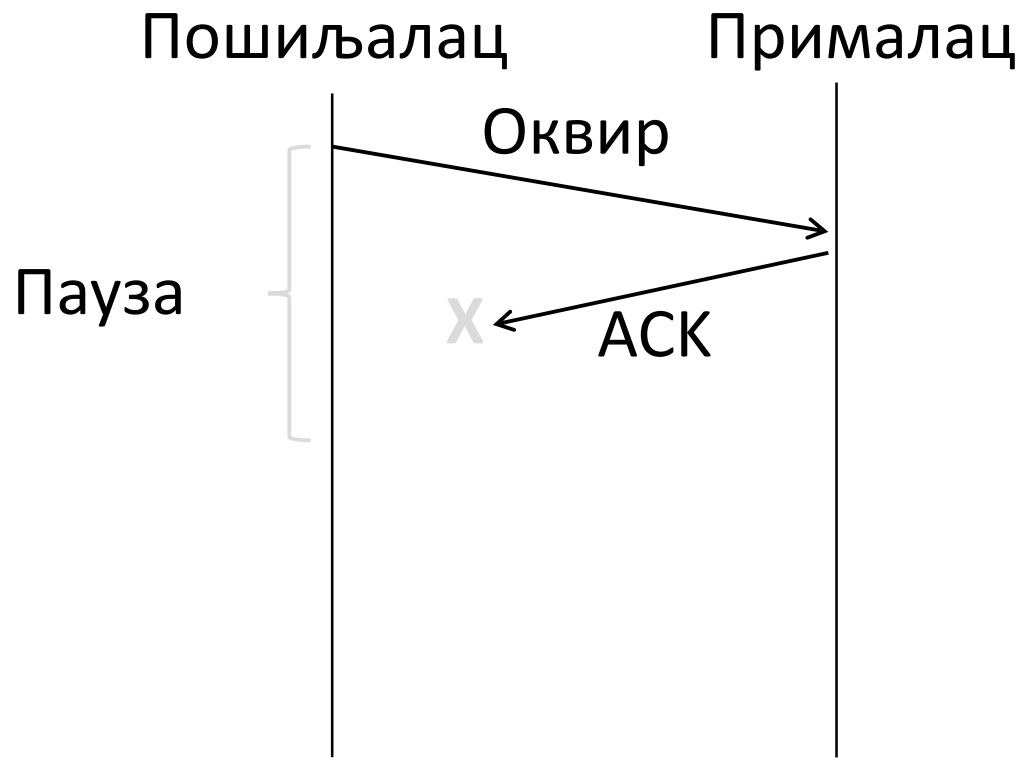
Паузе (тајмаути)

- Пауза треба да буде:
  - Не превише велика (неискоришћеност канала)
  - Не премала (непотребне трансмисије)

- Дужину паузе релативно једноставно одредити за LAN
  - Јасна је горња граница, мало одступање
- Тешко за Интернет
  - Велико одступање, нема јасне горње границе

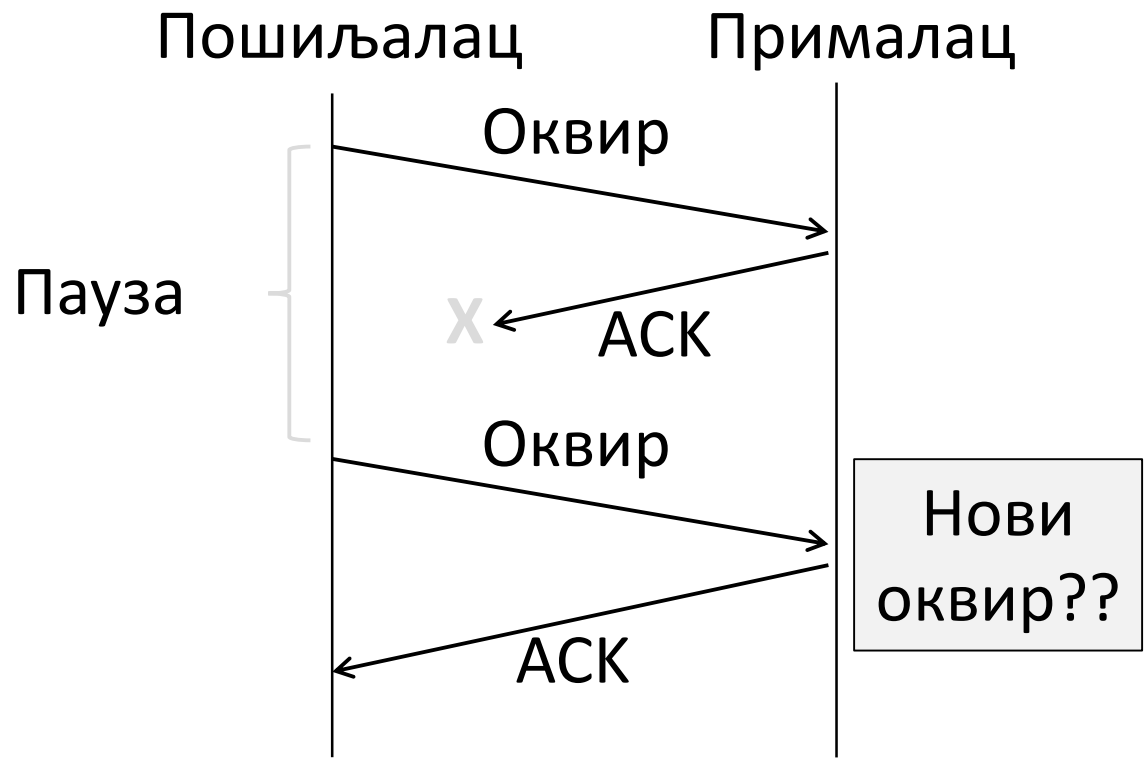
Дупликати

- Шта ако се АСК изгуби?



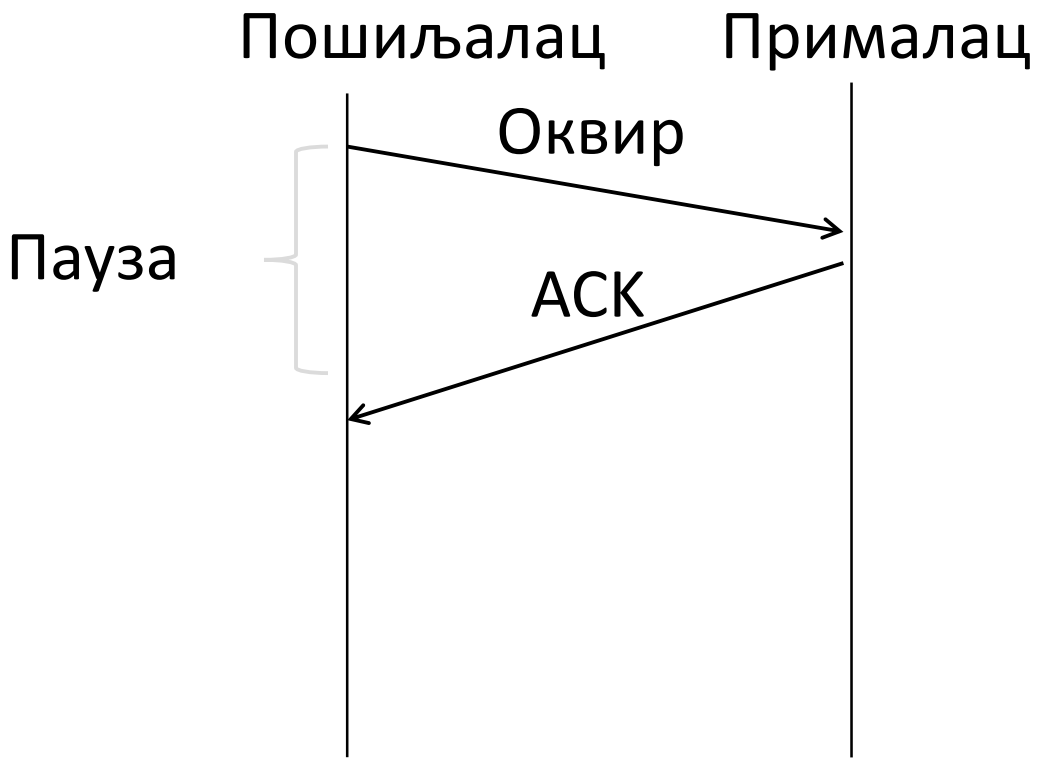
Дупликати (2)

- Шта ако се ACK изгуби?



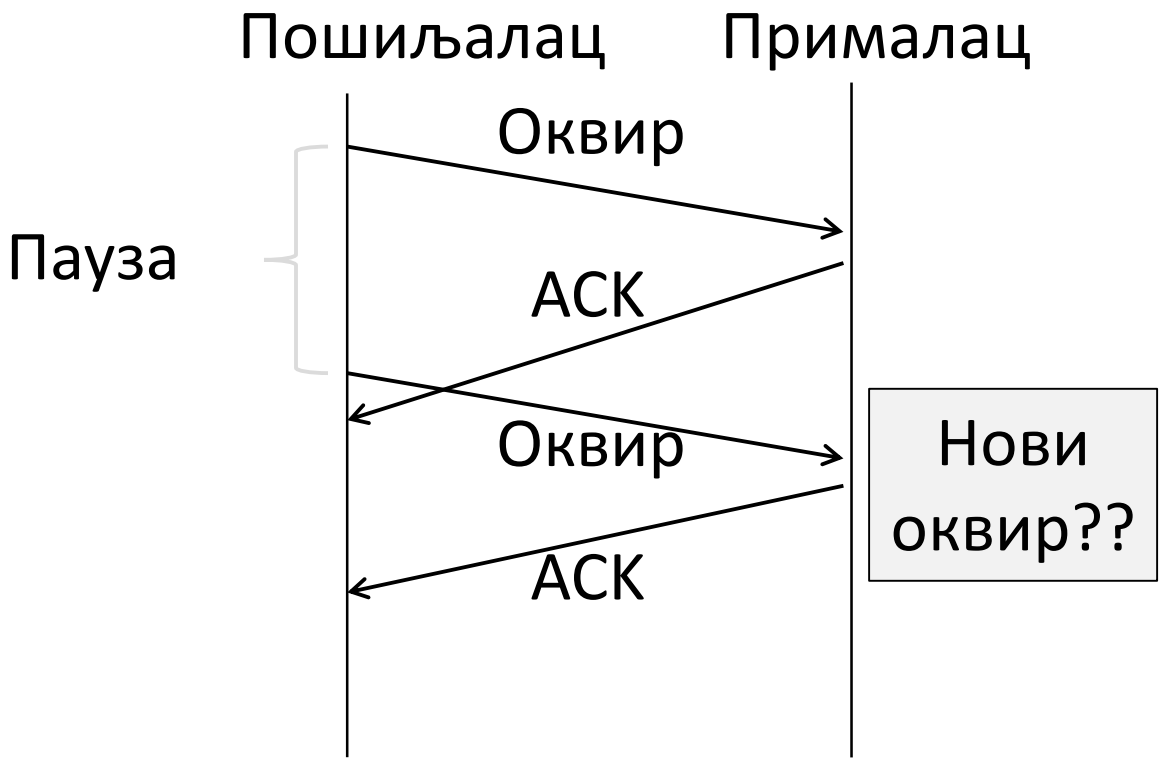
Дупликати (3)

- Или ако је пауза прекратка?



Дупликати (4)

- Или ако је пауза прекратка?



Индикатори оквира

- Оквири и ACK-ови морају да носе са собом ознаку како би се избегли дупликати
- Довољно је само разликовати тренутни оквир од наредног оквира (један бит)

Протокол „стани и чекај“ за несавршен канал

Пошиљалац:

```
void sender3(void) {
  seq_nr next_frame_to_send;
  frame s;
  packet buffer;
  event_type event;

  next_frame_to_send = 0;
  from_network_layer(&buffer);
  while (true) {
      s.info = buffer;
      s.seq = next_frame_to_send;
      to_physical_layer(&s);
      start_timer(s.seq);
      wait_for_event(&event);
      if (event == frame_arrival) {
          from_physical_layer(&s);
          if (s.ack == next_frame_to_send) {
              stop_timer(s.ack);
              from_network_layer(&buffer);
              inc(next_frame_to_send);
          }
      }
  }
}
```

Шаљи нови оквир (или ретрансмисију)

Активирај тајмер за ретрансмисију

Чекај на догађај

Ако је стигла потврда и добра је, узми нови оквир од мрежног слоја.
У супротном, остави исти оквир за ретрансмисију.

Протокол „стани и чекај“ за несавршен канал (2)

Прималац:

```
void receiver3(void)
{
  seq_nr frame_expected;
  frame r, s;
  event_type event;

  frame_expected = 0;
  while (true) {
      wait_for_event(&event);
      if (event == frame_arrival) {
          from_physical_layer(&r);
          if (r.seq == frame_expected) {
              to_network_layer(&r.info);
              inc(frame_expected);
          }
          s.ack = 1 – frame_expected;
          to_physical_layer(&s);
      }
  }
}
```

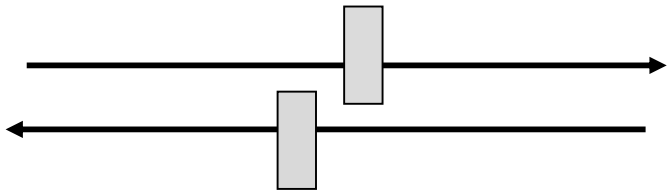
Чекај на оквир

Ако је онај који очекујеш, узми га и проследи мрежном слоју.

И пошаљи потврду за тренутни оквир

Ограничења „стани и чекај“ протокола

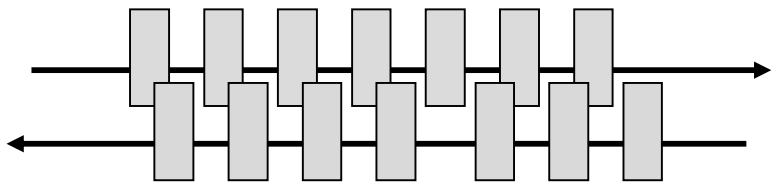
- Омогућен пренос само једног оквира дуж канал у једном тренутку:
  - Прихватљиво за LAN, али неефикасно за висок BDP



- Пример: B=1 Mb/s, D = 50 ms
  - Колико оквира у секунди се пренесе?
  - Колико оквира у секунди ако је B=10 Mbps?

Клизни (померајући) прозори

- Уопштење протокола „стани и чекај“
  - Омогућава да у сваком моменту W оквира буде у каналу
  - W оквира за RTT (round trip time) (=2D)
  - RTT је време потребно да оквир оде и да стигне потврда за њега



Протоколи „клизних прозора“

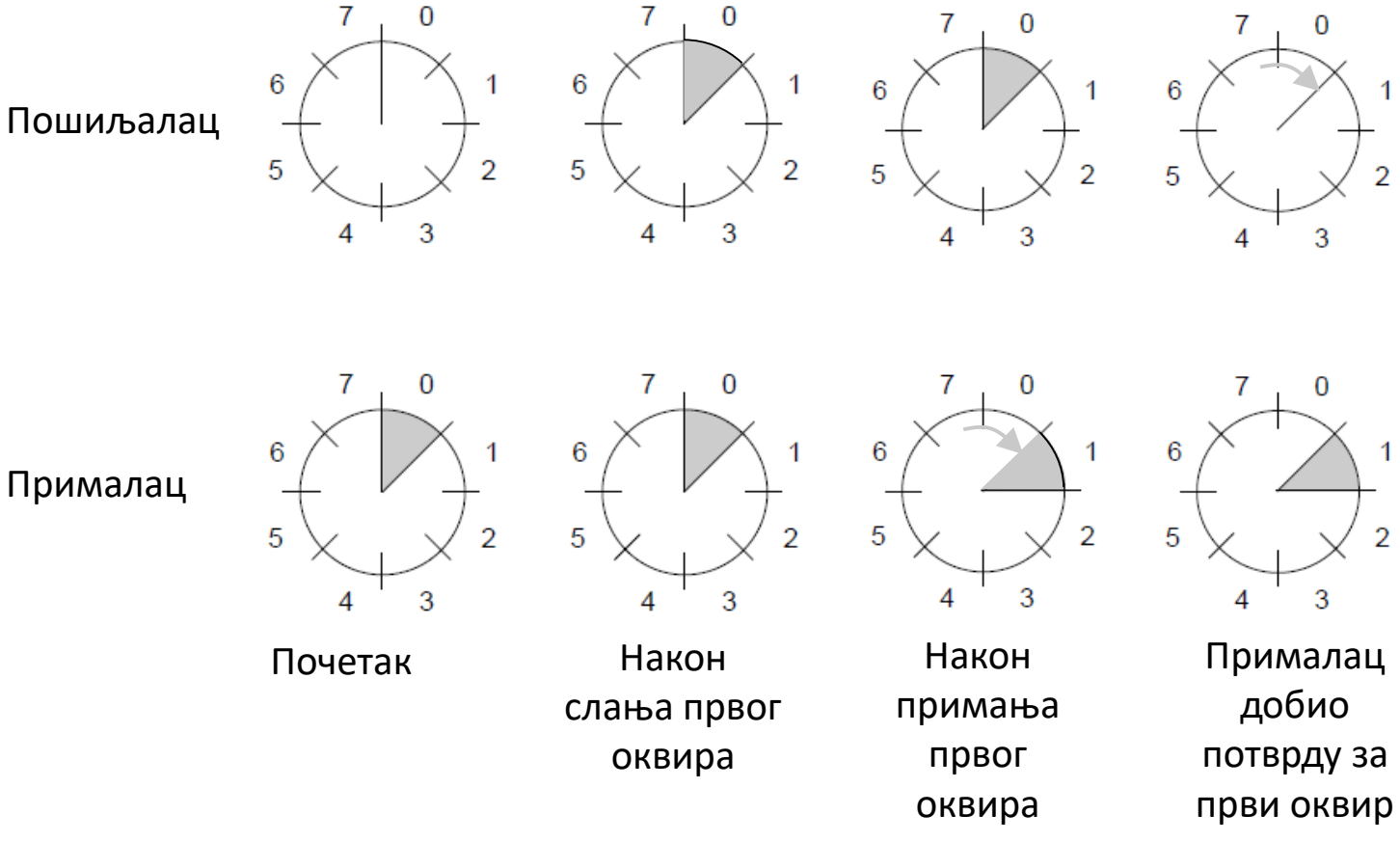
- Концепт „клизних прозора“
- 1-битни протокол клизних прозора
- „Врати се N“ протокол
- Протокол са селективним понављањем

Концепт клизних прозора

- Пошиљалац има на располагању неколико (W) узастопних оквира које може да пошаље
  - Мора да их баферује због евентуалне ретрансмисије
  - Са пристиглим ACK-овима пошиљалац помера списак узастопних расположивих оквира (помера прозор)
- Прималац има на располагању простор за неколико оквира које може да прихвати
  - Мора да поседује бафере за њих
  - Како стижу нови оквири, „прозор“ се помера

Концепт клизних прозора (2)

- Напредовање клизног прозора на страни пошиљаоца и примаоца
  - Пример: величина прозора је 1, а бројеви оквира 0-7 (3 бита)



Концепт клизних прозора (3)

- Већи прозори омогућавају проточну одбрану за ефикаснију употребу канала
  - Стани и чекај (W=1) је неефикасан, посебно за дуже канале
  - Оптималан (W) зависи од BDP
  - Желимо да је $W \geq 2BD+1$ у циљу што бољег искоришћења

- Различити начини за разрешавање грешака и за начине баферисања:
  - Размотрићемо „Врати се N“ протокол и протокол са селективним понављањем

1-битни протокол клизних прозора

- Нема одвојених алгоритама за пошиљаоца и примаоца, јер сада и један и други могу да шаљу и примају
  - За ACK користимо оквир из супротног правца („шлепање“)
  - Ради над несавршеним каналом
  - Омогућава контролу тока

```
void protocol4 (void) {
  seq_nr next_frame_to_send;
  seq_nr frame_expected;
  frame r, s;
  packet buffer;
  event_type event;
  next_frame_to_send = 0;
  frame_expected = 0;
  from_network_layer(&buffer);
  s.info = buffer;
  s.seq = next_frame_to_send;
  s.ack = 1 – frame_expected;
  to_physical_layer(&s);
  start_timer(s.seq);
```

Припрема оквира

Слање и постављање тајмера

. . .

1-битни протокол клизних прозора (2)

Чекамо на догађај

Ако је очекивани оквир са подацима, шаљи га мрежном слоју

Ако је АСК за последњи послати оквир, припреми наредни оквир за слање

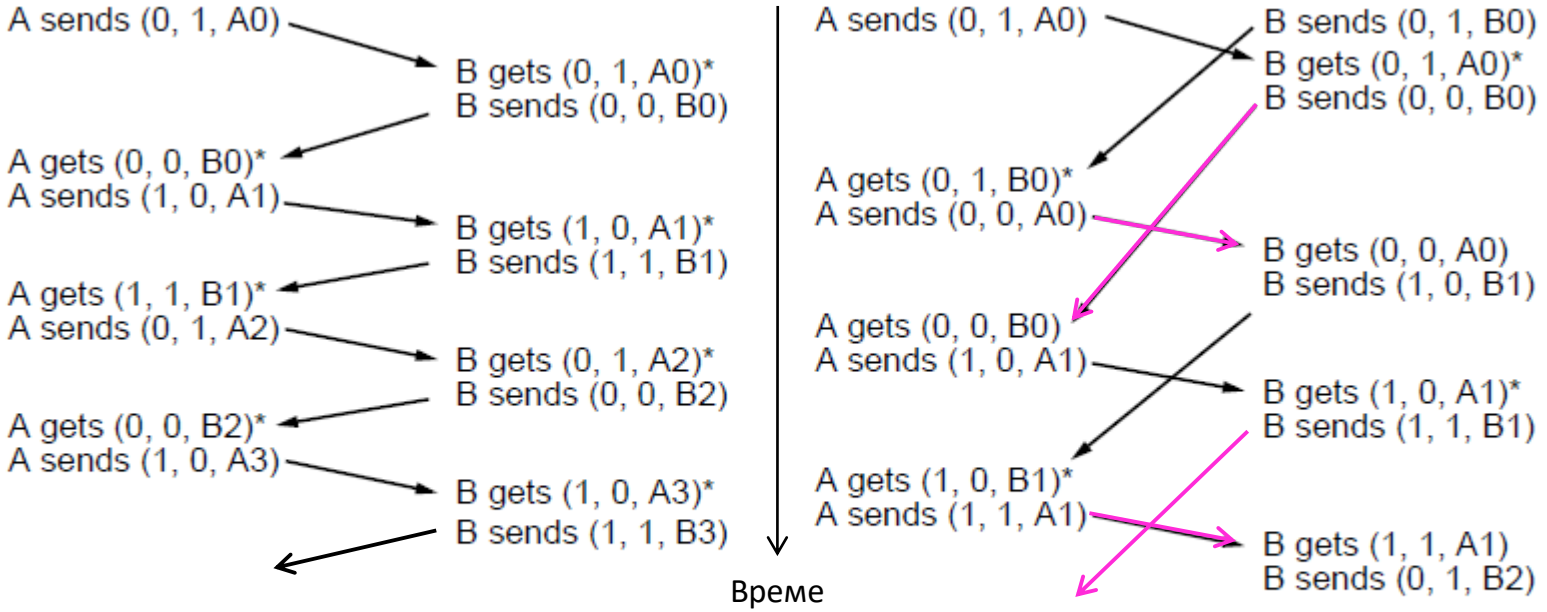
(Иначе је догађај био тајмоут)

Пошаљи следећи оквир. Ово ће бити нови оквир ако је у међувремену стигао АСК, иначе ће бити исти (ретрансмисија)

```
while (true) {
    wait_for_event(&event);
    if (event == frame_arrival) {
        from_physical_layer(&r);
        if (r.seq == frame_expected) {
            to_network_layer(&r.info);
            inc(frame_expected);
        }
        if (r.ack == next_frame_to_send) {
            stop_timer(r.ack);
            from_network_layer(&buffer);
            inc(next_frame_to_send);
        }
    }
    s.info = buffer;
    s.seq = next_frame_to_send;
    s.ack = 1 – frame_expected;
    to_physical_layer(&s);
    start_timer(s.seq);
  }
}
```

1-битни протокол клизних прозора(3)

- Ради коректно...
  - Али може да дође до успорења ако сви започну истовремено.



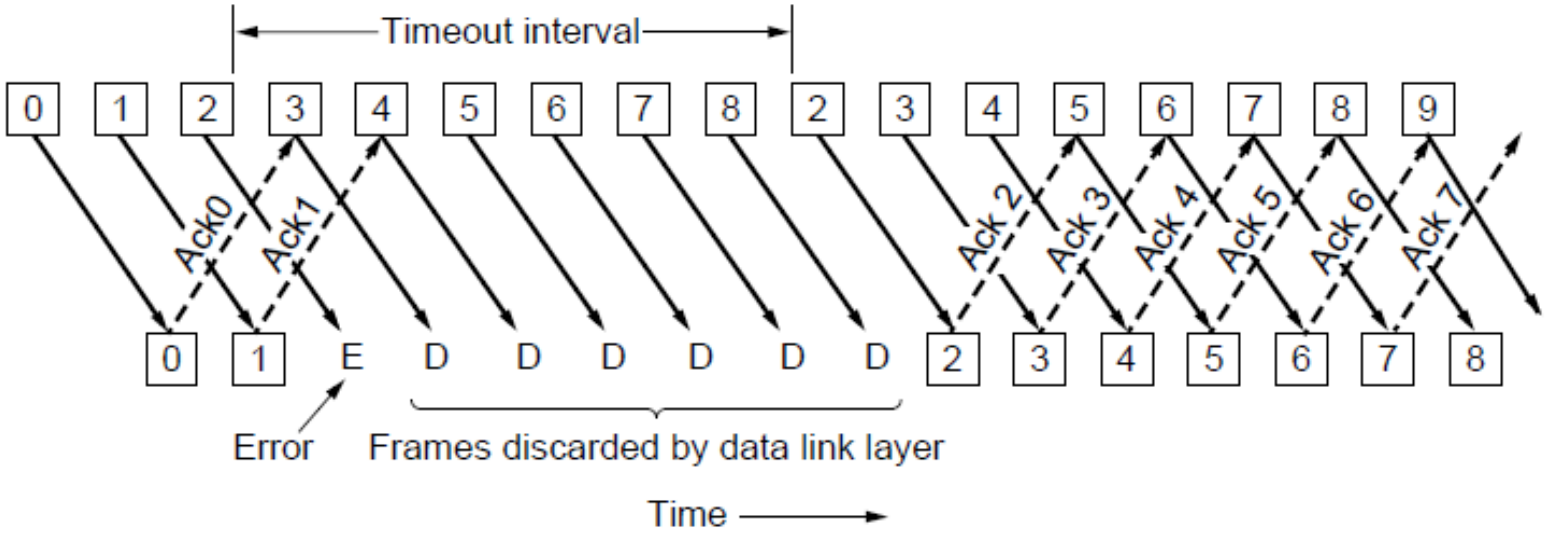
Нотација (индикатор пакета,индикатор за ACK, број оквира). (*) означава да је оквир успешно послат ка мрежном слоју

Нормалан сценарио

Сценарио са успорењем

„Врати се N“ протокол (1)

- Прималац прихвата само оквире који стижу редом:
  - Притом одбацује све који су уследили након спорног
  - Примаоцу истиче у неком моменту тајмоут и он шаље све непотврђене оквире до краја прозора поново

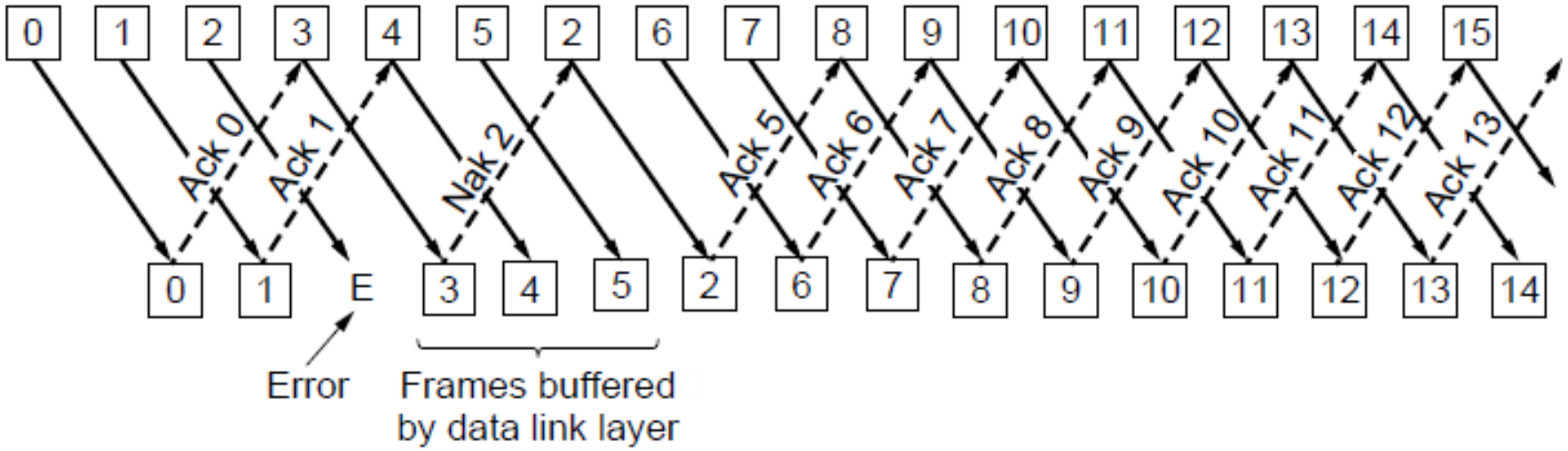


„Врати се N“ протокол (2)

- Карактеристике::
  - Једноставан алгоритам на страни примаоца, потребан му је бафер за само један оквир
  - Непотребно трошење протока у случају великих прозора, тада, у најгорем случају комплетни прозори (W оквира) морају да се шаљу поново

Протокол селективног понављања

- Прималац прихвата оквир све док је редни број оквира у опсегу дефинисаном клизним прозором
  - NAK (негативни ACK) информише пошиљаоца да поново пошаље проблематични оквир (зарад ефикасности), пре истека тајмаута за цео прозор



Протокол селективног понављања (2)

- Карактеристике:
  - Сложенији за имплементацију од „Врати се N“ због баферисања при примању и вишеструким тајмерима при слању
  - Ефикаснија употреба протока, јер се сада само изгубљени оквиру поново шаљу

Протокол селективног понављања (3)

- Додатни захтев:
  - Опсег бројева оквира (S) мора бити најмање два пута већи од величине прозора (W) како би се избегли проблеми са ретрансмисијом оквира