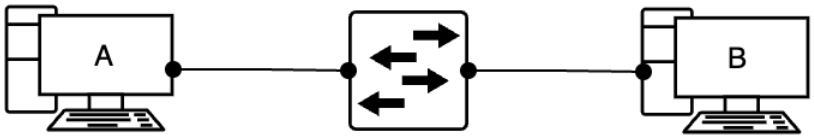
3. Контрола тока

(3.3 Основни протоколи слоја везе података, стр.209-237)

Како је раније наглашено, *Слој везе података* (*Data Link Layer)* задужен је за предају и пријем оквира (frames) између два учесника у рачунарској комуникацији. Поред уоквиравања и контроле грешака, које као прву етапу у пријему/предаји оквира врши овај слој, **контрола тока** је наредни корак.

Контрола тока (flow control) у рачунарским мрежама на **L2 слоју** односи се на механизме којима се регулише проток података између два директно повезана уређаја (нпр. два switch-а или рачунара) како би се избегло **загушење пријемника**, односно ситуација када пошиљалац шаље више података него што прималац може да обради или меморише. Размотрићемо кроз следећи пример:

Уколико рачунар А шаље оквире рачунару Б много већом брзином него што рачунар В може да обради оквире, доћиће да губитка одређеног броја оквира а самим тим и губитак синхронизације између страна у комуникацији што даље доводи до преноса непотпуних података.



Да би се избегли проблеми у преносу оквира, обе стране су уобавези да усагласе контролу тока пре него што започну међусобну комуникацију. Из претходно изложеног може се наслутити да је контрола тока у првом реду договор око брзине предаје/пријема оквира у коме прималац информише пошиљоца којом брзином је спреман да обрађује оквире након чега пошиљалац усалгашава брзину предаје оквира са брзином обраде оквира на страни примаоца.

Друга битна ставка јесте договор око количине података које пошиљалац треба да пошаље пре него што добије потврду о пријему послатих оквира. Ово значи да контрола тока представља низ процедура које пошиљалац треба да уважи у вези са емитовањем количине оквира и временском интервалу у коме се очекује потврда пријема послатих оквира.

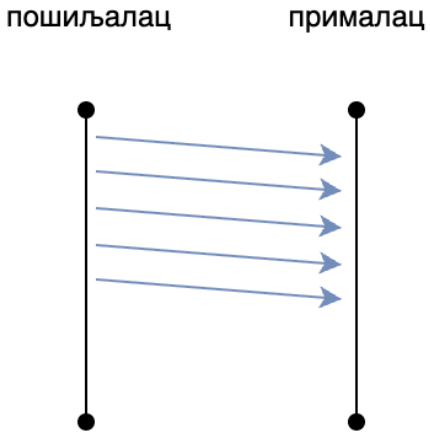
Суштински, у процесу контроле тока, фокус се поставља на примаоца. Примаоц који може да обрађује оквире брже него пошиљалац, не представља никакав проблем, али обрнуто, итекако имамо озбиљно нарушавање комуникације међу учесницима.

3.1 Подела протокола

Протоколи који се реализују на слоју везе података а тичу се контроле тока начелно се деле у две групе у зависности од типа комуникациног канала:

- протоколи у бешумним каналима (комуникациони канали су идеални, нема никаквих сметњи током преноса оквира)

- протоколи у бучним каналима (у комуникационим каналима уноси се одређени шум, тј, сметња која утиче на пренос оквира)



Протокол за неограничен једносмеран пренос је протокол који се ралазује у бешумном каналу. Ово је најједноставнији протокол зато што не води рачуна о могућим грешкама. Оквири се преносе само у једном смеру. Скица која описује дати протокол дата је у следећем примеру:

Дакле, подаци се преносе само у једном смеру на релацији пошиљалац→ прималац, података за пренос има неограничено и сви су спреми за слање, време обраде података је занемарено, комуникациони канал никада не изобличава оквире нити их губи. Ово јесте нереалан протокол, али је добра полазна основа за оно што будемо обрађивали у наредним одељцима.

Програмски к0д који би описао претходни протокол дат је на следећем примеру:

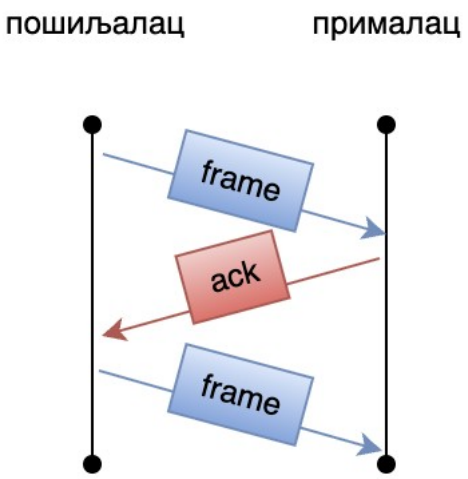
```
void sender(void) {
	frame s;				/*складиште за одлазни оквир*/
	packet buffer;				/*складиште за одлазни пакет*/
	while(true) {
		from_network_layer(&buffer); /*преузми пакет са мрежног слоја*/
		s.info = buffer;		/*уметни пакет у оквир*/
		to_physical_layer(&s);		/*проследи оквир мрежном слоју*/
	}
}

void receiver(void) {
	frame r;				/*складиште за долазни оквир*/
	event_type event;			/*догађај – оквир је стигао*/
	while(true) {
		wait_for_event(&event);		/*једина могућност је “frame_arrival”*/
		from_physycal_layer(&r);	/*преузми оквир*/
		to_network_layer(&r.info);	/*проследи пакет из оквира мрежном слоју*/
	}
}
```

Овај код ћемо поделити у две секције, пошиљоц (*sender*) и прималац (*receiver*), тј. функција пошиљаоца се извршава на рачуанру који емитује оквире, а функција примаоца се извршава на рачунару који дочекује оквире.

Функција пошиљаоца: на самом почекту функције дефинисане су две структуре података *frame s* и *packet buffer*. Ове структуре у себи садрже елементе које су битни за чување свих делова оквира (структура *frame*) и пакета (структура *packet*). С обзиром да само желимо да покажемо функционалност овог к0д-а, нећемо улазити у детаље везане за конструкцију ових структура. С обзиром да се ради о идеалном (утопијском) случају, пакет који је спреман за слање пружима се посебном функцијом *from_network_layer* и смешта у посебно место у структуру *frame* (*s.info*). Као последњи корак, функција *to_physical_layer* прослеђује оквир на мрежу.

Функција примаоца: на самом почекту дефинишемо структуру оквира али и догађај који ће означавати приспеће оквира. Догађај о приспећу оквира *wait_for_event* користићемо како би знали када да узмемо оквир са физичкој слоја *from_physycal_layer*, а затим, *to_network_layer* мрежном слоју проследити само пакет са којим он врши даљу обраду.

Stop-and-Wait је протокол *Data-Link Layer*-а намењен за трансмисију оквира путем бешумних канала. Пошиљалац емитује оквир и чека потврду, а тек када пријемник врати потврду да је оквир примљен, пошиљалац наставља са емитовањем наредног оквира. Скицу принципа рада дајемо на следећем примеру:



Пошиљалац, дакле, емитује оквир, затим чека потврду о пријему оквира након чега наставља емитовање наредног оквира и тд. На крају, назив овог протокола каже: ”Стани и чекај”, тј. чекај на потврду о пријему оквира. Ево како би то било написано у програмском к0д-у:

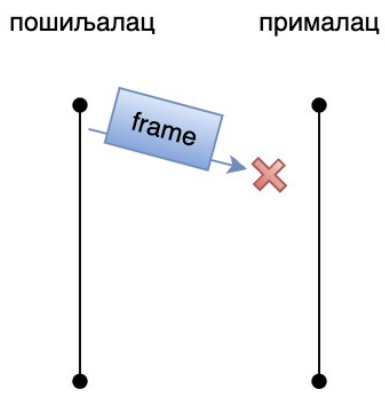
```
void sender(void) {
      frame s; /*складиште за одлазни оквир*/
      packet buffer; /*складиште за одлазни пакет*/
      event_type event; /*догађај о приспећу оквира*/
      while (true) {
            from_network_layer(&buffer); /*преузми пакет са мрежног слоја*/
            s.info = buffer; /*уметни пакет у оквир*/
```

```
            to_physical_layer(&s); /*проследи оквир физичком слоју*/
            wait_for_event(&event); /*чекај на потврду*/
            }
      }

void receiver(void) {
      frame r,s; /*складиште за одлазни оквир*/
      event_type event; /*догађај о приспећу оквира */
      while (true) {
            wait_for_event(&event); /*једина могућност је “frame_arrival”*/
            from_physycal_layer(&r); /*преузми оквир*/
            to_network_layer(&r.info); /*пакет из оквира проследи мрежном слоју*/
            to_physycal_layer(&s); /*проследи потврду о пријему*/
            }
      }
```

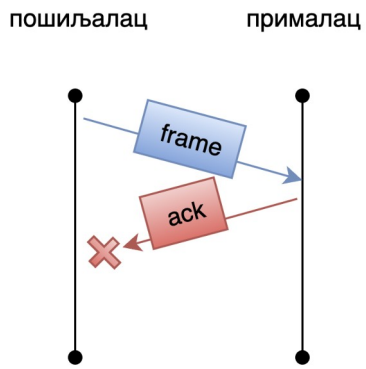
У односу на претходни пример, у функцији пошиљаоца имамо нову функцију *wait_for_event(&event)* која има задатак да прати приспеће потврде пријеме оквира, након чега пошиљаоц емитује наредни оквир. Када је реч о функцији на страни примаоца, овде имамо дефинисане две променљиве које ће бити типа структуре *frame*. Променљива *r* означава оквир који носи податак, а променљива *s* означава потврдни оквир (ack), односно оквир који се шаље пошиљаоцу као потврда о приспећу оквира у коме се налази податак.

Као што видимо из претходног примера, ”стани и чекај” протокол је боља верзија од његовог претходника из разлога што сада знамо да је оно што је пошиљалац послао, потврђено стигло. Међутим, на самом почетку смо извршли поделу на бешумне и бучне канале. Бешумни канали су идеални и готово се не могу наћи у реалном продукционом окружењу. Бучни канали су они са којима се сусрећемо свакодневно јер сваки комуникациони канал уноси не само шумове, већ и одређена слабљена током преноса сигнала. Ово нас наводи на размишљање о прилагођавању Stop-and-wait протокола у бучним каналима. Али пре свега хајде да погледамо који су проблеми овог протокола у бучном каналу:

Прво, може се десити да оквир који је послат не стигне до примаоца. Тада и пошиљалац и прималац чекају бесконачно време на приспеће оквира, тј. на оквир потврде.



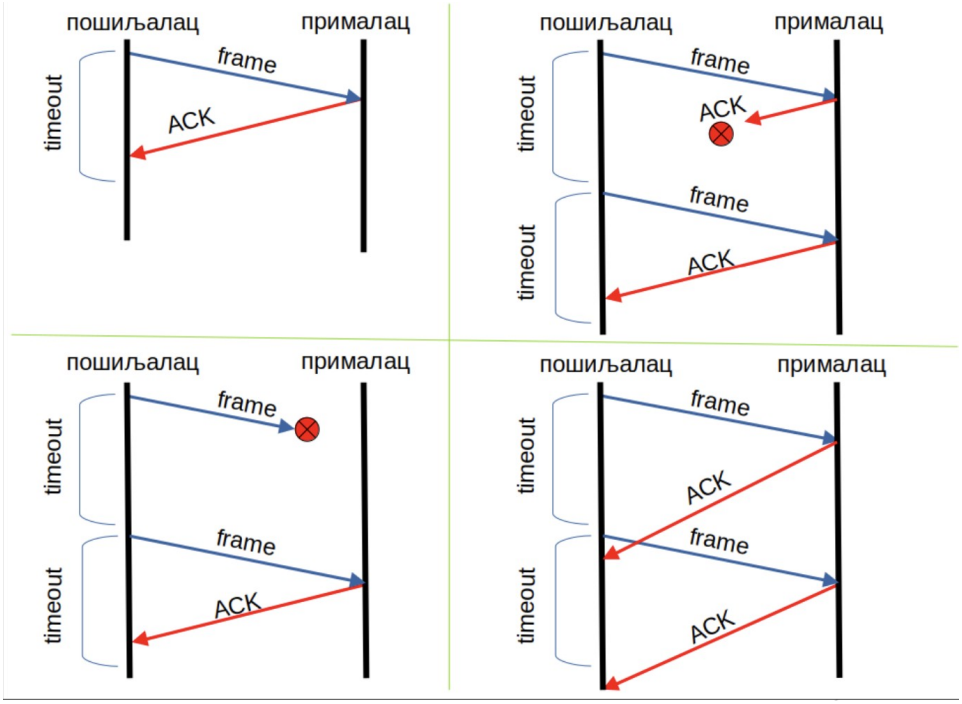
Друго, може се десити да се оквир потврде изгуби, па пошиљалац чека бесконачно време на приспеће оквира потврде како би емитовао наредни оквир.



Из претходно наведеног, потребно је урадити одређена побољшања како би се избегли дати проблеми у преносу оквира. Тако долазимо да наредног протокола **Stop-and-wait ARQ**.

Иницијални **Stop-and-wait** проширујемо тако што уводимо **тајмер** и **редни број оквира**. Дакле, након што пошаље оквир, пошиљалац чека одређено време, уколико се потврдни оквир појави, наставља са слањем наредног оквира, у супротном, понавља слање оквира за којег није стигла потврда.

Имамо неколико сценарија у вези са овим протоколом које дајемо на следећој скици:



Први квадрант показује како изгледа протокол када је све у реду. Други квадрант показује како се понаша протокол када *ack* не стигне. У трећем квадранту показан је пример када оквир не стиже на одредиште док је у четвртом квадранту показан пример када имамо одређено кашњење потврдног оквира услед чега настају проблеми у поновној ретрансмисији истог оквира. Дакле, овај протокол није савршен али у многоме решава проблеме свог претходника.

Кодна имплементација дата је на следећем примеру:

```
void sender(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);
			}
		}
	}
}
```

Напомена: код ”стани и чекај” протокола, секвенце оквира могу имати вредност 0 или 1. Код других протокола, редни број секвенце може имати и неку вишу вредност.

```
void receiver(void) {
	seq_nr next_frame_to_send;
	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);
			}
		}
	}
```

Протоколи клизних прозора (*Sliding Window*)

Протокол клизног прозора представља унапређену технику у односу на основни *Stop-and-Wait* протокол, са циљем побољшања ефикасности преноса података у мрежним системима.

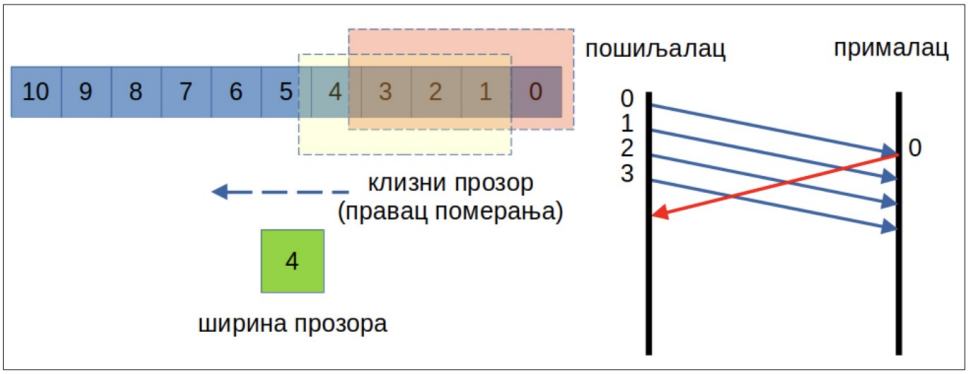
Разлози за побољшање Stop-and-Wait протокола:

- Истовремено се шаље само један оквир у јединици времена.
- Пропусни опсег канала се недовољно користи.
- Перформансе хардвера остају углавном неискоришћене.

Да би се ови недостаци отклонили, примењује се **метода клизног прозора**, која омогућава слање више оквира узастопно, без чекања потврде за сваки појединачни оквир.

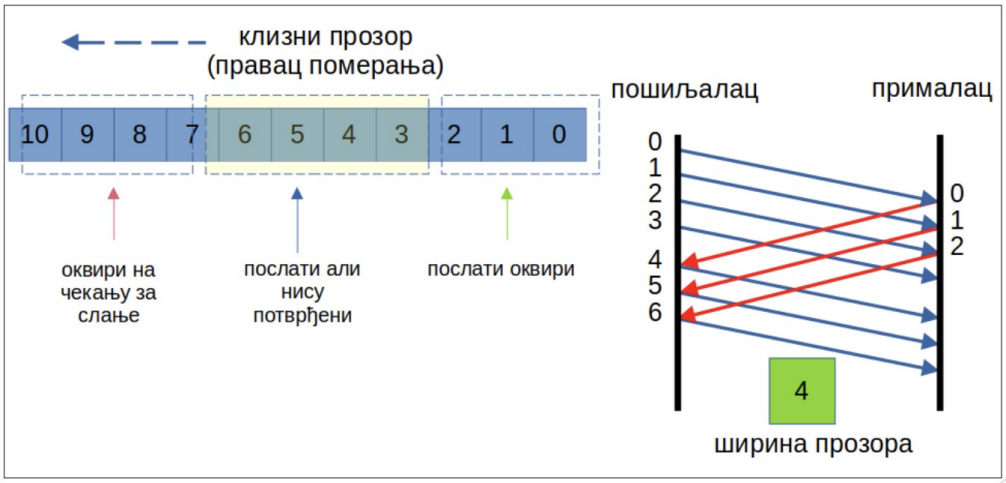
Сваки оквир који се шаље означен је јединственим редним бројем, такозваним ***Sequence Number***. Прималац, по пријему оквира, шаље повратну поруку (ACK) којом потврђује да је успешно примио оквир са одређеним редним бројем. По добијању потврде (ACK), пошиљалац „помера прозор“ и добија могућност да пошаље нове оквире, чиме се постиже континуирани ток података и боља искоришћеност мреже и хардвера.

Пример: Пошиљалац треба да пошаље одређени број оквира означених редом са 0,1,2...10. Параметар *Windows Size* поставимо на 4, то значи да се истовремено шаље 4 оквира. По пријему првог одговора, пошиљалац шаље оквир под редним бројем 4



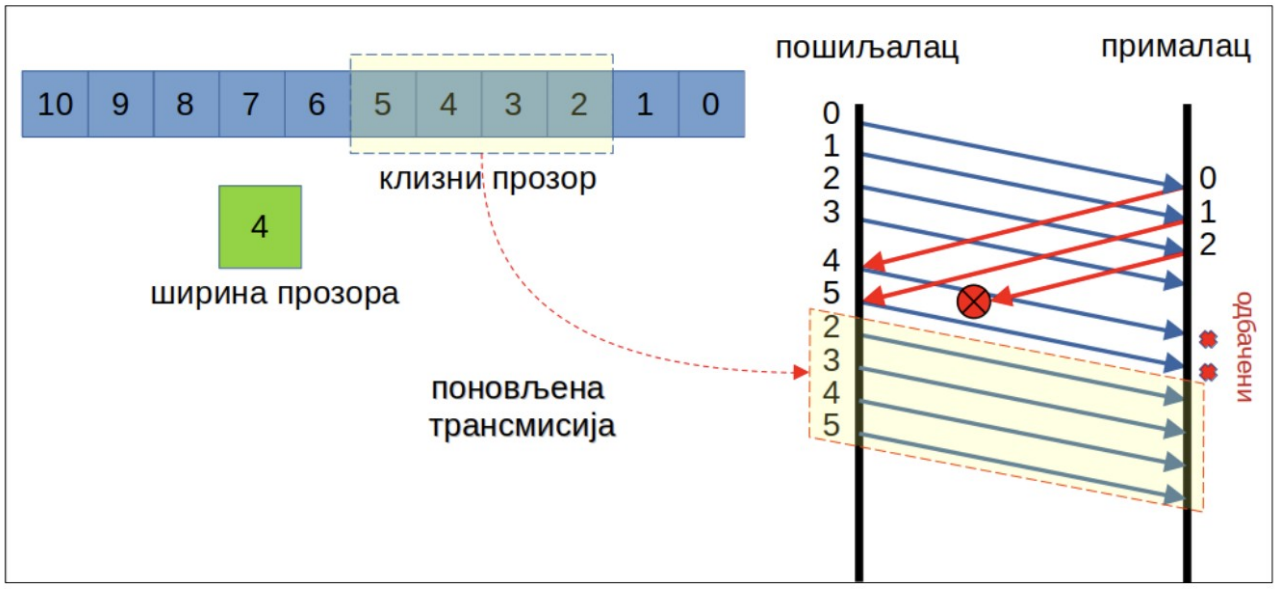
Протокол клизног прозора - пример

Слика приказује пример клизног прозора када су успешно потврђени оквири 0, 1 и 2, а још увек чекају потврде за оквире 3,4,5 и 6. Оквири 7,8,9 и 10 су спремни за слање и у стању су чекања.



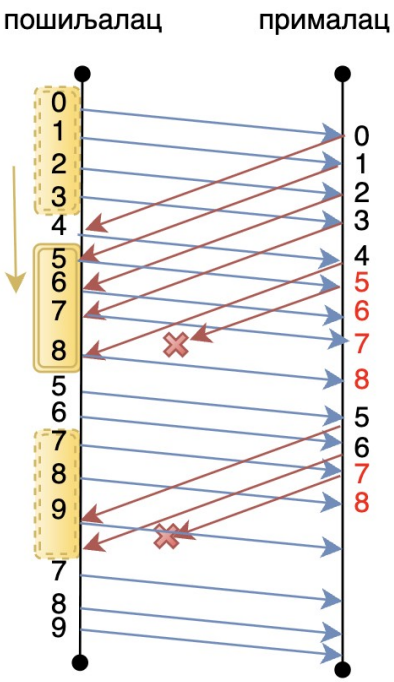
***Go-Back-N ARQ* протокол**

N у називу овог протокола означава величину прозора пошиљаоца (пример: *N-Back-4* када је величина прозора 4). Уколико потврда о послатом оквиру није стигла у одређеном времену, сви оквири у тренутном прозору биће ретрансмитовани.



Задатак 1: Рачунар А треба да пошаље поруку од 10 оквира рачунару В. Рачунари су се договорили да користе Go-Back-4 протокол. Уколико се сваки шести пакет којег шаље рачунар А изгуби (нема одговора од рачунара В), који је укупан број оквира које ће рачунар А послати рачунару В?

Решење.

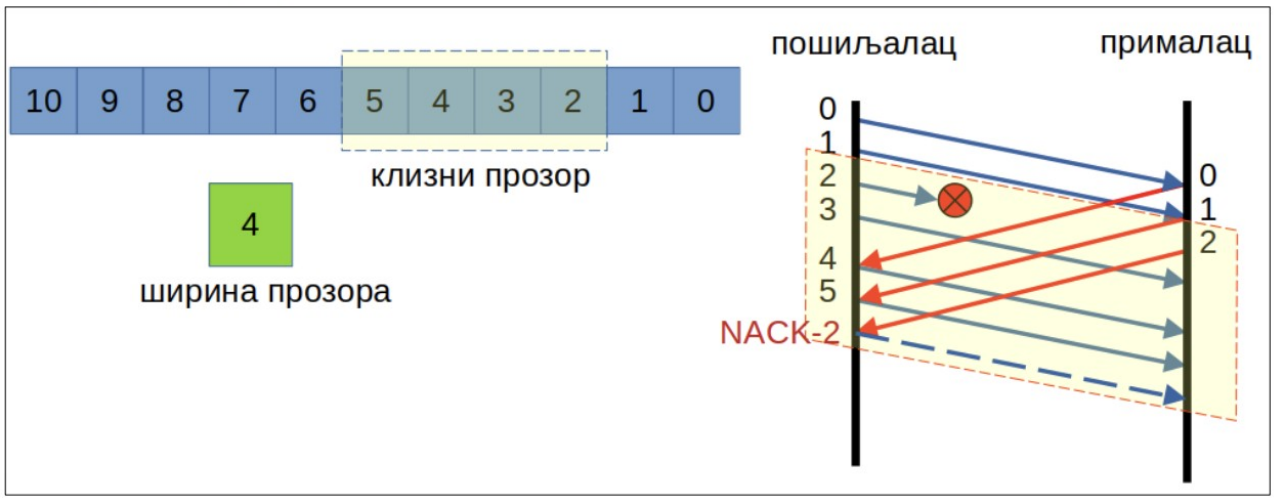


Укупан број оквира је 17.

Задатак 2: Рачунар А треба да пошаље поруку од 9 оквира рачунару В користећи клизни прозор величине 3 и Go-Back-N Error-Control стратегије. Сви оквири су спремни и одмах доступни за трансмисију. Уколико се сваки пети пакет којег шаље рачунар А изгуби (нема одговора од рачунара В), који је укупан број оквира које ће рачунар А послати рачунару В?

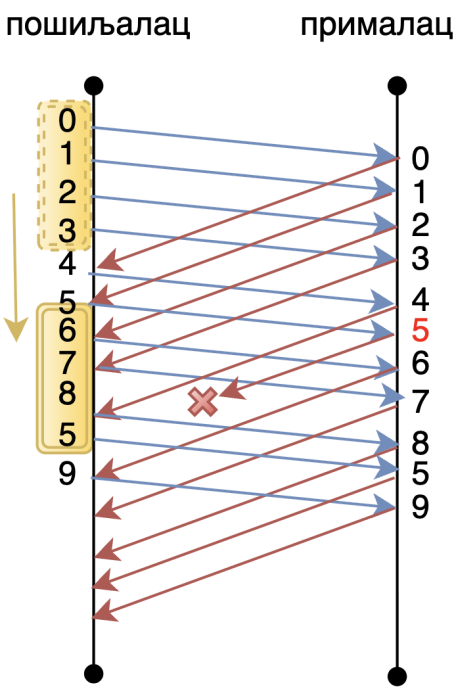
Селективно понављање (*Selective Repeat ARQ*)

За разлику од претходног протокола, протокол селективног понављања из клизног прозора ретрансмитује само оквир за којег пошиљалац није добио одговор (оштећен или изгубљени оквири), уместо свих оквира из тренутног клизног прозора. Уколико оквир са очекујућим *sequence* бројем не стигне (или стигне оштећен), прималац шаље *NACK* (*negative acknowledge*) пошиљаоцу. Пошиљалац ће ретрансмитовати само оквире за које је добио *NACK*.



Задатак 1: Рачунар А шаље 10 оквира рачунару В. Рачунари су се договорили да користе метод селективног понављања са клизним прозором величине 4. Колико оквира ће бити трансмитовано са Рачунара А уколико је сваки 6. оквир које се шаље изгубљен или оштећен? Упоредити број трансмисија на конкретном примеру између ове методе и методе Go-Back-4 ARQ.

Решење



Укупно 11 оквира је послато.

Додатак: Задатак за колоквијум (додатни): Рачунар А користи 32B (B - бајт) оквире за трансмисију поруке према рачунару Б користећи протокол клизног прозора. Време које је протекне од момента слања оквира до пријема потврде о пристиглом оквиру је 80 msec. Пропусни опсег канала је 128 kbps. Која је оптимална величина прозора коју рачунар А треба да одабере?