Transport

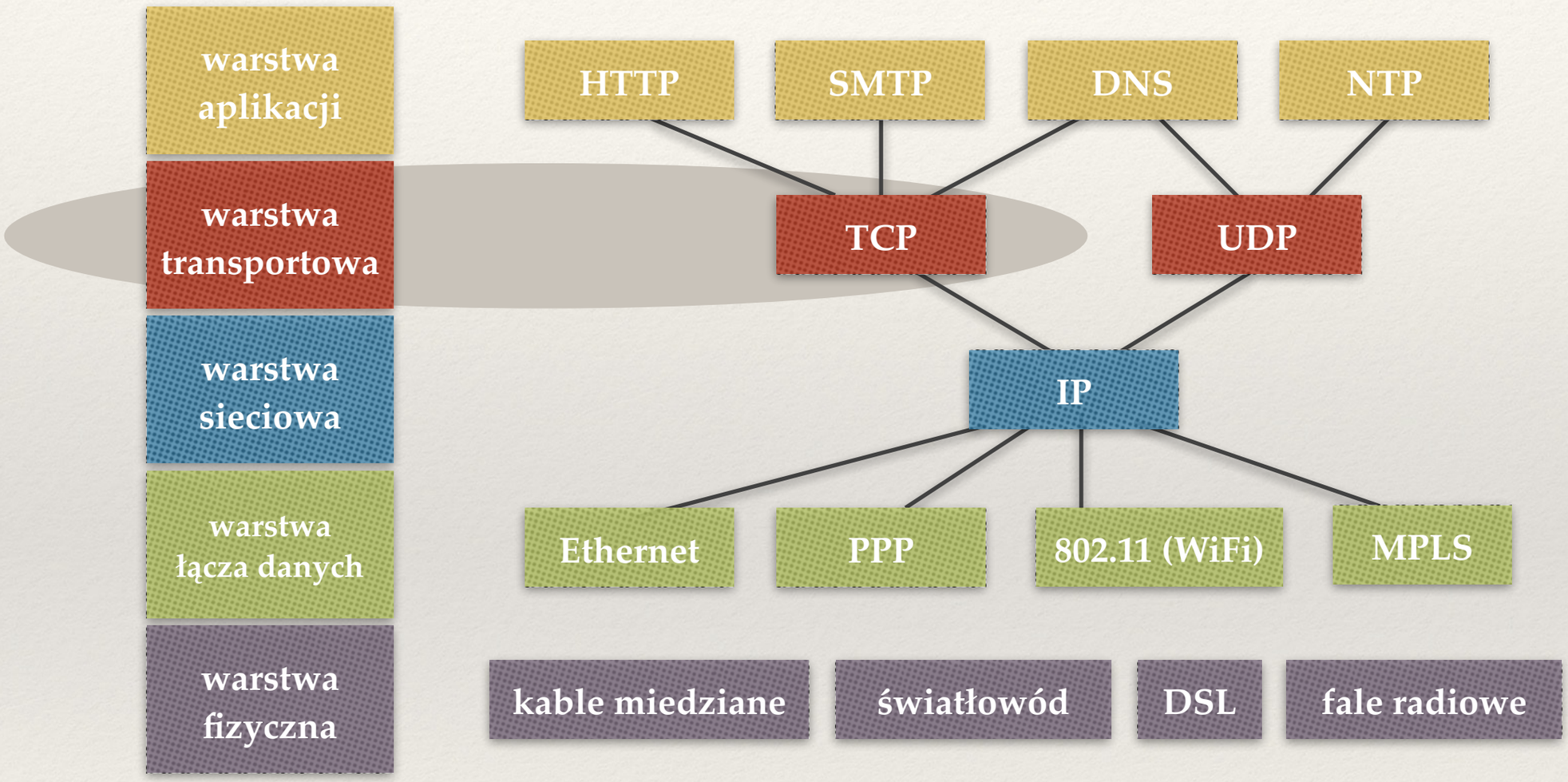
część 1: podstawy

Sieci komputerowe

Wykład 6

Marcin Bieńkowski

Protokoły w Internecie



Internetowy model warstwowy (1)

warstwa transportowa

zapewnia **globalne** dostarczanie danych pomiędzy **aplikacjami**

warstwa sieciowa

zapewnia **globalne** dostarczanie danych pomiędzy **komputerami**

warstwa łącza danych

zapewnia **lokalne** dostarczanie danych pomiędzy **komputerami**

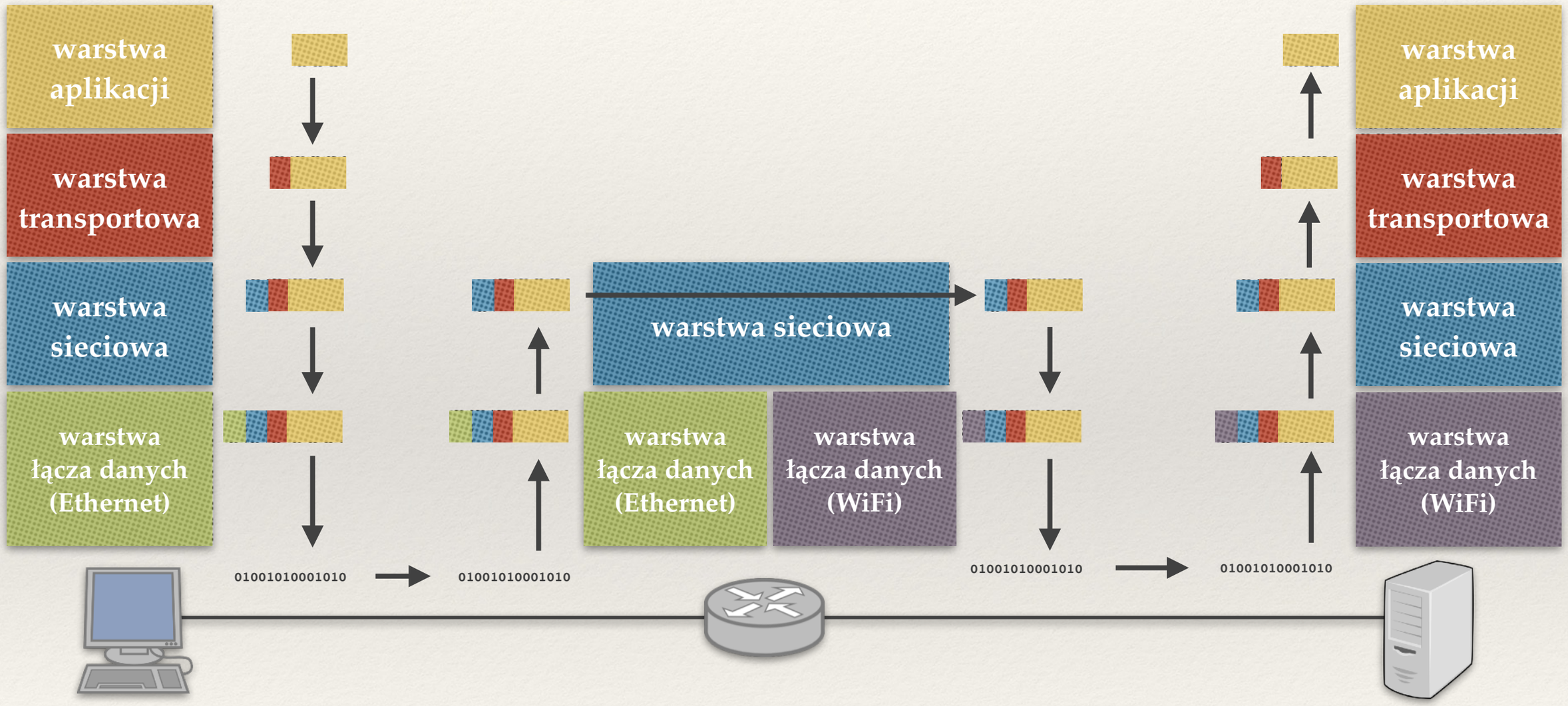
Porty

- **Port:**
  - liczba 16-bitowa;
  - identyfikuje aplikację wewnątrz danego komputera
    → multipleksowanie wielu strumieni danych w jednym ciągu pakietów.

- **W nagłówku warstwy transportowej znajduje się m.in.:**
  - port źródłowy;
  - port docelowy.

Internetowy model warstwowy (2)

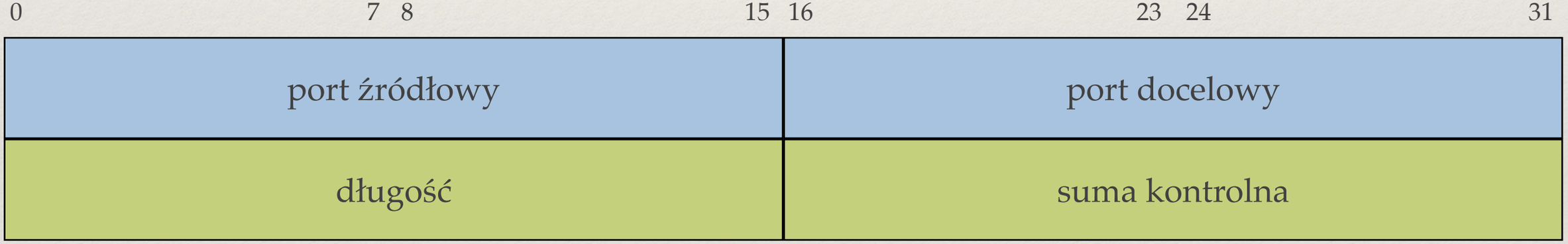
przesyłany pakiet:



Zawodny transport

UDP

- Najprostszy protokół warstwy transportowej
- Nagłówek UDP:



Gwarancje UDP (1)

- **Takie same jak IP, czyli żadne.**
  - Tylko zasada dołożenia wszelkich starań (*best effort*).

- **Pakiety mogą zostać:**
  - uszkodzone,
  - zgubione,
  - opóźnione,
  - zamienione (kolejność),
  - zduplikowane (przez wyższe lub niższe warstwy).

Gwarancje UDP (2)

- Czy te negatywne wydarzenia są częste?
- Lokalnie wysyłane pakiety nie będą przecież uszkadzane, gubione, duplikowane i będą przychodzić w tej samej kolejności?

demonstracja

kod programu na stronie wykładu

Gwarancje UDP (2)

- Czy te negatywne wydarzenia są częste?
- Lokalnie wysyłane pakiety nie będą przecież uszkadzane, gubione, duplikowane i będą przychodzić w tej samej kolejności?

demonstracja

kod programu na stronie wykładu

- **Kontrola przepływu** = nadawca powinien dostosowywać prędkość transmisji do szybkości odbiorcy.

Gdzie wykorzystujemy

Zawodny transport:

- Przesyłane są małe ilości danych (np. DNS, DHCP).
- Proste, ograniczone obliczeniowo urządzenia (np. TFTP wykorzystywany do aktualizacji firmware).
- Konieczna jest szybka reakcja (gry).
- Chcemy pełnej kontroli nad przesyłanymi danymi (NFS).

Niezawodny transport:

- Przesyłane są duże ilości danych (np. HTTP(S), nieinteraktywny streaming video)

Niezawodny transport: segmentacja

Segmentacja

Niezawodne przesyłanie ciągu bajtów:

- Zadanie warstwy transportowej.
- Wymaga dzielenia ciągu bajtów na **segmenty**.
  - W UDP użytkownik musi to robić sam.
- **Dlaczego nie przesyłamy wszystkiego w jednym segmencie?**

Słowo o nazewnictwie

warstwa transportowa

datagramy (gdy użytkownik sam dzieli na części, np. UDP)

segmenty (gdy warstwa dzieli na części, np. TCP)

warstwa sieciowa

pakiety

warstwa łącza danych

ramki

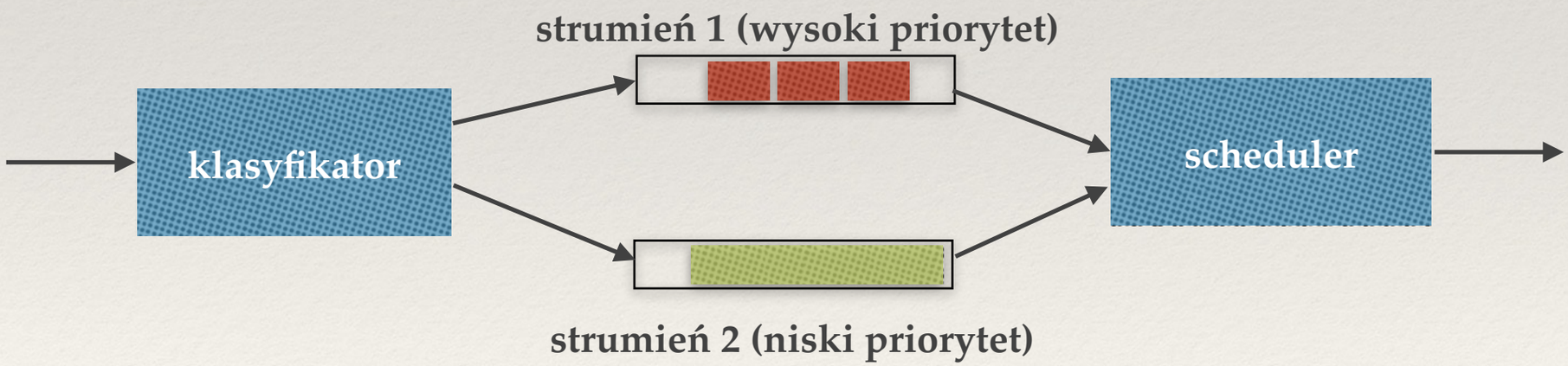
Niekonsekwentnie używane nazwy.

- Powszechnie stosowane „datagramy IP“.
- „Segment TCP“ często oznacza same dane TCP (bez nagłówka TCP), np. w definicji MSS (maximum segment size).

Dlaczego segmentacja jest potrzebna?

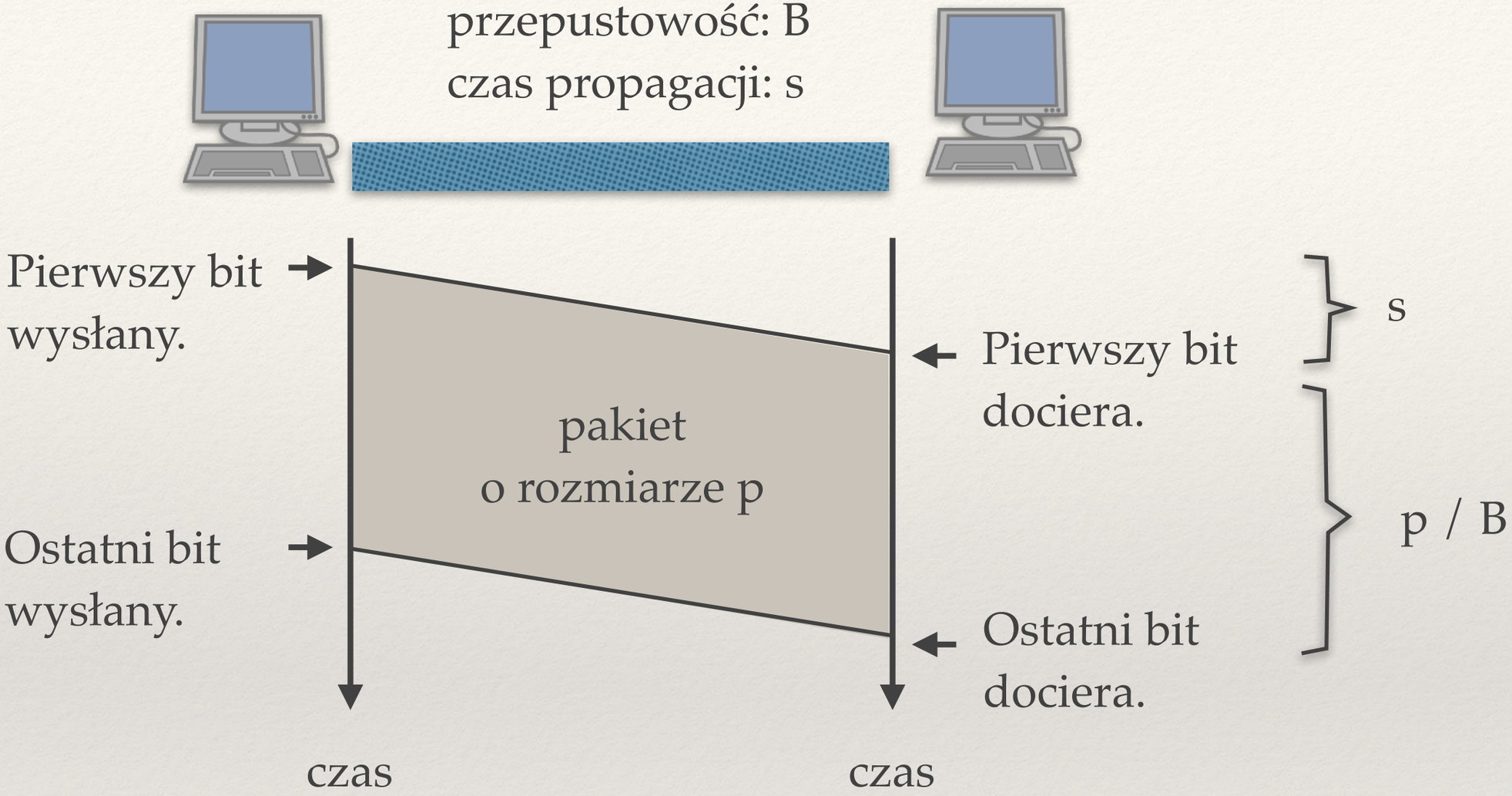
- **Skąd biorą się ograniczenia na rozmiar segmentu?**
  - MSS = MTU - rozmiar nagłówka IP - rozmiar nagłówka TCP

- **Dlaczego nie zwiększymy MTU (i rozmiaru pakietu IP)?**
  - Większa szansa na zakłócenia (sieci bezprzewodowe).
  - Duże pakiety: problem w szeregowaniu w kolejce wyjściowej routera (małe pakiety mają duże opóźnienie).



  - Główna przyczyna: **mniejsze opóźnienie przy długich ścieżkach**.

Opóźnienie pakietu na łączu (przypomnienie)

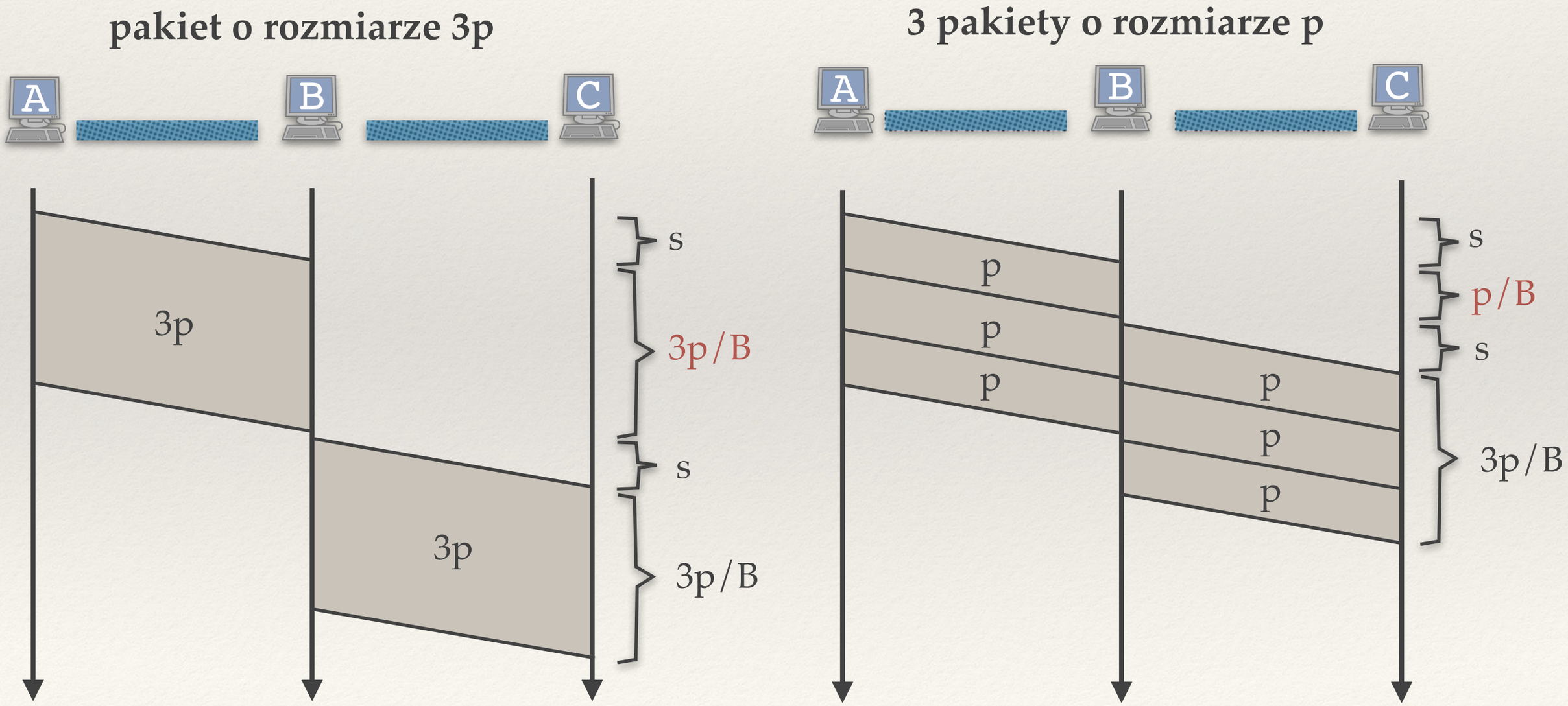


Opóźnienie na pojedynczym łączu = s + p / B.

- Ignorujemy czas kolejkowania pakietu w buforze.

Opóźnienie pakietu na ścieżce

- Dla łącz poniżej: przepustowość = B, czas propagacji = s.
- Im dłuższa ścieżka, tym większy efekt.
- Pomijamy tu narzut związany z rozmiarem nagłówka!



Niezawodny transport: ARQ

ARQ = Automatic Repeat reQuest

- **Zajmiemy się niezawodną transmisją jednokierunkową.**
  - Od nadawcy do odbiorcy.
  - W drugą stronę identyczny mechanizm.

- **ARQ**
  - Wysyłanie „do skutku“ (do otrzymania potwierdzenia od odbiorcy).
  - Odbiorca wysyła informacje zwrotne (otrzymałem pakiet / zwolnij / przyspiesz / …)
  - Niezawodny transport na bazie zawodnej usługi przesyłania pakietów.

- **Założymy, że strumień bajtów jest już posegmentowany.**

Dostępne podstawowe mechanizmy

- **Sumy kontrolne:** możemy wykrywać, czy pakiet został uszkodzony.

- **Potwierdzenia (ACK):** małe pakiety kontrolne potwierdzające otrzymanie danego segmentu.

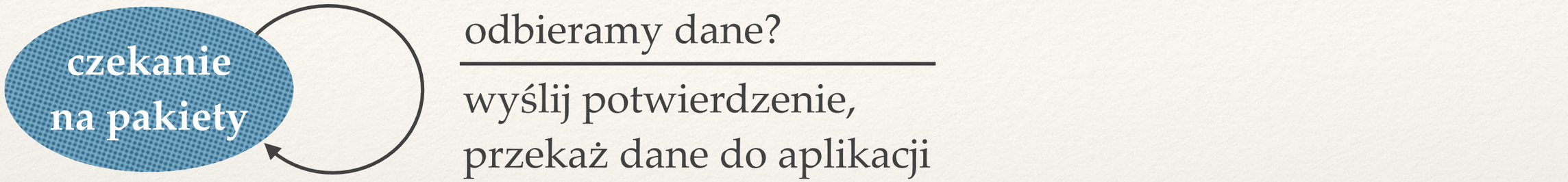
- **Timeout (przekroczenie czasu oczekiwania):** jeśli nie otrzymamy potwierdzenia przez pewien czas (typowy dla łącza, np. RTT — jak go mierzyć?)

- **Retransmisje:** ponowne wysłanie danego segmentu w przypadku przekroczenia czasu oczekiwania.

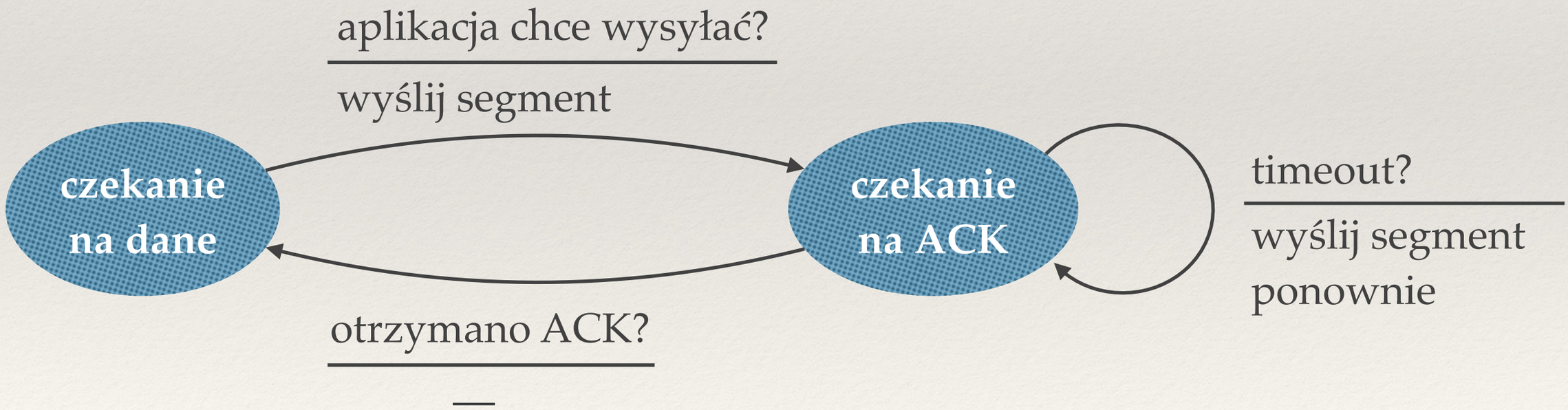
ARQ: Stop-and-Wait

Protokół Stop-and-Wait

Algorytm odbiorcy:

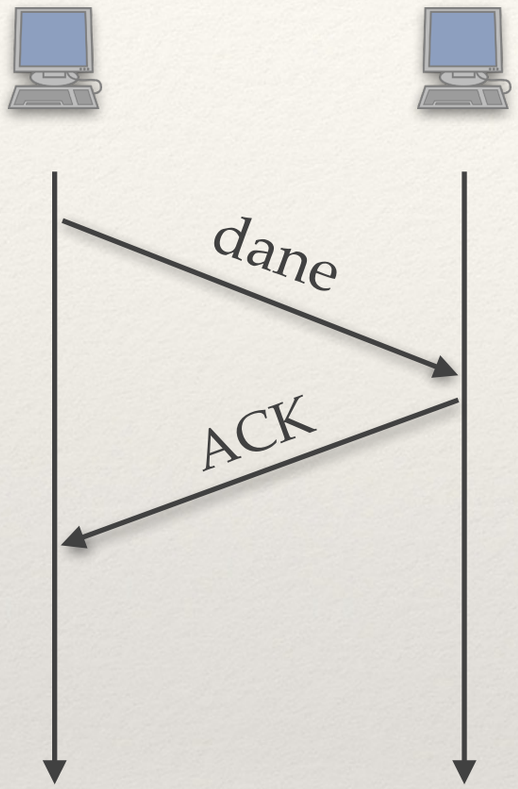


Algorytm nadawcy:



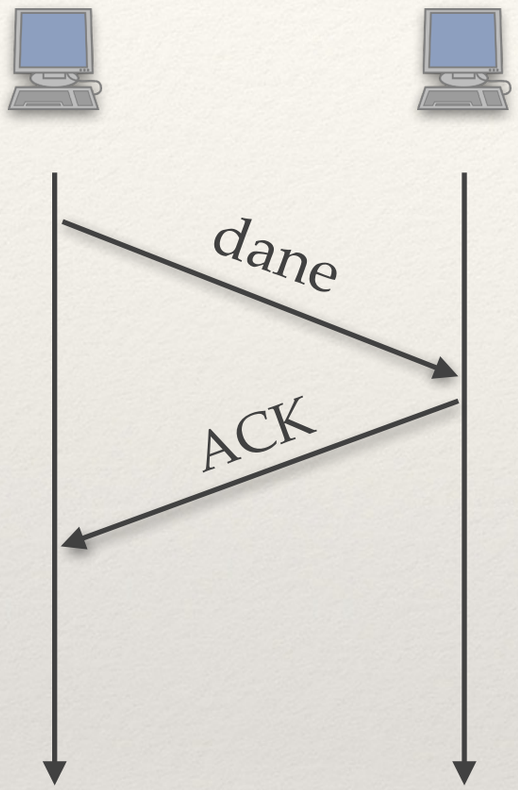
Stop-and-wait: typowe wykonania

1. bez utraty pakietów

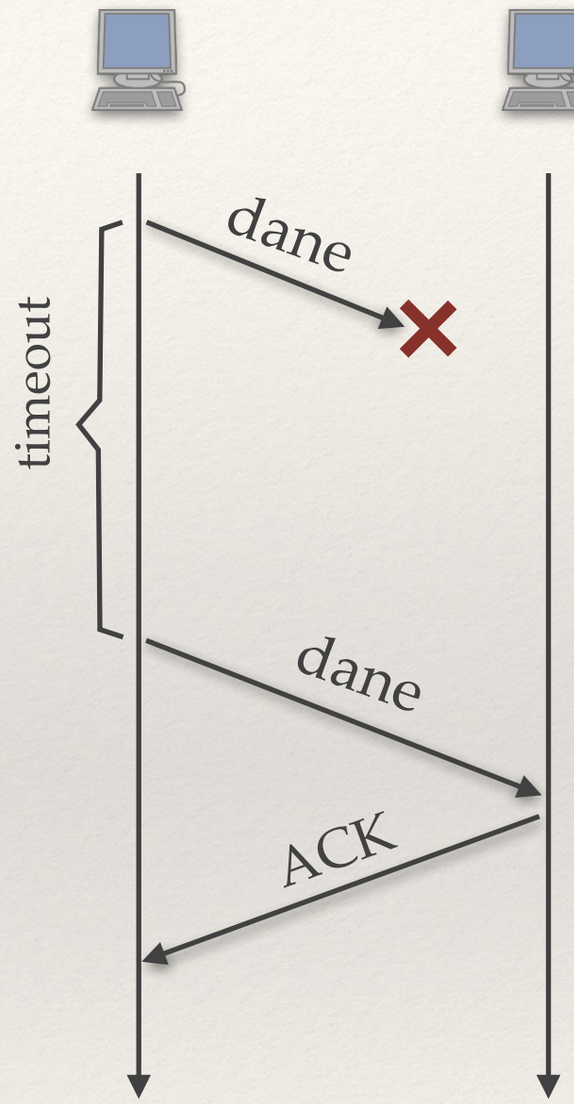


Stop-and-wait: typowe wykonania

1. bez utraty pakietów

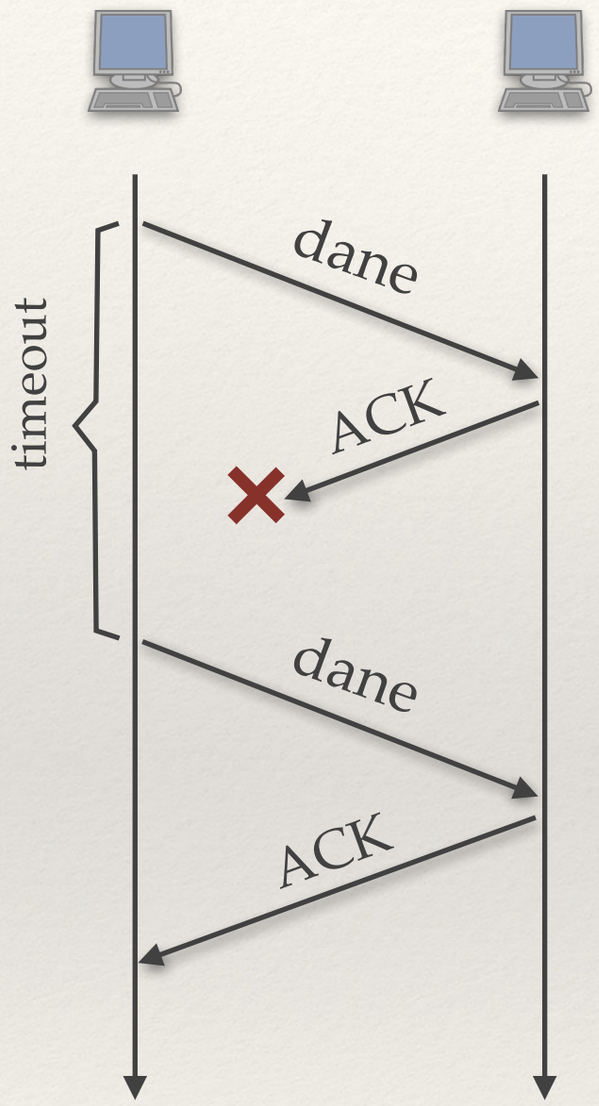


2. utrata danych



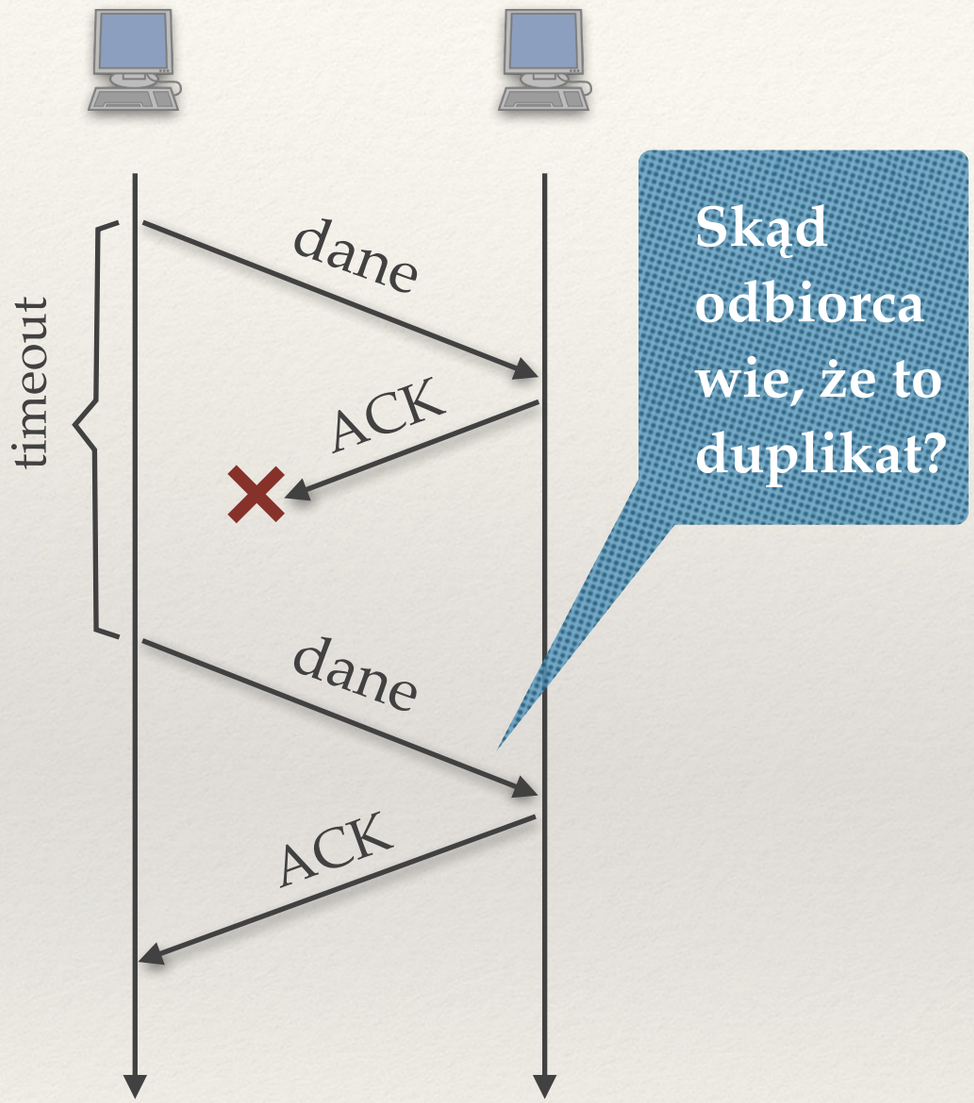
Stop-and-wait: możliwe wykonania

3. utrata ACK



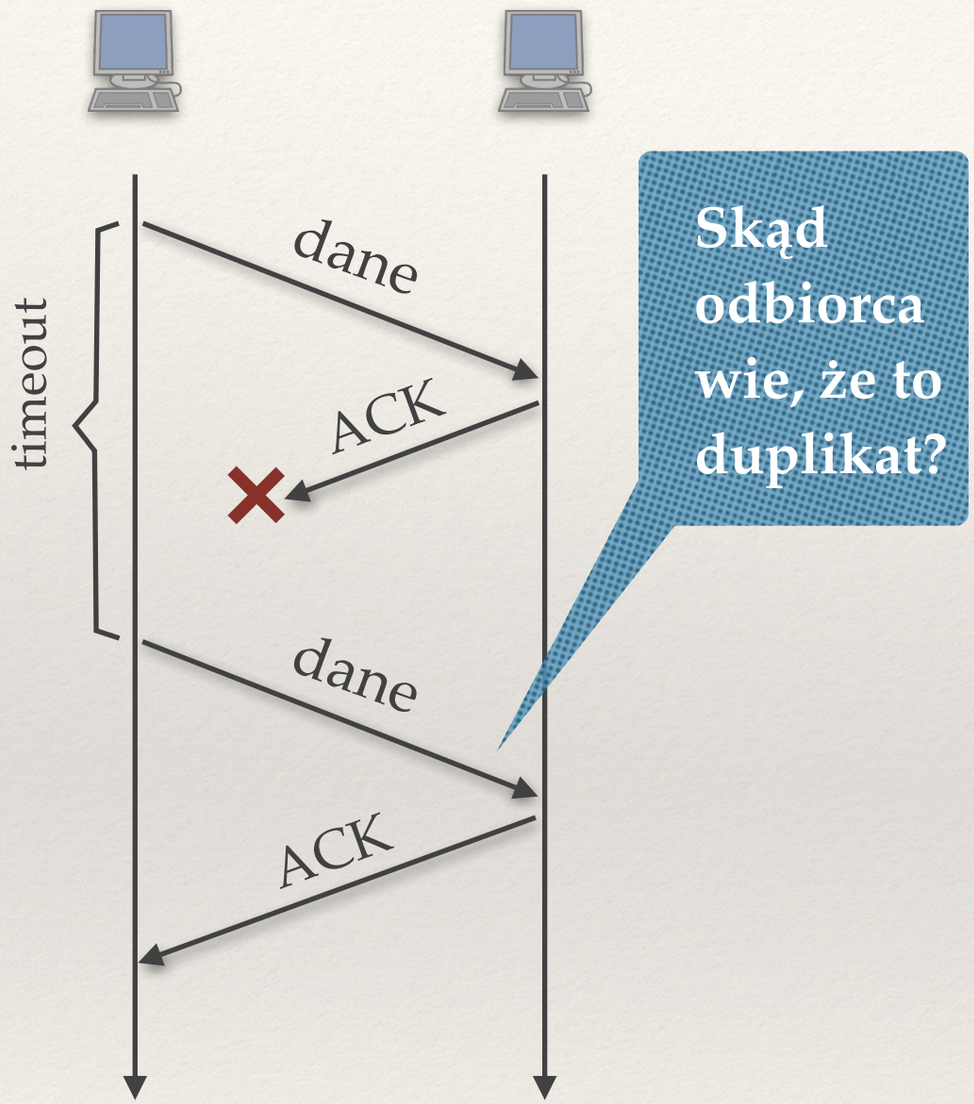
Stop-and-wait: możliwe wykonania

3. utrata ACK

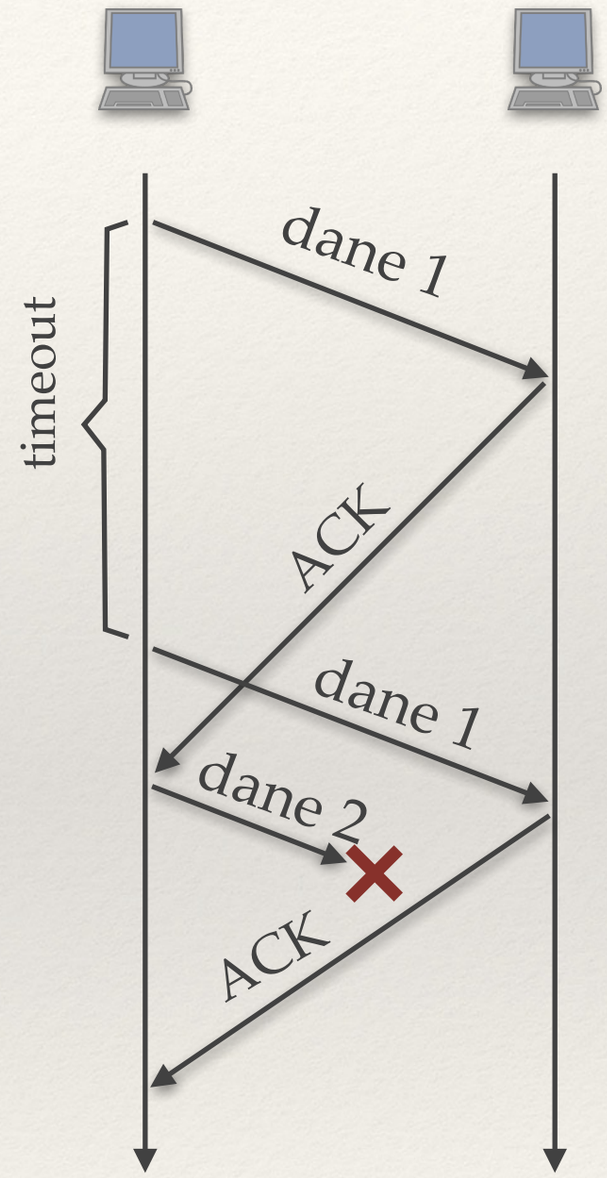


Stop-and-wait: możliwe wykonania

3. utrata ACK

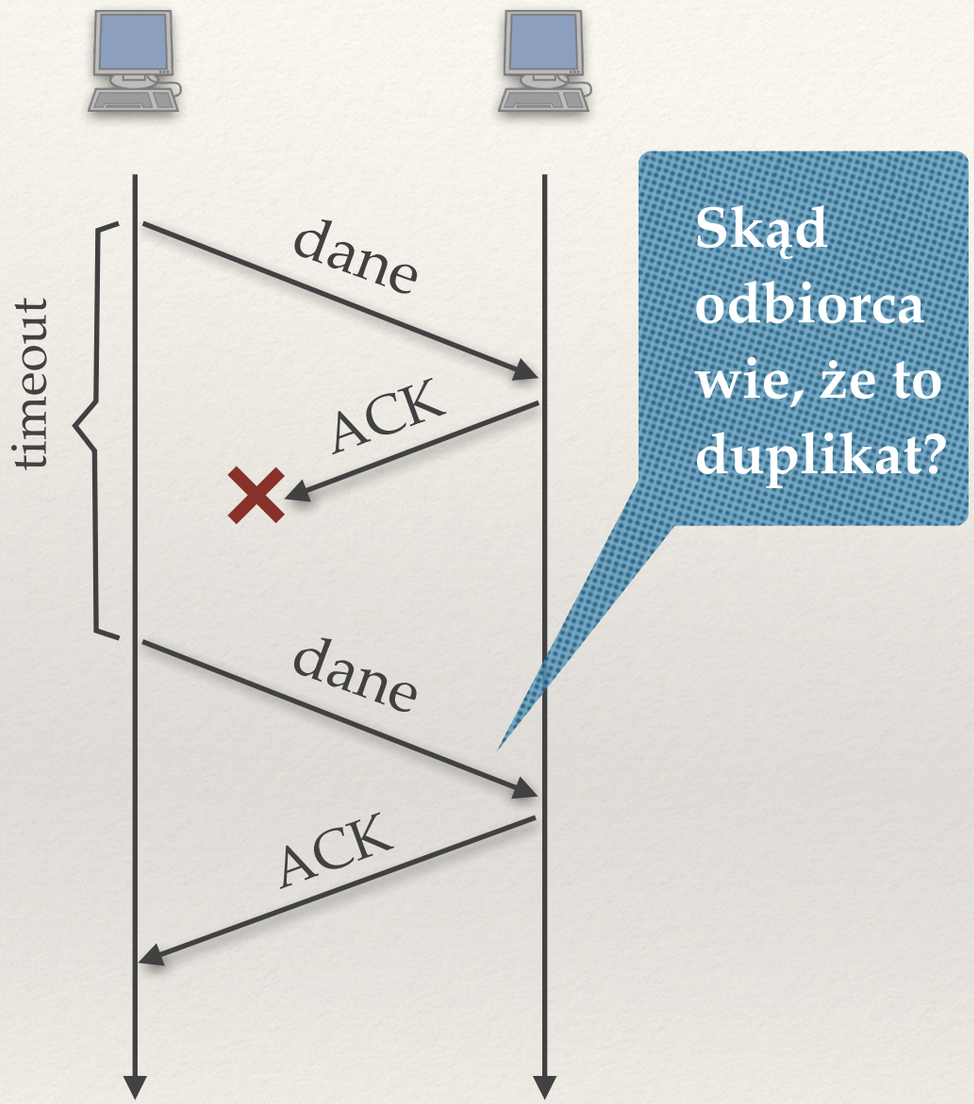


4. opóźnienie ACK + utrata danych

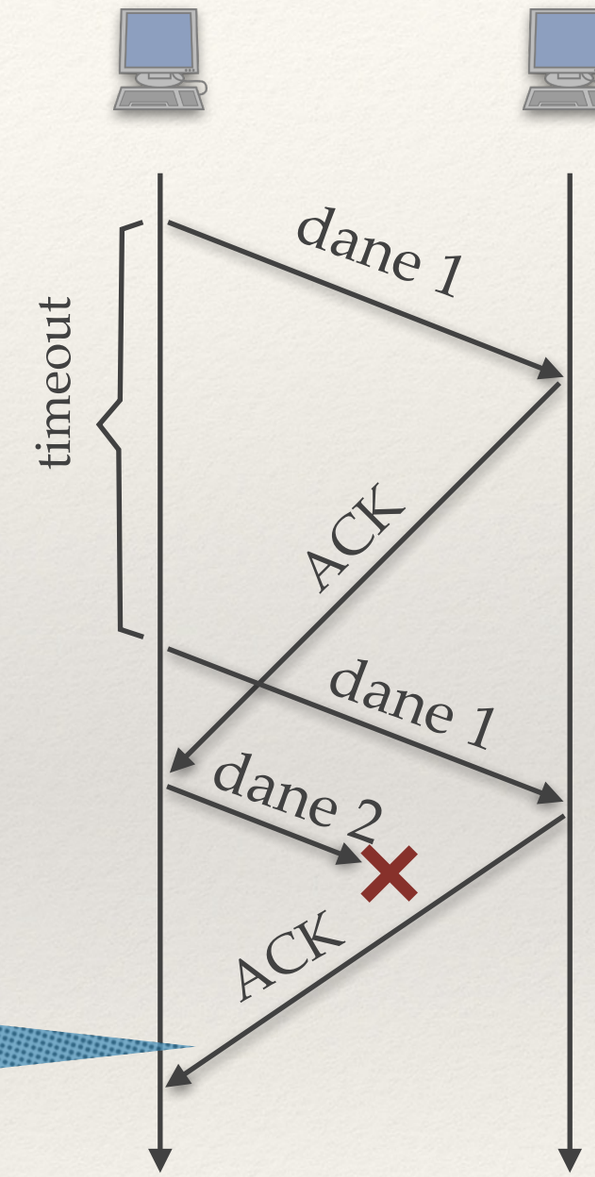


Stop-and-wait: możliwe wykonania

3. utrata ACK

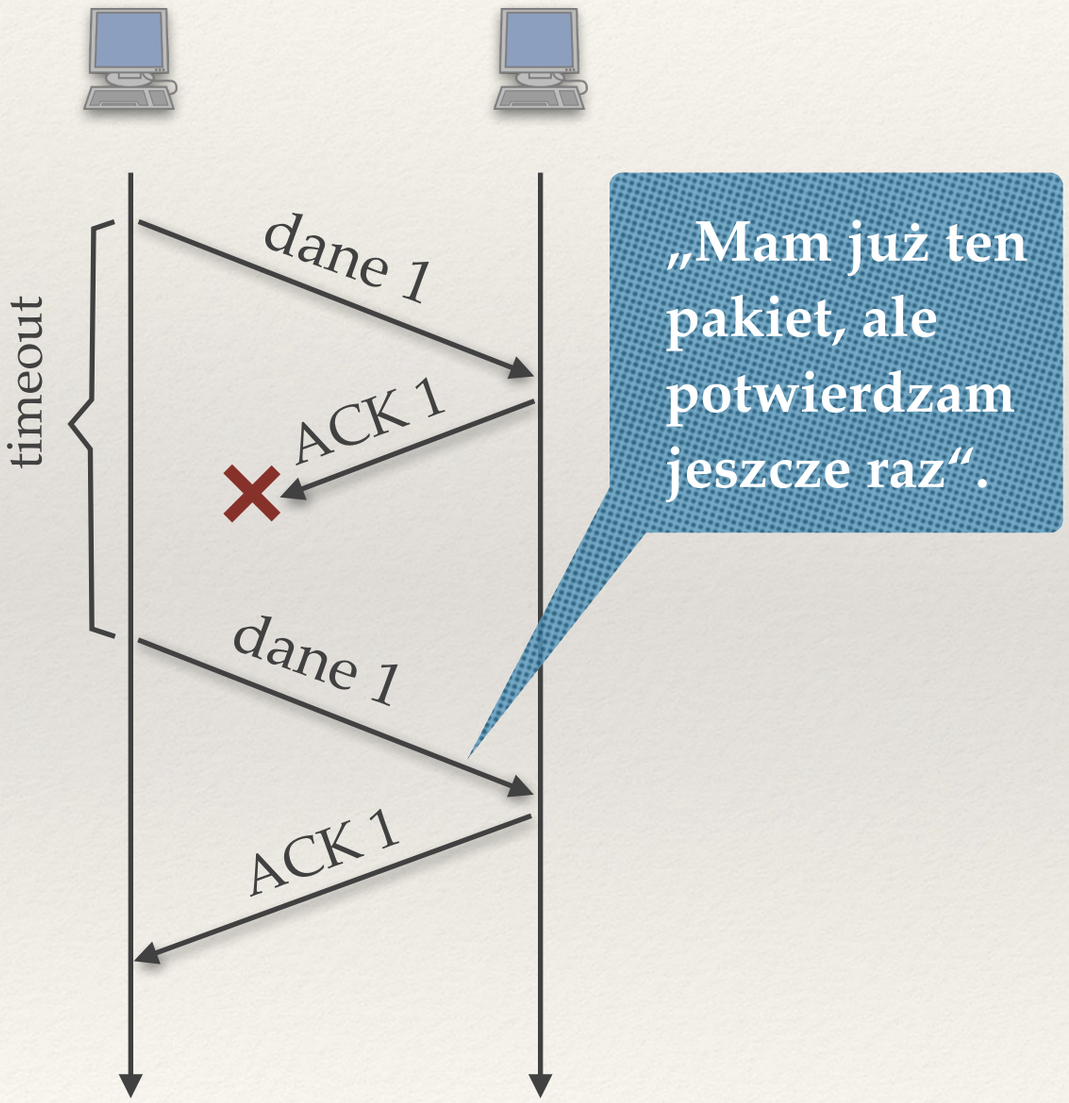


4. opóźnienie ACK + utrata danych



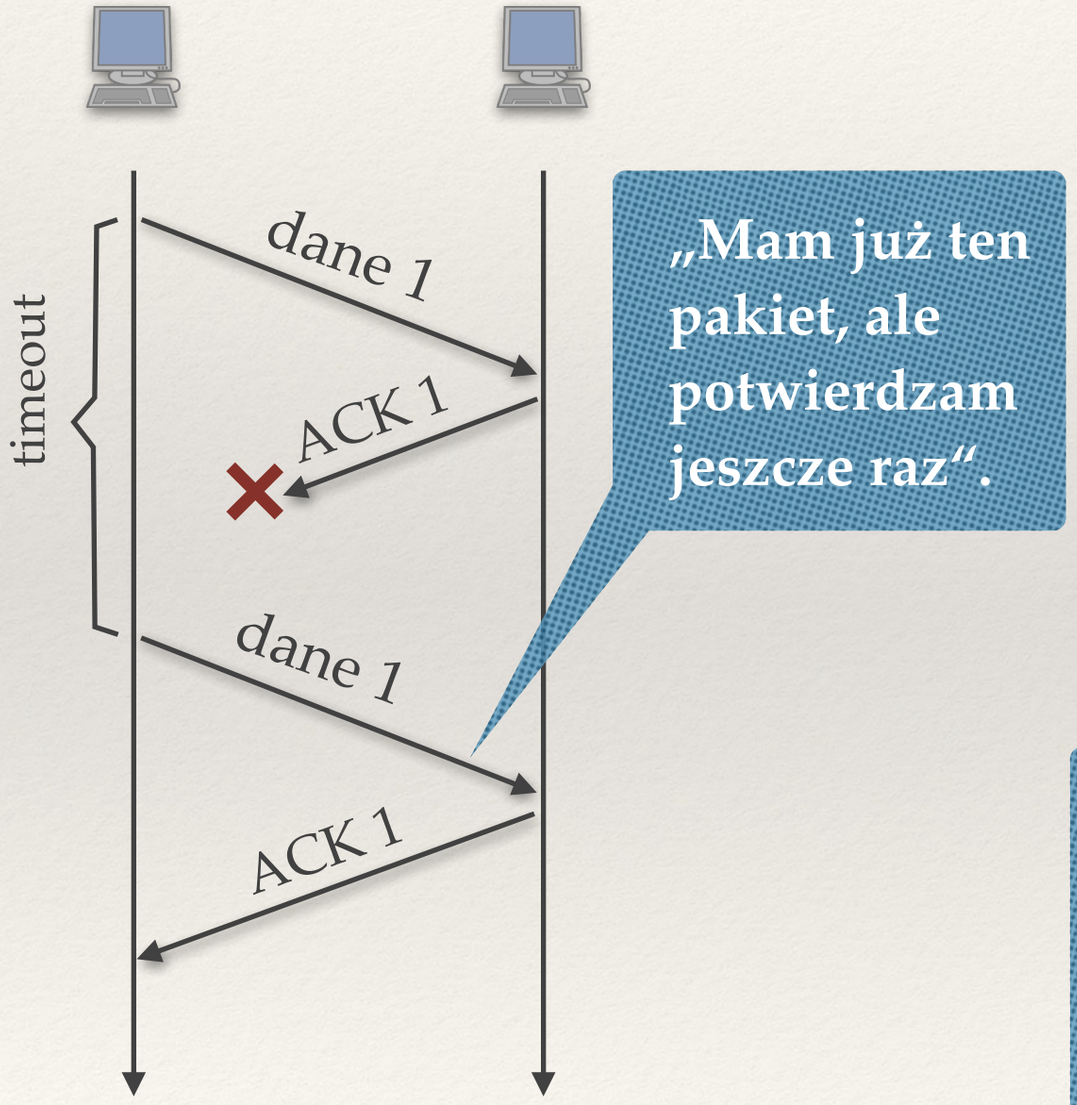
Numery sekwencyjne

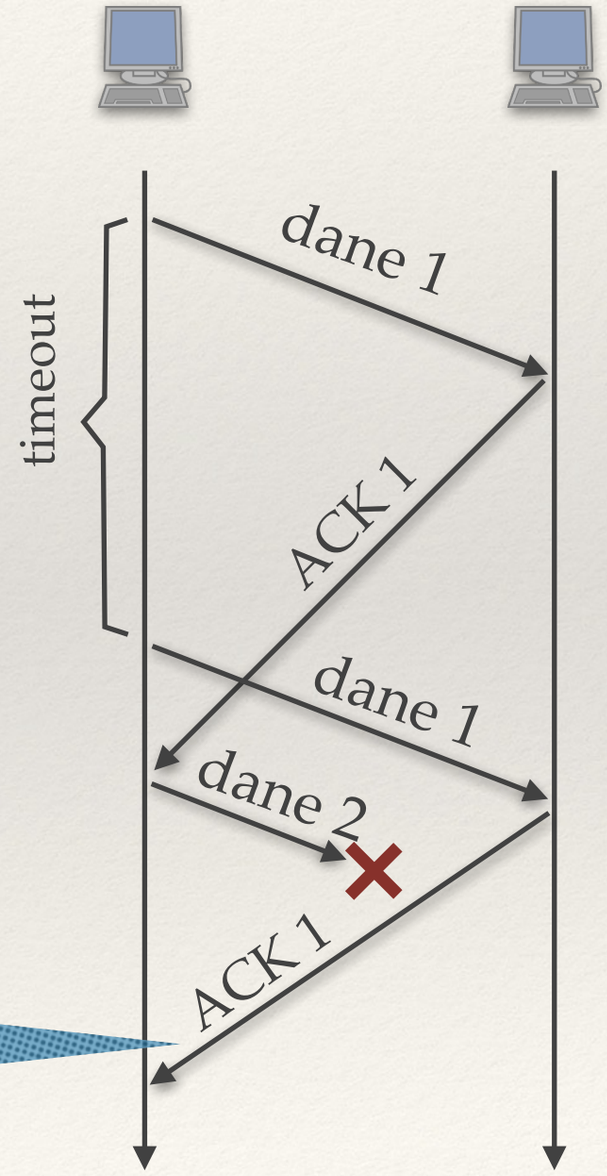
- Każdy segment jest numerowany.
- ACK zawiera numer potwierdzanego segmentu.



Numery sekwencyjne

- Każdy segment jest numerowany.
- ACK zawiera numer potwierdzanego segmentu.

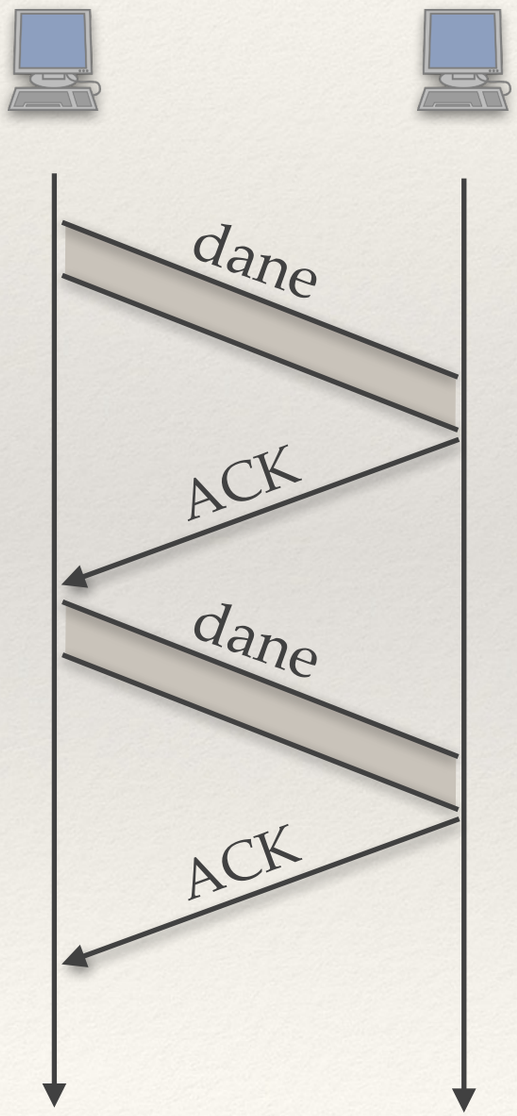




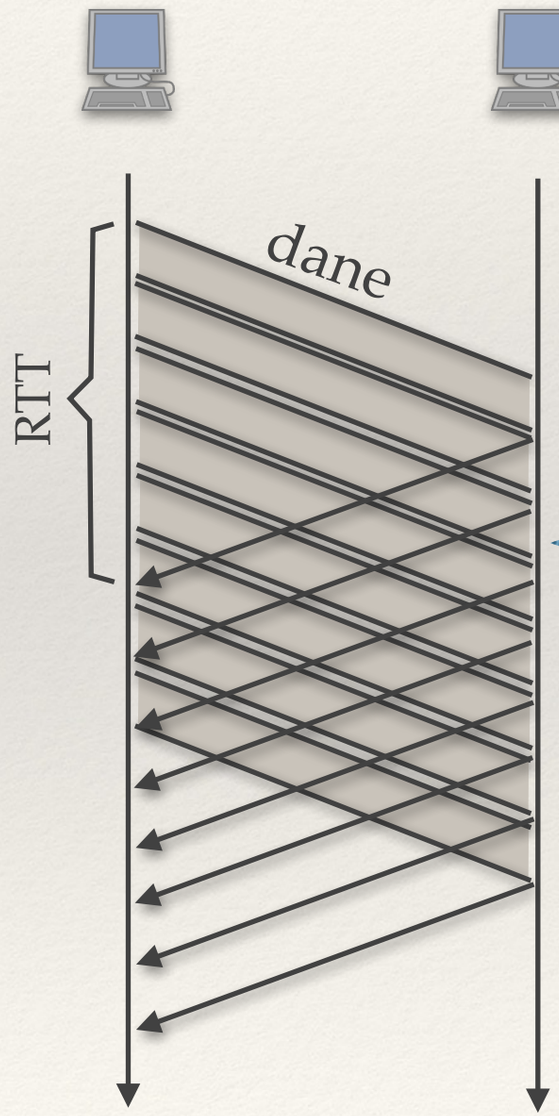
Stop-and-wait + numery sekwencyjne

- Działa, ale przy długich łączach o dużej przepustowości wykorzystuje ułamek ich możliwości!

stop-and-wait



lepiej: (jak?)



Nadawca powinien móc nadawać bez czekania na ACK:

- **co najmniej przez RTT;**
- **równoważnie: nadać co najmniej BDP danych. (BDP = *bandwidth-delay product*)**

ARQ: Okno przesuwne

Przesuwne okno nadawcy (1)

- SWS (*sender window size*) = maksymalna liczba wysłanych i niepotwierdzonych segmentów.
- Segmenty od 1 do LAR są potwierdzone, a LAR+1 nie.
- Nadawca może wysyłać tylko segmenty leżące w oknie.

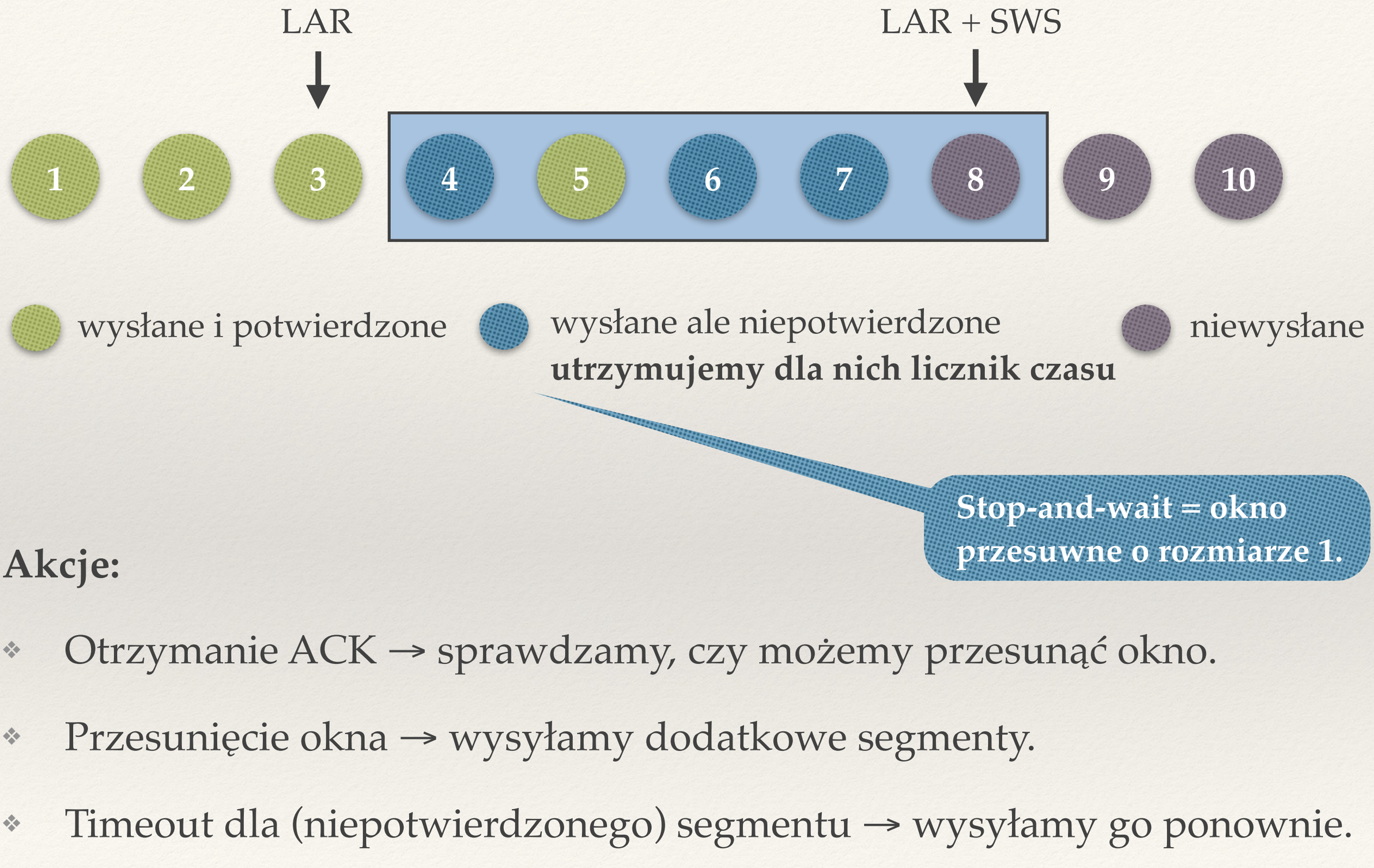
Przesyłany ciąg segmentów (okno nadawcy):

Przesuwne okno nadawcy (2)

Akcje:

- Otrzymanie ACK → sprawdzamy, czy możemy przesunąć okno.
- Przesunięcie okna → wysyłamy dodatkowe segmenty.
- Timeout dla (niepotwierdzonego) segmentu → wysyłamy go ponownie.

Przesuwne okno nadawcy (2)



Akcje:

- Otrzymanie ACK → sprawdzamy, czy możemy przesunąć okno.
- Przesunięcie okna → wysyłamy dodatkowe segmenty.
- Timeout dla (niepotwierdzonego) segmentu → wysyłamy go ponownie.

Mechanizmy potwierdzania

Nadawca: mechanizm przesuwnego okna.

Trzy typowe mechanizmy dla odbiorcy:

- Go-Back-N
- Potwierdzanie selektywne
- Potwierdzanie skumulowane

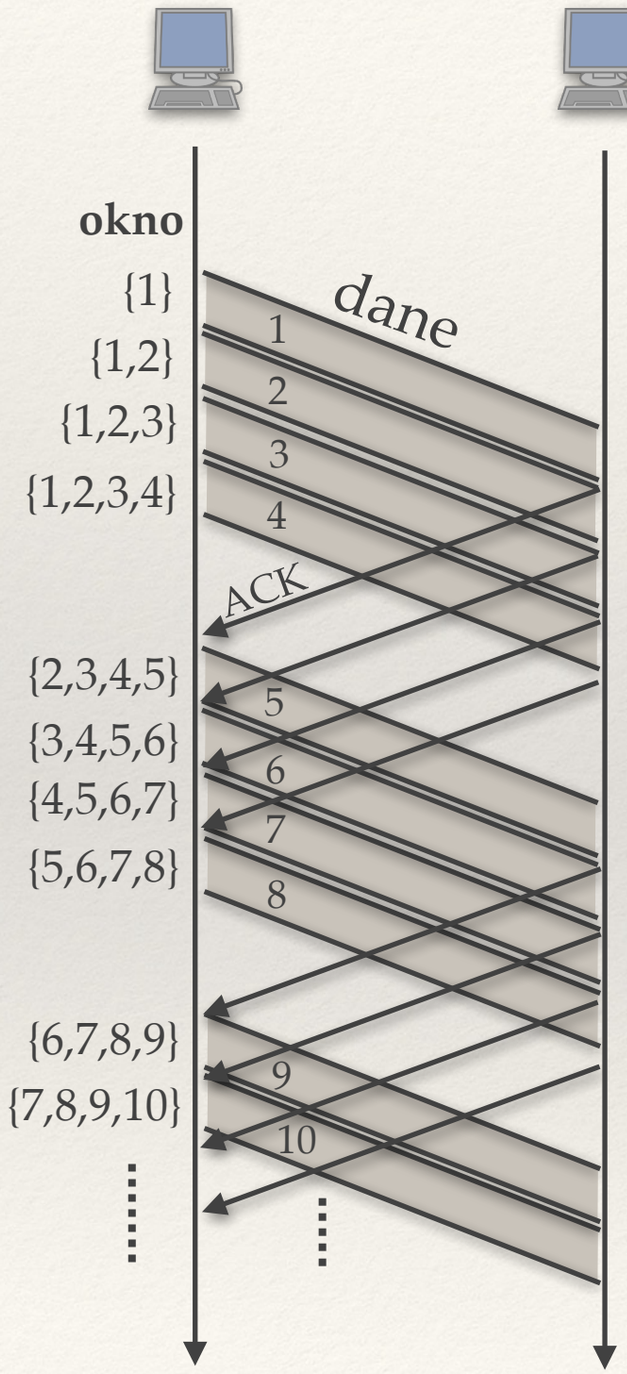
Potwierdzanie Go-Back-N

- Załóżmy, że odbiorca dostał już segmenty do P włącznie.

- Czy wysłać ACK dla otrzymanego segmentu S?
  - Tak dla S = P+1.
  - Tak dla S ≤ P (ponowne potwierdzenie, prawdopodobnie poprzedni ACK zaginął).
  - Nie dla S > P+1.
    - Prosta implementacja.
    - Jeśli warstwa transportowa odbiorcy sama przetwarzałaby segmenty (byłaby aplikacją), to nie potrzebowalibyśmy bufora odbiorcy.

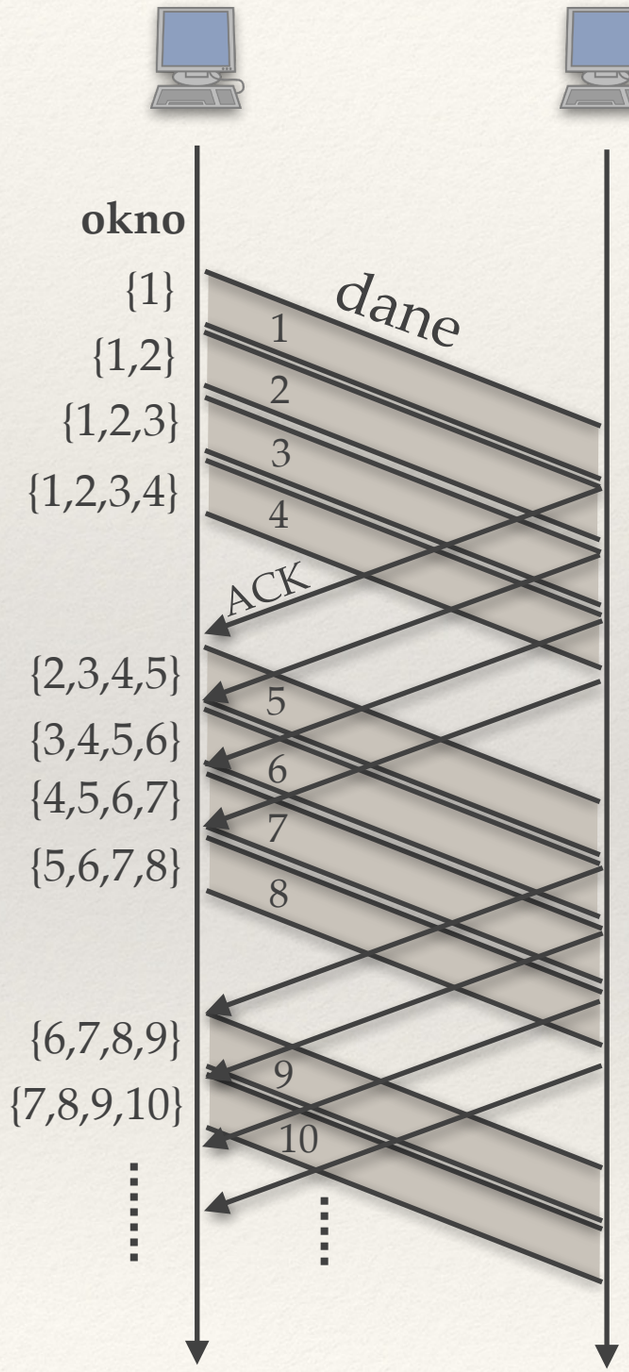
Potwierdzanie Go-Back-N: przykłady

SWS = 4, pakiety nie giną

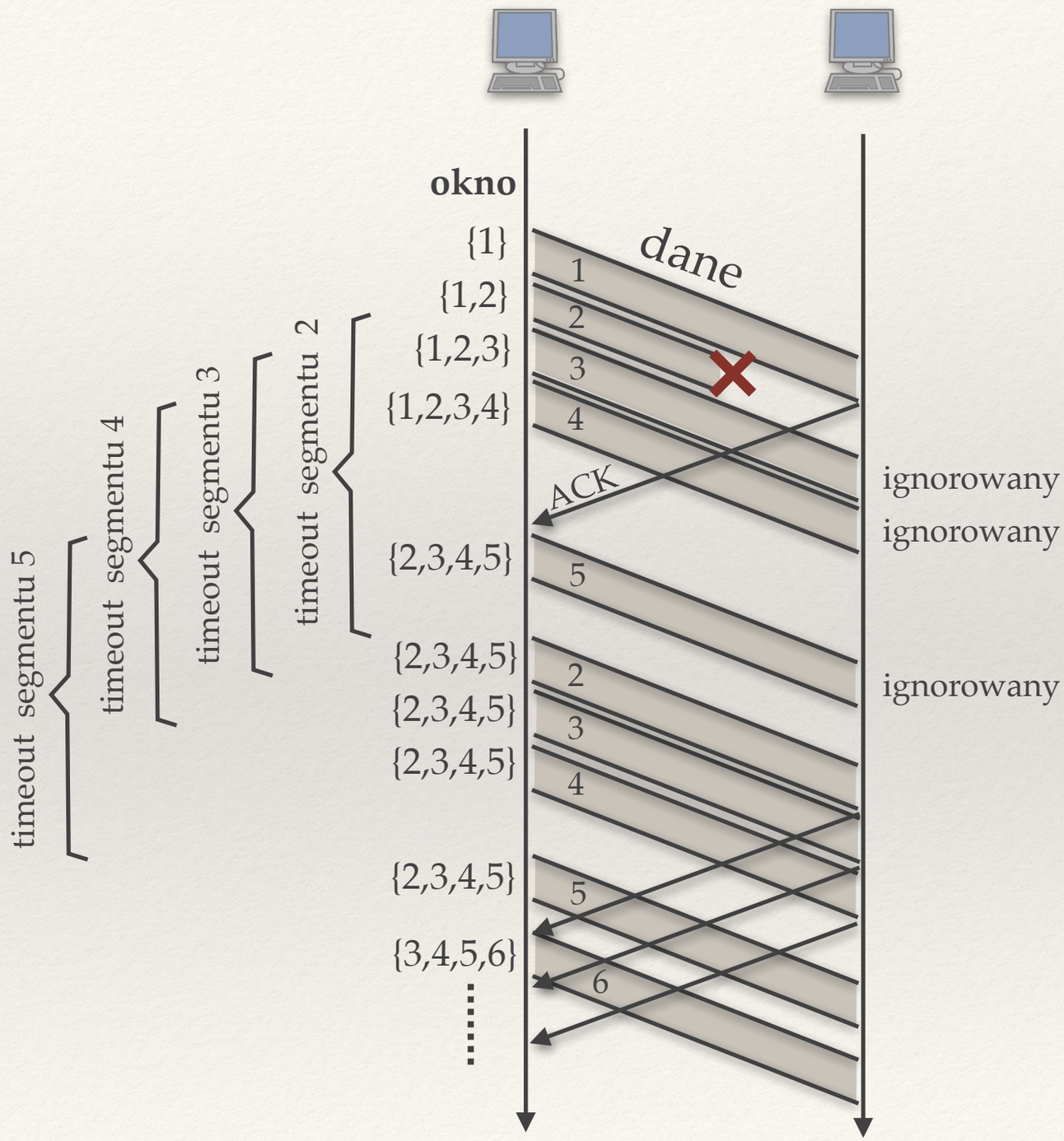


Potwierdzanie Go-Back-N: przykłady

SWS = 4, pakiety nie giną

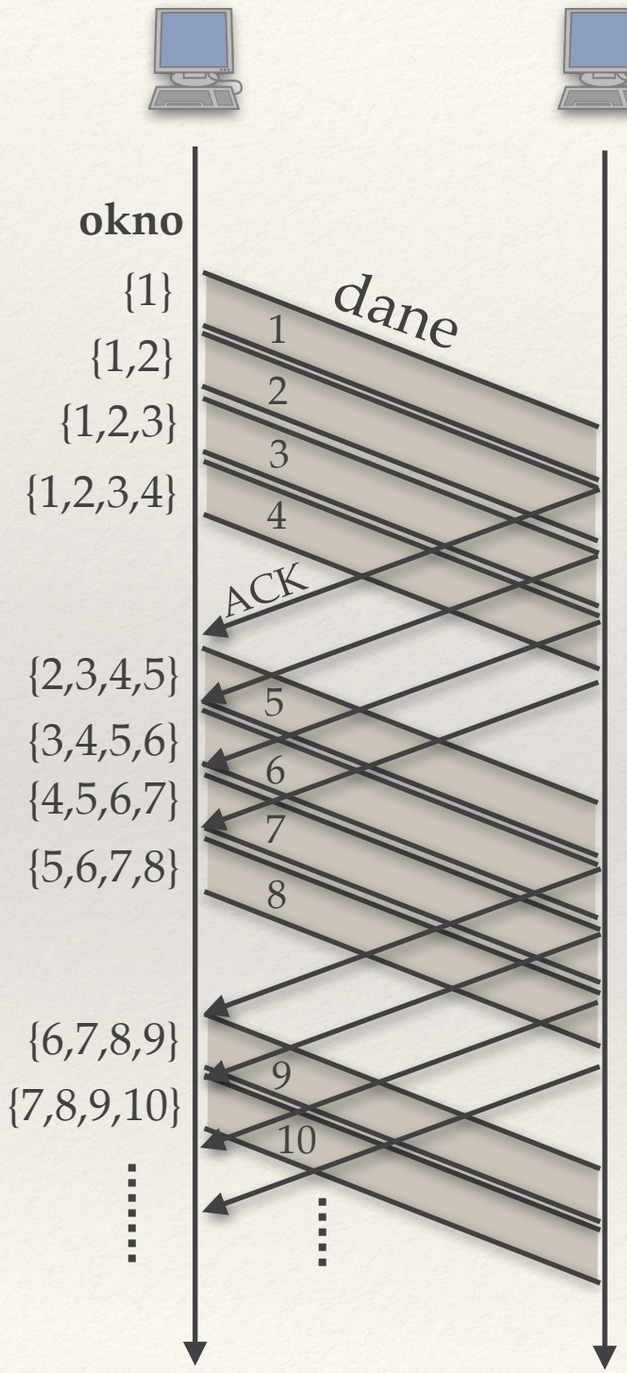


SWS = 4, pakiet ginie

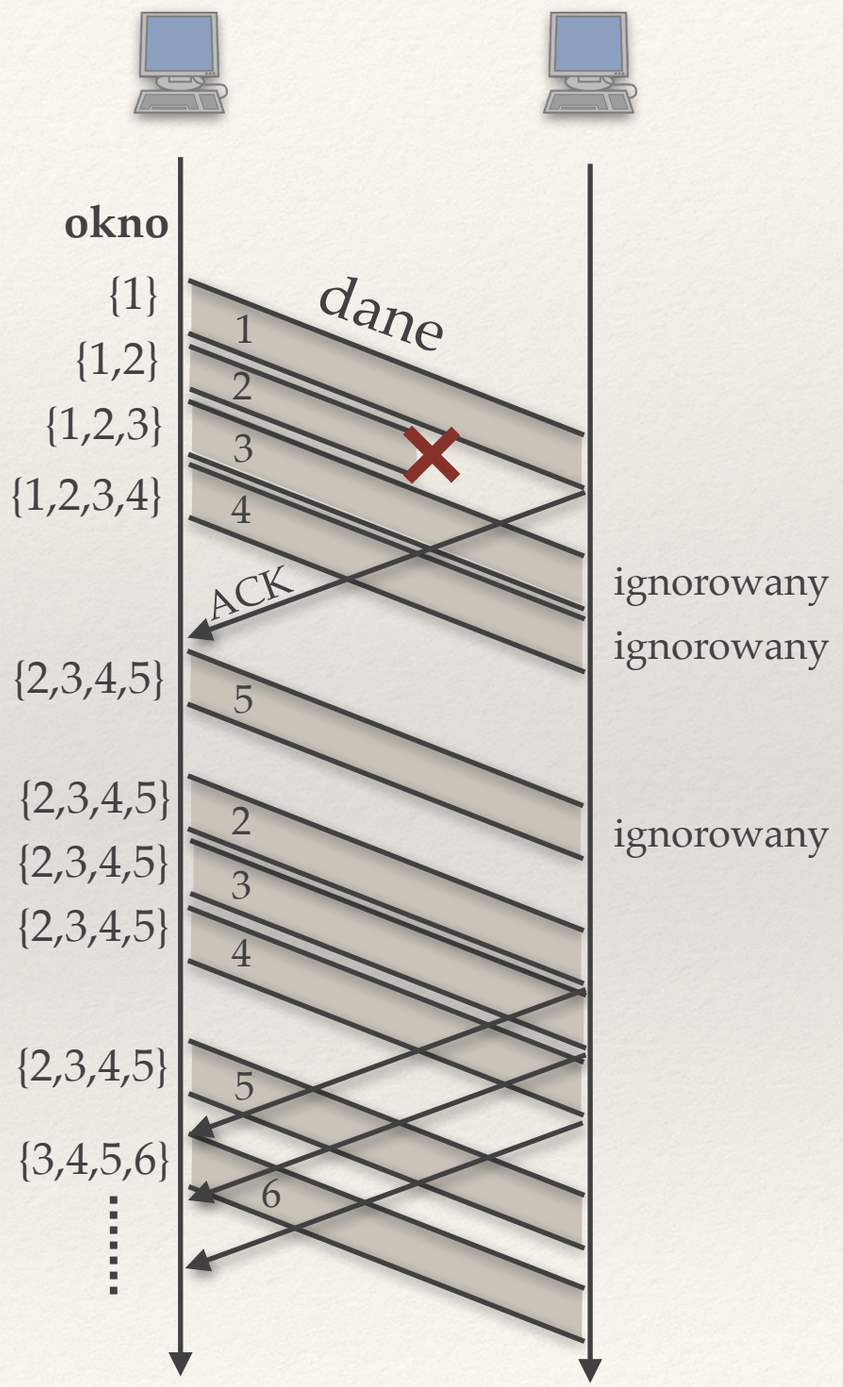


Potwierdzanie Go-Back-N: przykłady

SWS = 4, pakiety nie giną



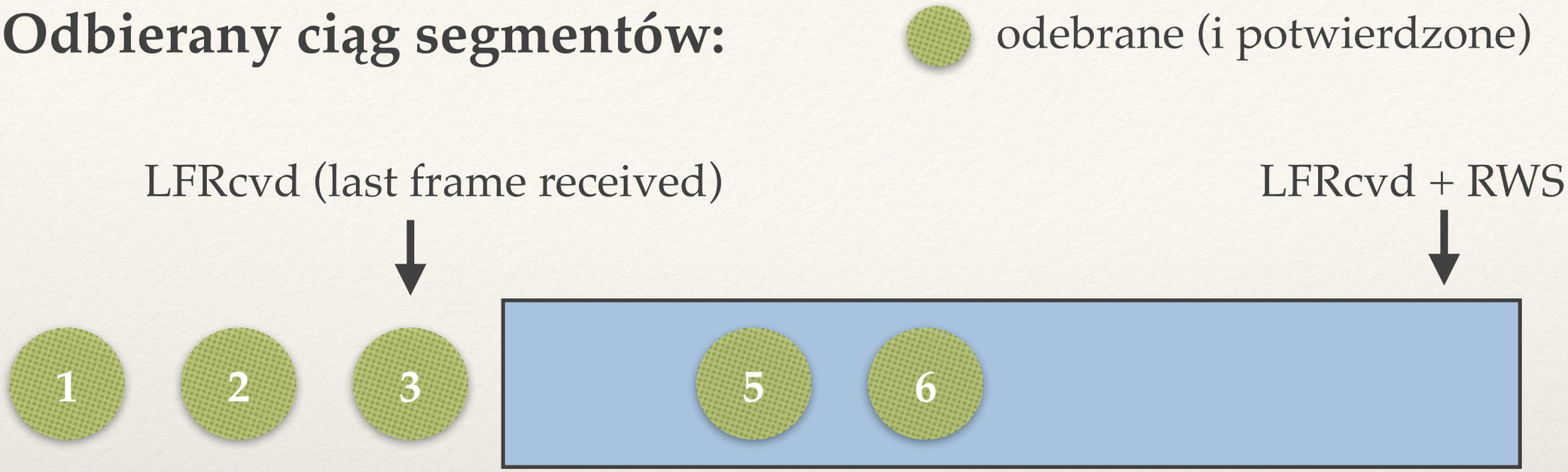
SWS = 4, pakiet ginie



Zaginięcie jednego segmentu → konieczność ponownego wysłania całego okna.

Potwierdzanie selektywne: okno odbiorcy (1)

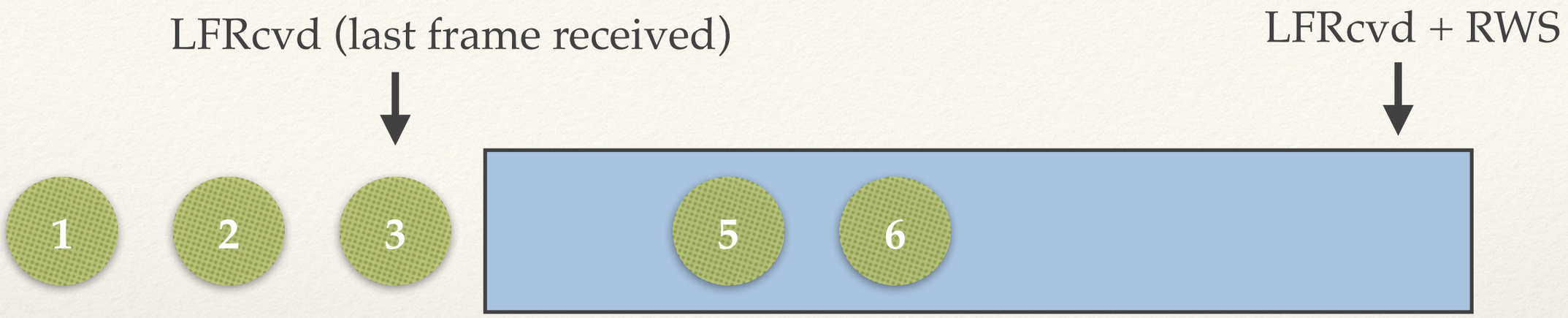
Odbierany ciąg segmentów:



Okno odbiorcy:

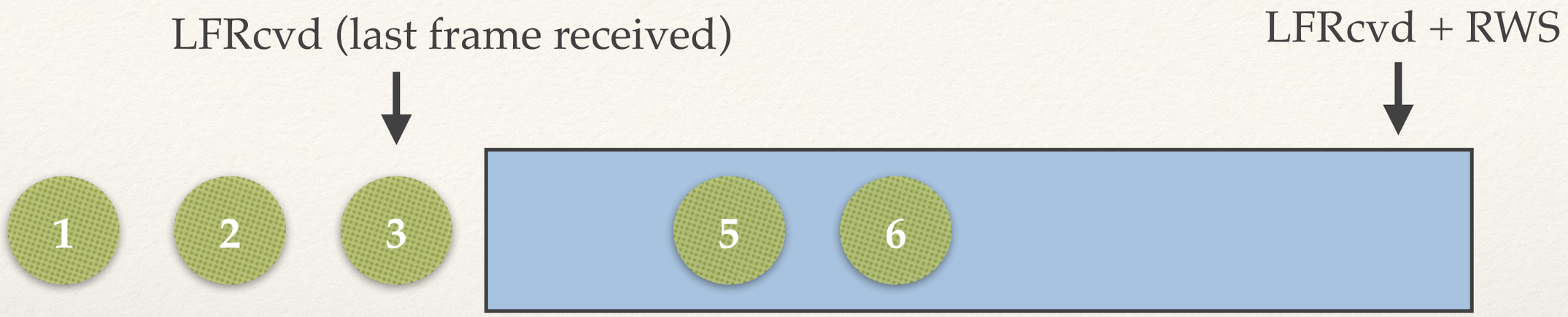
- Rozmiar RWS (*receiver window size)*.
- Segmenty od 1 do LFRcvd (*last frame received*) są otrzymane, a segment LFRcvd+1 nie.

Potwierdzanie selektywne: okno odbiorcy (2)



- **Otrzymujemy segment S**
  - LFRcvd < S ≤ LFRcvd + RWS → zapisz segment w buforze odbiorcy.
  - S ≤ LFRcvd + RWS → odeślij ACK.
  - S > LFRcvd + RWS → ignoruj segment.
  - S = LFRcvd + 1 → aktualizuj LFRcvd (przesuń okno).

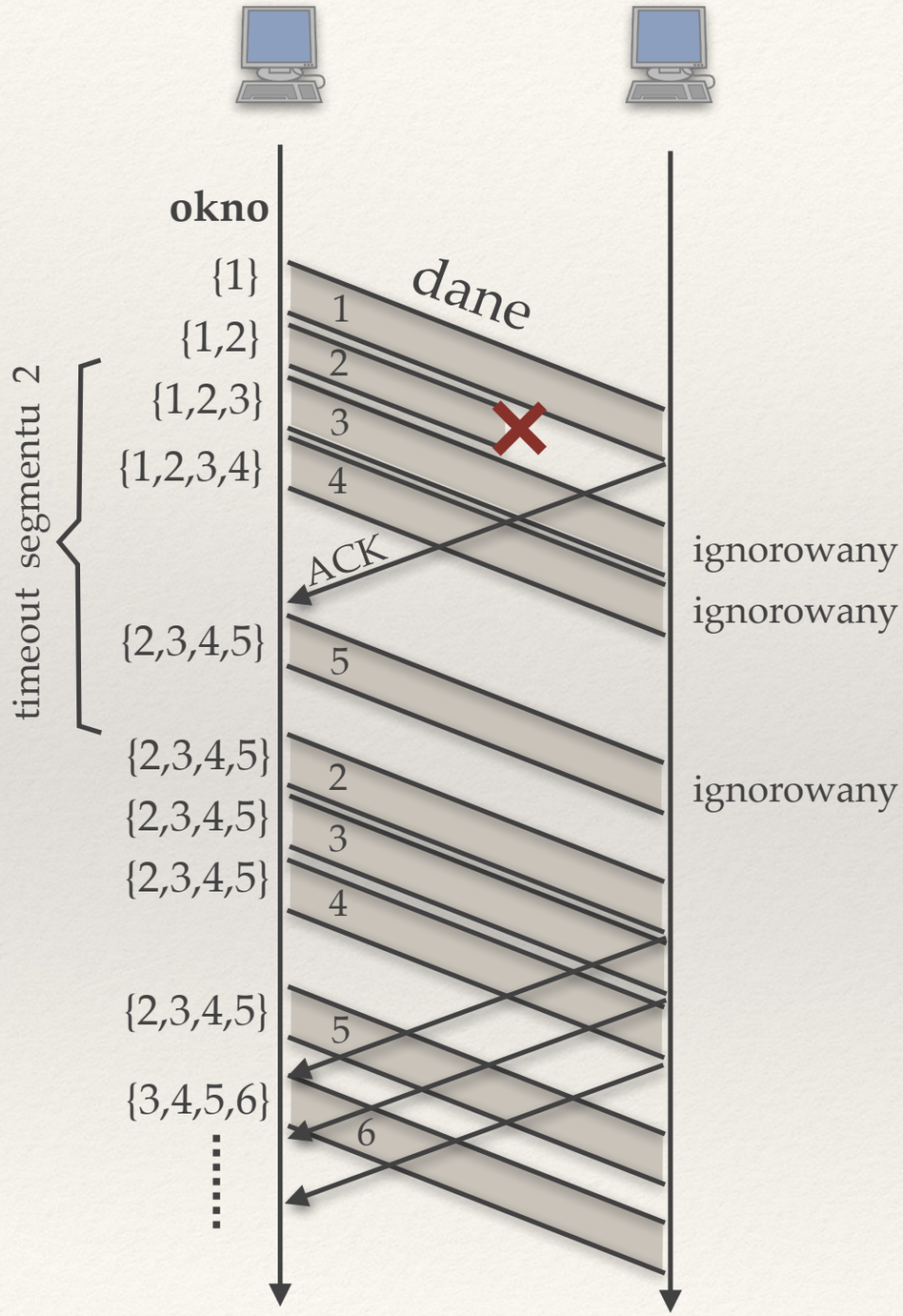
Potwierdzanie selektywne: okno odbiorcy (2)



- **Otrzymujemy segment S**
  - LFRcvd < S ≤ LFRcvd + RWS → zapisz segment w buforze odbiorcy.
  - S ≤ LFRcvd + RWS → odeślij ACK.
  - S > LFRcvd + RWS → ignoruj segment.
  - S = LFRcvd + 1 → aktualizuj LFRcvd (przesuń okno).
- **Dla RWS = 1, potwierdzanie selektywne = Go-Back-N.**

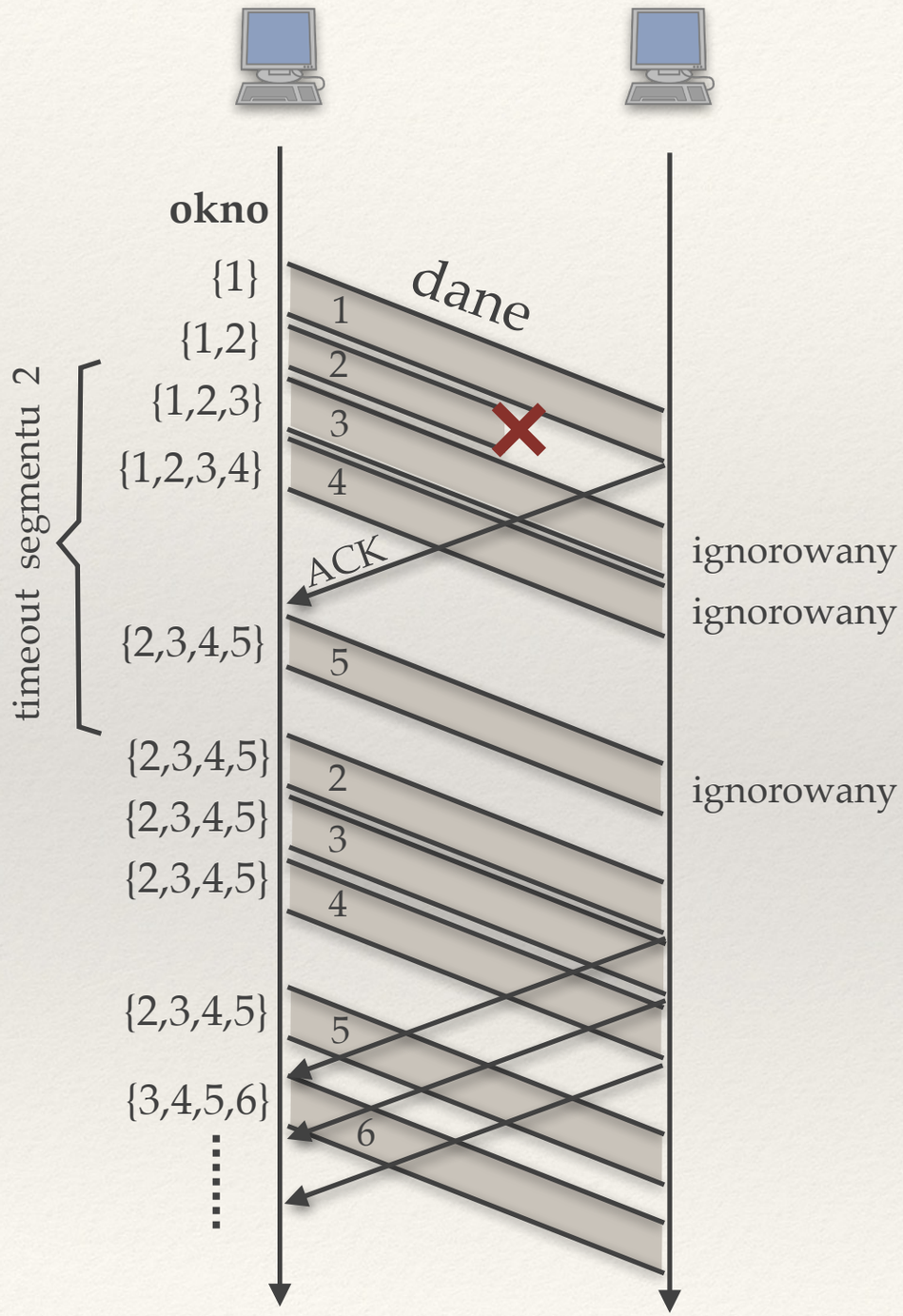
Go-Back-N vs. potwierdzanie selektywne

SWS = 4, Go-Back-N

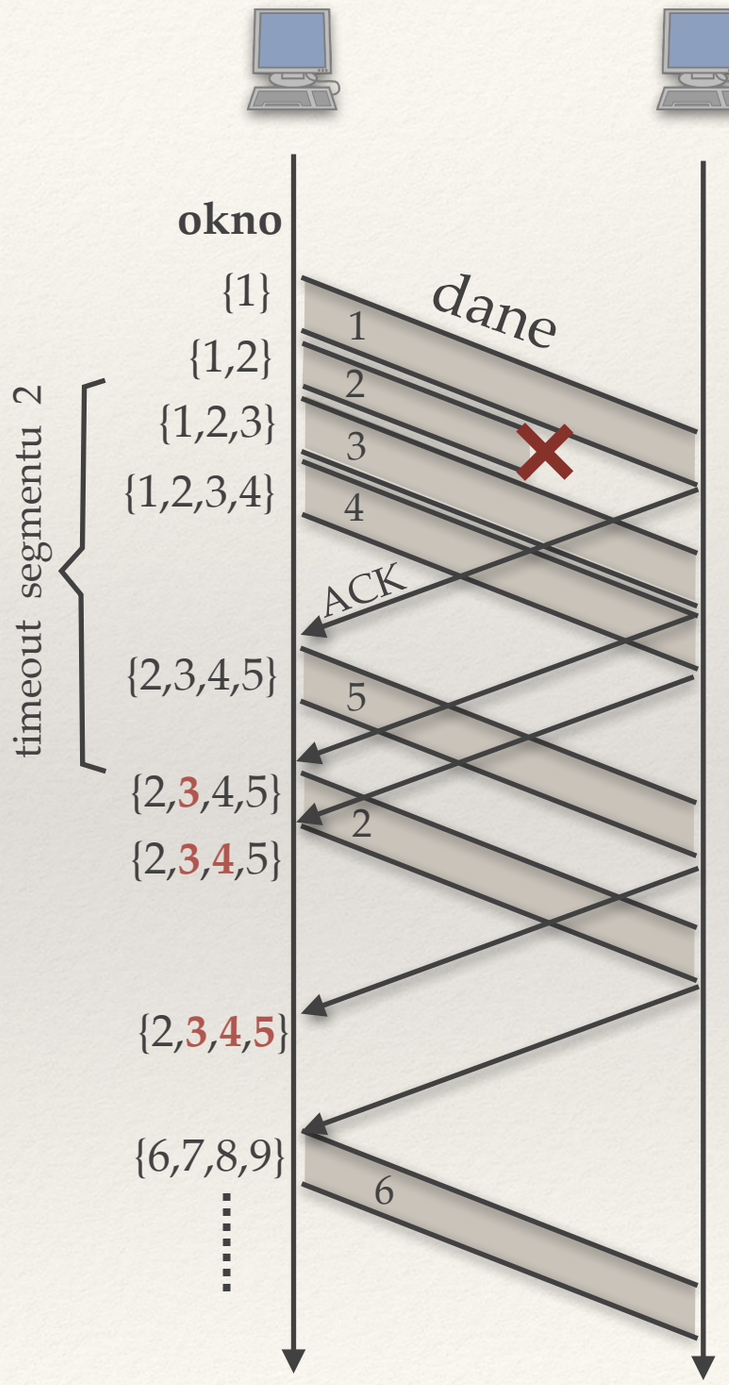


Go-Back-N vs. potwierdzanie selektywne

SWS = 4, Go-Back-N

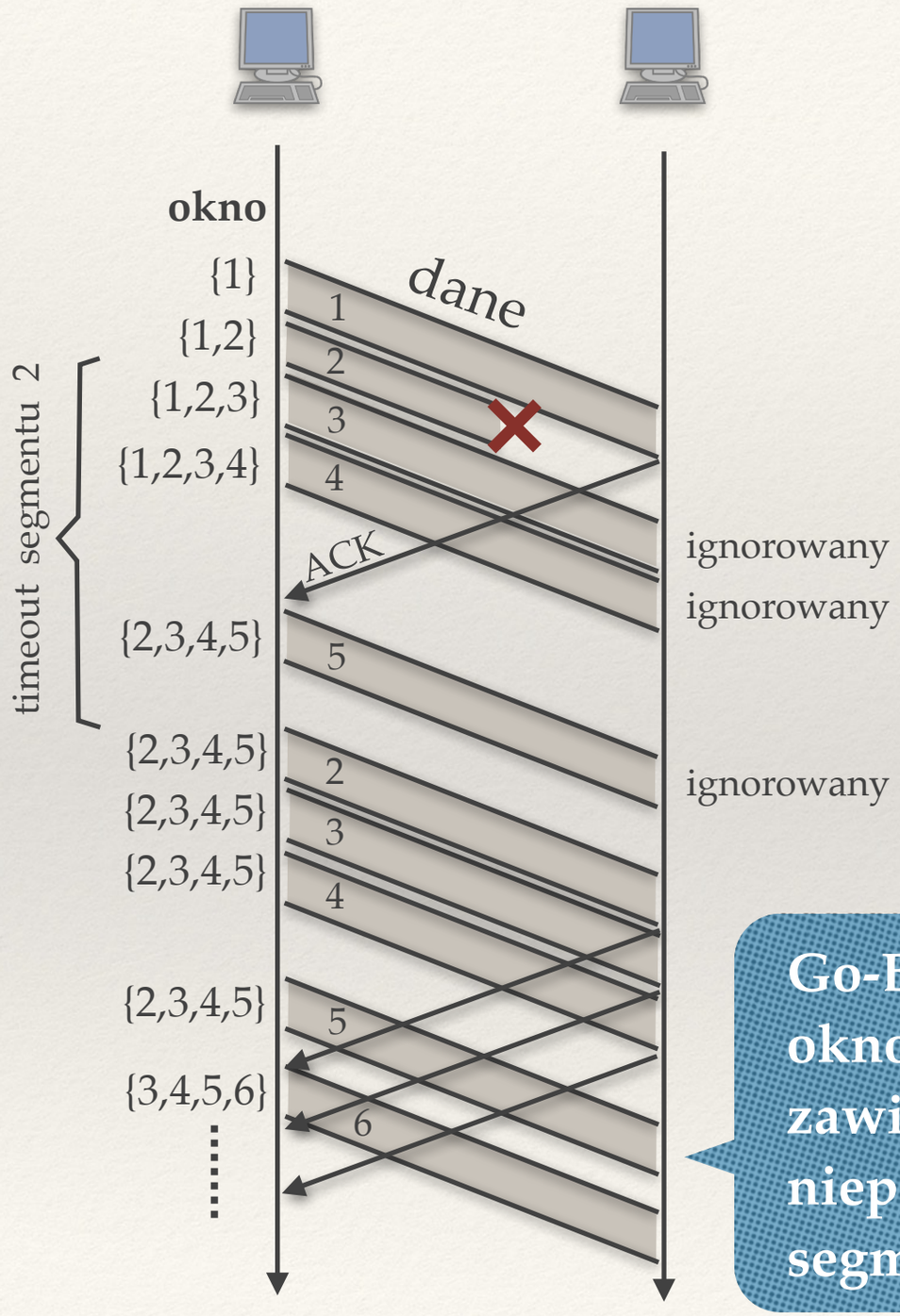


SWS = 4, potwierdzanie selektywne

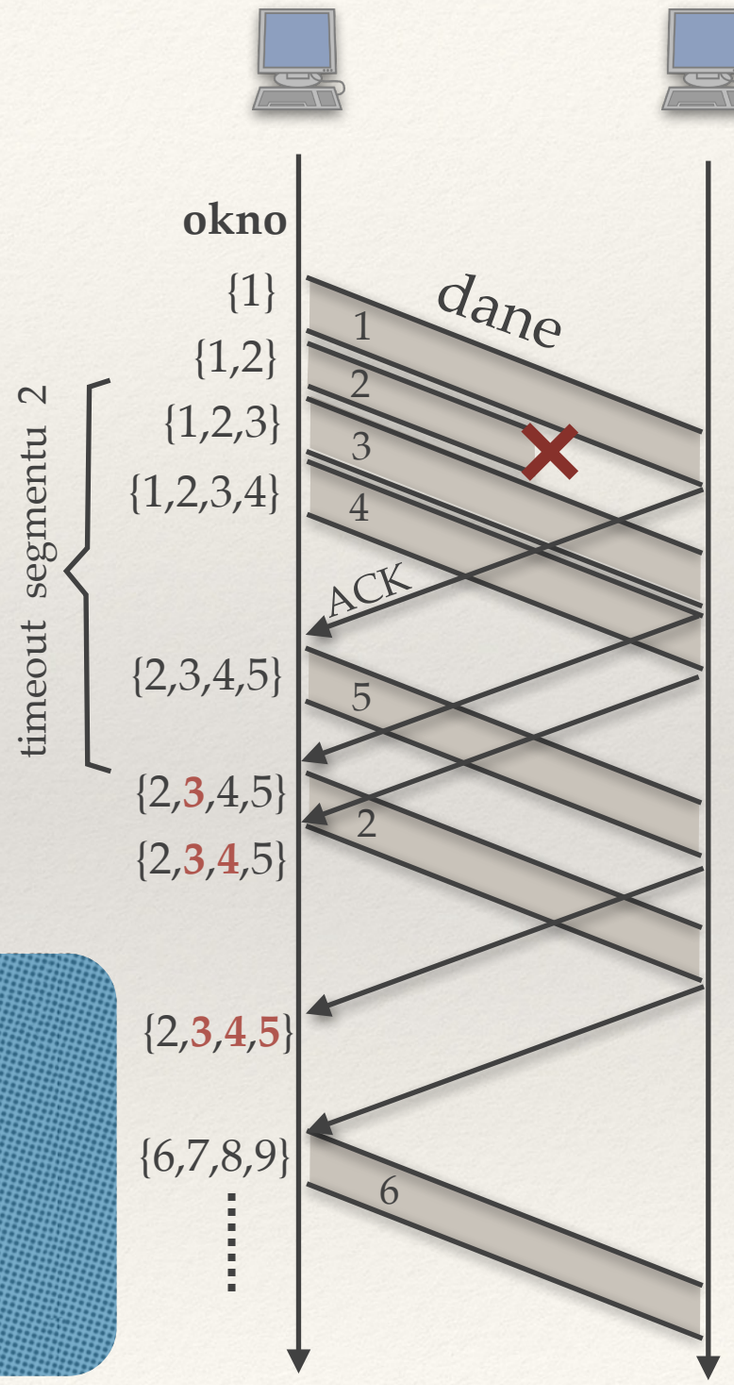


Go-Back-N vs. potwierdzanie selektywne

SWS = 4, Go-Back-N

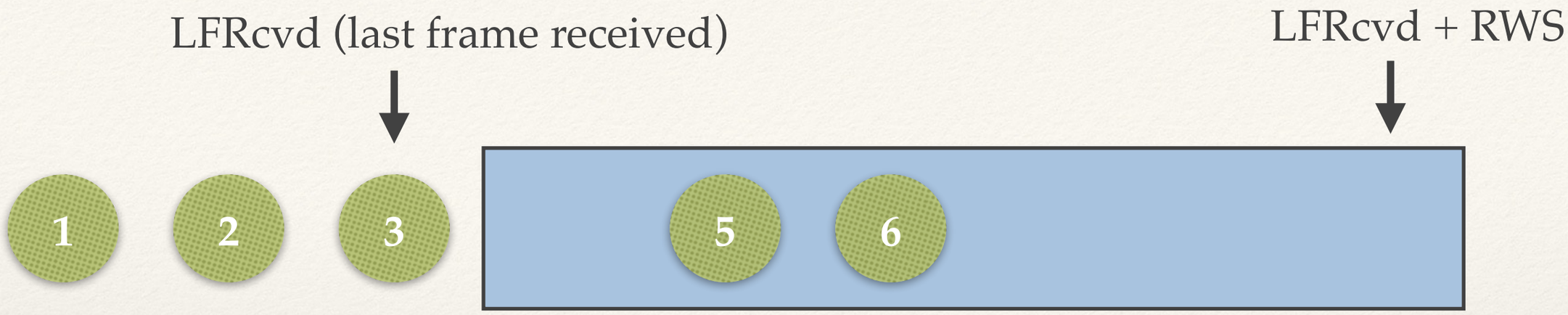


SWS = 4, potwierdzanie selektywne



Go-Back-N:
okno nadawcy zawiera wyłącznie niepotwierdzone segmenty

Potwierdzanie skumulowane



- **Poza ACK wszystko jak przy potwierdzaniu selektywnym.**

- **Wysyłanie ACK:**
  - Wysyłamy tylko jeśli otrzymamy segment S ≤ LFRcvd + RWS (jak poprzednio)
  - W razie potrzeby aktualizujemy LFRcvd (przesuwamy okno w prawo) a następnie wysyłamy **ACK dla LFRcvd**.

animacje

Potwierdzanie skumulowane vs. selektywne

- **Selektywne.** Wiemy dokładnie które pakiety dotarły

- **Skumulowane.** Możemy wprowadzić mechanizm opóźnionych potwierdzeń:
  - Jeśli są dane do wysłania w drugą stronę, to wysyłamy ACK z tymi danymi.
  - W p.p. wymuszamy określony czas (ułamek RTT) pomiędzy kolejnymi ACK.
  - Zmniejsza liczbę potwierdzeń bez generowania timeoutów.

- **Selektywne i skumulowane.** Wnioskowanie na podstawie „dziur“ w ACK:
  - Selektywne: ACK 1, ACK 2, ACK 4, ACK 5.
  - Skumulowane: ACK 1, ACK 2, ACK 2, ACK 2, ACK 2.
  - Być może warto wysłać segment 3 bez czekania na jego timeout?

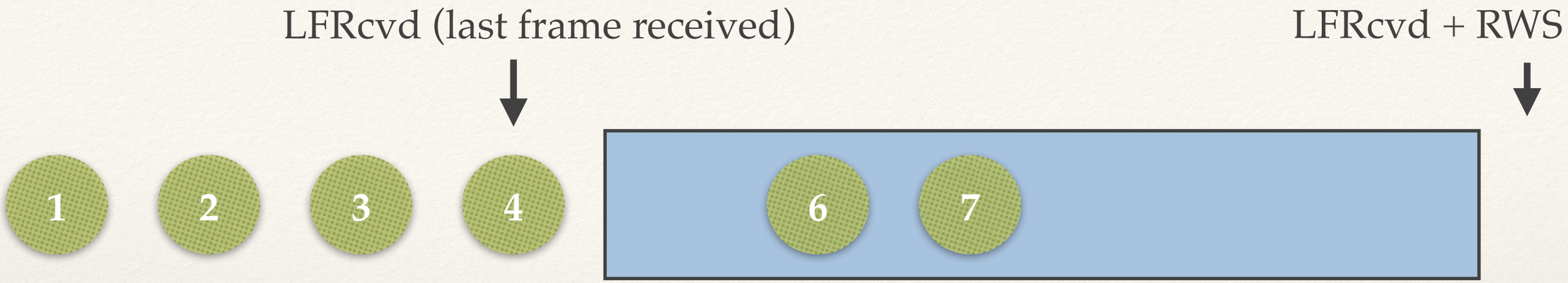
Kontrola przepływu

Kontrola przepływu a warstwa aplikacji (1)

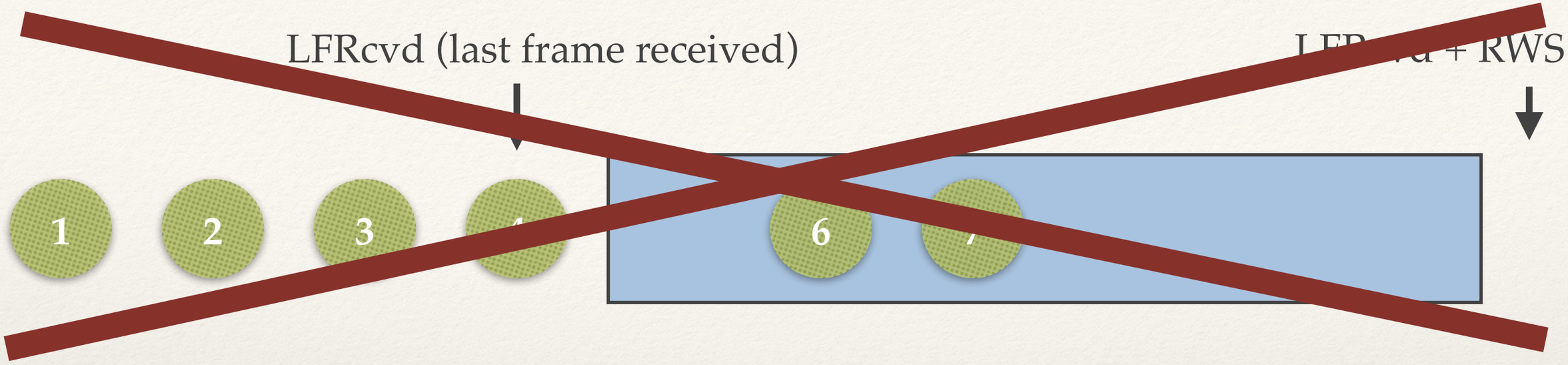
Co to znaczy „przekazujemy dane do warstwy aplikacji“?

- To warstwa aplikacji pobiera dane funkcją `read()`.
- Okno odbiorcy to bufor na odebrane pakiety, jeszcze nieprzeczytane przez aplikację.

Kontrola przepływu a warstwa aplikacji (2)

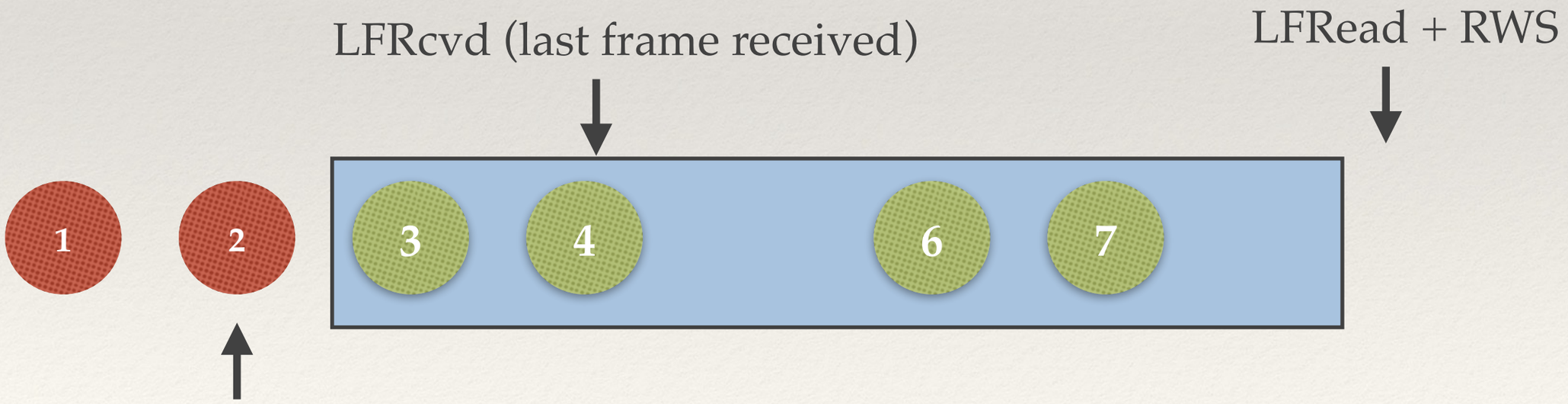


Kontrola przepływu a warstwa aplikacji (2)



odebrane, potwierdzone, przeczytane przez aplikację

odebrane (i potwierdzone) ale nieprzeczytane przez aplikację



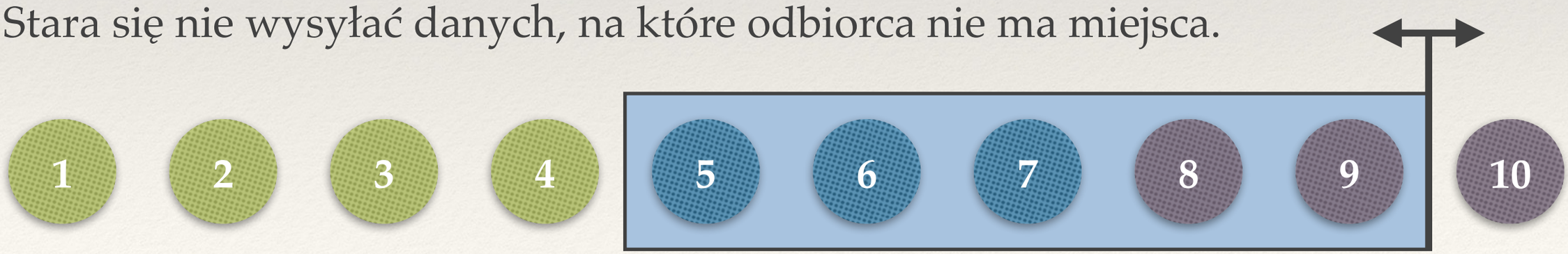
Kontrola przepływu: oferowane okno

Odbiorca:

- Oferowane okno = wolne miejsce w buforze
- Oferowane okno wysyłane nadawcy (zazwyczaj razem z ACK).
- Np.: pakiety potwierdzane, ale aplikacja wolno czyta → oferowane okno jest małe.

Nadawca:

- Zmienia SWS (rozmiar swojego okna) na rozmiar oferowanego okna.
- Stara się nie wysyłać danych, na które odbiorca nie ma miejsca.

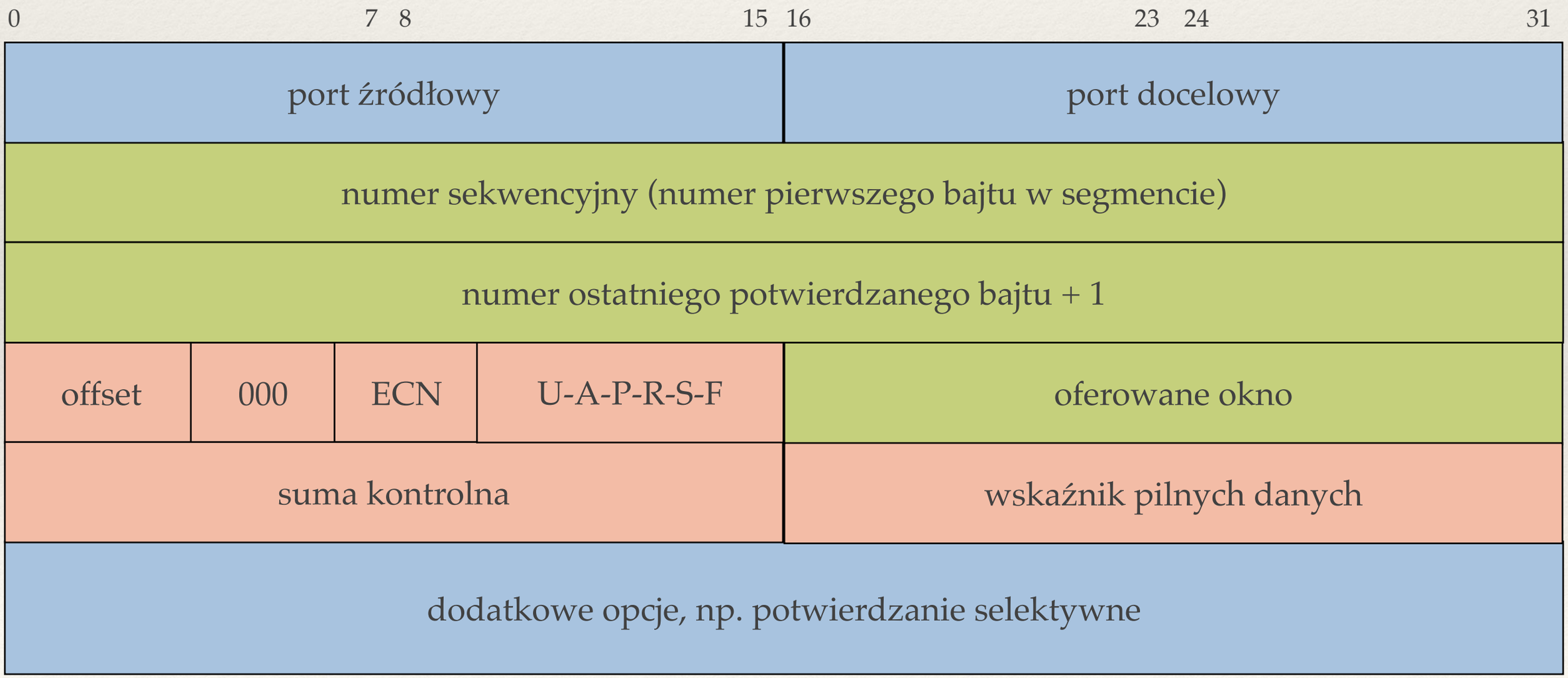


Niezawodny transport: TCP

TCP

- **Numeruje bajty, a nie segmenty.**
- **Potwierdzanie skumulowane:**
  - Uwaga: ACK n = oznacza „mam wszystko do bajtu n-1 włącznie“.
  - ACK zazwyczaj wysyłany w pakiecie razem z danymi w drugą stronę.

demonstracja



Numerowanie bajtów: przykładowe komplikacje

- Oferowaliśmy okno = 0 i aplikacja zwolniła miejsce?
  → Wyślij osobno rozmiar okna (bez ACK).

- Nie mamy danych do wysłania w drugą stronę?
  → Opóźnione wysyłanie ACK.

- Oferowane okno jest mniejsze niż MSS?
  → Czekamy z nadawaniem.

- Aplikacja generuje dane mniejsze niż MSS
  → Wyślij dopiero jeśli kiedy wszystkie poprzednie dane zostaną potwierdzone (mechanizm Nagle'a).
  → Co z interaktywnymi programami (praca zdalna)?

Ustawianie timeoutu dla segmentu

- **Obliczamy średnie RTT ważone wykładniczo czasem**
  - avg-RTT = α × avg-RTT + (1 - α) × zmierzone-RTT
  - RTT segmentów stałe → avg-RTT zbiega do tej wartości.
  - W podobny sposób mierzymy wariancję RTT (var-RTT).

- **RTO (*retransmission timeout*) = 2 × avg-RTT + 4 × var-RTT**

Gdzie implementować?

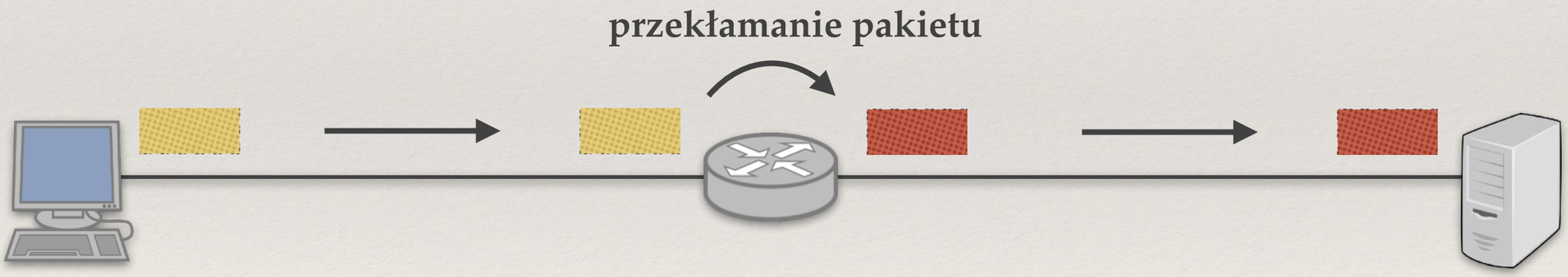
Gdzie implementować warstwę transportową?

Warstwa sieciowa:

- Implementowana na routerach i urządzeniach końcowych.

Warstwa transportowa:

- Wczesna filozofia: niezawodność dostarczania zapewniać na każdym łączu.
- Problem: błąd może nie być związany z łączem!



- Wniosek: niezawodność dostarczania i tak musi być kontrolowana na urządzeniach końcowych.

Zasada end-to-end (1)

Słaba wersja

- Niezawodne przesyłanie danych musi być implementowane na urządzeniach końcowych, ale *warstwy niższe mogą w tym pomagać.*

Silna wersja

- Niezawodne przesyłanie danych musi być implementowane na urządzeniach końcowych, *warstwy niższe nie powinny się tym w ogóle zajmować.*

Zasada end-to-end (2)

Która wersja zasady? Spór filozoficzny na przykładzie TCP + WiFi.

- TCP działa dobrze tylko jeśli łącza są w miarę niezawodne.
- Łącza bezprzewodowe tracą średnio 20-80% pakietów
- Potwierdzanie i retransmisja na poziomie warstwy łącza danych.

- Łamiemy model warstwowy i silną wersję zasady.
  - Krótkoterminowe duże korzyści.
  - Ale być może trudności we wprowadzeniu innej wersji warstwy transportowej.

Lektura dodatkowa

- Kurose & Ross: rozdział 3.
- Tanenbaum: rozdział 6.
- Dokumentacja online:
  - `http://www.networksorcery.com/enp/protocol/tcp.htm`

Zagadnienia

- Co może stać sie z przesyłanym ciągiem pakietów IP podczas zawodnego i niezawodnego transportu?
- Co to jest kontrola przepływu?
- Czym różnią się protokoły UDP i TCP? Podaj zastosowania każdego z nich.
- Co to jest segmentacja? Dlaczego segmenty mają ograniczoną wielkość? Rozwiń skrót MSS.
- Jak nazywają się jednostki danych przesyłane w kolejnych warstwach?
- Jak małe pakiety zmniejszają opóźnienie przesyłania danych?
- Wytłumacz znaczenie skrótów RTT i RTO. Na jakiej podstawie ustalana jest wartość RTO?
- Jak protokoły niezawodnego transportu wykrywają duplikaty pakietów i potwierdzeń?
- Opisz algorytm Stop-and-Wait. Jakie są jego wady i zalety?
- Do czego służą numery sekwencyjne w niezawodnym protokole transportowym?
- Opisz algorytm okna przesuwnego.
- Jaki jest związek między rozmiarem okna a BDP (bandwidth-delay product)?
- Opisz i porównaj następujące mechanizmy potwierdzania: Go-Back-N, potwierdzanie selektywne, potwierdzanie skumulowane.
- Dlaczego istotne jest potwierdzanie odbioru duplikatów segmentów?
- Co to jest okno oferowane? Jak pomaga w kontroli przepływu?
- Jakie mechanizmy niezawodnego transportu i kontroli przepływu implementowane są w protokole TCP?
- Na czym polega opóźnione wysyłanie ACK w protokole TCP?
- Na czym polega mechanizm Nagle'a? Kiedy nie należy go stosować?
- Co oznaczają pola ,,numer sekwencyjny'' i ,,numer potwierdzenia'' w nagłówku TCP?
- Czy warstwa transportowa implementowana jest na routerach? Dlaczego?
- Sformułuj słabą i silną zasadę end-to-end.