Transport

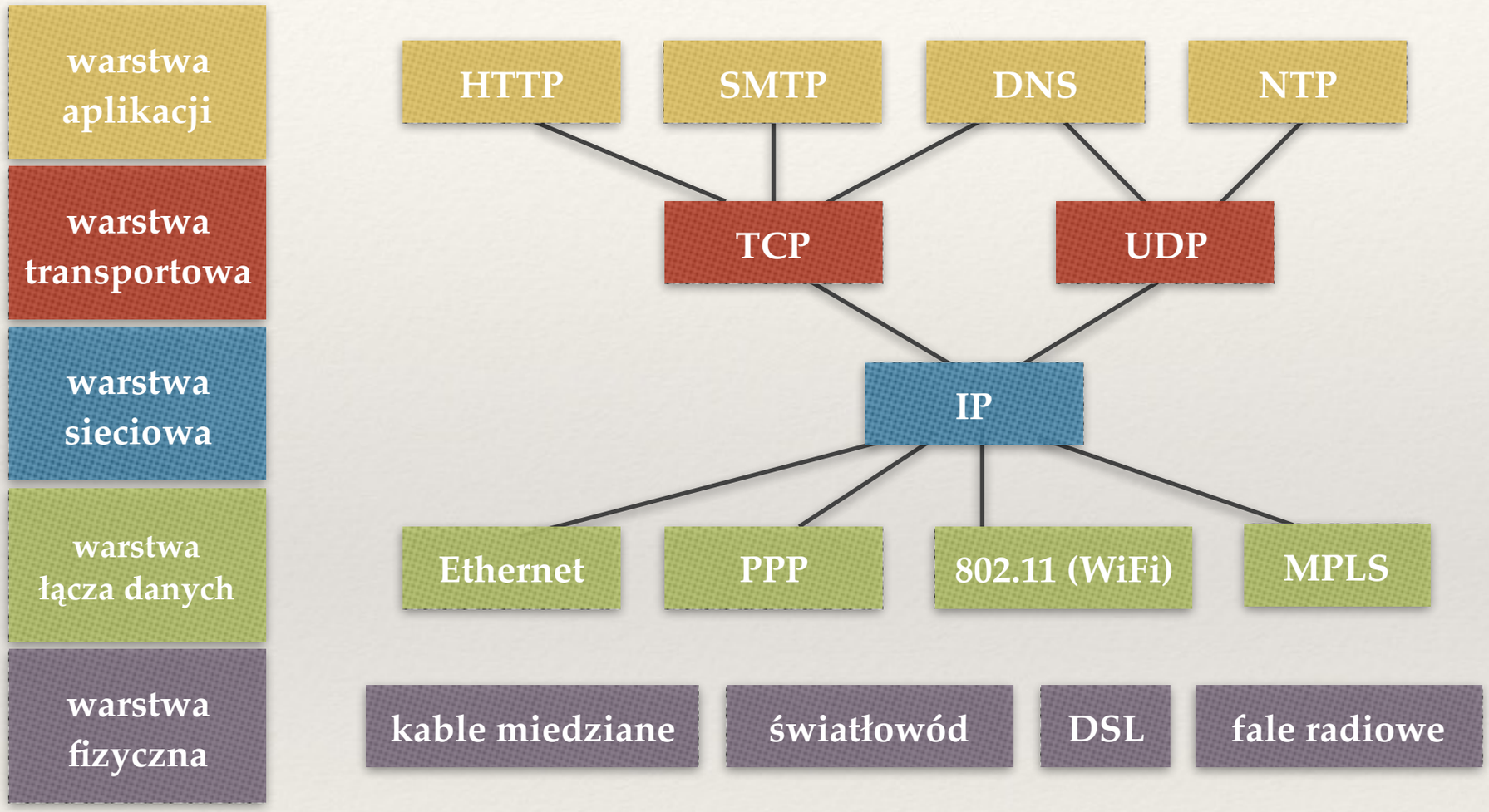
część 3: kontrola przeciążenia

Sieci komputerowe

Wykład 13

Marcin Bieńkowski

Protokoły w Internecie



Podsumowanie mechanizmów warstwy transportowej

Warstwa sieci = zawodna usługa przesyłania pakietów

- Tylko zasada dołożenia wszelkich starań (*best effort*)

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

Podstawowe mechanizmy w warstwie transportowej

- **Segmentacja:** dzielimy przesyłany strumień danych na kawałki; **dla uproszczenia będziemy wszystko liczyć w segmentach**.

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

- **Timeout (przekroczenie czasu oczekiwania):** jeśli nie otrzymamy potwierdzenia przez pewien czas (typowy dla łącza, np. 2 * RTT).

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

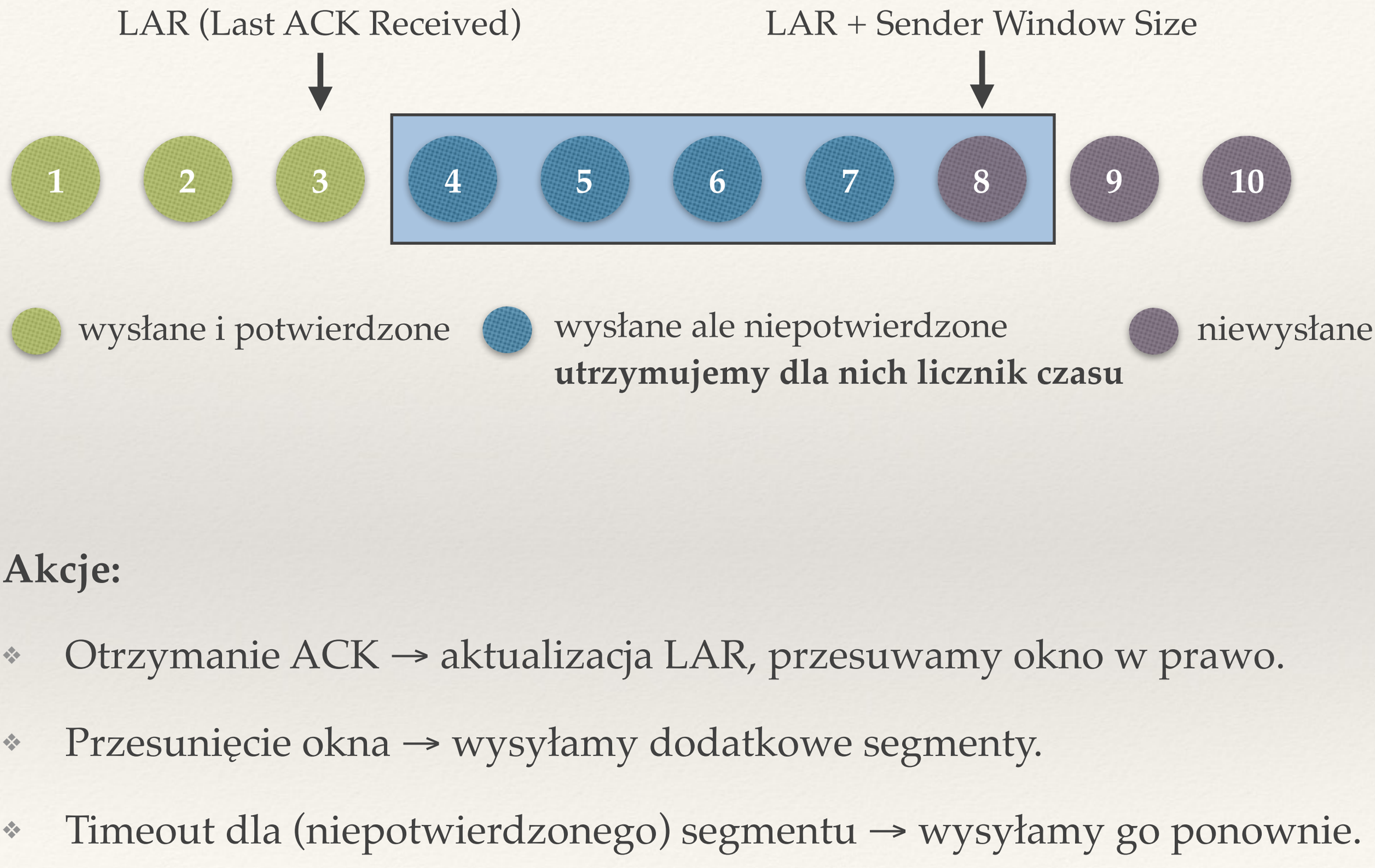
Co umiemy już zapewniać?

- **Niezawodny transport**
  - Mechanizmy ARQ (Automatic Repeat reQuest) = wysyłanie do skutku

- **Kontrola przepływu**
  - Nadawca powinien dostosowywać prędkość transmisji do szybkości z jaką odbiorca może przetwarzać dane.

- **Jak?**
  - Najczęściej: okno przesuwne + potwierdzenia skumulowane.

Okno nadawcy (przy potwierdzeniach skumulowanych)



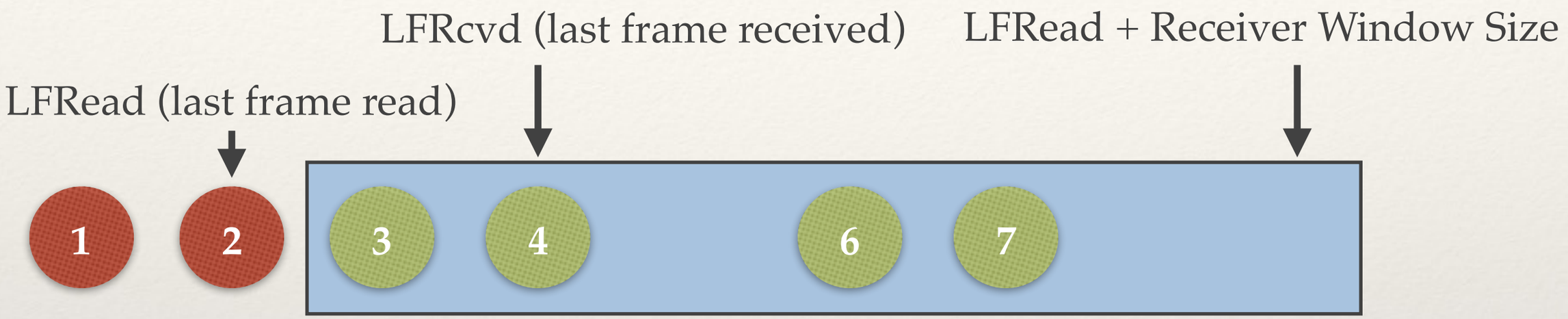
Akcje:

- Otrzymanie ACK → aktualizacja LAR, przesuwamy okno w prawo.
- Przesunięcie okna → wysyłamy dodatkowe segmenty.
- Timeout dla (niepotwierdzonego) segmentu → wysyłamy go ponownie.

Okno odbiorcy (przy potwierdzeniach skumulowanych)

odebrane, potwierdzone, przeczytane przez aplikację

odebrane (i potwierdzone) ale nieprzeczytane przez aplikację



- **Otrzymujemy segment S**
  - LFRead < S ≤ LFRead + RWS → zapisz segment w buforze odbiorczym.
  - Zaktualizuj LFRcvd.
  - S ≤ LFRead + RWS → odeślij ACK dla segmentu LFRcvd.

Jeszcze jeden ważny mechanizm: opóźnione potwierdzenia

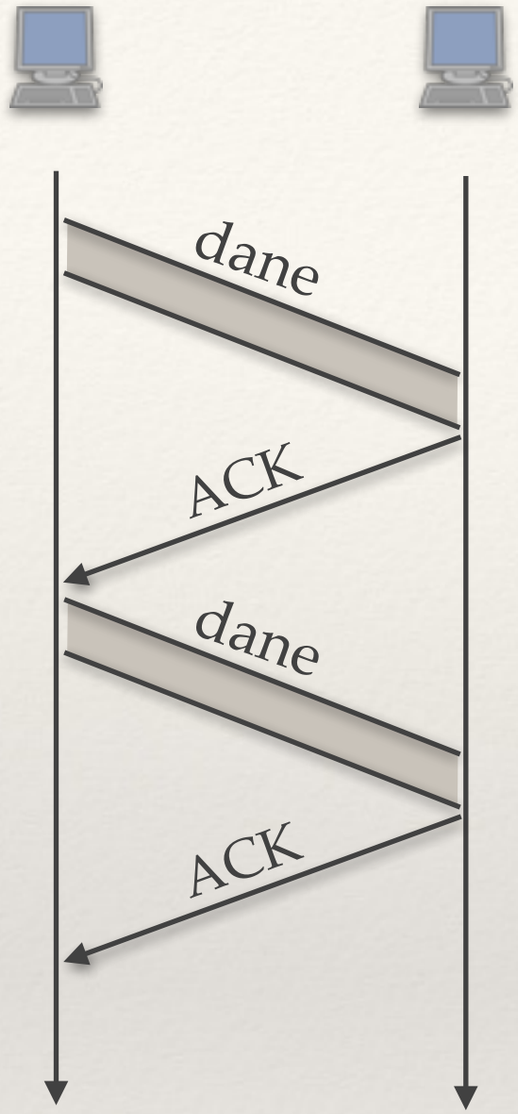
- Potwierdzanie każdego segmentu osobno jest nieefektywne
- Potwierdzenia można wysyłać „przy okazji“ razem z danymi.
- Jeśli nie ma danych do wysłania, to mechanizm opóźnionych potwierdzeń wymusza upłynięcie pewnego czasu (np. 200 ms) między kolejnymi potwierdzeniami

SWS (rozmiar okna nadawcy)

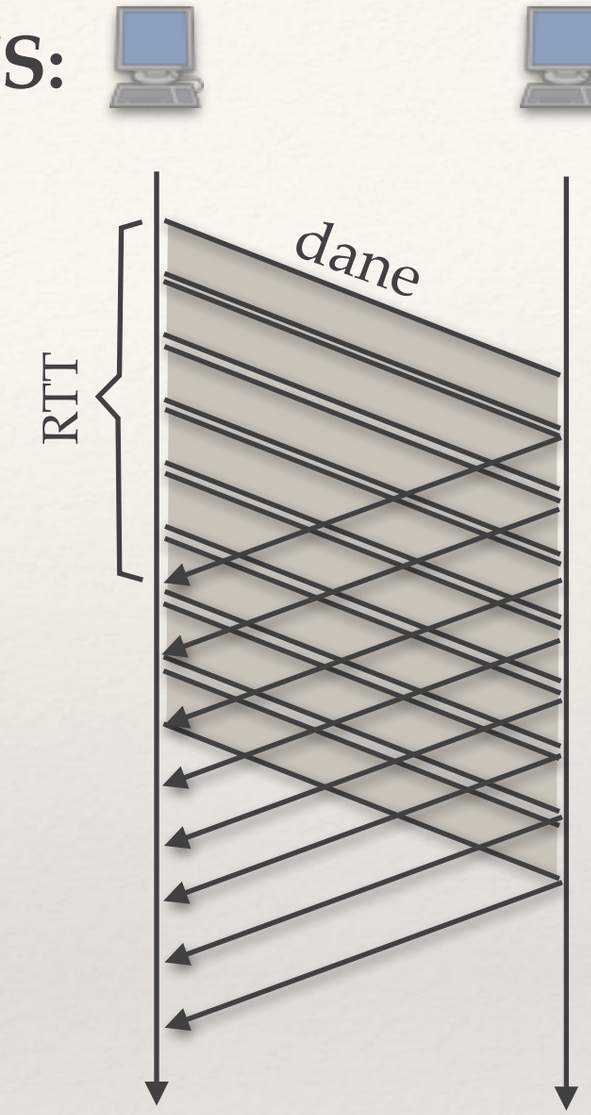
- O*ferowane okno* = rozmiar wolnego miejsca w buforze odbiorcy.
- Wysyłane przez odbiorcę z każdym ACK.
- Nadawca ustala: SWS = oferowane okno.

SWS a prędkość transmisji

SWS = 1:



większe SWS:

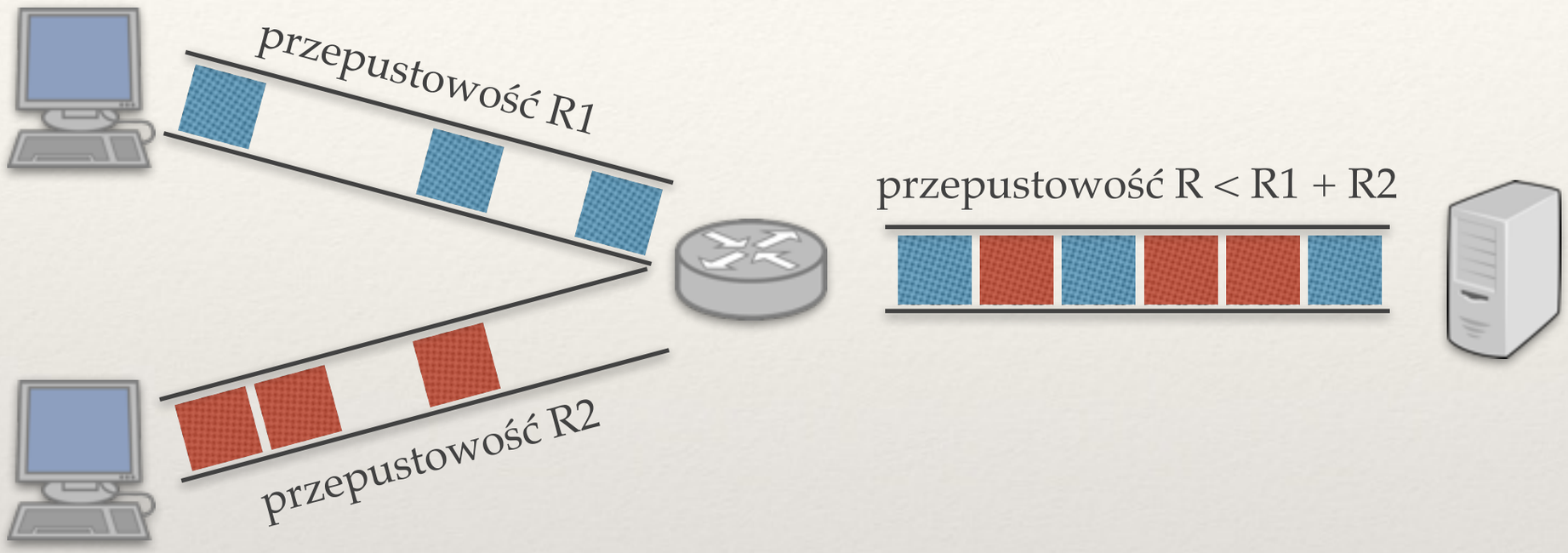


- Dane wysyłane są ze średnią prędkością SWS / RTT.
- Okno mniejsze od BDP = przepustowość * RTT
  → nadawca nie jest w stanie wykorzystać całego łącza.

Problem przeciążenia

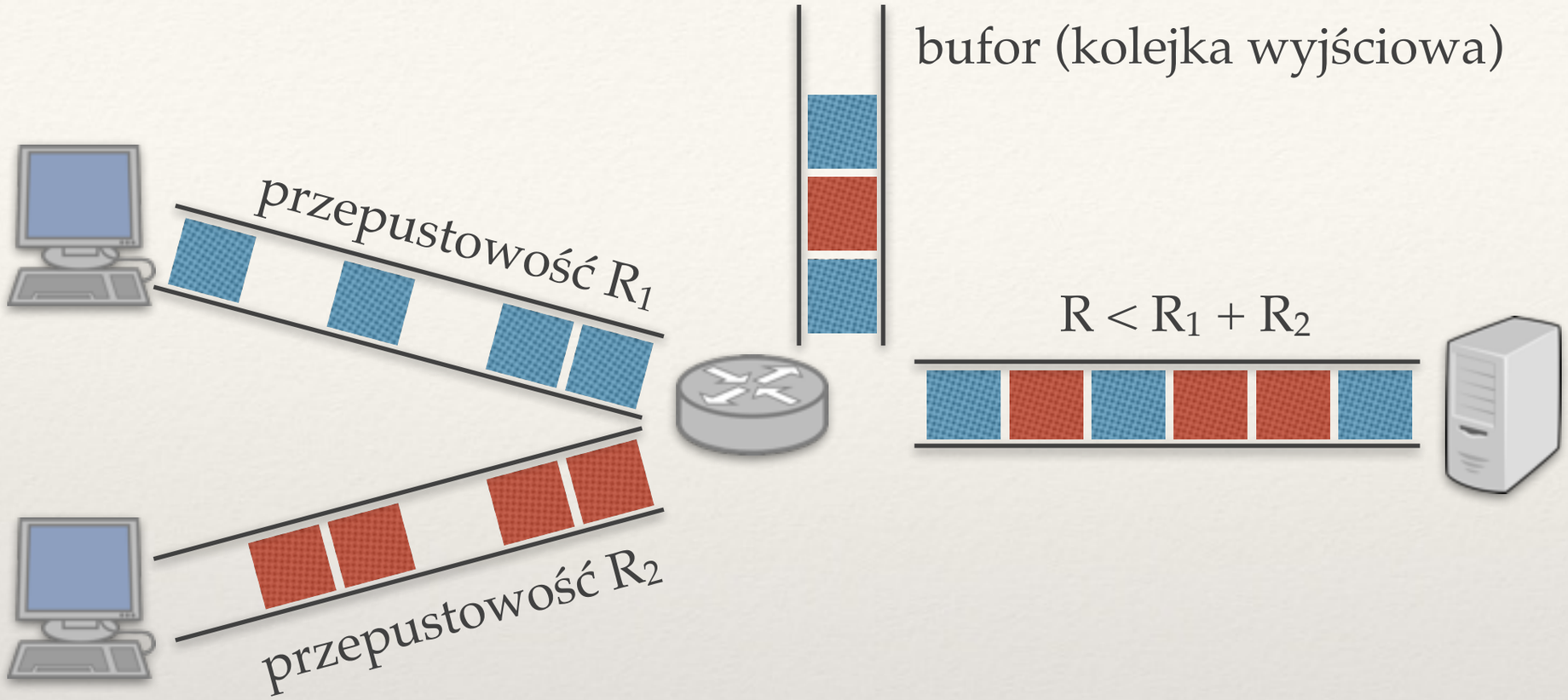
Statystyczny multipleksing

Różne strumienie danych przesyłane tym samym łączem.



Założenie: różne komputery wykorzystują łącze w innych momentach
→ lepsze wykorzystanie łącza.

Bufory



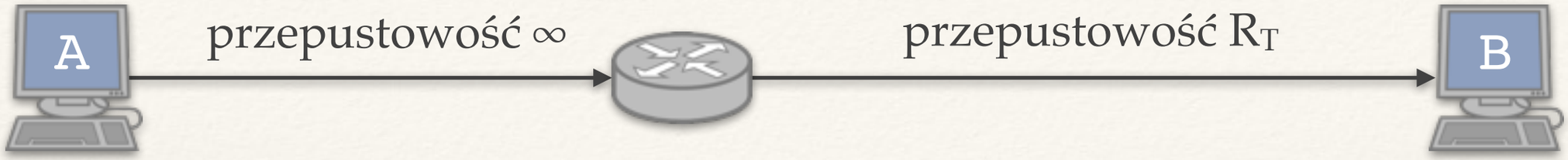
Bufory przy łączach wyjściowych:

- Pomagają przy **przejściowym** nadmiarze pakietów.
- Jeśli bufor się przepełni (**przeciążenie**) → pakiety są odrzucane.

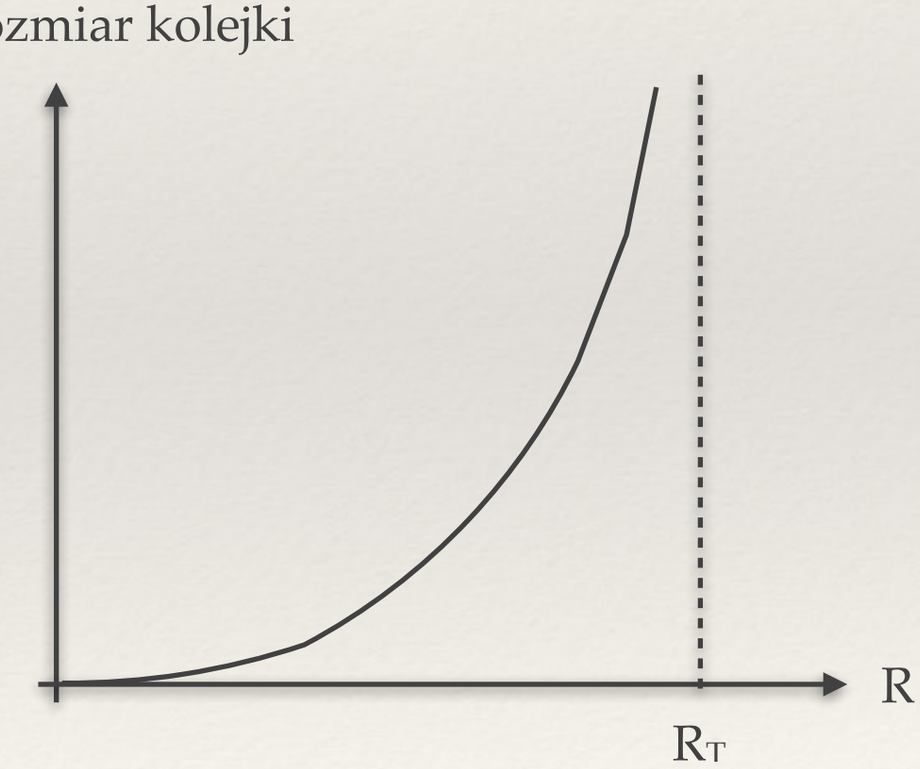
Większy bufor?

- Pozwoli przetrwać dłuższy czas bez utraty pakietów…
- … ale w przypadku stałego przeładowania pakietami:
  - Kolejki rosną.
  - Opóźnienie rośnie.
  - Mechanizmy retransmisji zaczną wysyłać ponownie pakiety.
  - Retransmitowane pakiety jeszcze bardziej zwiększają długość kolejki…

Średni rozmiar kolejki

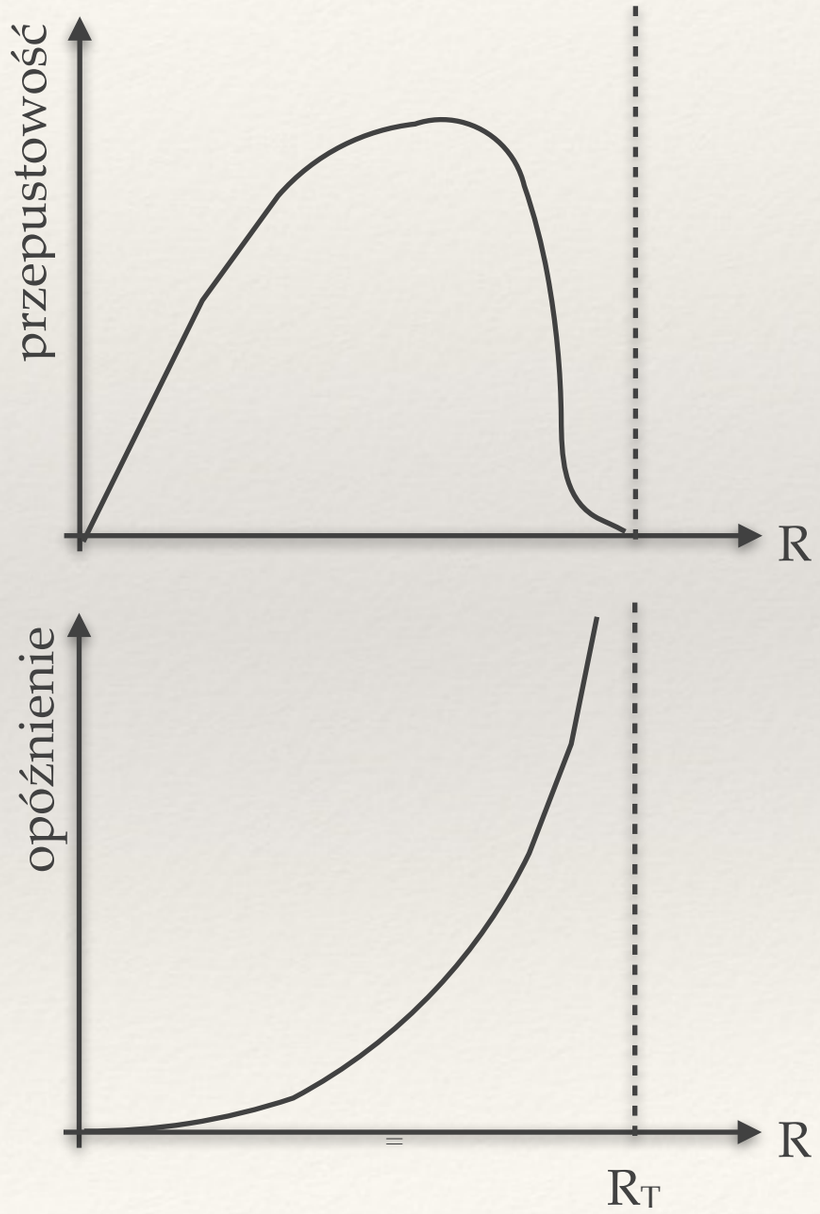


- Zakładamy, że pakiety wysyłane są od `A` do `B` losowo, ze **średnią** prędkością R.
- Matematyczna teoria kolejek → wykres rozmiaru kolejki.
- Opóźnienie jest liniową funkcją rozmiaru kolejki.



Opóźnienie i faktyczna przepustowość

- Potrzebujemy sposobu na powstrzymanie strumieni danych.
- W przeciwnym przypadku:
  - bardzo duże opóźnienia,
  - bardzo małą faktyczną przepustowość (dużo duplikatów).

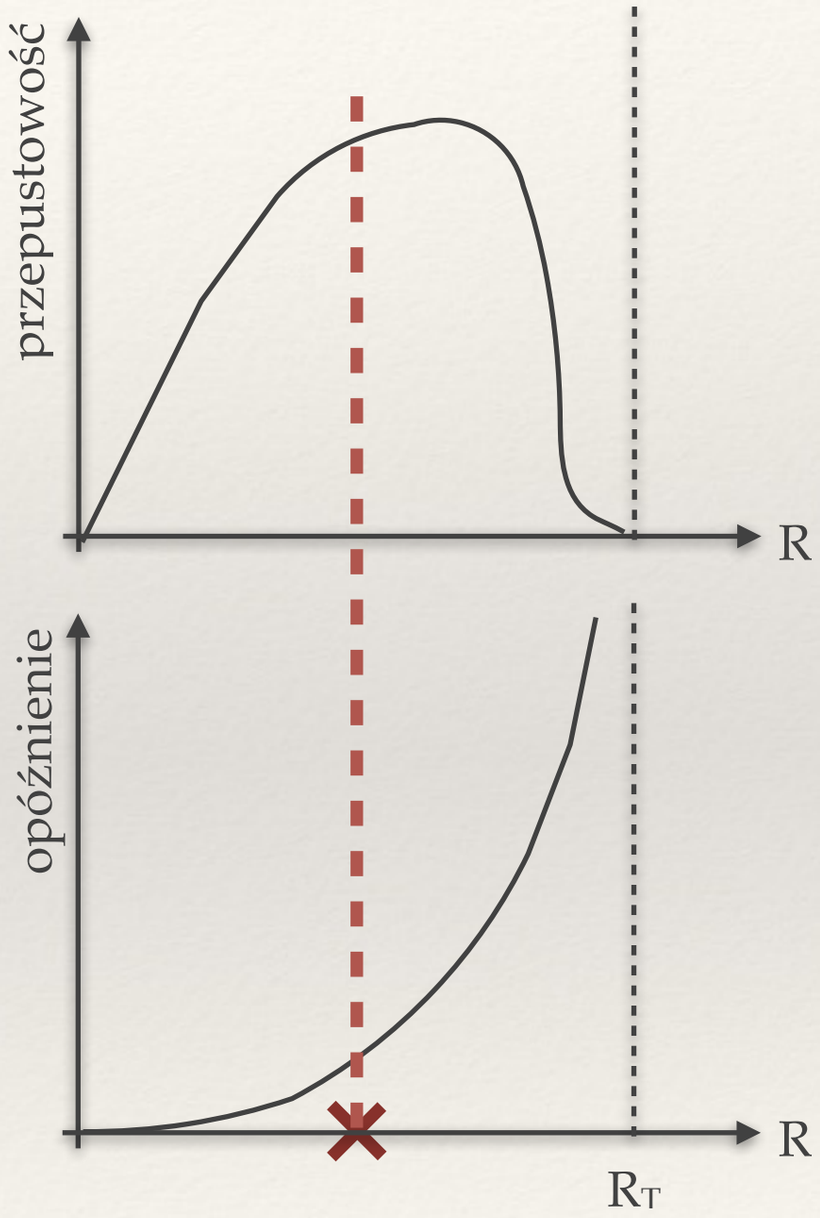


Bufory w routerach

Stan bliski przeciążenia jest dobry.

- Pełne kolejki
  → większe opóźnienia.
- Puste kolejki
  → moglibyśmy nadawać szybciej!

- Chcemy mieć mechanizm, który będzie utrzymywać obciążenie w okolicach optymalnego punktu.



Cele kontroli przeciążenia

- **Wysokie wykorzystanie łączy.**
  - Zajęte łącza = szybkie przesyłanie danych.

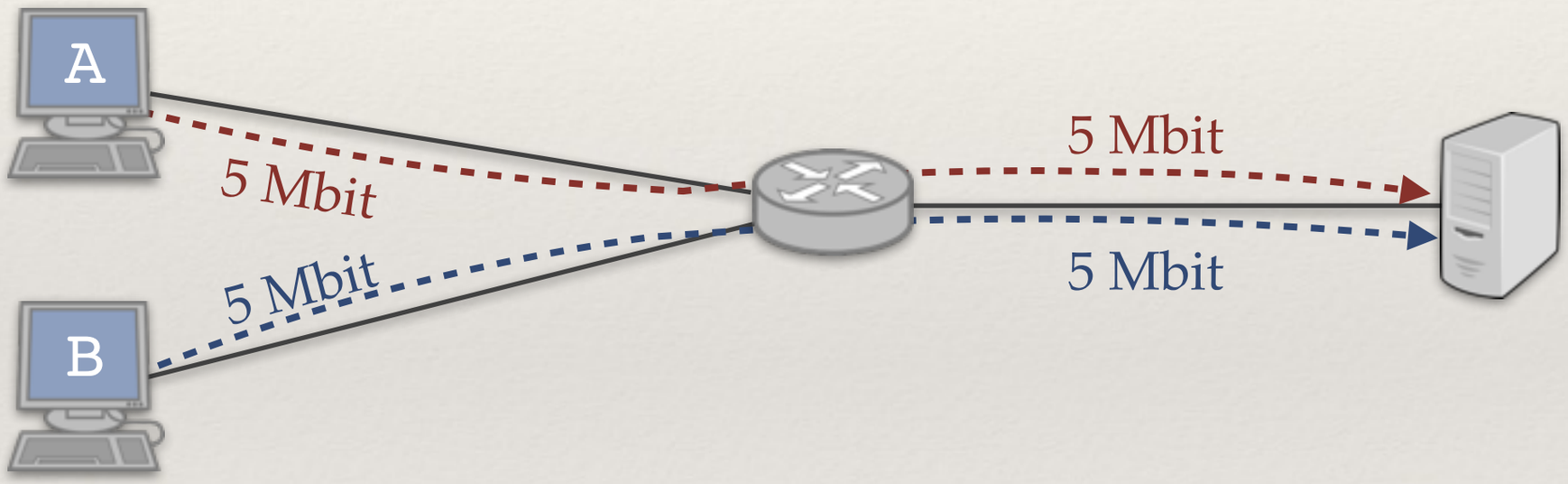
- **Sprawiedliwy podział łącza (*fairness*).**
  - Co to znaczy?

- **Dodatkowe cele**
  - Rozproszony algorytm.
  - Szybko reaguje na zmieniające się warunki.

Sprawiedliwy podział łącza

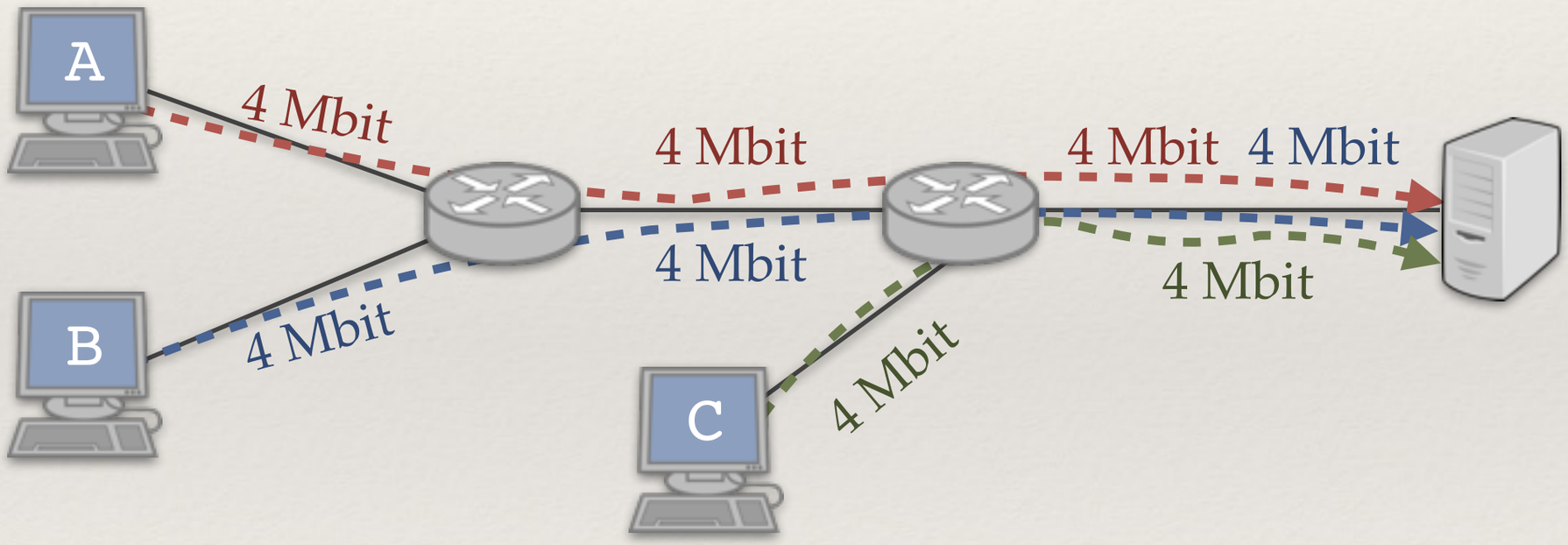
Sprawiedliwy podział łącza: przykład 1

- Każde łącze ma przepustowość 10 Mbit/s.
- Każdy komputer chce wysyłać do serwera jak najszybciej.



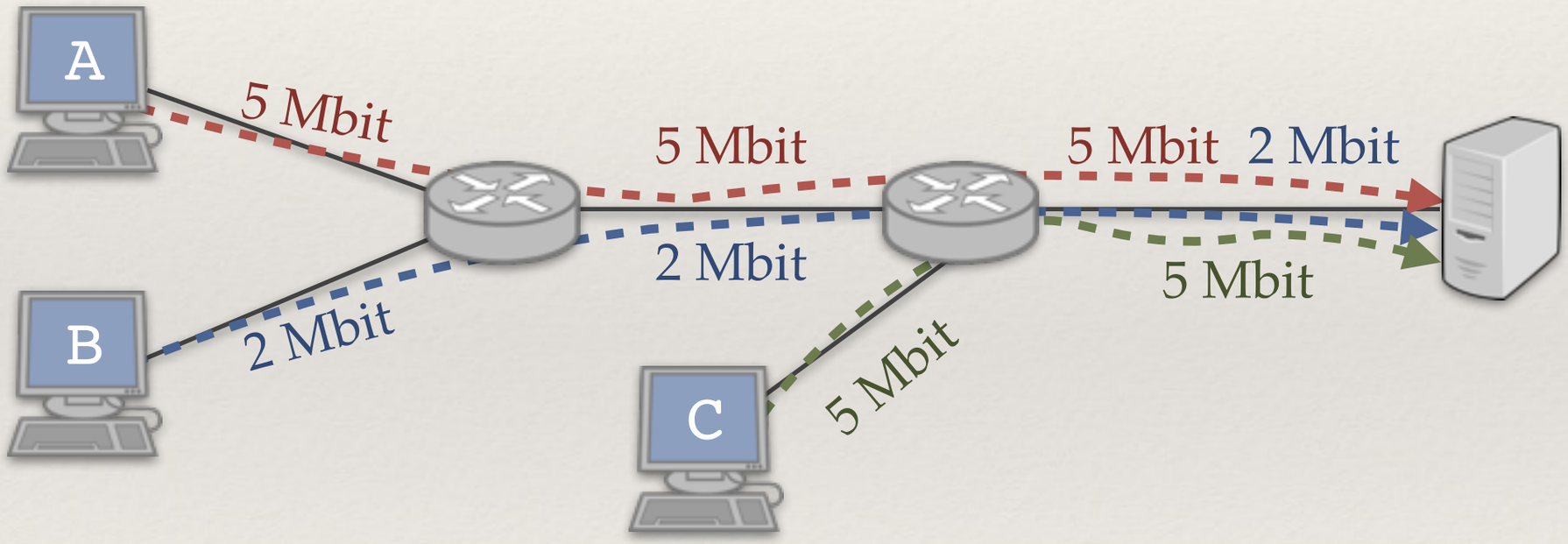
Sprawiedliwy podział łącza: przykład 2

- Każde łącze ma przepustowość 12 Mbit/s.
- Każdy komputer chce wysyłać do serwera jak najszybciej.



Sprawiedliwy podział łącza: przykład 3

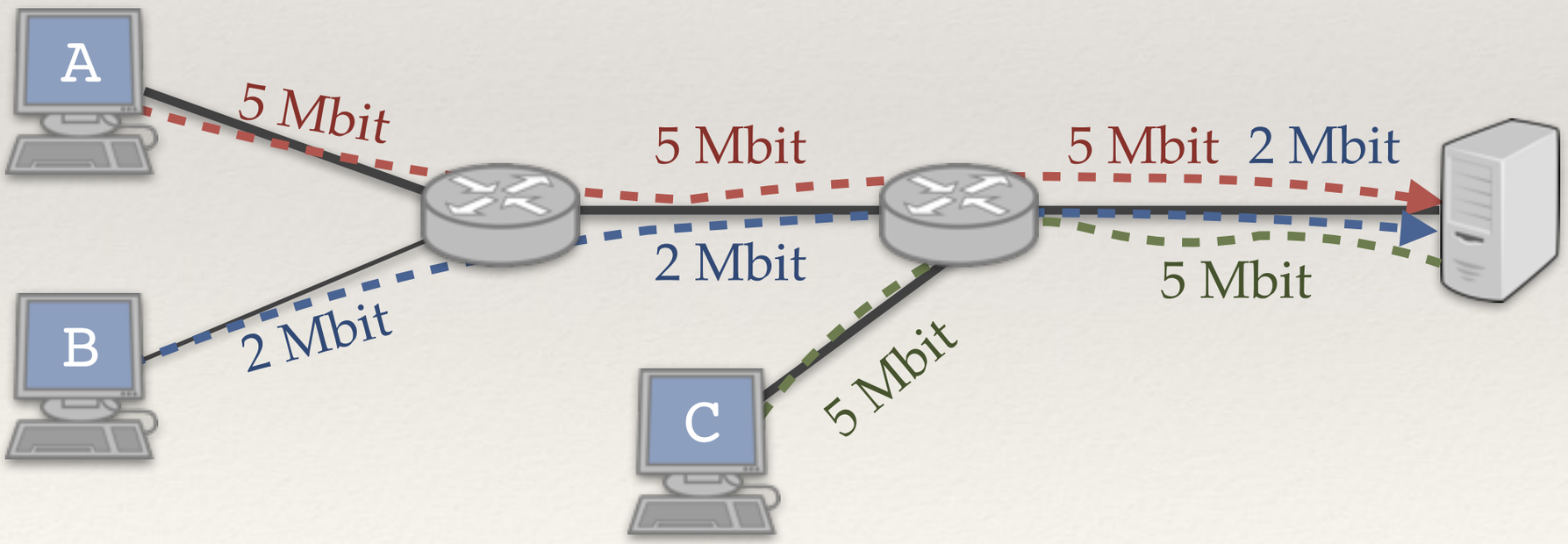
- Każde łącze ma przepustowość 12 Mbit/s, poza łączem między B a routerem, które ma przepustowość 2 Mbit/s.
- Każdy komputer chce wysyłać do serwera jak najszybciej.



- Czy to przypisanie jest „sprawiedliwe“, czy też powinniśmy dać B proporcjonalnie mniej łącza?

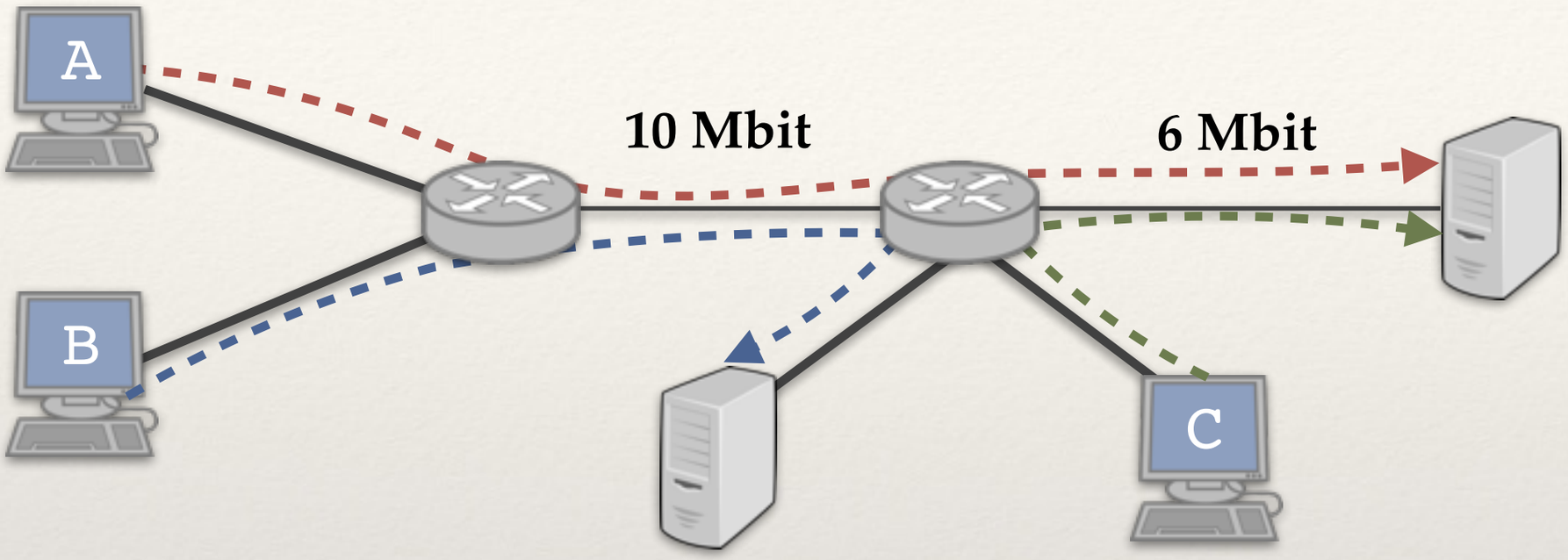
Max-Min fairness

- Jedna z możliwych, często stosowanych definicji sprawiedliwego podziału łącza.
- Przypisanie jest *max-min fair,* jeśli nie można zwiększyć szybkości żadnego ze strumieni bez spowolnienia innego strumienia, który jest wolniejszy lub tak samo szybki.



Sprawiedliwy podział łącza vs. przepustowość

Nieoznaczone łącza mają nieskończoną przepustowość.



	„sprawiedliwe“	„niesprawiedliwe“
A	3 Mbit	1 Mbit
B	7 Mbit	9 Mbit
C	3 Mbit	5 Mbit
suma	13 Mbit	**15 Mbit**

AIMD

Kontrola przeciążenia w warstwie transportowej

- Algorytm dla nadawcy.
- Wykorzystuje istniejące mechanizmy (okno przesuwne).
- Reaguje na obserwowane zdarzenia (utratę pakietów).
- Wylicza, z jaką prędkością może wysyłać, tj. rozmiar okna (ile pakietów może być wysłanych i niepotwierdzonych).

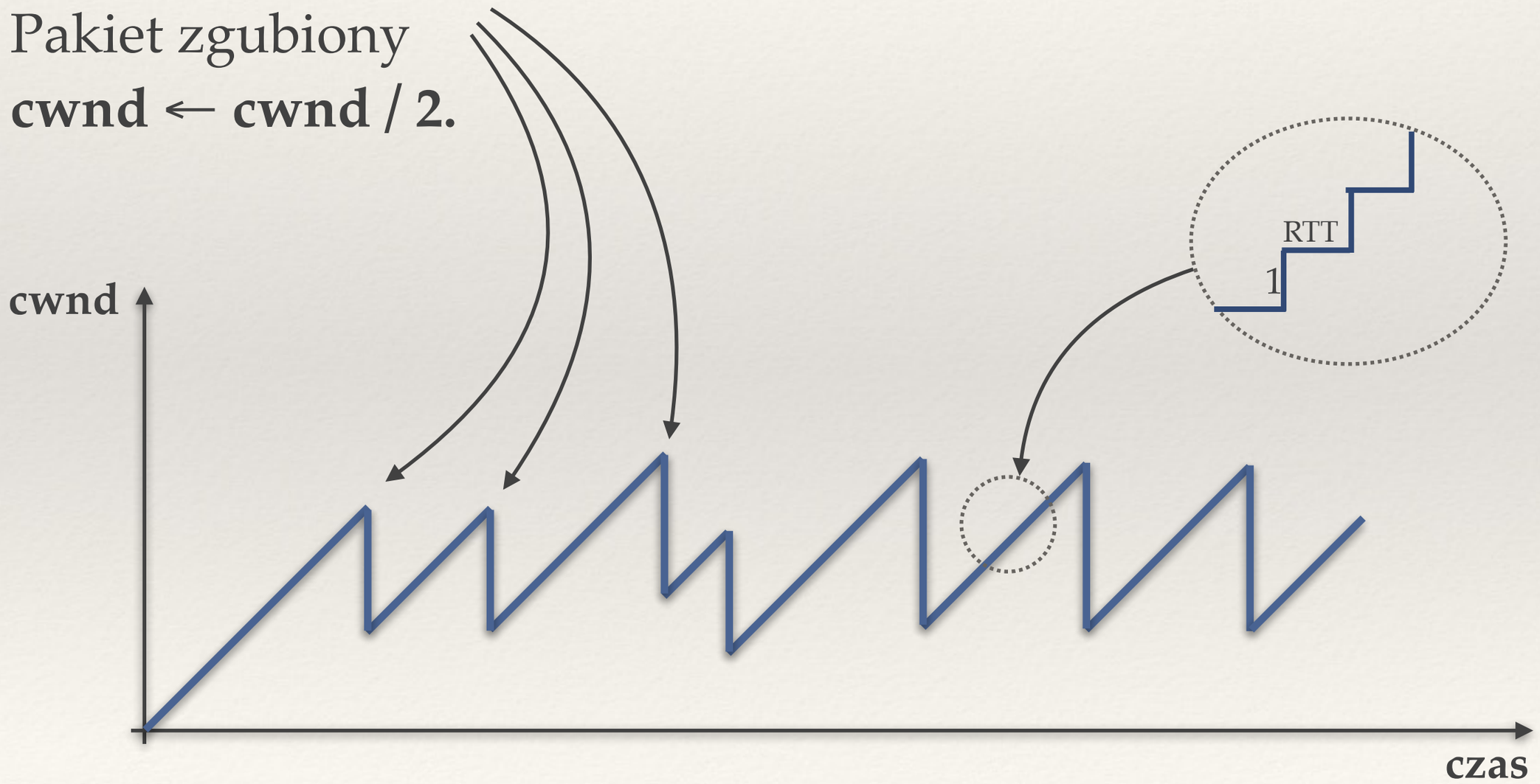
Kontrola przepływu vs. kontrola przeciążenia

- **Kontrola przepływu = nie chcemy zalać odbiorcy danymi.**
  - SWS = oferowane okno.
- **Kontrola przeciążenia = nie chcemy zalać sieci danymi.**
  - Parametr `cwnd` (*congestion window*) obliczany przez nadawcę.
  - SWS = min { oferowane okno, `cwnd` }.
- **Będziemy zakładać, że oferowane okno = ∞.**

AIMD (Additive Increase, Multiplicative Decrease)

- Pakiet wysłany poprawnie (otrzymaliśmy ACK):
  cwnd ← cwnd + 1 / cwnd.
  (W ciągu RTT wysyłane cwnd segmentów, więc cwnd zwiększa się o 1).

- Pakiet zgubiony
  cwnd ← cwnd / 2.



AIMD

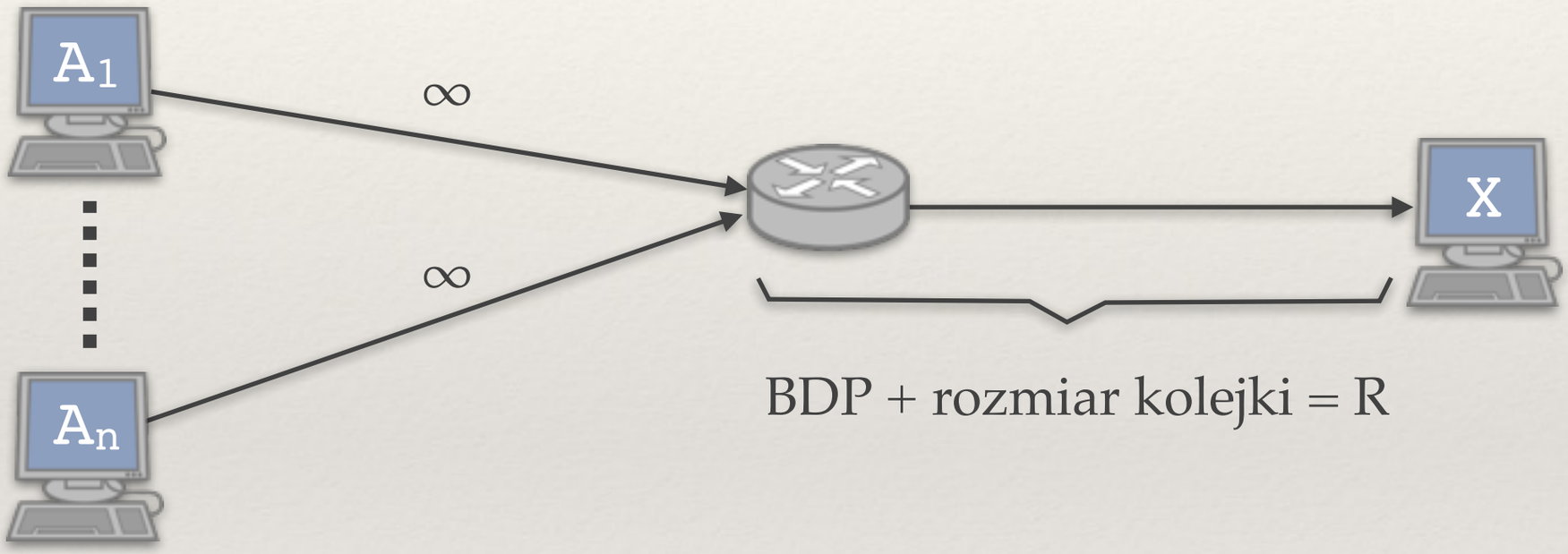
- **Inny sposób patrzenia:**
  - AIMD nie kontroluje szybkości wysyłania.
  - AIMD kontroluję liczbę pakietów (danego strumienia), która jednocześnie może być w sieci.

animacja

  - Przy odpowiednio dużych buforach, najbardziej krytyczne łącze jest cały czas wykorzystane w 100%.

AIMD vs. sprawiedliwy podział i efektywność

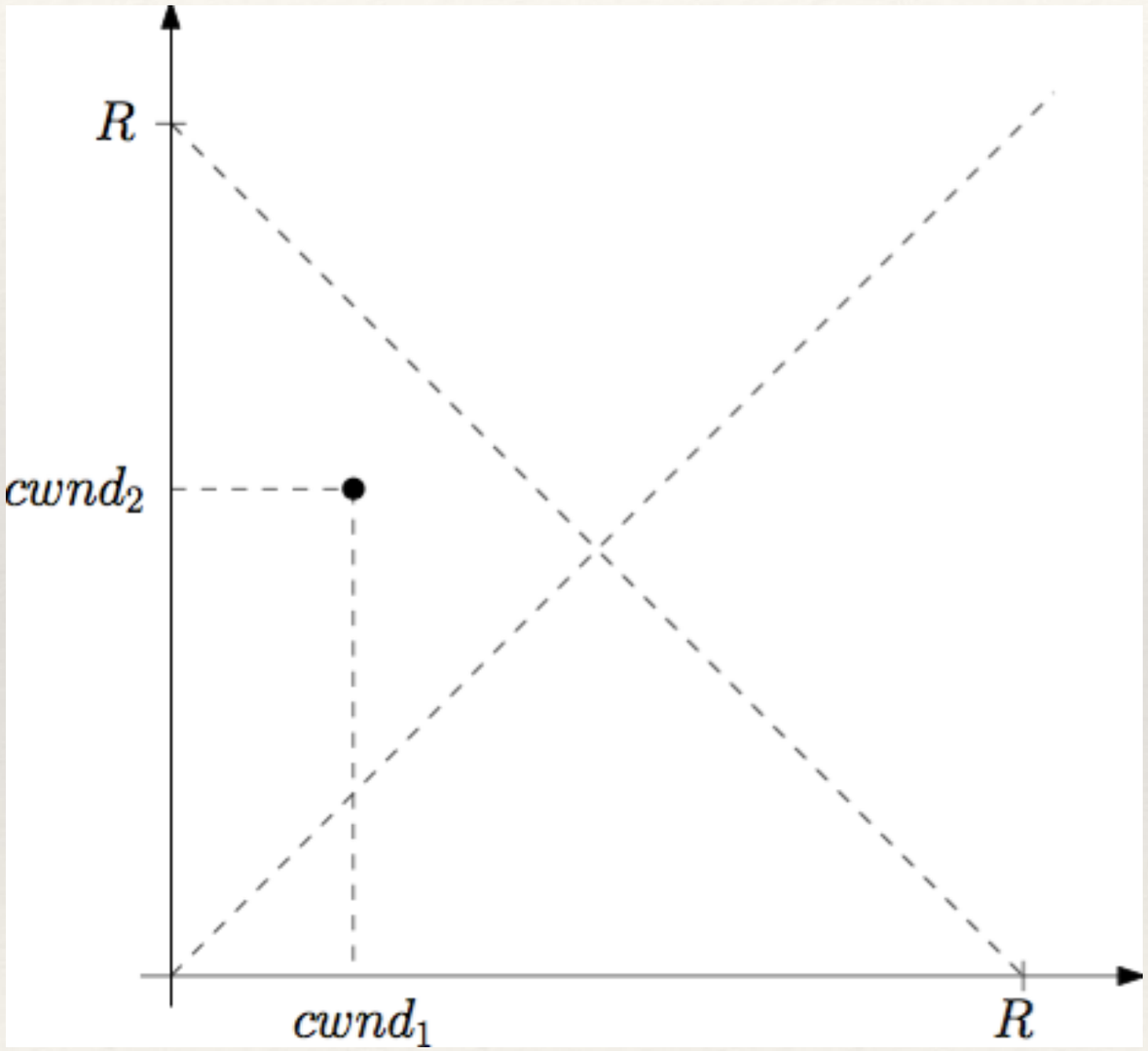
Własność AIMD: $A_1 \ldots A_n$ rozpoczyna transmisje do X w dowolnych momentach → ich rozmiary okien zbiegną do R/n.



Pokażemy to dla n = 2.

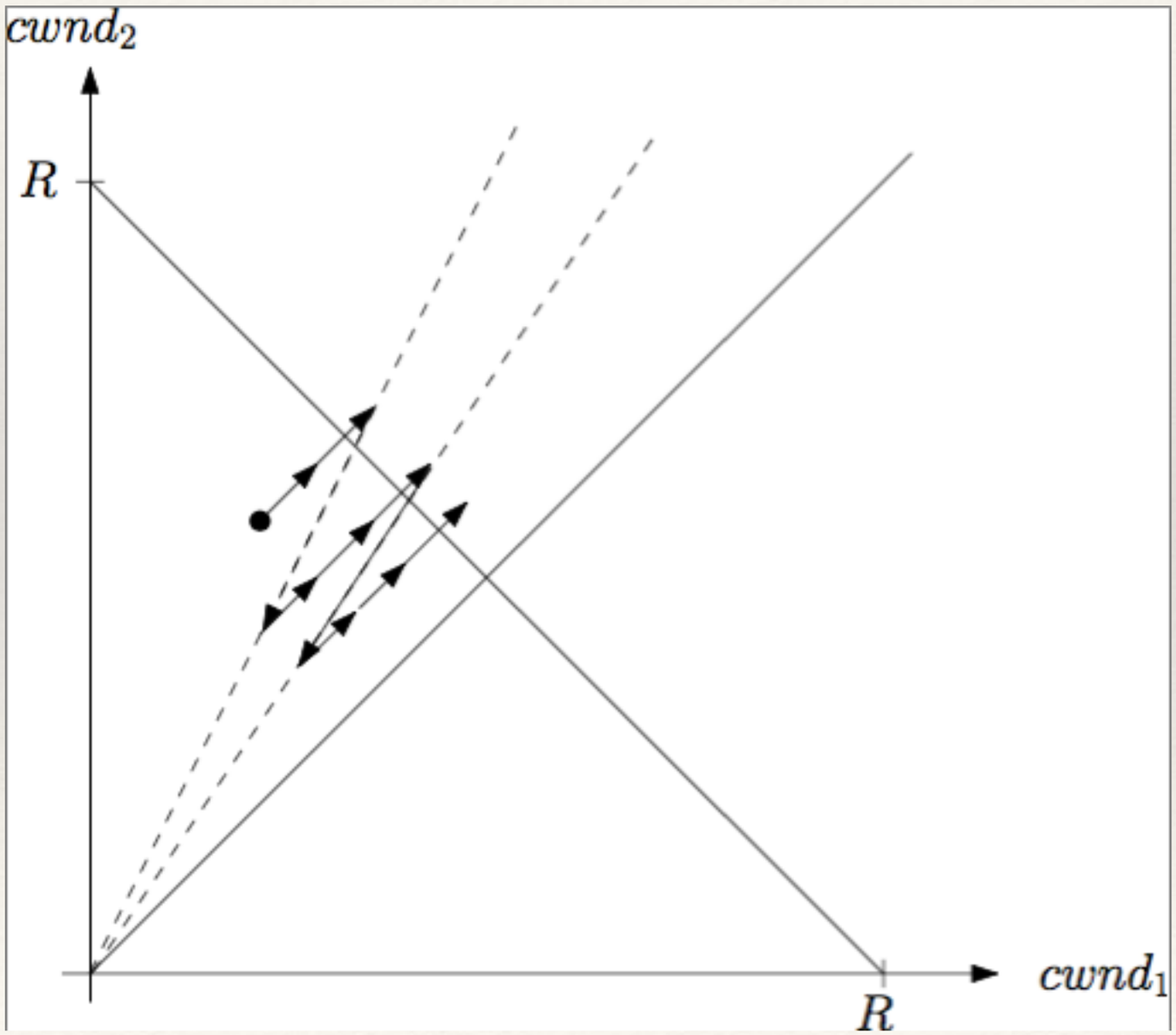
Sprawiedliwość podziału i efektywność (1)

- Pakiety gubią się wtedy i tylko wtedy, jeśli $\texttt{cwnd}_1 + \texttt{cwnd}_2 > R$



Sprawiedliwość podziału i efektywność (2)

- Rozmiary $\texttt{cwnd}_1$ i $\texttt{cwnd}_2$ jednocześnie rosną o 1 lub maleją dwukrotnie
- Docelowo mamy sprawiedliwy podział łącza i $\texttt{cwnd}_1 + \texttt{cwnd}_2$ oscylujące w przedziale [R/2, R].



Kontrola przeciążenia w TCP

Kontrola przeciążenia w TCP

- **AIMD jest używane w fazie „unikania przeciążenia“.**
  - **ACK pakietu** (explicite albo wnioskowany z potwierdzenia skumulowanego) → zwiększamy `cwnd` o MSS * MSS / `cwnd`.
    - Co RTT wysyłane jest `cwnd` / MSS segmentów, więc jeśli wszystkie będą potwierdzone, to zwiększymy `cwnd` o MSS.
  - **Pakiet zaginął** (przekroczony timeout albo otrzymaliśmy podwójne potwierdzenie) → zmniejszamy `cwnd` dwukrotnie.

Nie do końca prawda.

- **Dodatkowo TCP wprowadza fazę „wolnego startu“.**

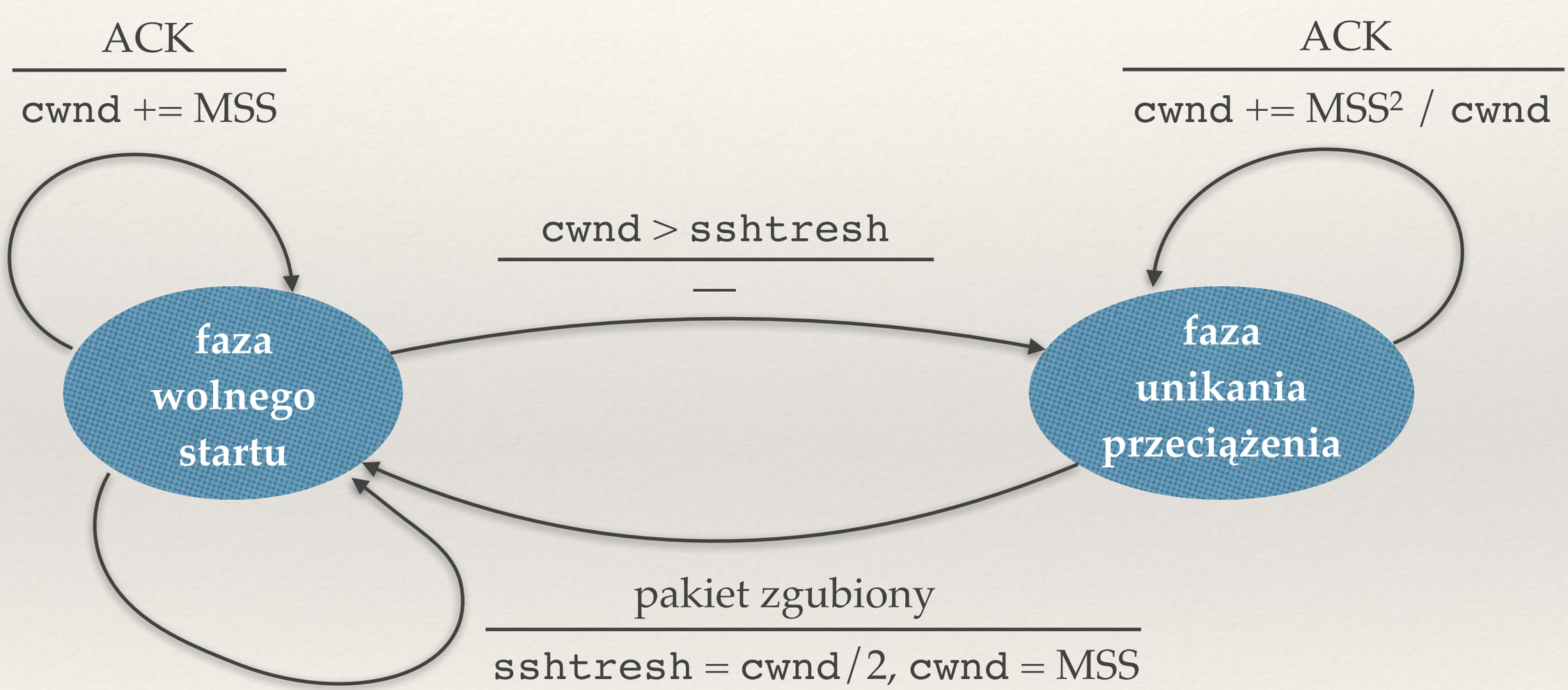
Wolny start w TCP

- **Faza wolnego startu:**
  - Zaczynamy od `cwnd` = 1.
  - Do momentu pierwszej utraty pakietu po każdym ACK zwiększamy `cwnd` o MSS.
  - → Co RTT `cwnd` zwiększa się dwukrotnie.
  - Trwa do utraty pierwszego pakietu.

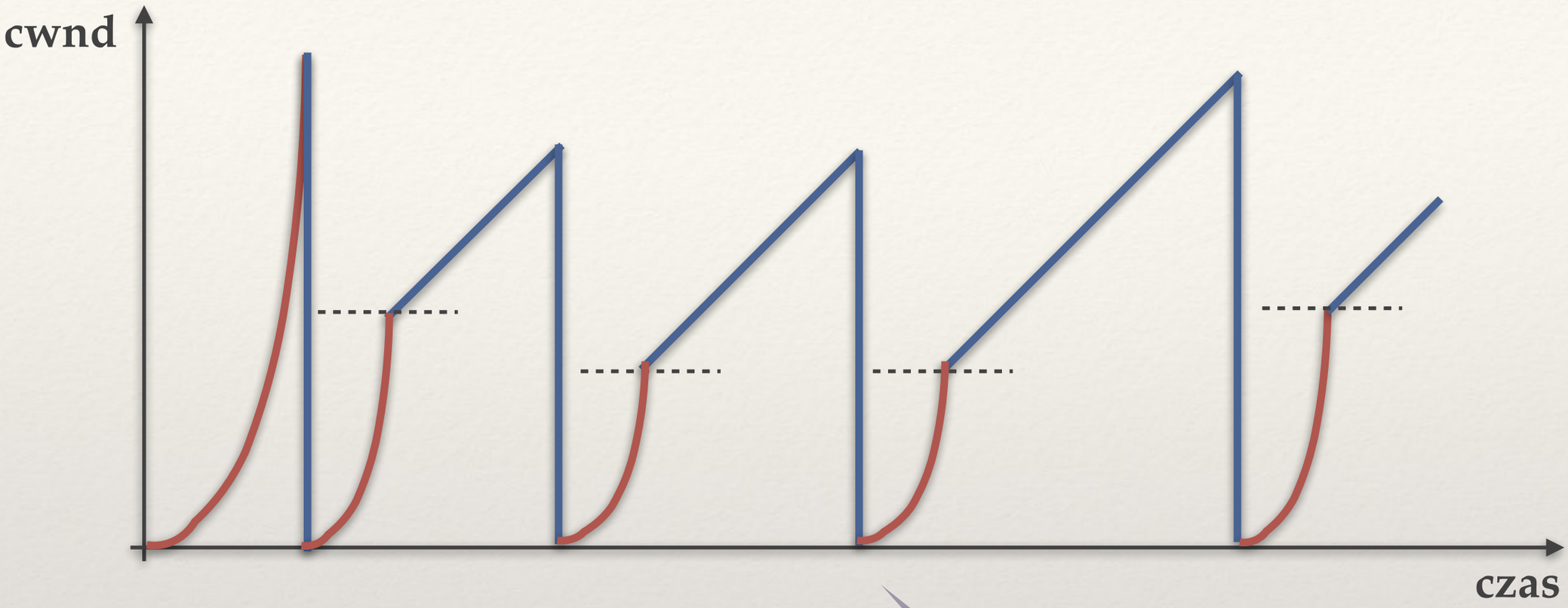
- **Strata pakietu w dowolnej fazie:**
  - `ssthresh` ← `cwnd` / 2.
  - uruchamiamy fazę wolnego startu do momentu, gdy `cwnd` > `ssthresh`.

AIMD w TCP: diagram przejścia

Inicjalizacja: faza wolnego startu, `sshtresh` = ∞, `cwnd` = MSS.



AIMD w TCP: przykład



faza wolnego startu

faza unikania przeciążenia

`sshtresh`

Fazy wolnego startu mają logarytmiczną długość, poza nimi TCP wykonuje AIMD.

Źródła informacji o utracie pakietu

- **Timeout dla pakietu**

- **Wielokrotny ACK**
  - Przykładowo: odbiorca dostaje segmenty 1, 2, 3, 5, 6
    → wysyła ACK 1, ACK 2, ACK 3, ACK 3, ACK 3.
  - Statystycznie mniejsza szansa, że mamy do czynienia z dłuższym przeciążeniem (kolejne pakiety doszły do odbiorcy)!
  - **Szybka retransmisja** (wysyłamy brakujący segment bez czekania na timeout).
  - **Szybkie przywracanie** (pomijamy fazę krótkiego startu): `ssthresh` = `cwnd` / 2; `cwnd` = `ssthresh`.

Wspomaganie przez routery

RED

RED (Random Early Detection)

- Router na trasie wyrzuca losowe pakiety.
- Prawdopodobieństwo wyrzucenia ustalane jako rosnąca funkcja **średniej** długości kolejki.
  - Nie reaguje na krótkotrwałe zwiększenia kolejki.

- Krótsze kolejki → mniejsze opóźnienia.
- Desynchronizacja strumieni (zmniejszają prędkości w różnych momentach).

ECN (Explicit Congestion Notification)

- Prawdopodobne przeciążenie → router ustawia bity ECN w nagłówku IP.
- Odbiorca ustawia bity ECN w nagłówku TCP ACK.
- Nadawca reaguje tak, jak na utratę pakietu.

nagłówek IP:

wersja	IHL	typ usługi	całkowita długość pakietu
pola związane z fragmentacją pakietu			
TTL		protokół	suma kontrolna nagłówka IP
źródłowy adres IP			
docelowy adres IP			

nagłówek TCP:

port źródłowy				port docelowy
numer sekwencyjny (numer pierwszego bajtu w segmencie)				
numer ostatniego potwierdzanego bajtu + 1				
offset	000	ECN	U-A-P-R-S-	oferowane okno
suma kontrolna				wskaźnik pilnych danych
dodatkowe opcje, np. potwierdzanie selektywne				

Problemy z kontrolą przeciążenia

Problemy

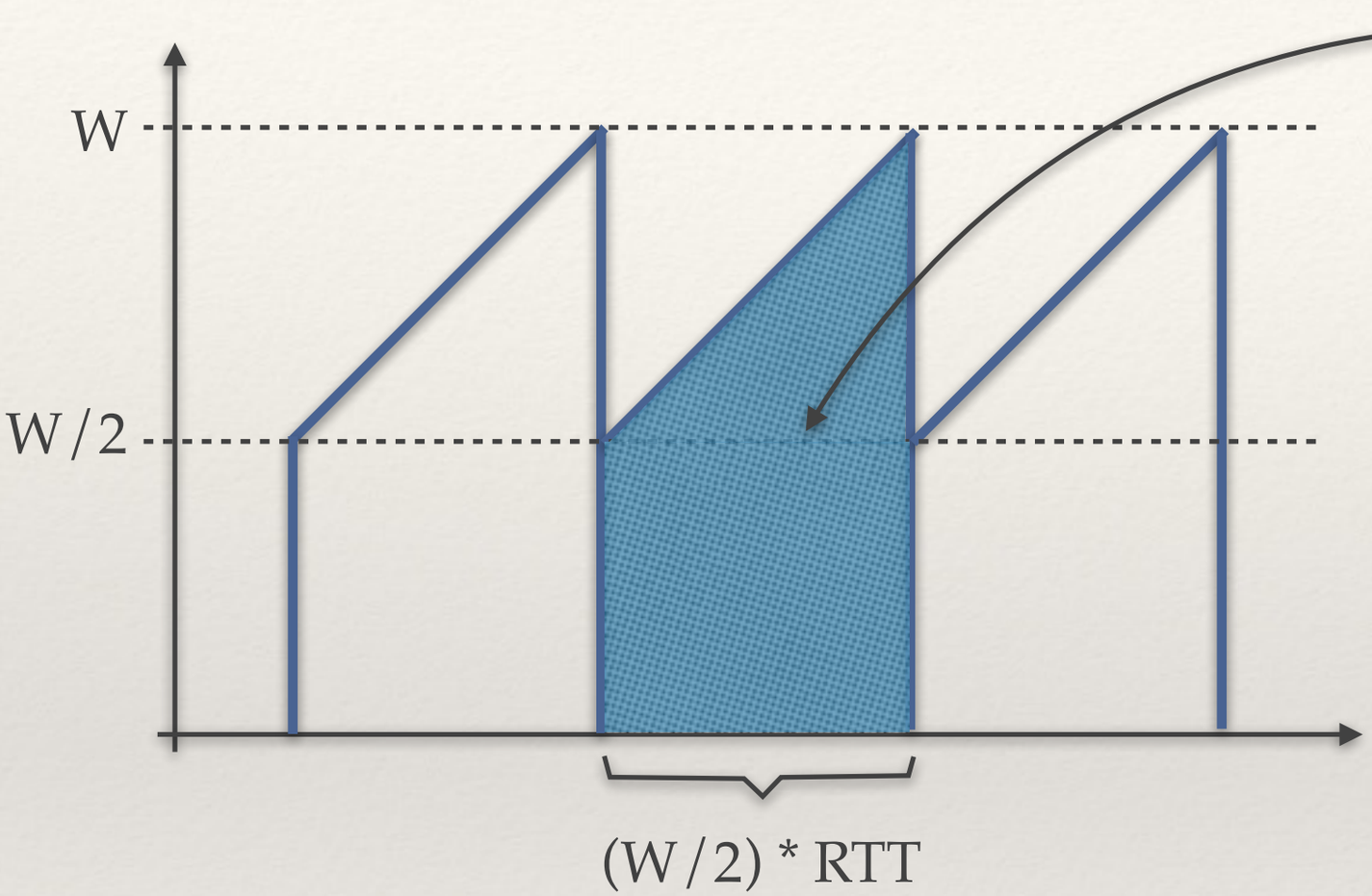
- Krótkie połączenia.
- Przepustowość proporcjonalna do $1/\sqrt{p}$, gdzie p jest frakcją traconych pakietów.
- Podatność na oszukiwanie.

Krótkie połączenia

- 90% połączeń przesyła poniżej 100 KB danych.
- Nie wychodzą poza fazę wolnego startu!
- Remedium: trwałe połączenia, HTTP 2.0, …

Przepustowość vs. straty segmentów (1)

- Rozważmy stabilny stan i pomińmy fazy wolnego startu.



Średnio (3/4) W segmentów przesyłanych w jednym RTT.

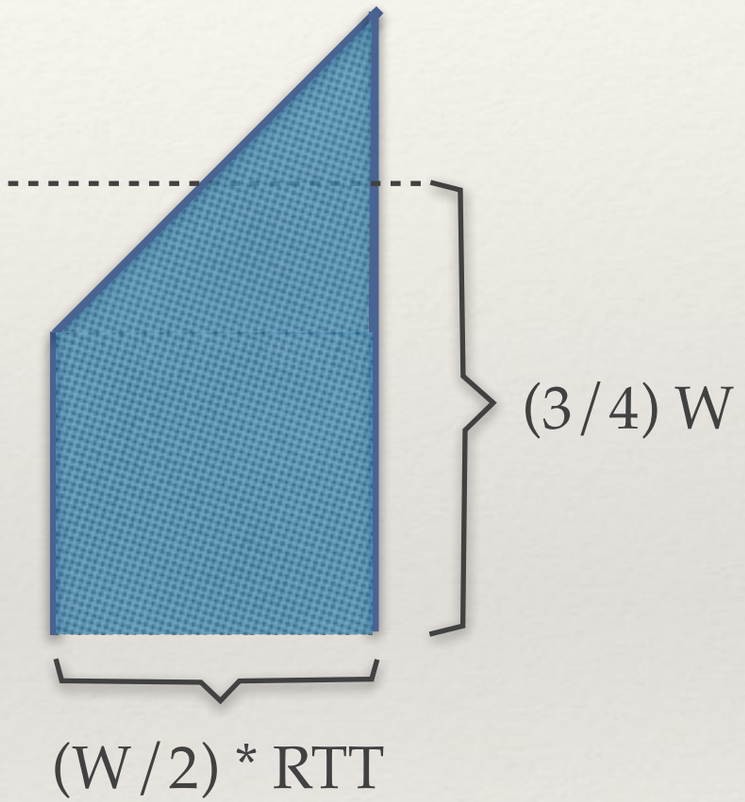
- W jednej fazie: przesyłanych (3/8) W^2 segmentów, jeden gubiony.
- Średnia prędkość przesyłania: (3/4) W / RTT.

Przepustowość vs. straty segmentów (2)

- Frakcja traconych pakietów: $p = 8/3\ (1/W^2)$.

- Średnia prędkość przesyłania (segment / sek.)

$$T = \frac{3}{4} \cdot \frac{W}{\mathrm{RTT}} = \sqrt{\frac{3}{2}} \cdot \frac{1}{\mathrm{RTT} \cdot \sqrt{p}}$$



Transmisje o mniejszym RTT są preferowane

- Prędkość przesyłania: $T = \sqrt{\frac{3}{2}} \cdot \frac{1}{\mathrm{RTT} \cdot \sqrt{p}}$ segmentów / sekundę.
- Dwie transmisje korzystającego z tego samego łącza o małej przepustowości: ich okna przeciążenia i frakcja traconych pakietów taka sama.
- Prędkości transmisji proporcjonalne do 1/RTT: transmisje o mniejszym RTT przesyłane szybciej!

Łącza o dużym BDP

- Prędkość przesyłania: $T = \sqrt{\frac{3}{2}} \cdot \frac{1}{\mathrm{RTT} \cdot \sqrt{p}}$ segmentów / sekundę.
- Transmisja RTT = 100 ms, MSS = 1500 bajtów, łącze 10 Gbit/s.
- Żeby osiągnąć T = 10 Gbit/s, p ≈ 2 * 10^{-10}:
  - Co najwyżej jeden tracony segment na 2 * 10^{10} segmentów.
  - Nie do zrealizowania w praktyce.

- Dla takich łączy FastTCP ze zmodyfikowanym AIMD: powyżej pewnej wartości `cwnd` zwiększane szybciej i zmniejszane wolniej.

TCP w WiFi

- Przepustowość proporcjonalna do $1/\sqrt{p}$, gdzie p jest frakcją traconych pakietów.
- Pakiety tracone niekoniecznie przez przeciążenie, np. w sieciach bezprzewodowych tracone przez interferencje!
- Żeby TCP działało sensownie konieczne są retransmisje dokonywane przez warstwy niższe (warstwę łącza danych).

Nadużycia

- Zaczynanie wolnego startu z `cwnd` > 1 MSS.
- Szybsze zwiększanie `cwnd` → niesprawiedliwy podział łącza.
- Otwieranie wielu połączeń (np. aplikacje P2P), bo każde połączenie ma takie samo `cwnd`.

Lektura dodatkowa

- Kurose & Ross: rozdział 3.
- Tanenbaum: rozdział 6.
- TCP Congestion Control RFC: `https://tools.ietf.org/html/rfc5681`