Routing

część 2: tworzenie tablic

Sieci komputerowe

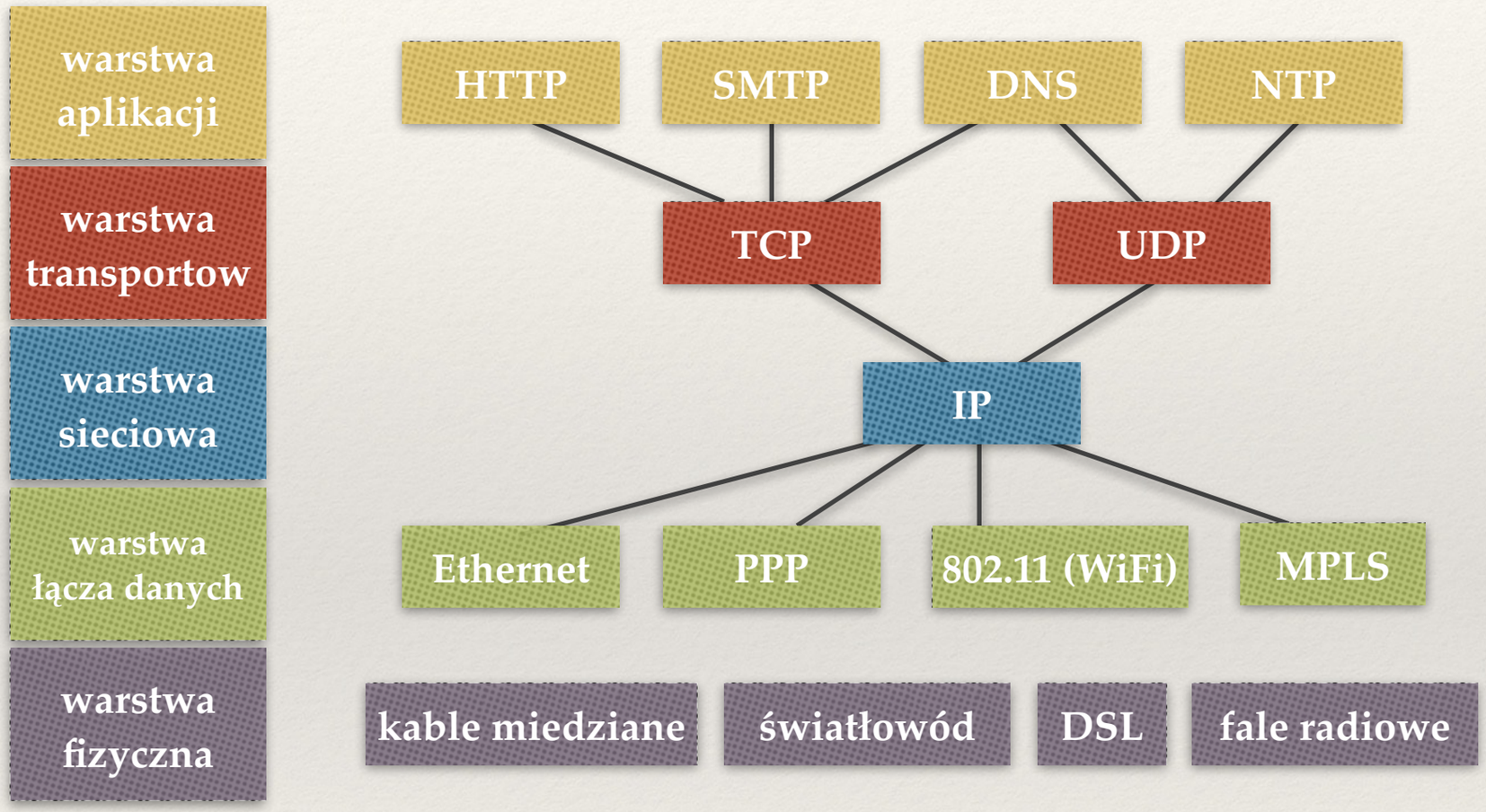
Wykład 3

Marcin Bieńkowski

W poprzednim odcinku

Jedna warstwa sieci i globalne adresowanie

- Każde urządzenie w sieci posługuje się **tym samym protokołem warstwy sieci**. W Internecie: protokół IP.

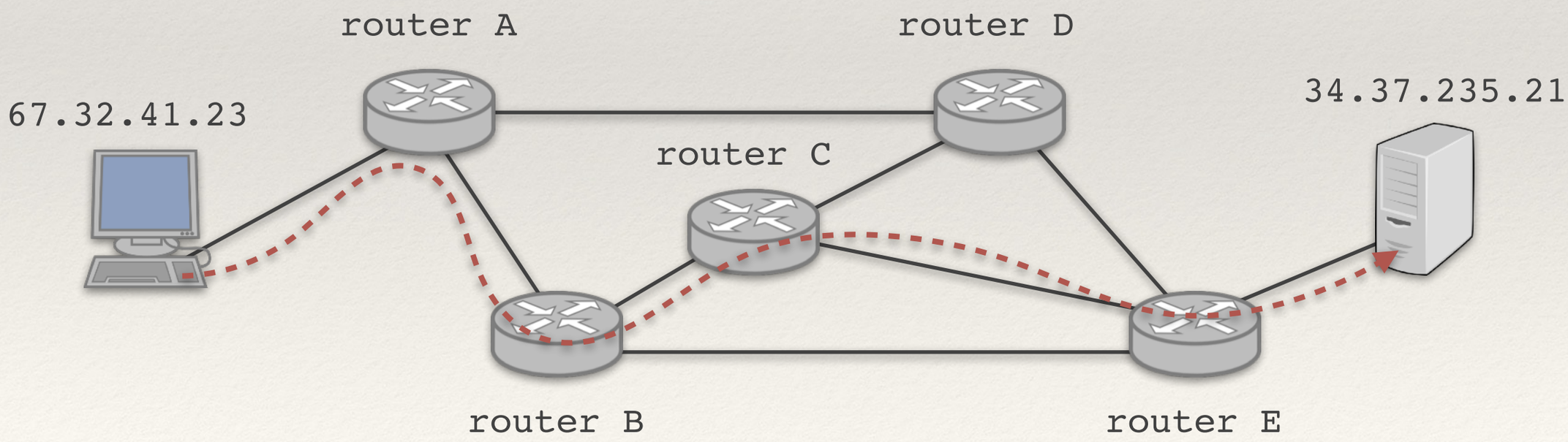


- Każde urządzenie ma **unikatowy adres**. W Internecie: adresy IP.

Przełączanie pakietów

- Chcemy przesyłać między aplikacjami strumień danych.
- Wysyłany strumień danych dzielimy na małe porcje: pakiety.

- Każdy pakiet przesyłany niezależnie.



Notacja CIDR

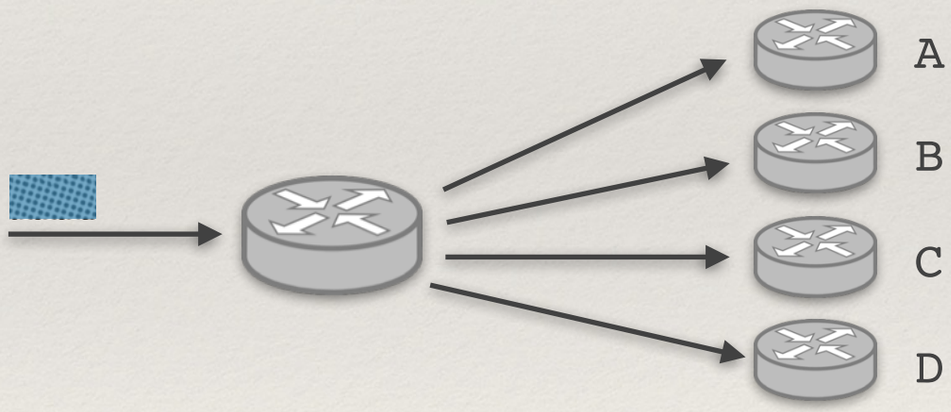
- CIDR opisuje prefiksy adresów IP:
  - `156.17.4.32 = 10011100.00010001.00000100.00100000`
  - `156.17.4.32/28` = adresy zaczynające się od prefiksu 28-bitowego `10011100.00010001.00000100.0010`

- Zazwyczaj sieć może być opisana jednym prefiksem CIDR.

Tablice routingu

- Router podejmuje decyzję na podstawie nagłówka pakietu w oparciu o tablice routingu.
- Zawiera reguły typu „jeśli adres docelowy pakietu zaczyna się od prefiksu *A*, to wyślij pakiet do *X*“.

prefiks CIDR	akcja
10.20.30.0/24	do routera A
8.0.0.0/8	do routera B
9.9.9.0/24	do routera C
156.17.0.0/16	do routera C
156.18.0.0/16	do routera D



- Pakiet niepasujący do żadnej reguły jest odrzucany.

Dziś: tworzenie tablic

Ręczna konfiguracja routingu

- Sprawdza się w przypadku małej sieci.
- W Internecie bez szans powodzenia:
  - dodawane lub usuwane routery i łącza;
  - zmiana parametrów i awarie łączy.
- Chcemy zapewniać łączność i unikać *cykli w routingu* (pakietów krążących w kółko)

Tablica przekazywania i routingu

- **Tablica przekazywania (*forwarding table*)**
  - Przez dwa ostatnie wykłady nazywaliśmy ją (potocznie) tablicą routingu.
  - Informacje **o następnym routerze na trasie**.
  - Używana do podejmowania decyzji o pakietach na podstawie najdłuższego pasującego prefiksu.
    - Silnie zoptymalizowana struktura danych wspomagana sprzętowo.

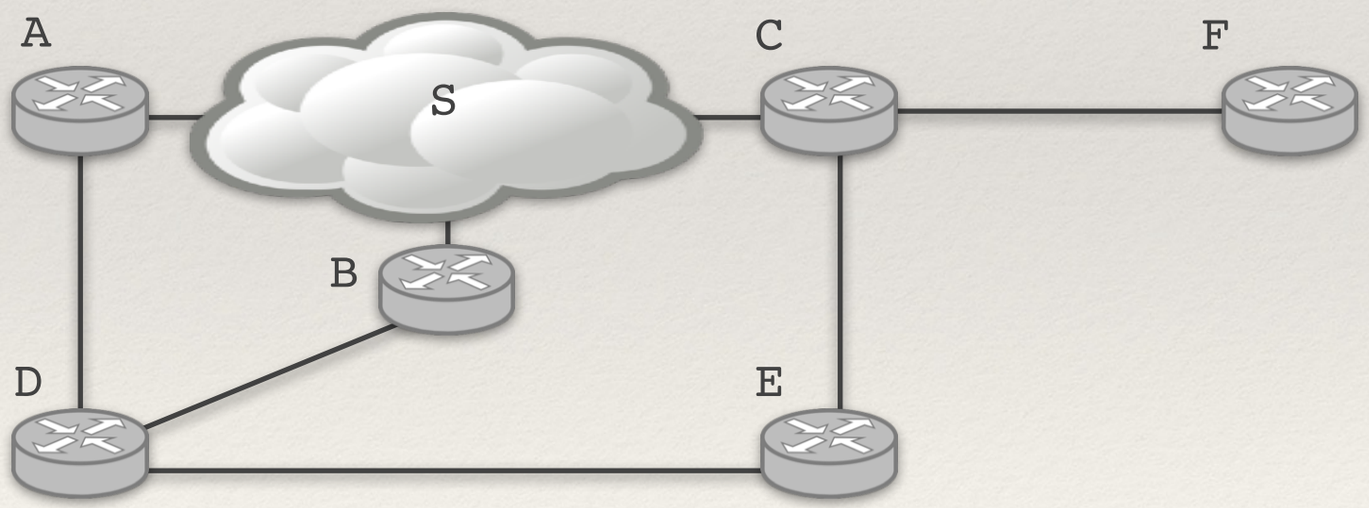
prefiks CIDR	akcja
`10.20.30.0/24`	do routera A
`8.0.0.0/8`	do routera B
`9.9.9.0/24`	do routera C
`156.17.0.0/16`	do routera C
`156.18.0.0/16`	do routera D

- **Tablica routingu (*routing table*)**
  - Informacje o **trasach**.
  - Zawiera dodatkowe informacje, np. zapasowe trasy routingu.

Cel

Chcemy skonfigurować poprawnie tablice przekazywania.

- Do dowolnego miejsca w sieci.
- Algorytm nie powinien tworzyć cykli w routingu.
- Algorytm trzeba zaimplementować w rozproszonym środowisku.



Wiele różnych rozwiązań

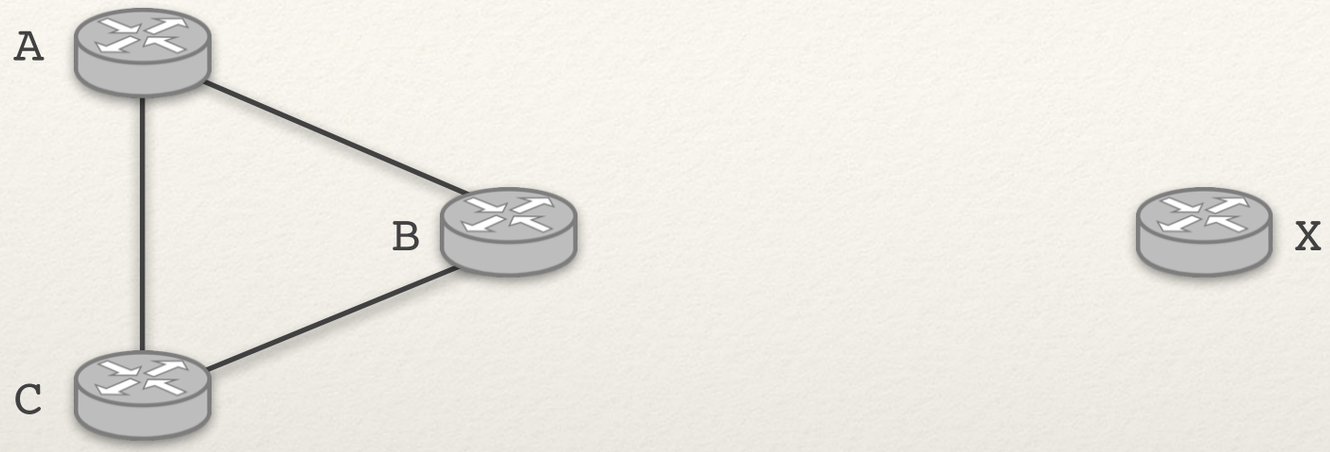
- **Chcemy dodatkowo minimalizować „odległość do celu“.**

- **Odległość = minimalna przepustowość na trasie?**
  - Problem: minimalna przepustowość na cyklu może być stała (trasa zawierająca wielokrotnie jakiś cykl może być optymalna wg. tej definicji).

- **Odległość = suma wartości krawędzi do celu (najkrótsza ścieżka).**
  - Nie zawiera cykli.
  - Jak zdefiniować wartości krawędzi? (**metryka**)
    - czas propagacji;
    - koszt pieniężny;
    - 1 → odległość pomiędzy dwoma routerami = liczba routerów na trasie (*hops*).

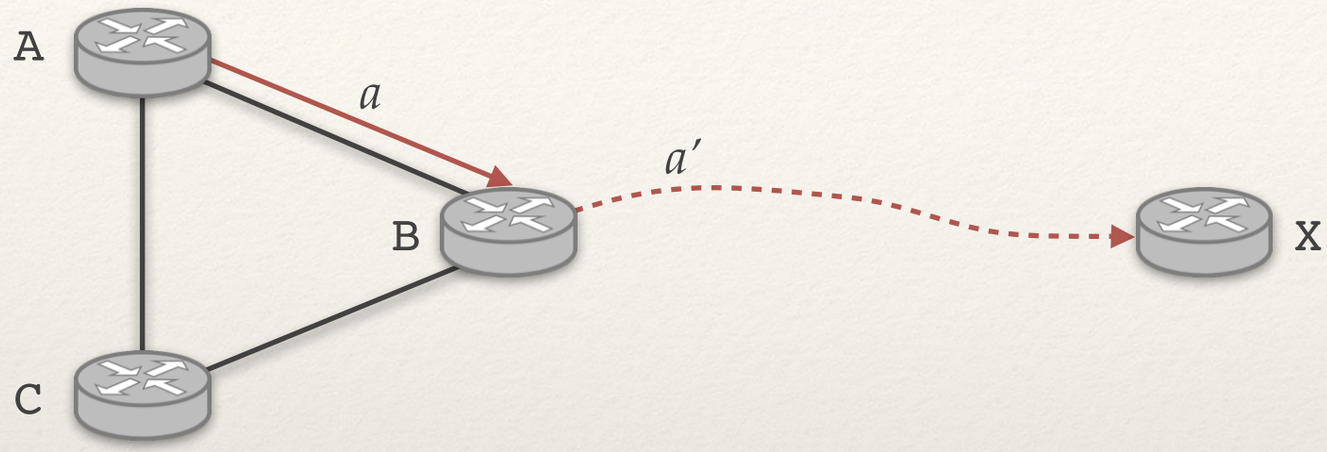
Routing według najkrótszych ścieżek → brak cykli

- Trasy do *X* (obliczone lokalnie na poszczególnych routerach)



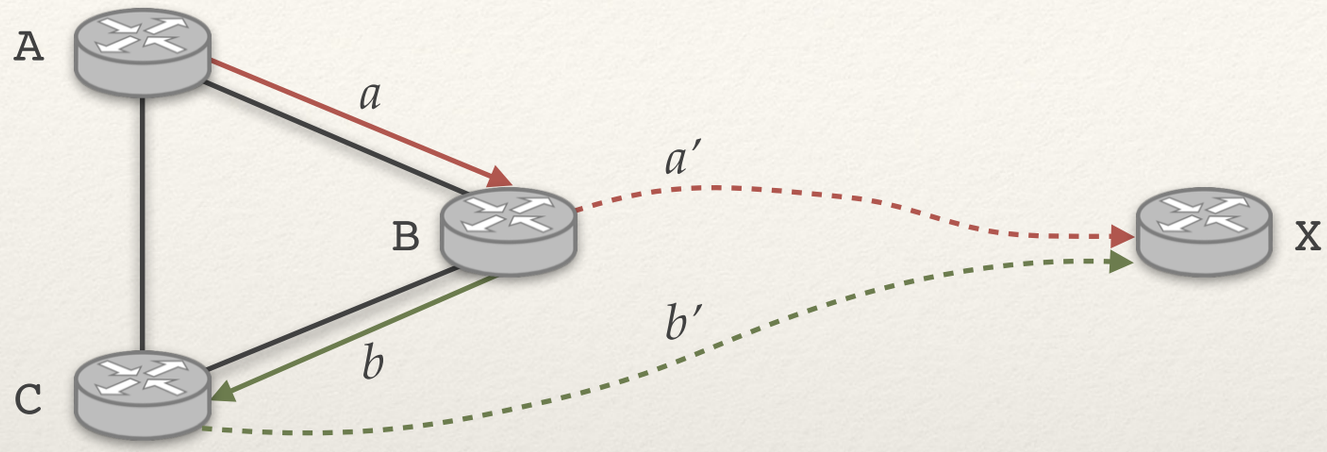
Routing według najkrótszych ścieżek → brak cykli

- Trasy do *X* (obliczone lokalnie na poszczególnych routerach)



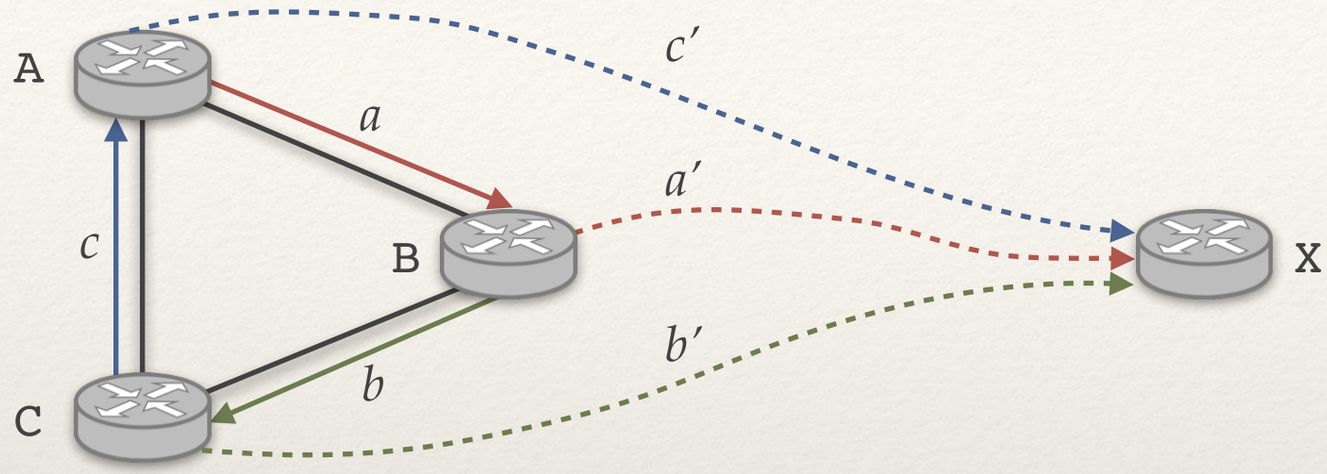
Routing według najkrótszych ścieżek → brak cykli

- Trasy do *X* (obliczone lokalnie na poszczególnych routerach)



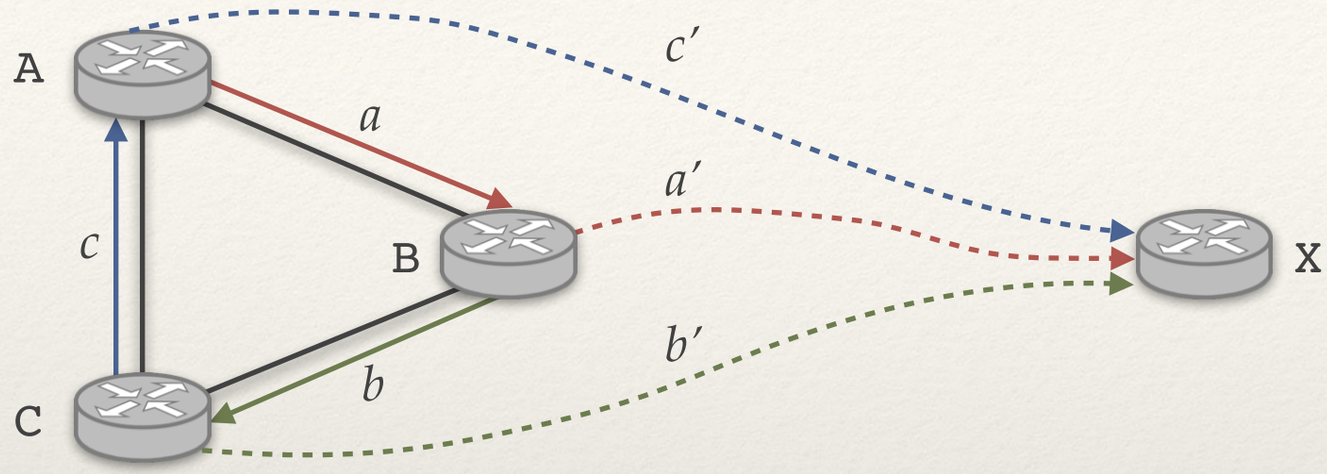
Routing według najkrótszych ścieżek → brak cykli

- Trasy do *X* (obliczone lokalnie na poszczególnych routerach)



Routing według najkrótszych ścieżek → brak cykli

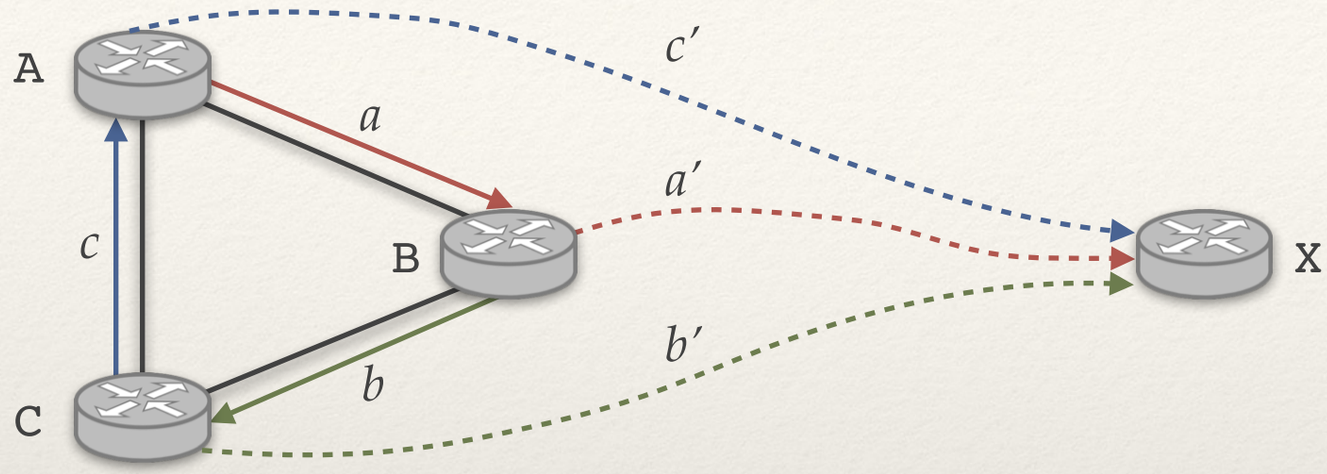
- Trasy do *X* (obliczone lokalnie na poszczególnych routerach)



- Załóżmy że mamy cykl.

Routing według najkrótszych ścieżek → brak cykli

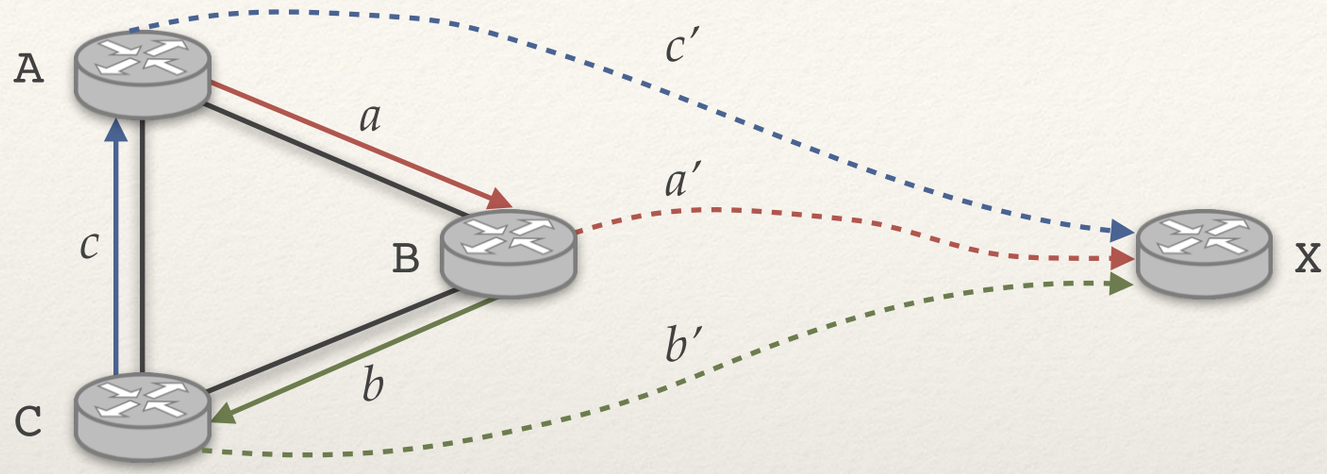
- Trasy do X (obliczone lokalnie na poszczególnych routerach)



- Załóżmy że mamy cykl.
- Wybrane ścieżki są najkrótsze:
  - $a + a' \leq c'$
  - $b + b' \leq a'$
  - $c + c' \leq b'$

Routing według najkrótszych ścieżek → brak cykli

- Trasy do *X* (obliczone lokalnie na poszczególnych routerach)

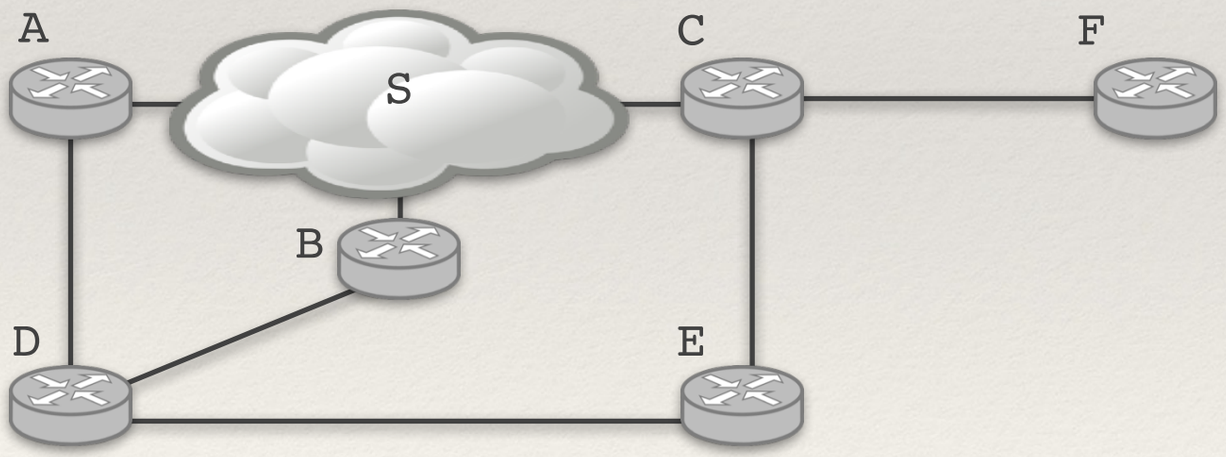


- Załóżmy że mamy cykl.
- Wybrane ścieżki są najkrótsze:
  - $a + a' \leq c'$
  - $b + b' \leq a'$
  - $c + c' \leq b'$
- A zatem: $a + b + c \leq 0$ → sprzeczność.

Sąsiedztwo

Warunek wstępny:

- Każdy router zna swoje bezpośrednie otoczenie (sieci i routery).
- Router zna **stan** sąsiadujących łączy przez okresowy monitoring, np. wymiana pakietów co 30 sekund z sąsiadem.



sąsiedztwo routera B	
sieć / router	odległość
B	0
S	1
A	1
C	1
D	1

Najkrótsze ścieżki w rozproszony sposób

- **Algorytmy stanu łączy**
  - Powiadom **wszystkich o swoim bezpośrednim sąsiedztwie.**
  - Na podstawie sąsiedztw zbuduj graf sieci i oblicz lokalnie najkrótsze ścieżki.

- **Algorytmy wektora odległości**
  - Okresowo powiadamiaj **sąsiadów o całej swojej tablicy przekazywania**.
  - Aktualizuj swoją tablicę routingu na tej podstawie.

Stan łączy

Dwa elementy

Wysłanie informacji o sąsiedztwie do wszystkich routerów.

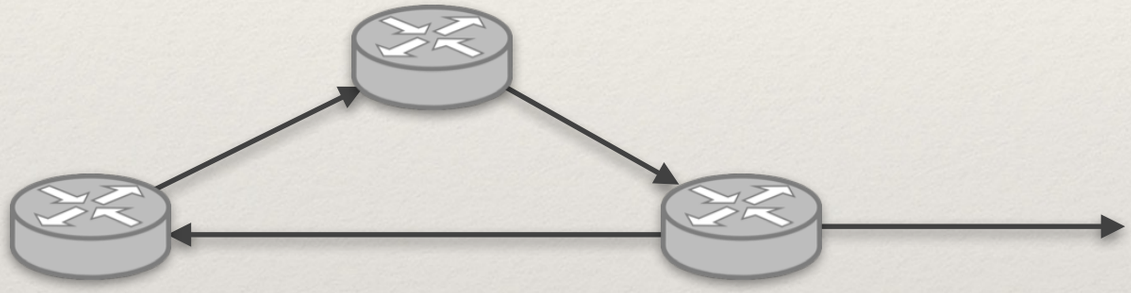
- Ogólny problem: jak wysłać coś do wszystkich bez mapy sieci?

Lokalne obliczenie najkrótszych ścieżek.

- Algorytm Dijkstry (najkrótsze ścieżki od jednego źródła).
- Router musi przechowywać cały graf: $O(|V|+|E|)$ danych.
- Czas działania: $O(|V| \log |V| + |E|)$.

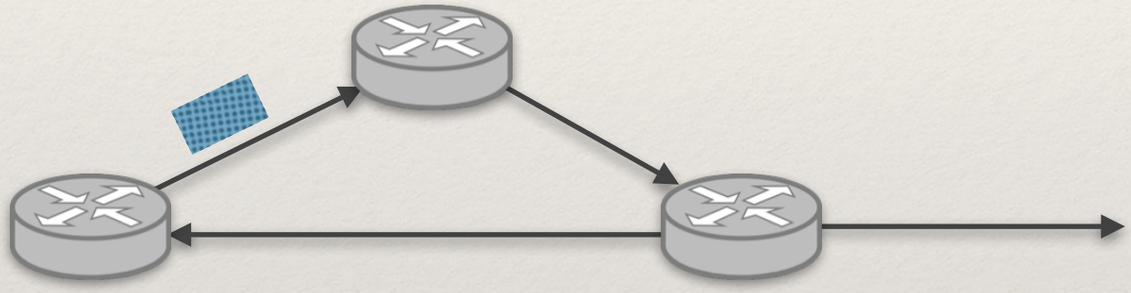
Niekontrolowane „zalewanie“ sieci informacją

- Reguła: po odebraniu informacji *E* od routera X, wyślij *E* do wszystkich sąsiadów poza X.
- Problem:



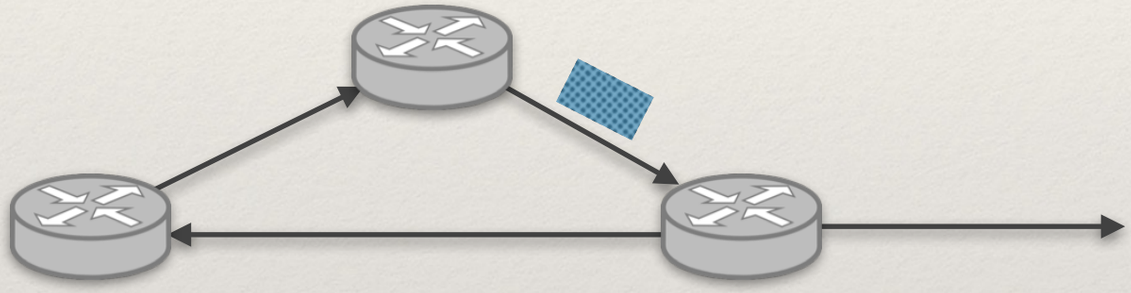
Niekontrolowane „zalewanie“ sieci informacją

- Reguła: po odebraniu informacji *E* od routera X, wyślij *E* do wszystkich sąsiadów poza X.
- Problem:



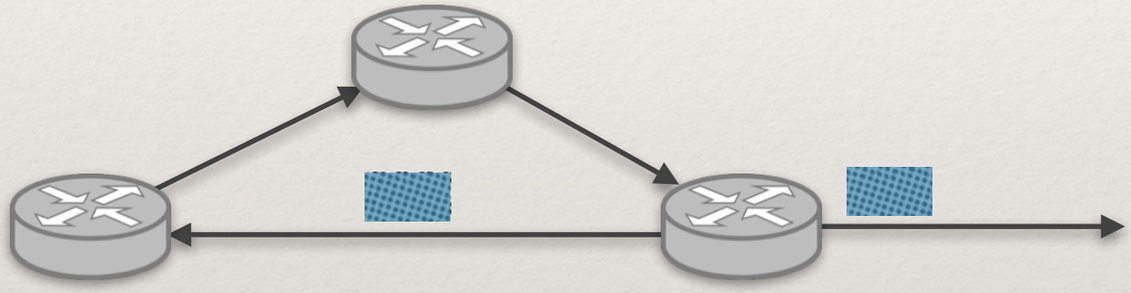
Niekontrolowane „zalewanie“ sieci informacją

- Reguła: po odebraniu informacji *E* od routera X, wyślij *E* do wszystkich sąsiadów poza X.
- Problem:



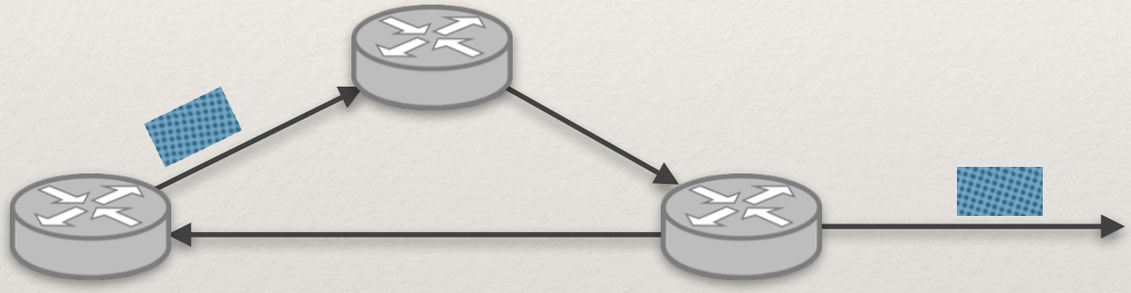
Niekontrolowane „zalewanie“ sieci informacją

- Reguła: po odebraniu informacji *E* od routera X, wyślij *E* do wszystkich sąsiadów poza X.
- Problem:



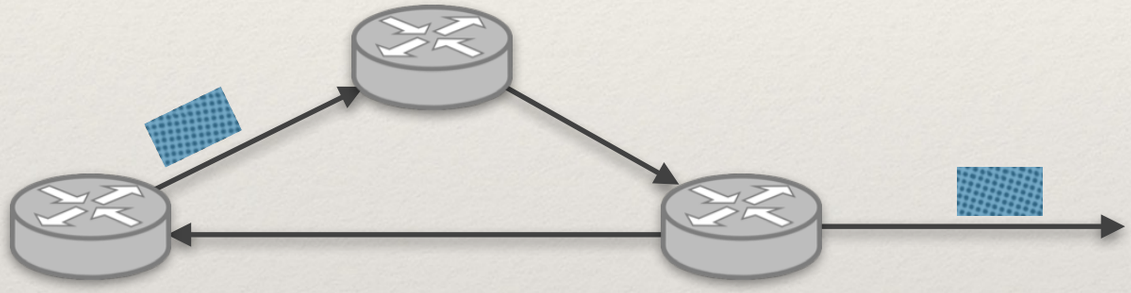
Niekontrolowane „zalewanie“ sieci informacją

- Reguła: po odebraniu informacji *E* od routera X, wyślij *E* do wszystkich sąsiadów poza X.
- Problem:



Niekontrolowane „zalewanie“ sieci informacją

- Reguła: po odebraniu informacji *E* od routera X, wyślij *E* do wszystkich sąsiadów poza X.
- Problem:



- Nawet jeśli w grafie nie ma cykli: wiele kopii pakietu może dotrzeć do jednego routera i każda z nich zostanie przesłany dalej.
- Trzeba pamiętać, jakie informacje już rozsyłaliśmy.

Kontrolowane „zalewanie” sieci informacją

- Router źródłowy dodaje do informacji *E*:
  - swój adres *s*,
  - numer sekwencyjny *n*.

- Reguła: po odebraniu informacji *(E,s,n)* od routera X:
  - sprawdź, czy już przekazywaliśmy jakąś informację z adresu *s* i o numerze *n*;
  - jeśli nie, to wyślij (*E,s,n)* do wszystkich sąsiadów poza X.
  - Jak długo trzymać numery *n*? (Globalny TTL).

Zbieżność do stanu stabilnego

- Jeśli sieć nie zmienia się przez pewien czas, to:
  - każdy router będzie miał ten sam obraz sieci;
  - stworzone tablice przekazywania będą bez cykli w routingu.

- Możliwe cykle, jeśli niektóre routery już wiedzą o awarii łącza a inne nie → ćwiczenie.

Algorytm stanu łączy w Internecie

Protokół OSPF (Open Shortest Path First).

- Komunikaty LSA = Link State Advertisement (stan pojedynczego łącza).
- Przesyłane na początku + przy zmianie + co jakiś czas (30 min.)
- LSA zawiera źródło i numer sekwencyjny.
- Po 1h otrzymane LSA są wyrzucane z pamięci.

Wektory odległości

Co robi router

- **Przechowuje *wektor odległości* V zawierający odległości do znanych mu routerów i sieci:**
  - początkowo: V = tylko sąsiedztwo.
- **Co pewien czas:**
  - wysyła V do sąsiednich routerów;
  - uaktualnia tablicę routingu na podstawie informacji od sąsiadów.
  - tablica routingu = tablica przekazywania + informacja z V o odległościach do celu

Uaktualnianie tablicy routingu

Aktualizacja tablicy dla routera X.

A mówi: „mam do B odległość d(A,B)“.

$$d(X,B) \leftarrow \min \{\, d(X,B),\ \ s(X,A) + d(A,B) \,\}$$

Aktualna odległość od X do B.

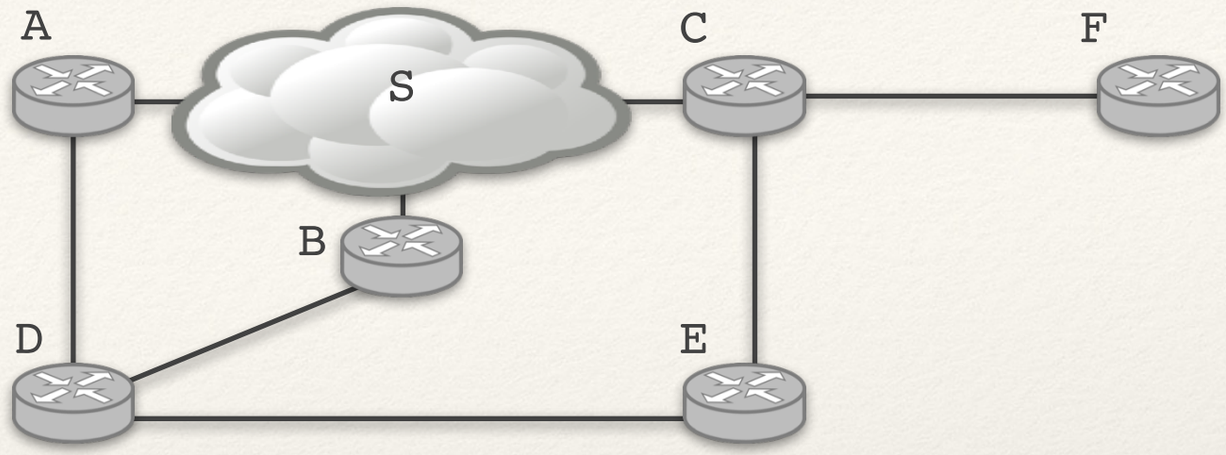
A jest sąsiadem X odległym o s(X,A).

Uwagi:

- Przy aktualizacji d(X,B) ustawiamy też A jako pierwszy router na trasie do B.
- Jeśli X nie zna B, to aktualna wartość $d(X,B) = \infty$.
- Rozproszony wariant algorytmu Bellmana-Forda.
- Przechowujemy tylko jedną (najlepszą) ścieżkę.

Przykład tworzenia tablic

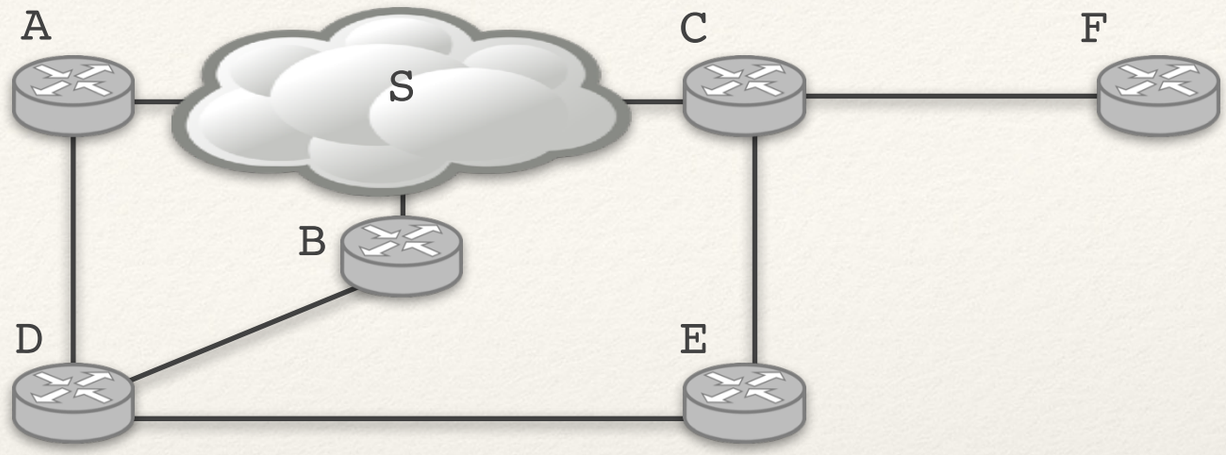
Krok 0.



	A	B	C	D	E	F
trasa do A	-	1	1	1		
trasa do B	1	-	1	1		
trasa do C	1	1	-		1	1
trasa do D	1	1		-	1	
trasa do E			1	1	-	
trasa do F			1			-
trasa do S	1	1	1			

Przykład tworzenia tablic

Krok 0.

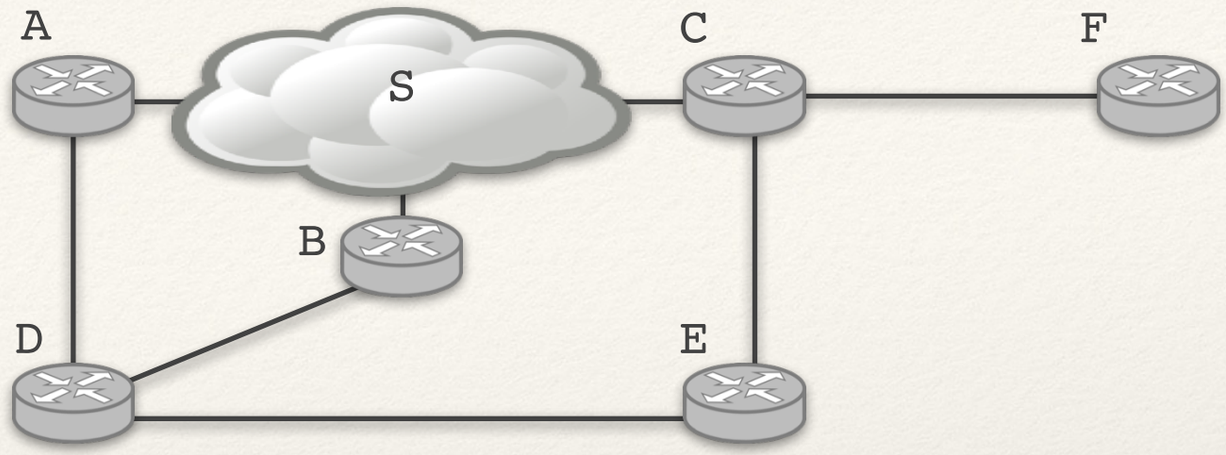


	A	B	C	D	E	F
trasa do A	-	1	1	1		
trasa do B	1	-	1	1		
trasa do C	1	1	-		1	1
trasa do D	1				1	
trasa do E			1	1	-	
trasa do F			1			-
trasa do S	1	1	1			

„mam ścieżkę do E długości 1“

Przykład tworzenia tablic

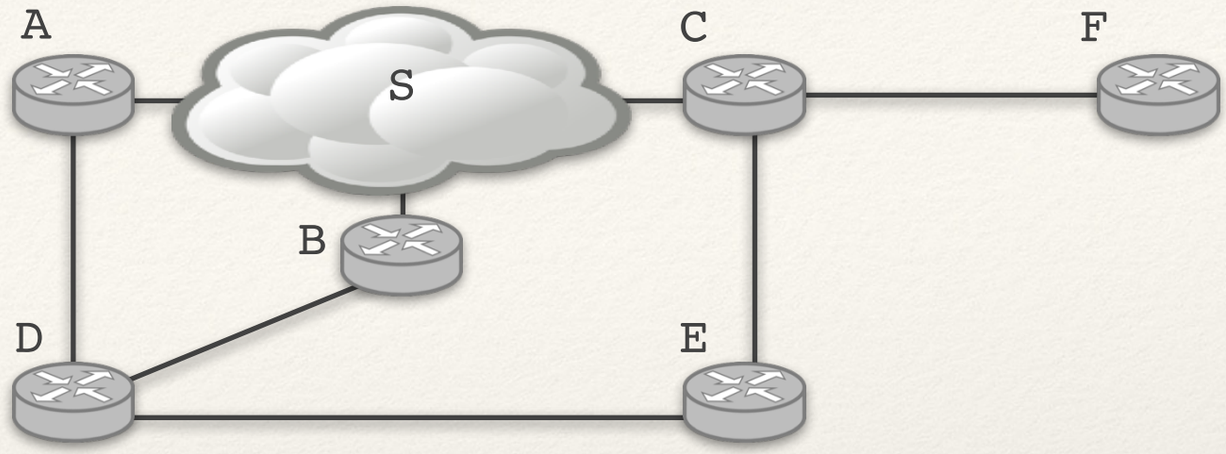
Krok 1.



	A	B	C	D	E	F
trasa do A	-	1	1	1	2 (via D)	2 (via C)
trasa do B	1	-	1	1	2 (via D)	2 (via C)
trasa do C	1	1	-	2 (via E)	1	1
trasa do D	1	1	2 (via E)	-	1	
trasa do E	2 (via C)	2 (via D)	1	1	-	2 (via C)
trasa do F	2 (via C)	2 (via C)	1		2 (via C)	-
trasa do S	1	1	1	2 (via A)	2 (via C)	2 (via C)

Przykład tworzenia tablic

Krok 1.

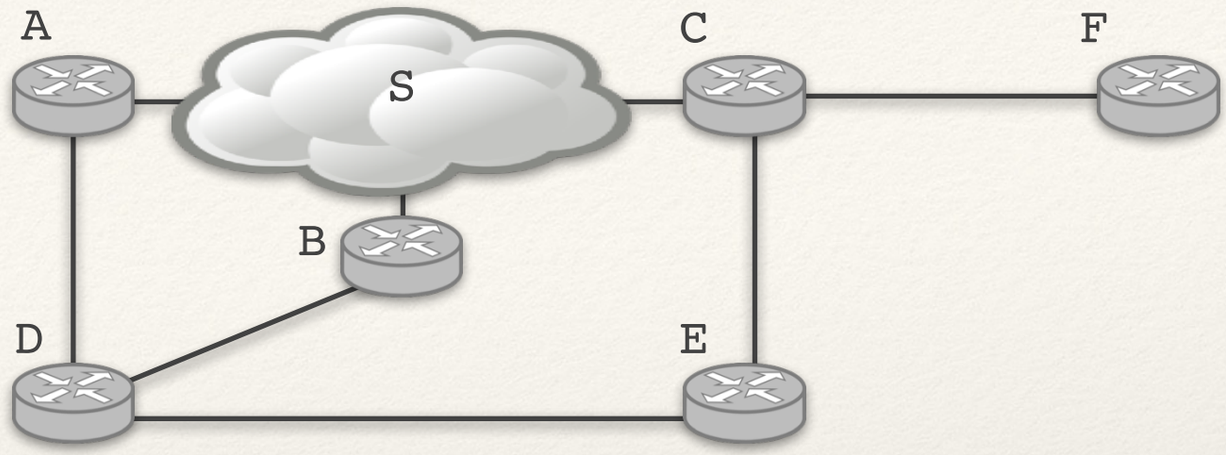


E dowiedział się też od *C*, że *A* jest sąsiadem *C*, ale *D* był szybszy.

	A	B	C	D	E	F
trasa do A	-	1	1	1	2 (via D)	2 (via C)
trasa do B	1	-	1	1	2 (via D)	2 (via C)
trasa do C	1	1	-	2 (via E)	1	1
trasa do D	1	1	2 (via E)	-	1	
trasa do E	2 (via C)	2 (via D)	1	1	-	2 (via C)
trasa do F	2 (via C)	2 (via C)	1		2 (via C)	-
trasa do S	1	1	1	2 (via A)	2 (via C)	2 (via C)

Przykład tworzenia tablic

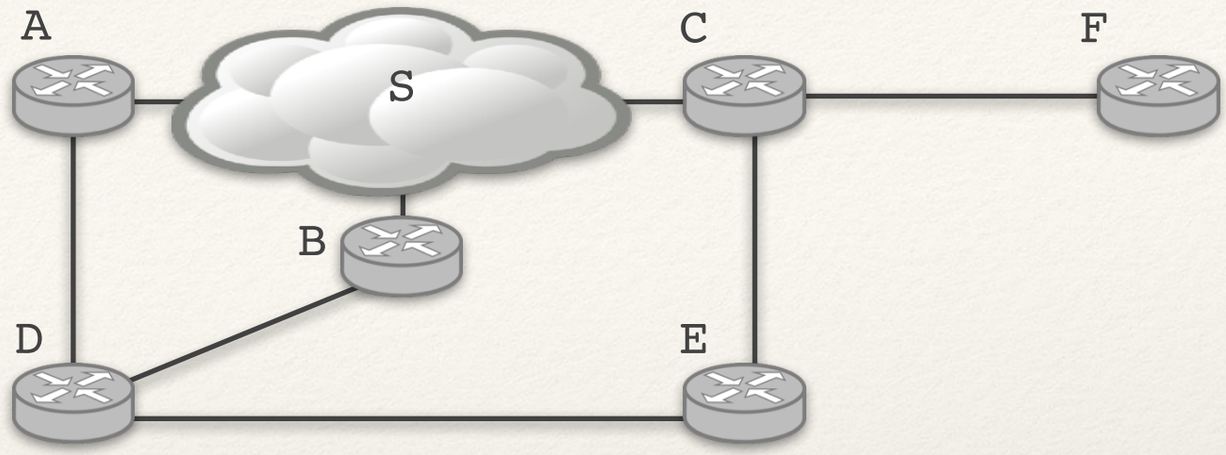
Krok 1.



	A	B	C	D	E	F
trasa do A	-	1	1	1	2 (via D)	2 (via C)
trasa do B	1	-	1	1	2 (via D)	2 (via C)
trasa do C	1	1	-	2 (via E)	1	1
trasa do D	1	1	2 (via E)	-	1	
trasa do E	2 (via C)	2 (via D)	1	1	-	2 (via C)
trasa do F	2 (via C)	2 (via C)	1		2 (via C)	-
trasa do S	1	1	1	2 (via A)	2 (via C)	2 (via C)

Przykład tworzenia tablic

Krok 1.



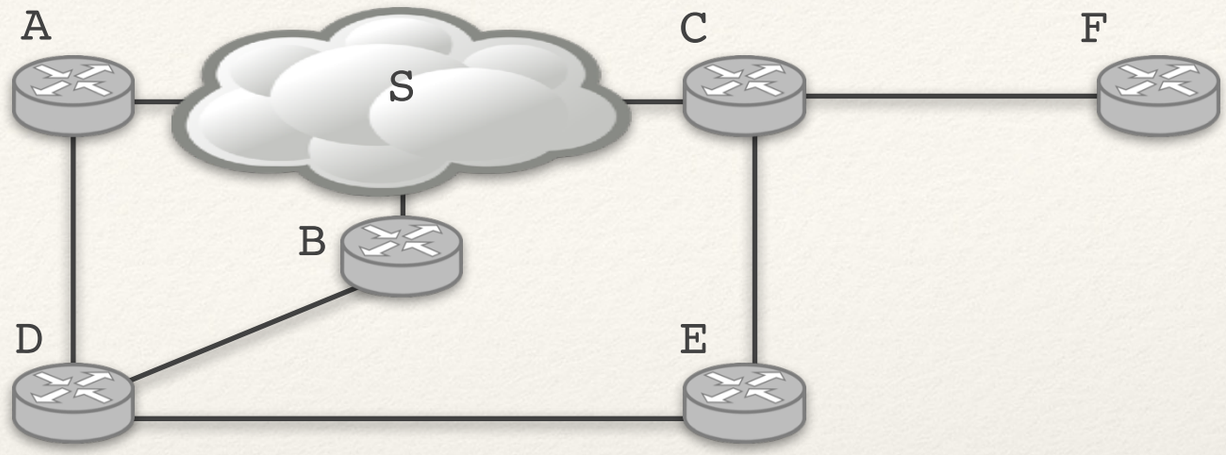
	A	B	C	D	E	F
trasa do A	-	1	1	1	2 (via D)	2 (via C)
trasa do B	1	-	1	1	2 (via D)	2 (via C)
trasa do C	1	1	-	2 (via E)	1	1
trasa do D	1	1	2 (via E)	-	1	
trasa do E	2 (via C)	2 (via D)	1	1	-	2 (via C)
trasa do F	2 (via C)	2 (via C)	1		2 (via C)	-
trasa do S				2 (via A)	2 (via C)	2 (via C)

„mam ścieżkę do F długości 2“

Przykład tworzenia tablic

Krok 2.

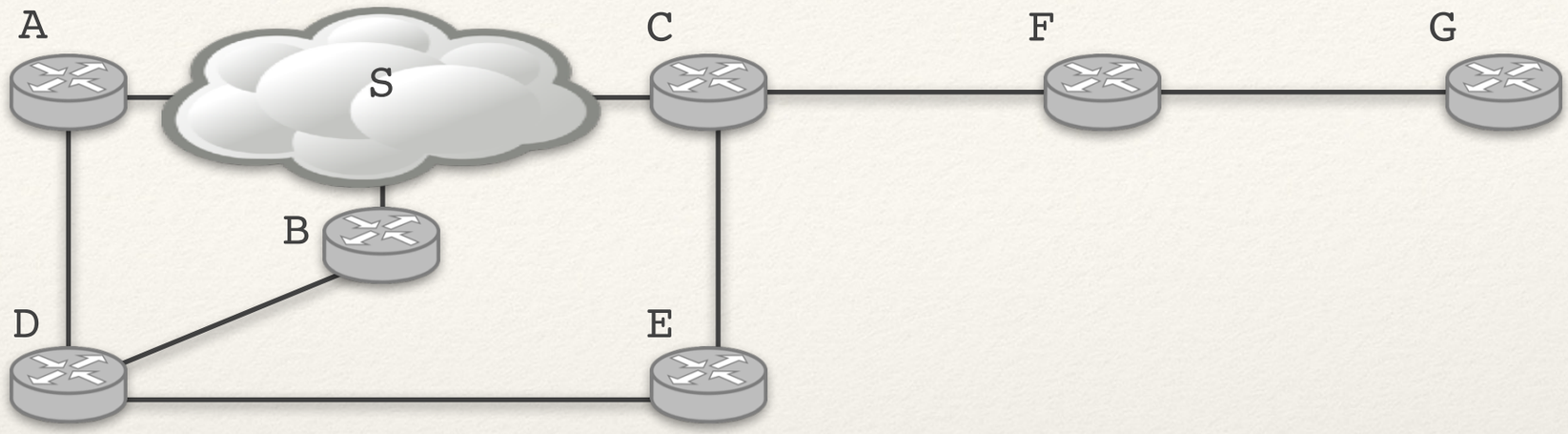
- stan stabilny: kolejne wysłania wektorów nie powodują aktualizacji



	A	B	C	D	E	F
trasa do A	-	1	1	1	2 (via D)	2 (via C)
trasa do B	1	-	1	1	2 (via D)	2 (via C)
trasa do C	1	1	-	2 (via E)	1	1
trasa do D	1	1	2 (via E)	-	1	3 (via C)
trasa do E	2 (via C)	2 (via D)	1	1	-	2 (via C)
trasa do F	2 (via C)	2 (via C)	1	3 (via A)	2 (via C)	-
trasa do S	1	1	1	2 (via A)	2 (via C)	2 (via C)

Dodawanie routera

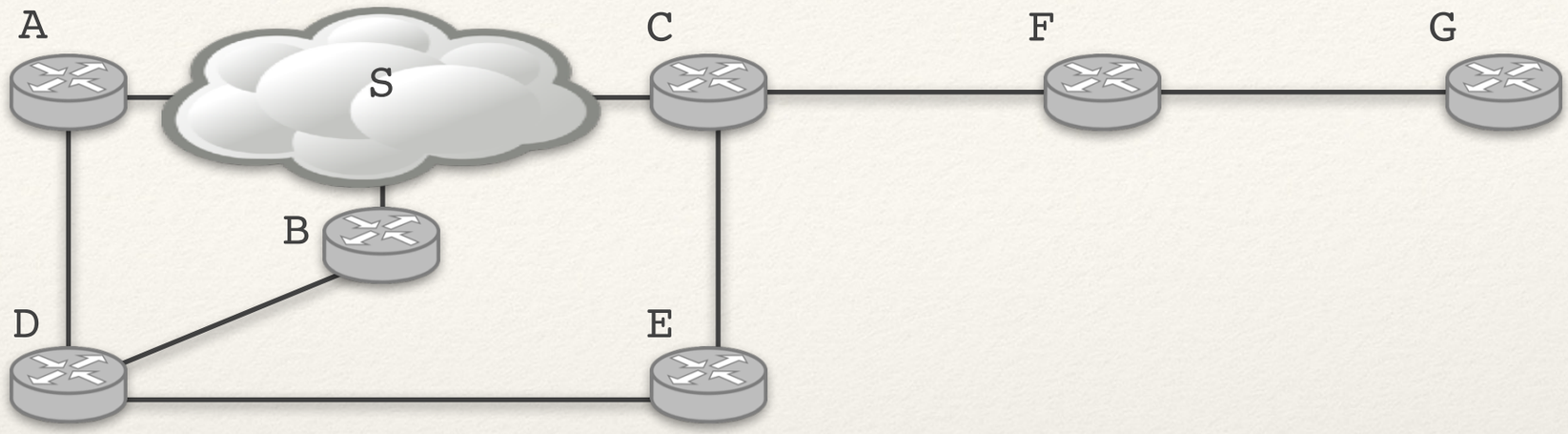
Krok 0.



	A	B	C	D	E	F	G
trasa do A	-	1	1	1	2 (via D)	2 (via C)	
trasa do B	1	-	1	1	2 (via D)	2 (via C)	
trasa do C	1	1	-	2 (via E)	1	1	
trasa do D	1	1	2 (via E)	-	1	3 (via C)	
trasa do E	2 (via C)	2 (via D)	1	1	-	2 (via C)	
trasa do F	2 (via C)	2 (via C)	1	3 (via A)	2 (via C)	-	1
trasa do S	1	1	1	2 (via A)	2 (via C)	2 (via C)	
trasa do G						1	

Dodawanie routera

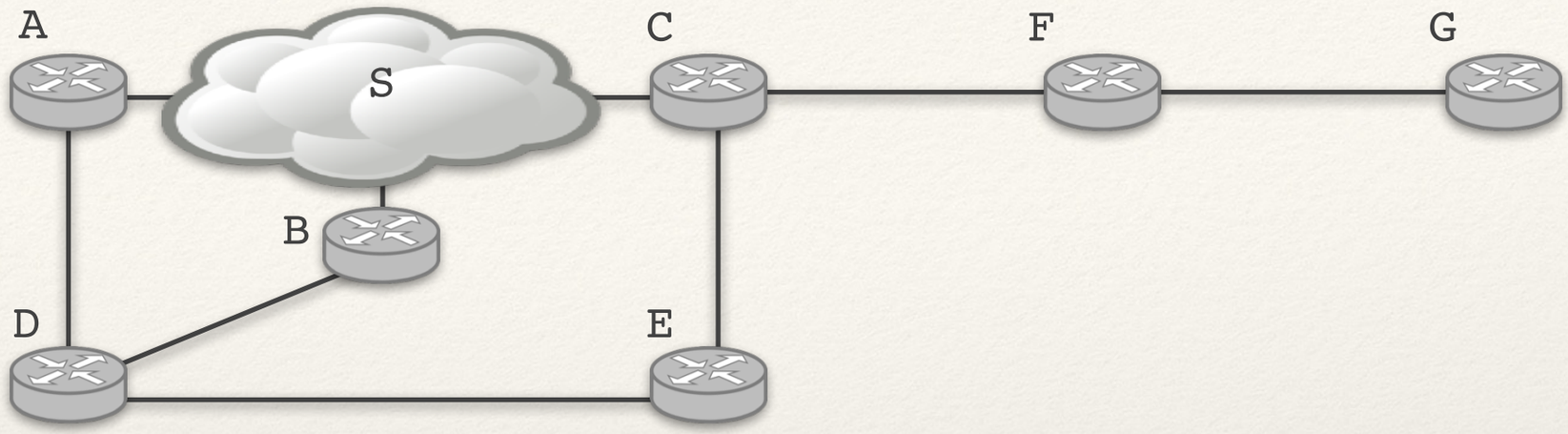
Krok 1.



	A	B	C	D	E	F	G
trasa do A	-	1	1	1	2 (via D)	2 (via C)	3 (via F)
trasa do B	1	-	1	1	2 (via D)	2 (via C)	3 (via F)
trasa do C	1	1	-	2 (via E)	1	1	2 (via F)
trasa do D	1	1	2 (via E)	-	1	3 (via C)	4 (via F)
trasa do E	2 (via C)	2 (via D)	1	1	-	2 (via C)	3 (via F)
trasa do F	2 (via C)	2 (via C)	1	3 (via A)	2 (via C)	-	1
trasa do S	1	1	1	2 (via A)	2 (via C)	2 (via C)	3 (via F)
trasa do G			2 (via F)			1	-

Dodawanie routera

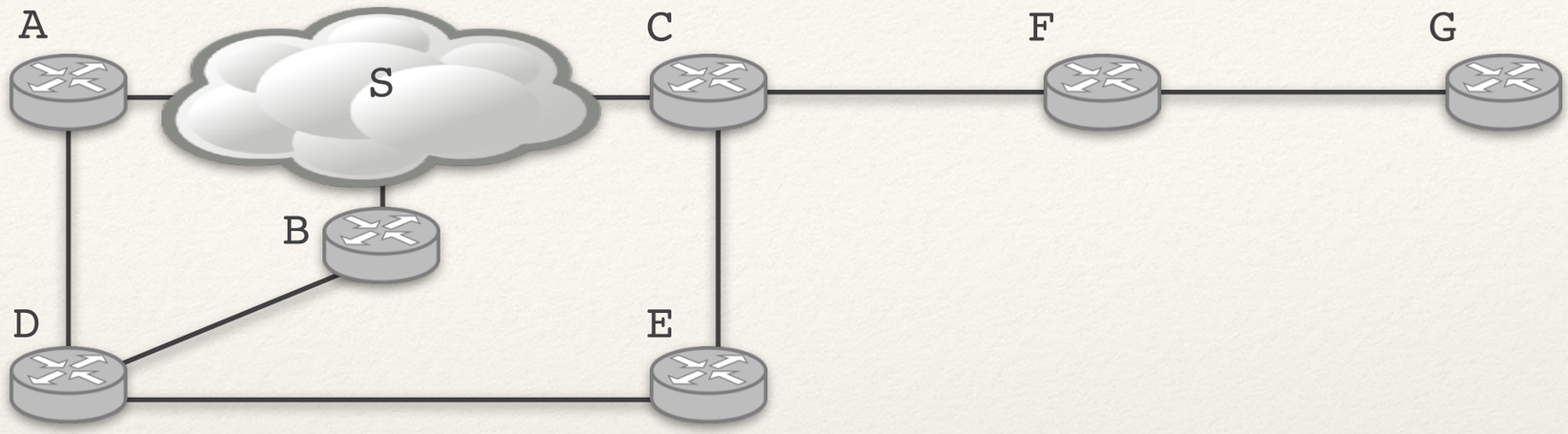
Krok 2.



	A	B	C	D	E	F	G
trasa do A	-	1	1	1	2 (via D)	2 (via C)	3 (via F)
trasa do B	1	-	1	1	2 (via D)	2 (via C)	3 (via F)
trasa do C	1	1	-	2 (via E)	1	1	2 (via F)
trasa do D	1	1	2 (via E)	-	1	3 (via C)	4 (via F)
trasa do E	2 (via C)	2 (via D)	1	1	-	2 (via C)	3 (via F)
trasa do F	2 (via C)	2 (via C)	1	3 (via A)	2 (via C)	-	1
trasa do S	1	1	1	2 (via A)	2 (via C)	2 (via C)	3 (via F)
trasa do G	3 (via C)	3 (via C)	2 (via F)		3 (via C)	1	-

Dodawanie routera

Krok 3.



	A	B	C	D	E	F	G
trasa do A	-	1	1	1	2 (via D)	2 (via C)	3 (via F)
trasa do B	1	-	1	1	2 (via D)	2 (via C)	3 (via F)
trasa do C	1	1	-	2 (via E)	1	1	2 (via F)
trasa do D	1	1	2 (via E)	-	1	3 (via C)	4 (via F)
trasa do E	2 (via C)	2 (via D)	1	1	-	2 (via C)	3 (via F)
trasa do F	2 (via C)	2 (via C)	1	3 (via A)	2 (via C)	-	1
trasa do S	1	1	1	2 (via A)	2 (via C)	2 (via C)	3 (via F)
trasa do G	3 (via C)	3 (via C)	2 (via F)	4 (via B)	3 (via C)	1	-

Szybkość zbieżności

- Odległości będą poprawne po D turach, gdzie D jest średnicą sieci.
- Informacja o dodaniu routera lub łącza propaguje się z prędkością jednej krawędzi na turę.
- A informacja o awarii?

Awaria łącza

Aktualizacja sąsiedztwa:

- Wpisujemy odległość nieskończoną do nieosiągalnego routera / sieci.

Aktualizacja tablicy dla routera X jeśli A jest pierwszym routerem na trasie do B:

$$d(X,B) \leftarrow s(X,A) + d(A,B)$$

A mówi: „mam do B odległość d(A,B)".

Aktualizujemy, nawet jeśli nowa trasa jest gorsza niż posiadana!

A jest sąsiadem X odległym o s(X,A).

Przykład awarii łącza (1)

trasa do D	A	B	C
czas = 0	3 (via B)	2 (via C)	1

Przykład awarii łącza (1)

- **Łącze pomiędzy C i D psuje się.**

trasa do D	A	B	C
czas = 0	3 (via B)	2 (via C)	1
czas = 1	3 (via B)	2 (via C)	∞

Przykład awarii łącza (1)

- **Łącze pomiędzy C i D psuje się.**
- **Dobry przypadek:**
  - C przekazuje swoją tablicę do B wcześniej niż B do C.
  - B przekazuje swoją tablicę do A wcześniej niż A do B.

trasa do D	A	B	C
czas = 0	3 (via B)	2 (via C)	1
czas = 1	3 (via B)	2 (via C)	∞
czas = 2	3 (via B)	∞	∞

Przykład awarii łącza (1)

- **Łącze pomiędzy C i D psuje się.**
- **Dobry przypadek:**
  - C przekazuje swoją tablicę do B wcześniej niż B do C.
  - B przekazuje swoją tablicę do A wcześniej niż A do B.

trasa do D	A	B	C
czas = 0	3 (via B)	2 (via C)	1
czas = 1	3 (via B)	2 (via C)	∞
czas = 2	3 (via B)	∞	∞
czas = 3	∞	∞	∞

Przykład awarii łącza (2)

- **Łącze pomiędzy C i D psuje się.**
- **Zły przypadek:** B przekazuje najpierw swoją tablicę do C.

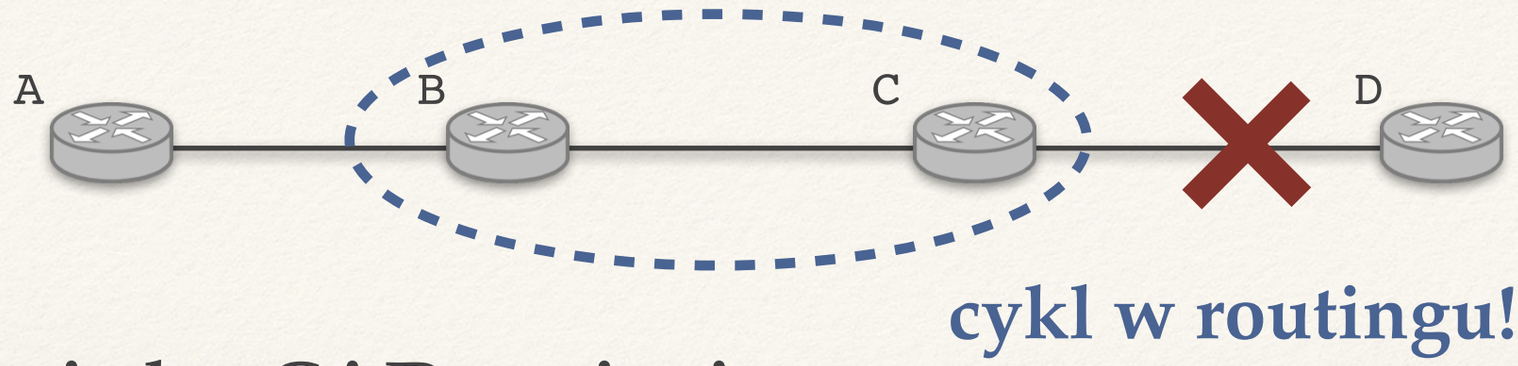
trasa do D	A	B	C
czas = 0	3 (via B)	2 (via C)	1
czas = 1	3 (via B)	2 (via C)	∞

Przykład awarii łącza (2)

- **Łącze pomiędzy C i D psuje się.**
- **Zły przypadek:** B przekazuje najpierw swoją tablicę do C.

trasa do D	A	B	C
czas = 0	3 (via B)	2 (via C)	1
czas = 1	3 (via B)	2 (via C)	∞
czas = 2	3 (via B)	2 (via C)	**3 (via B)**

Przykład awarii łącza (2)

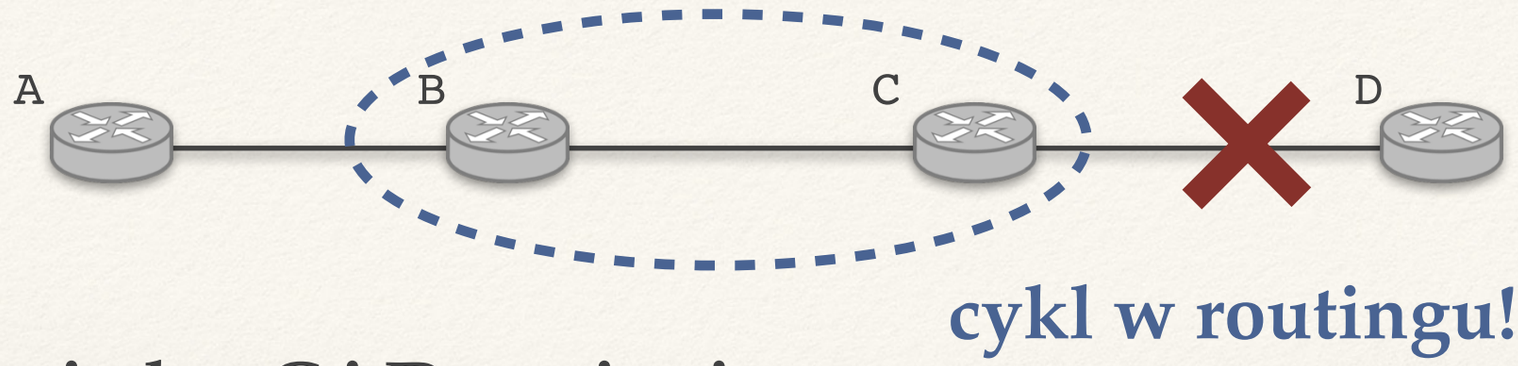


cykl w routingu!

- **Łącze pomiędzy C i D psuje się.**
- **Zły przypadek:** B przekazuje najpierw swoją tablicę do C.

trasa do D	A	B	C
czas = 0	3 (via B)	2 (via C)	1
czas = 1	3 (via B)	2 (via C)	∞
czas = 2	3 (via B)	2 (via C)	**3 (via B)**

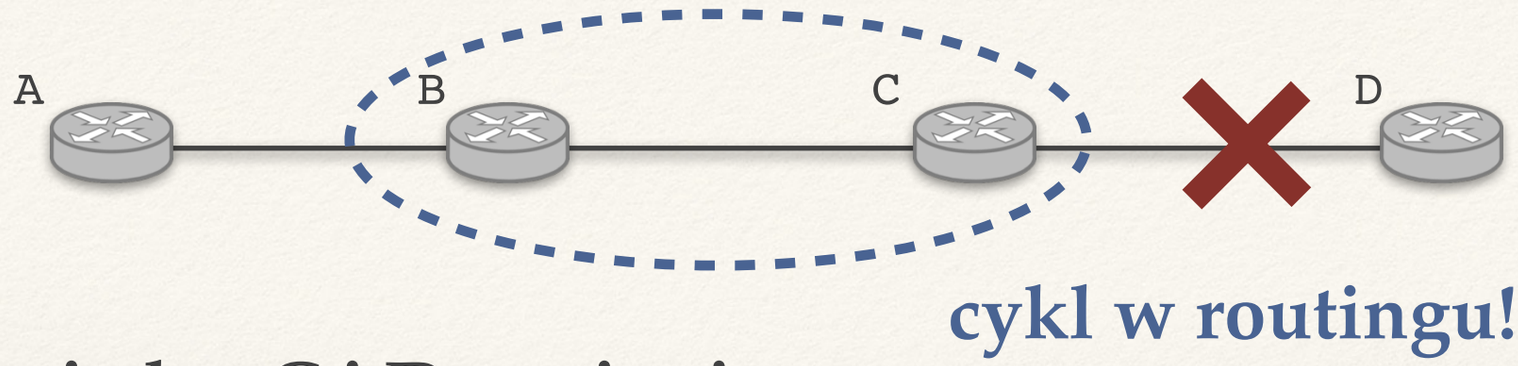
Przykład awarii łącza (2)



- **Łącze pomiędzy C i D psuje się.**
- **Zły przypadek:** B przekazuje najpierw swoją tablicę do C.

trasa do D	A	B	C
czas = 0	3 (via B)	2 (via C)	1
czas = 1	3 (via B)	2 (via C)	∞
czas = 2	3 (via B)	2 (via C)	**3 (via B)**
czas = 3	3 (via B)	4 (via C)	3 (via B)

Przykład awarii łącza (2)



cykl w routingu!

- **Łącze pomiędzy C i D psuje się.**
- **Zły przypadek:** B przekazuje najpierw swoją tablicę do C.

trasa do D	A	B	C
czas = 0	3 (via B)	2 (via C)	1
czas = 1	3 (via B)	2 (via C)	∞
czas = 2	3 (via B)	2 (via C)	**3 (via B)**
czas = 3	3 (via B)	4 (via C)	3 (via B)
czas = 4	5 (via B)	4 (via C)	5 (via B)

Przykład awarii łącza (2)

cykl w routingu!

- **Łącze pomiędzy C i D psuje się.**
- **Zły przypadek:** B przekazuje najpierw swoją tablicę do C.

trasa do D	A	B	C
czas = 0	3 (via B)	2 (via C)	1
czas = 1	3 (via B)	2 (via C)	∞
czas = 2	3 (via B)	2 (via C)	**3 (via B)**
czas = 3	3 (via B)	4 (via C)	3 (via B)
czas = 4	5 (via B)	4 (via C)	5 (via B)
czas = 5	5 (via B)	6 (via C)	5 (via B)

Przykład awarii łącza (2)

cykl w routingu!

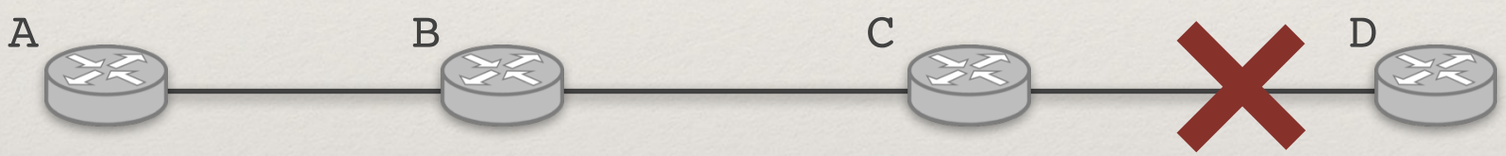
- **Łącze pomiędzy C i D psuje się.**
- **Zły przypadek:** B przekazuje najpierw swoją tablicę do C.

trasa do D	A	B	C
czas = 0	3 (via B)	2 (via C)	1
czas = 1	3 (via B)	2 (via C)	∞
czas = 2	3 (via B)	2 (via C)	**3 (via B)**
czas = 3	3 (via B)	4 (via C)	3 (via B)
czas = 4	5 (via B)	4 (via C)	5 (via B)
czas = 5	5 (via B)	6 (via C)	5 (via B)
…	…	…	…

Zliczanie do nieskończoności (1)

- **Problem zliczania do nieskończoności:**
  - Routery zwiększają znaną odległość do *D* średnio o 1 na turę.

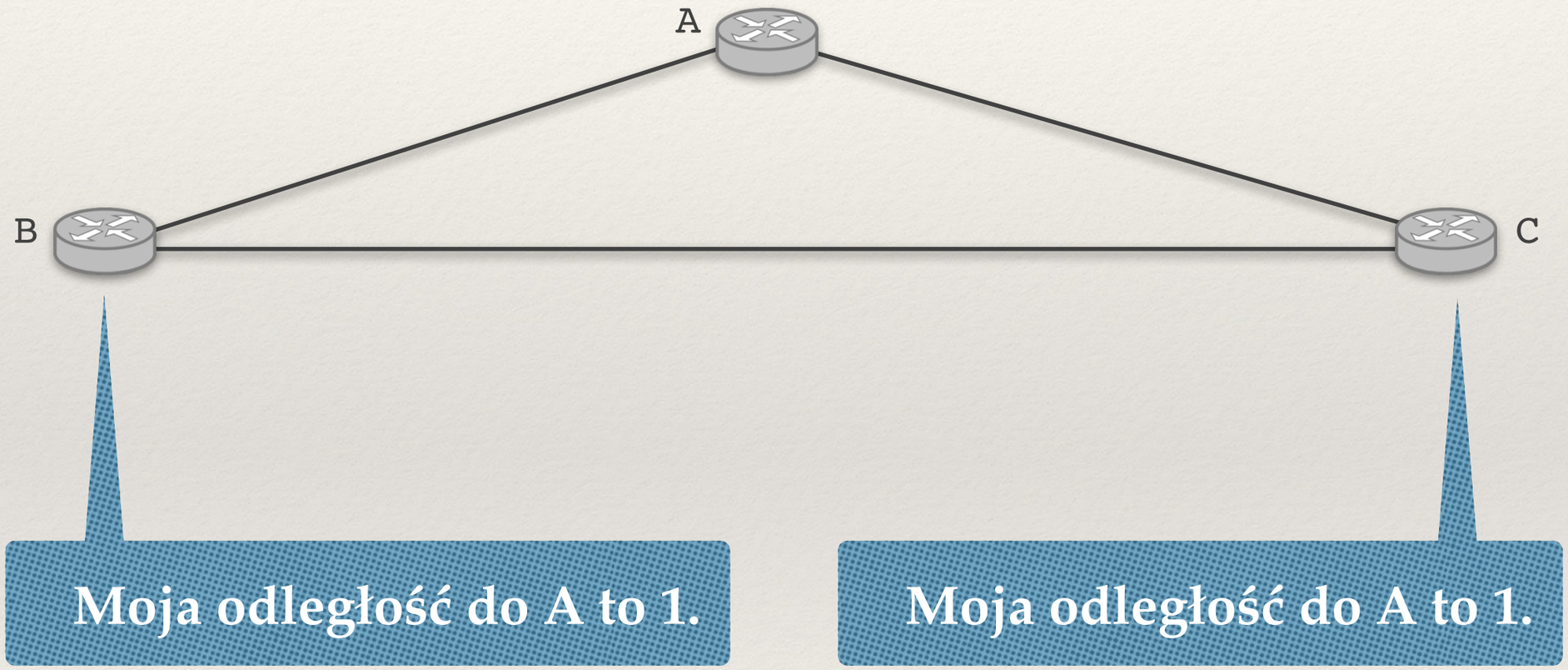
- **Dlaczego problem wystąpił:**
  - B wysłał do C informację o odległości do D ale C jest na tej trasie!



- **Zatruwanie ścieżki zwrotnej (*poison reverse*)**:
  - Jeśli X jest wpisany jako następny router na ścieżce do Y,
    to wysyłamy do X informację „mam do Y ścieżkę nieskończoną“.
  - Może nie pomóc w większych sieciach → ćwiczenie.

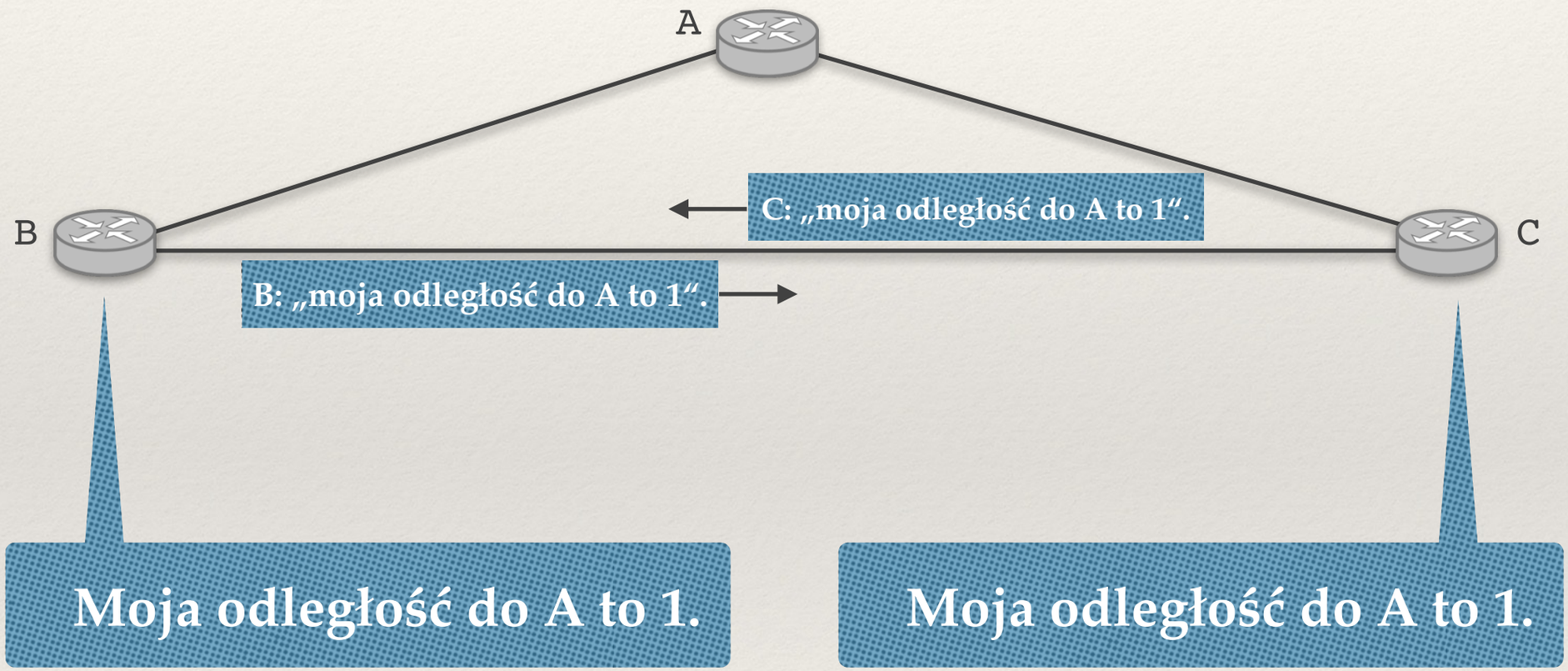
Zatruwanie ścieżki zwrotnej (poison reverse)

- Jeśli X jest wpisany jako następny router na ścieżce do Y to wysyłamy do X informację „mam do Y ścieżkę nieskończoną“.
- Po co w ogóle coś wysyłać?



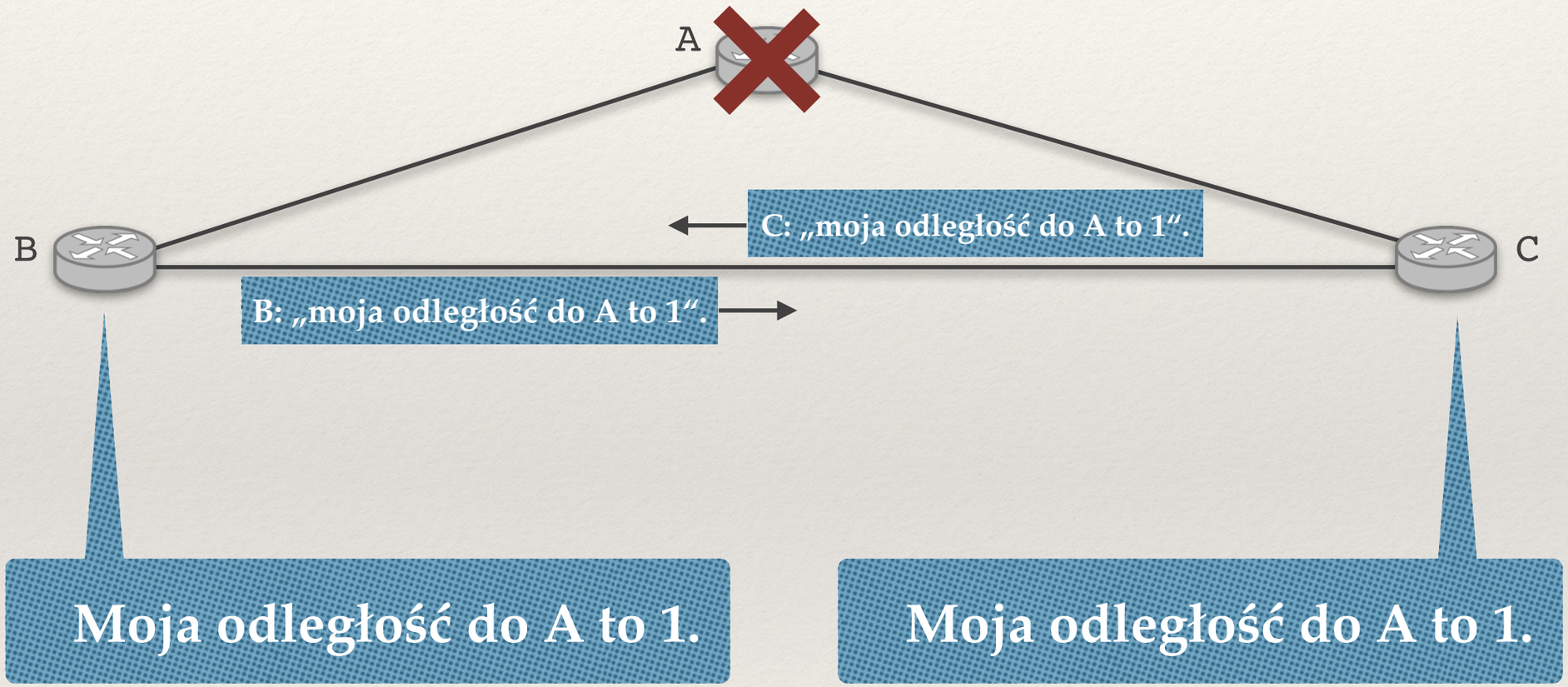
Zatruwanie ścieżki zwrotnej (poison reverse)

- Jeśli X jest wpisany jako następny router na ścieżce do Y to wysyłamy do X informację „mam do Y ścieżkę nieskończoną“.
- Po co w ogóle coś wysyłać?



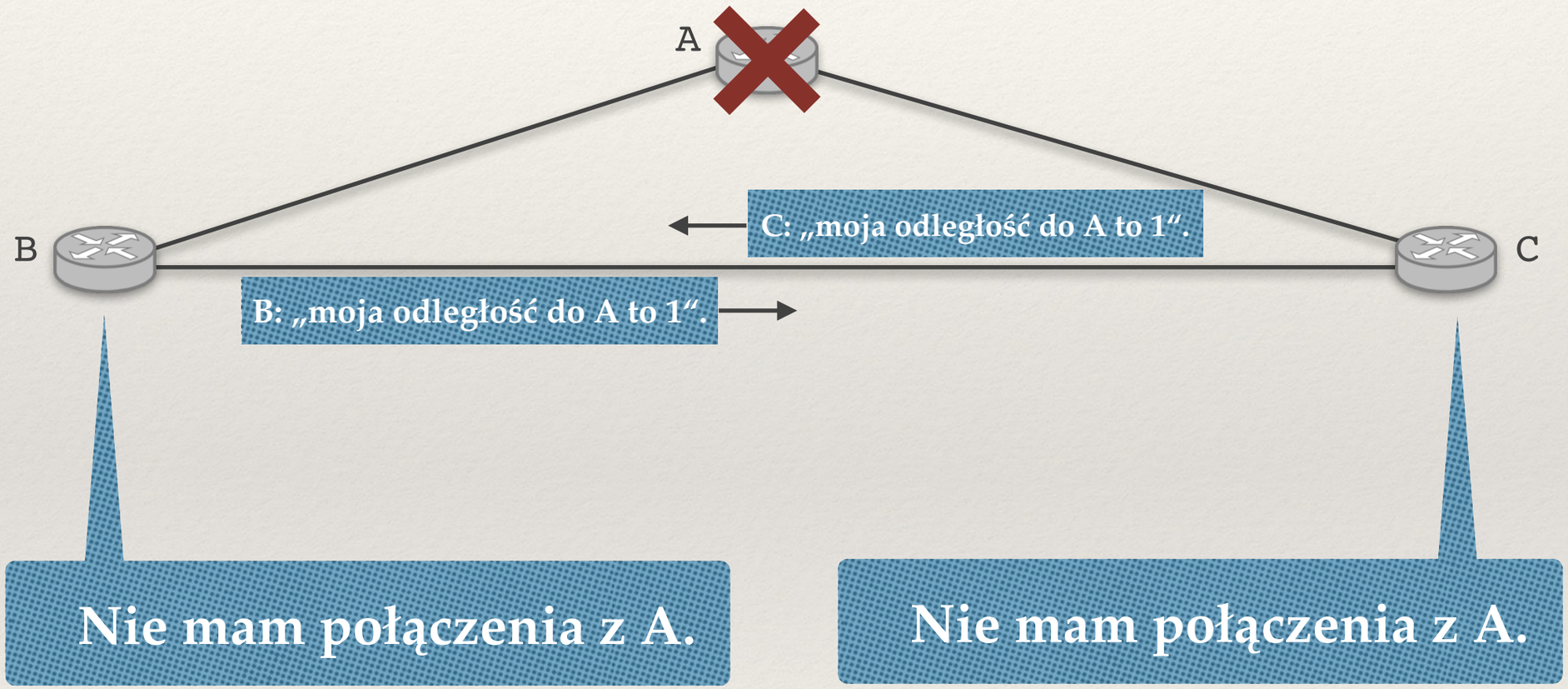
Zatruwanie ścieżki zwrotnej (poison reverse)

- Jeśli X jest wpisany jako następny router na ścieżce do Y to wysyłamy do X informację „mam do Y ścieżkę nieskończoną“.
- Po co w ogóle coś wysyłać?



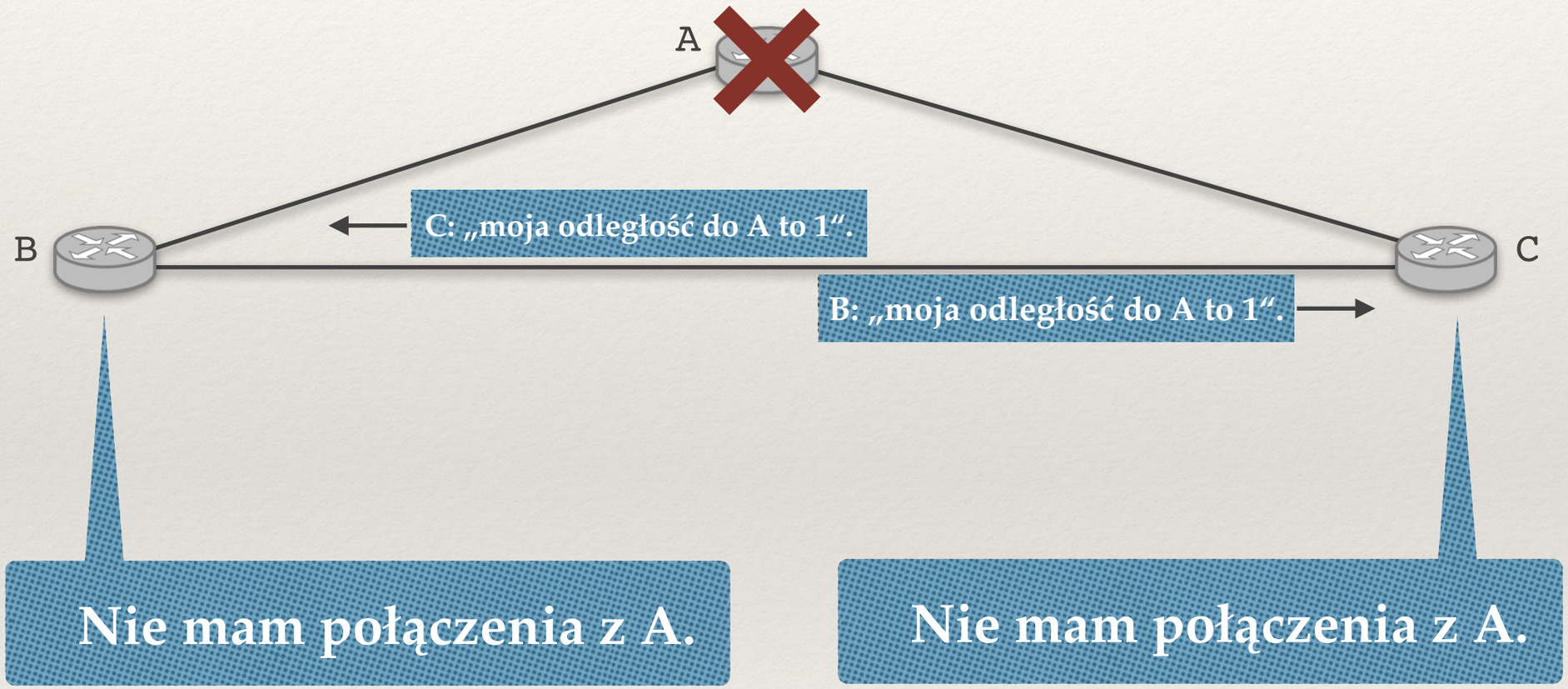
Zatruwanie ścieżki zwrotnej (poison reverse)

- Jeśli X jest wpisany jako następny router na ścieżce do Y to wysyłamy do X informację „mam do Y ścieżkę nieskończoną“.
- Po co w ogóle coś wysyłać?



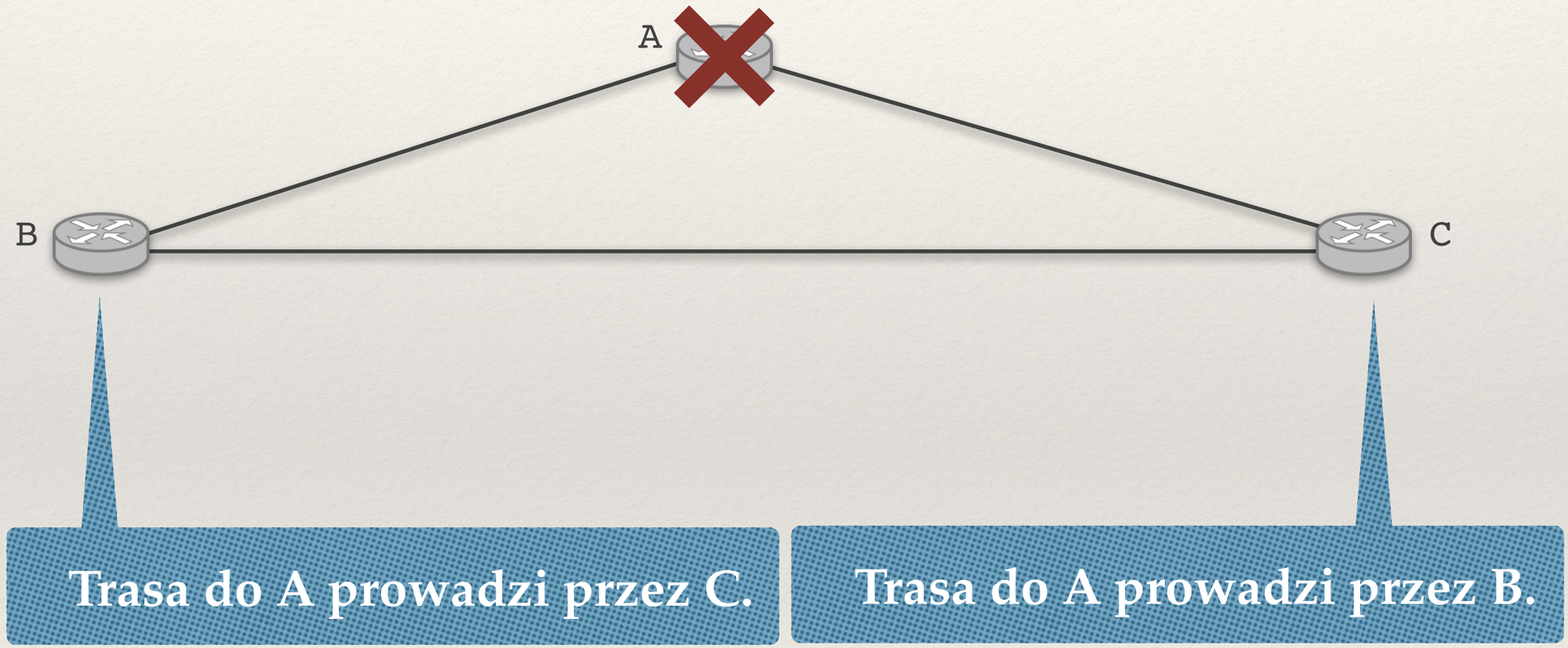
Zatruwanie ścieżki zwrotnej (poison reverse)

- Jeśli X jest wpisany jako następny router na ścieżce do Y to wysyłamy do X informację „mam do Y ścieżkę nieskończoną“.
- Po co w ogóle coś wysyłać?



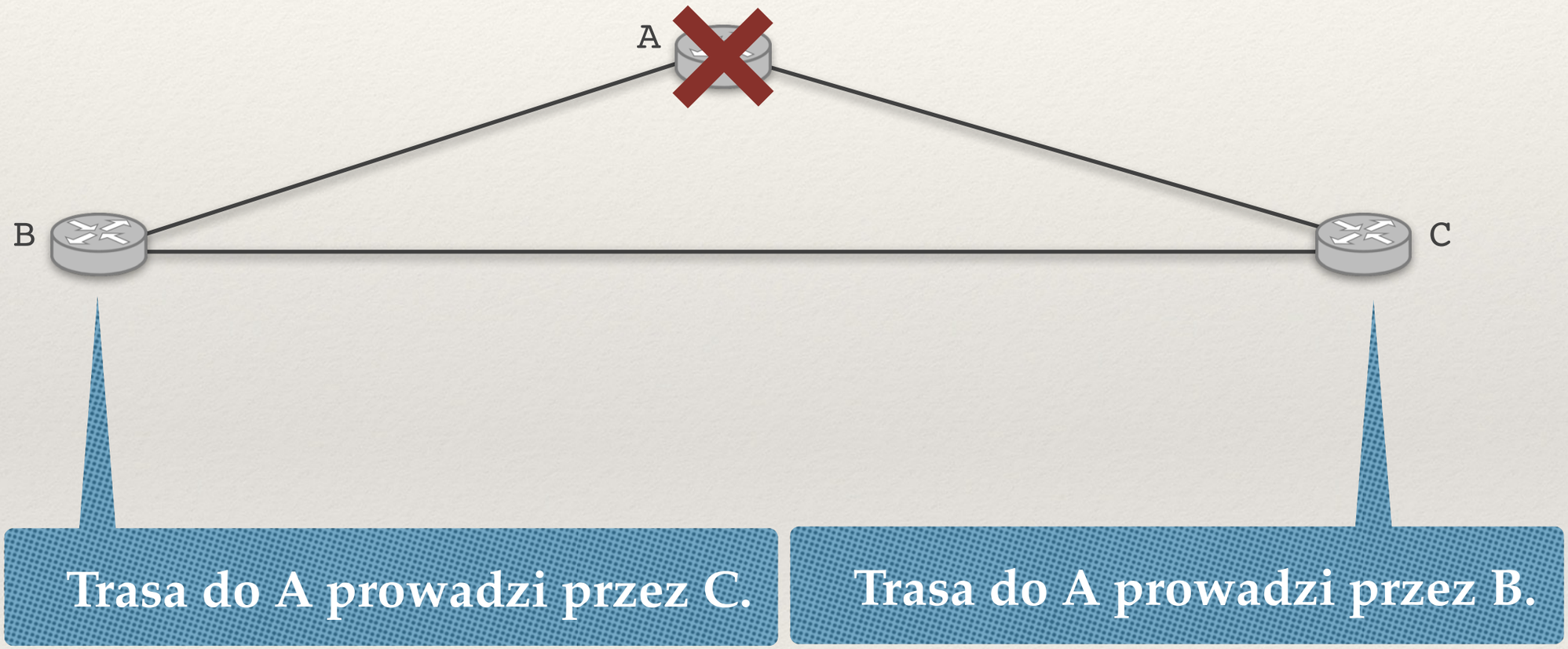
Zatruwanie ścieżki zwrotnej (poison reverse)

- Jeśli X jest wpisany jako następny router na ścieżce do Y to wysyłamy do X informację „mam do Y ścieżkę nieskończoną“.
- Po co w ogóle coś wysyłać?



Zatruwanie ścieżki zwrotnej (poison reverse)

- Jeśli X jest wpisany jako następny router na ścieżce do Y to wysyłamy do X informację „mam do Y ścieżkę nieskończoną“.
- Po co w ogóle coś wysyłać?



- Teraz wysłanie zatrutej trasy zwrotnej rozwiąże ten problem, w przeciwnym przypadku sytuacja pozostanie niezmieniona!

Zliczanie do nieskończoności (2)

Dodatkowe pomysły rozwiązania:

- Wysyłanie również pierwszego routera na trasie (nie pomaga w większych sieciach).
- Szybsza aktualizacja w momencie wykrycia awarii.
- Jeśli wszystko inne zawiedzie: ustalić wartość graniczną odległości: powyżej niej router jest już uważany za nieosiągalny.

Algorytmy wektora odległości w Internecie

Protokół RIP (Routing Information Protocol)

- wysyłanie wektora odległości co 30 sek + w momencie zmiany;
- zatruwanie ścieżki zwrotnej;
- $\infty = 16$ (w RIPv1);
- nieefektywny dla większych sieci.

Porównanie algorytmów

	stan łączy	wektory odległości
pamięć	O(|V|+|E|)	O(|V|)
implementacja	trudniejszy (zalewanie)	łatwiejszy (tylko kontakt z sąsiadami)
szybkość zbieżności (w praktyce)	szybsza	wolniejsza
zapotrzebowanie na moc obliczeniową	większe (algorytm Dijkstry)	mniejsze (tylko aktualizacja odległości)

Routing w Internecie

Routing w Internecie

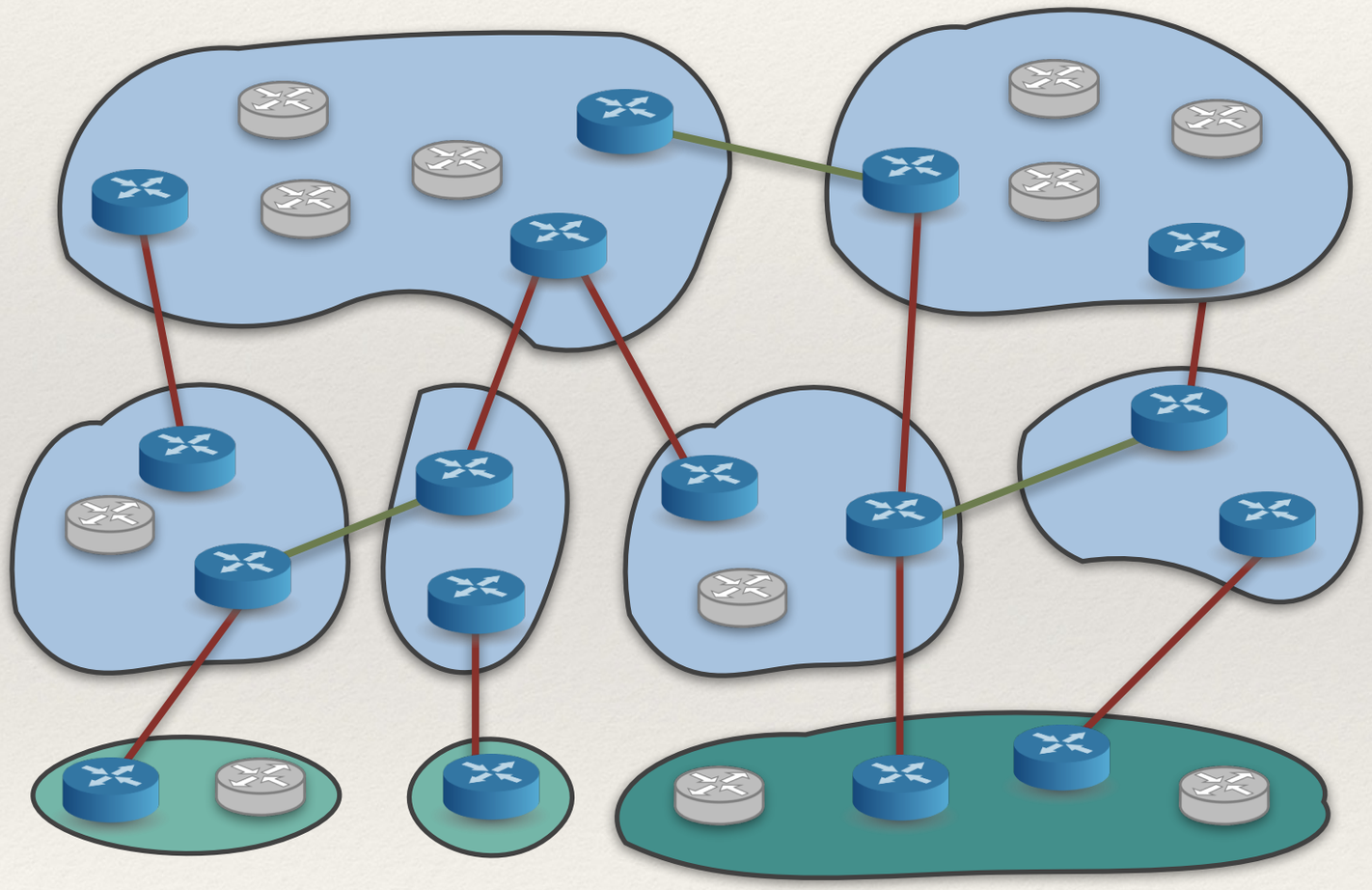
- **Omówiliśmy dwa podejścia minimalizujące pewną funkcję celu:**
  - przesyłają wszystko ścieżką najkrótszą;
  - łączom o małej przepustowości można przypisywać duże wagi.

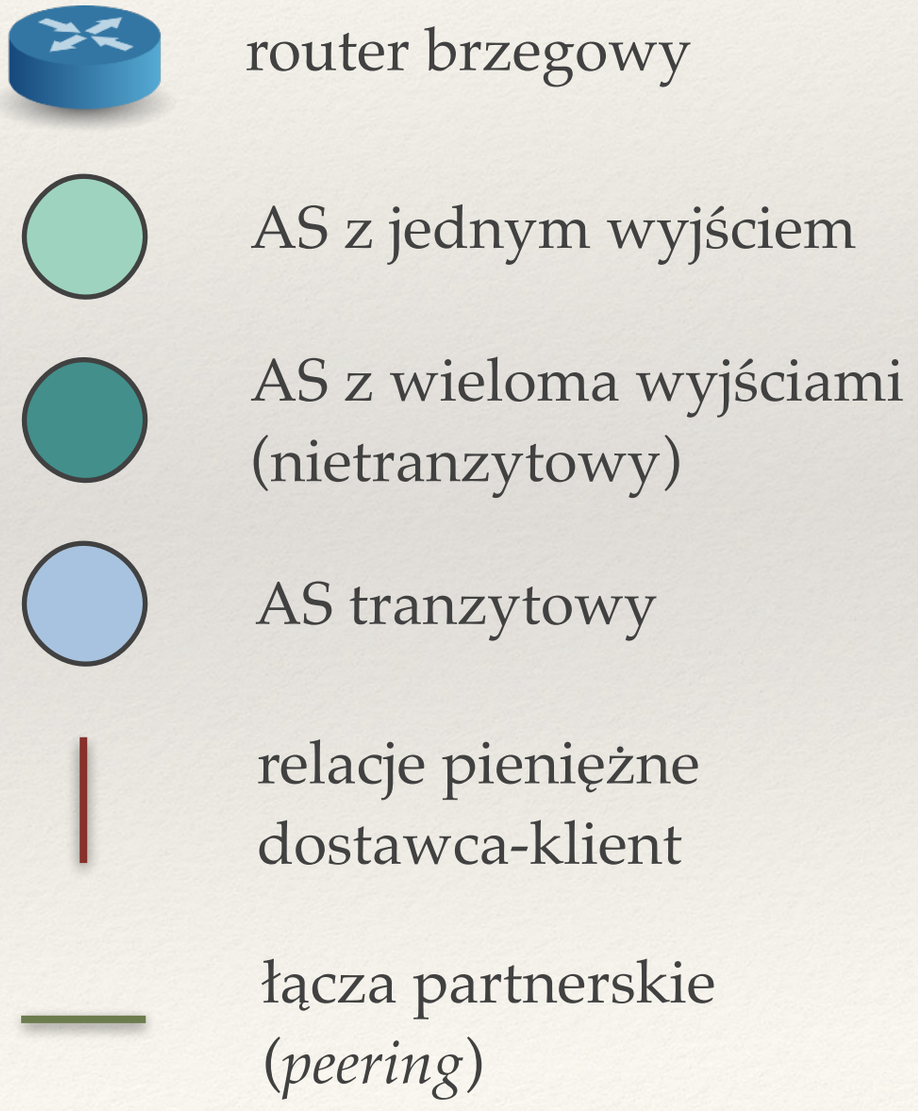
- **Nie to, co chciałby optymalizować ISP!**
  - ISP = Internet Service Provider (dostawca Internetu)

Systemy autonomiczne

Każdy ISP posiada jeden lub więcej system autonomiczny (AS).

- ~18 tys. ISP, ~47 tys. AS.
- Spójna polityka wewnętrznego routingu (często OSPF, rzadziej RIP).





Gdzie jesteśmy na mapie: http://bgp.he.net/AS8970

Przykładowa trasa wraz z AS

```
$> traceroute -A google.com

traceroute to google.com (172.217.20.206), 30 hops max, 60 byte packets
 1 172.16.16.254 (172.16.16.254) [*]  0.206 ms
 2 info.wask.wroc.pl (156.17.4.254) [AS8970]  1.579 ms
 3 matchem-vprn509-curie-uni.wask.wroc.pl (156.17.252.26) [AS8970]  0.597 ms
 4 uwrvprn509-unir2.wask.wroc.pl (156.17.252.37) [AS8970]  1.207 ms
 5 unir2-uwrvprn509.wask.wroc.pl (156.17.252.36) [AS8970]  0.777 ms
 6 z-Wroclaw.lodz-gw2.10Gb.rtr.pionier.gov.pl (212.191.240.121) [AS8501] 10.684 ms
 7 poznan-gw3.z-lodz-gw2.rtr.pionier.gov.pl (212.191.126.70) [AS8501]  9.588 ms
 8 72.14.203.178 (72.14.203.178) [AS15169]  17.730 ms
 9 108.170.250.209 (108.170.250.209) [AS15169]  15.131 ms
10 216.239.41.171 (216.239.41.171) [AS15169]  13.652 ms
   216.239.41.169 (216.239.41.169) [AS15169]  13.721 ms
11 waw02s08-in-f14.1e100.net (172.217.20.206) [AS15169]  13.640 ms
```

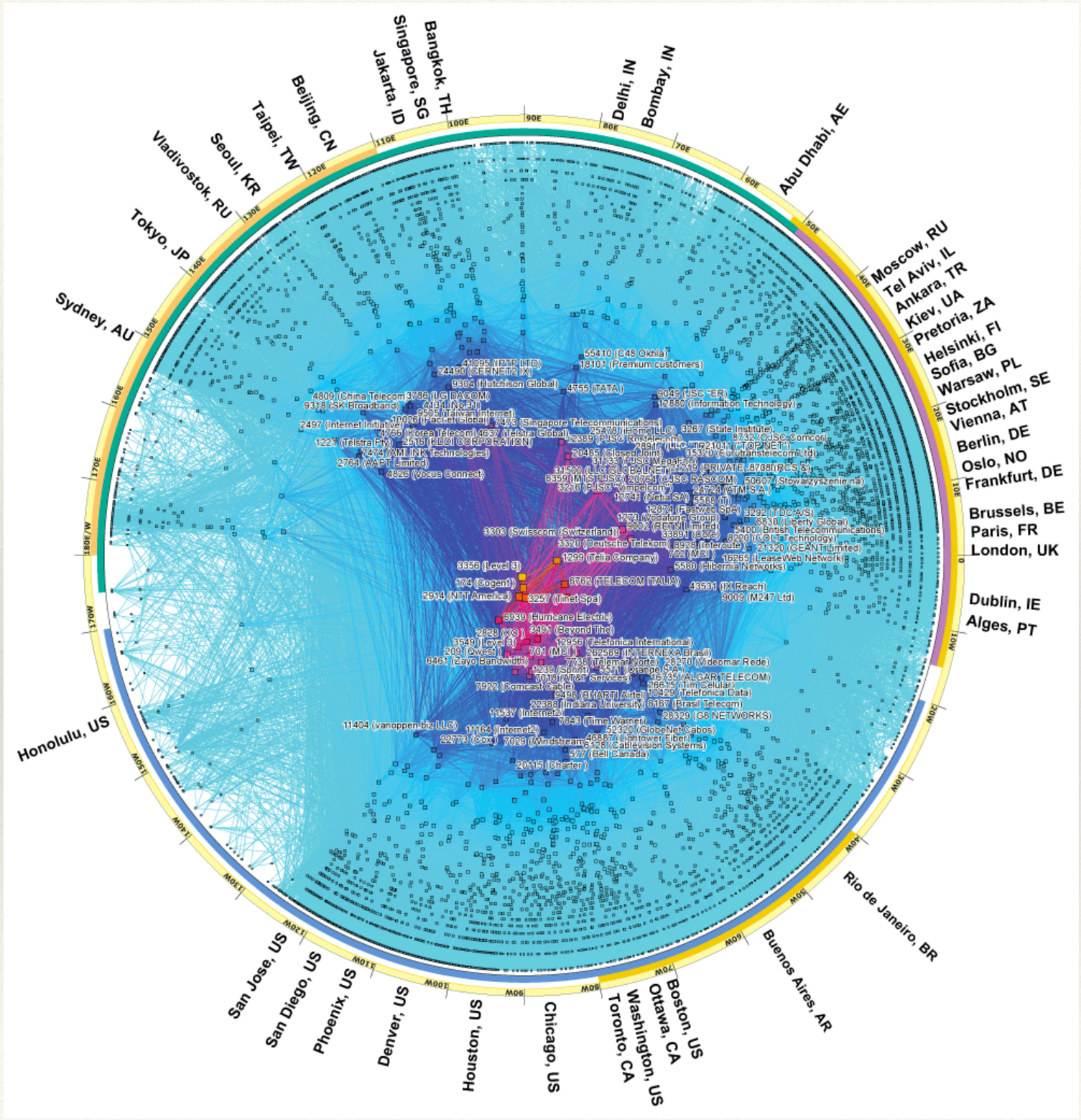
Przykładowa trasa wraz z AS

```
$> traceroute -A google.com

traceroute to google.com (172.217.20.206), 30 hops max, 60 byte packets
 1 172.16.16.254 (172.16.16.254) [*]  0.206 ms
 2 info.wask.wroc.pl (156.17.4.254) [AS8970]  1.579 ms
 3 matchem-vprn509-curie-uni.wask.wroc.pl (156.17.252.26) [AS8970]  0.597 ms
 4 uwrvprn509-unir2.wask.wroc.pl (156.17.252.37) [AS8970]  1.207 ms
 5 unir2-uwrvprn509.wask.wroc.pl (156.17.252.36) [AS8970]  0.777 ms
 6 z-Wroclaw.lodz-gw2.10Gb.rtr.pionier.gov.pl (212.191.240.121) [AS8501] 10.684 ms
 7 poznan-gw3.z-lodz-gw2.rtr.pionier.gov.pl (212.191.126.70) [AS8501]  9.588 ms
 8 72.14.203.178 (72.14.203.178) [AS15169]  17.730 ms
 9 108.170.250.209 (108.170.250.209) [AS15169]  15.131 ms
10 216.239.41.171 (216.239.41.171) [AS15169]  13.652 ms
   216.239.41.169 (216.239.41.169) [AS15169]  13.721 ms
11 waw02s08-in-f14.1e100.net (172.217.20.206) [AS15169]  13.640 ms
```

demonstracja

Mapa ISP z 2017 roku



Obrazek ze strony
https://www.caida.org/research/topology/as_core_network/2017/

Czego chcą ISP?

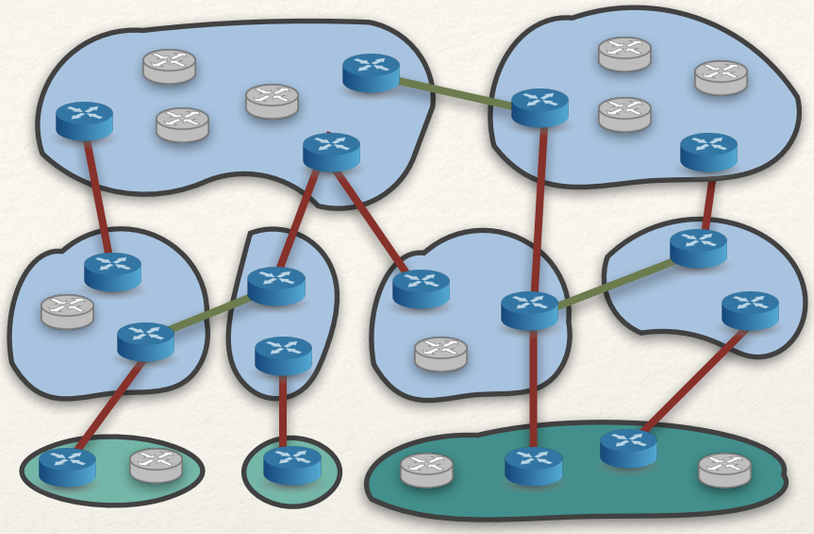
- Wybór tras routingu na podstawie **polityki ISP**, np.:
  - „Chcę płacić jak najmniej“.
  - „Nie chcę udostępniać wewnętrznych szczegółów na temat AS“.
  - „Nie chcę żeby ktoś przesyłał dane przez mój AS, jeśli nie mam z tego zysku“.
- Względy ekonomiczne, prywatności, autonomii.
- Polityki nie są realizowane przez najkrótsze ścieżki!

Border Gateway Protocol (BGP)

Algorytm routingu pomiędzy AS.

- Bazuje na algorytmach wektora odległości.
  - Bo algorytmy stanu łączy nie gwarantują prywatności i wymagają uzgodnień pomiędzy ISP.
- Rozgłaszane są całe poznane trasy *„Sieć 123.123.0.0/16 jest osiągalna przez trasę {AS3, AS21, AS13}, pierwszy router to 34.34.34.34“* → łatwe unikanie cykli.
- **ISP sam decyduje:**
  - czy i komu rozgłosić poznaną trasę;
  - które trasy wykorzysta do tworzenia tablic przekazywania.

Filtrowanie tras: które trasy warto rozgłaszać?

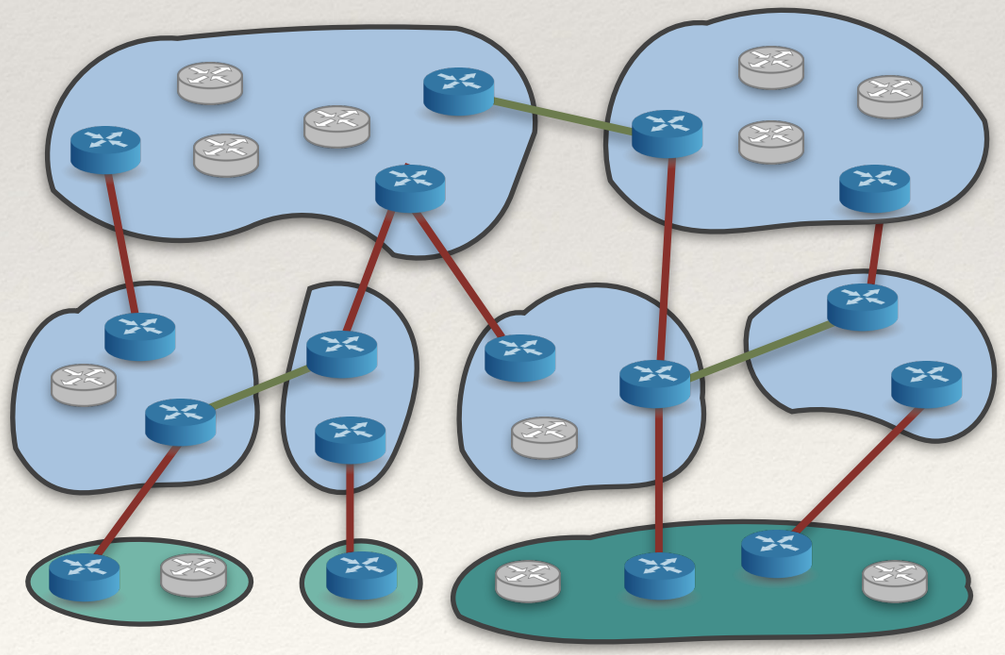


- **Zawartość naszego AS (prefiksy CIDR):**
  - Inaczej nikt do nas nie trafi.
- **Trasy do naszych klientów:**
  - Tak, bo klienci nam płacą, za przesyłane dane.
  - Szczególnie warto rozgłaszać je naszym partnerom, bo to jest ruch za który nie płacimy.
- **Trasy do naszych dostawców:**
  - Naszym klientom tak.
  - Poza tym nie: nie chcemy, żeby inni przesyłali przez nasz AS ruch do naszego dostawcy (my płacimy, nam nie płacą).
- **Trasy do naszych partnerów:**
  - Naszym klientom tak.
  - Poza tym zazwyczaj nie.

Wybór tras

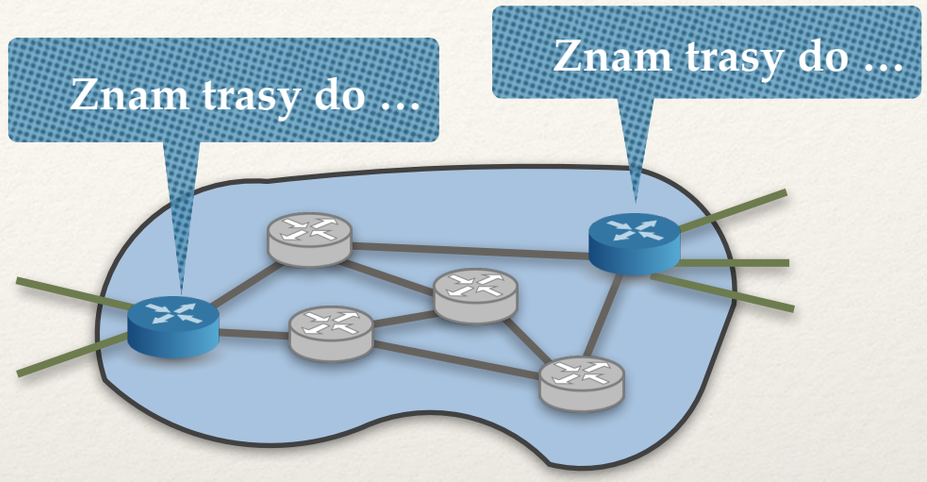
Wiele możliwości dotarcia do jakiejś sieci (prefiksu CIDR)

- Zazwyczaj wybór najkrótszej trasy (najmniejsza liczba AS).
- Ale można zmienić taki wybór. Częsta polityka:
  - wybierz najpierw trasę przez swojego klienta,
  - … potem przez partnera,
  - … a na końcu trasę przez dostawcę.



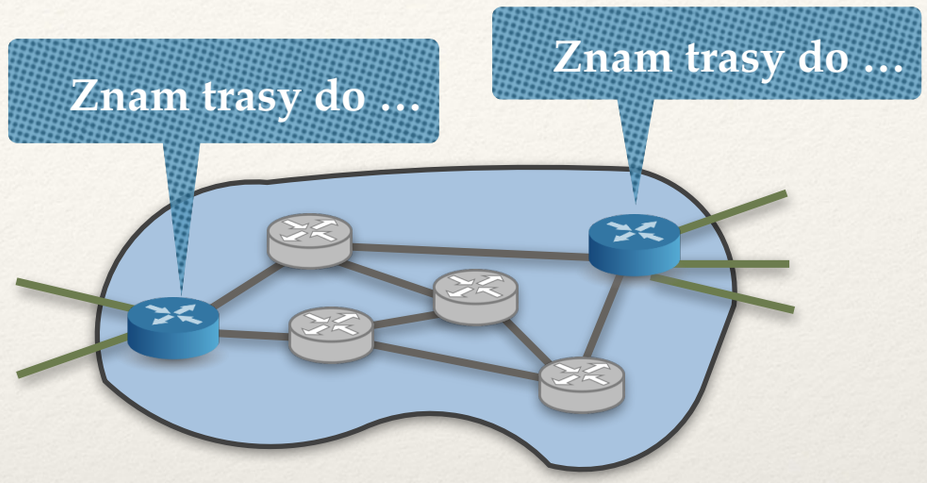
Routing pomiędzy i wewnątrz AS, idea (1)

- **Routery brzegowe danego AS** (via BGP):
  - rozgłoś prefiksy CIDR tego AS;
  - dowiedz się o trasach do innych AS.



Routing pomiędzy i wewnątrz AS, idea (1)

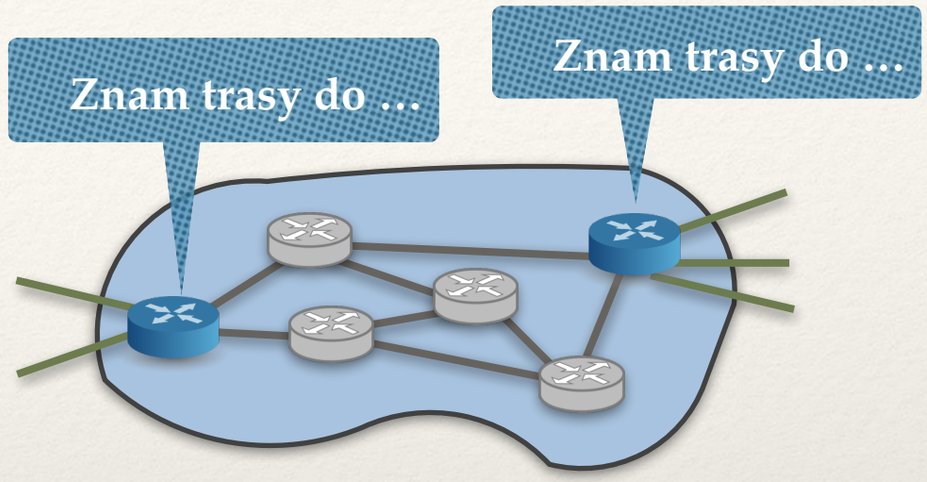
- **Routery brzegowe danego AS** (via BGP):
  - rozgłoś prefiksy CIDR tego AS;
  - dowiedz się o trasach do innych AS.



- **AS z jednym wyjściem X:**
  - Ustal routing wewnątrz AS (OSPF lub RIP lub IS-IS lub …)
  - Dodaj X na wszystkich routerach jako bramę domyślną.

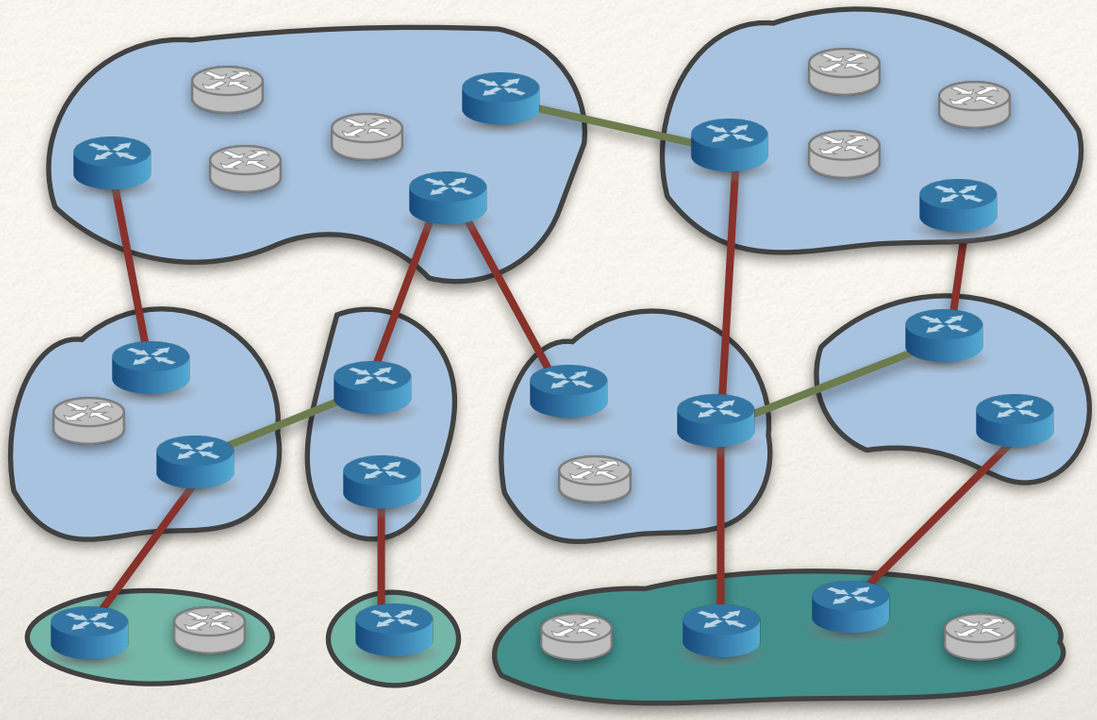
Routing pomiędzy i wewnątrz AS, idea (2)

- **Routery brzegowe danego AS** (via BGP):
  - rozgłoś prefiksy CIDR tego AS;
  - dowiedz się o trasach do innych AS.



- **AS z wieloma wyjściami $X_1, X_2, X_3, \ldots$**
  - Routery X_i biorą udział w protokole routingu wewnątrz AS udostępniając trasy, których nauczyły się przez BGP jako swoje „sąsiedztwo“ (z odpowiednimi odległościami).
  - Każdy router musi przechowywać informacje o wielu sieciach.

Czego brakuje na obrazku



- **IXP (Internet Exchange Point)**: punkt wymiany ruchu, łączy ze sobą wiele routerów brzegowych, często w relacji peering.
- **CDN (Content Delivery Networks)**: jak AS, ale celem jest dostarczanie treści jak najbliżej użytkowników końcowych (Akamai, Google, Netflix, …)

Lektura dodatkowa

- Kurose & Ross: rozdział 4.
- Tanenbaum: rozdział 5.

- Dokumentacja RIP i OSPF:
  - http://www.networksorcery.com/enp/protocol/rip.htm
  - http://www.networksorcery.com/enp/protocol/ospf.htm

- Różne ciekawostki:
  - Jak ekonomia ukształtowała BGP: http://web.mit.edu/6.829/www/currentsemester/papers/AS-bgp-notes.pdf
  - Jak Pakistan przejął YouTube: https://www.youtube.com/watch?v=IzLPKuAOe50
  - Wizualizacja AS: https://www.caida.org/research/topology/as_core_network/2017/

Zagadnienia

- Co to jest cykl w routingu? Co go powoduje?
- Czym różni się tablica routingu od tablicy przekazywania?
- Dlaczego w algorytmach routingu dynamicznego obliczamy najkrótsze ścieżki?
- Co to jest metryka? Jakie metryki mają sens?
- Czym różnią się algorytmy wektora odległości od algorytmów stanów łączy?
- Jak router może stwierdzić, że sąsiadujący z nim router jest nieosiągalny?
- Co to znaczy, że stan tablic routingu jest stabilny?
- Jak zalewać sieć informacją? Co to są komunikaty LSA?
- Co wchodzi w skład wektora odległości?
- W jaki sposób podczas działania algorytmu routingu dynamicznego może powstać cykl w routingu?
- Co to jest problem zliczania do nieskończoności? Kiedy występuje?
- Na czym polega technika zatruwania ścieżki zwrotnej (poison reverse)?
- Po co w algorytmach wektora odległości definiuje się największą odległość w sieci (16 w protokole RIPv1)?
- Po co stosuje się przyspieszone uaktualnienia?
- Co to jest system autonomiczny (AS)? Jakie znasz typy AS?
- Czym różnią się połączenia dostawca-klient pomiędzy systemami autonomicznymi od łącz partnerskich (peering)?
- Dlaczego w routingu pomiędzy systemami autonomicznymi nie stosuje się najkrótszych ścieżek?
- Które trasy w BGP warto rozgłaszać i komu? A które wybierać?
- Jak BGP współpracuje z algorytmami routingu wewnątrz AS?