SIECI KOMPUTEROWE

wykład dla kierunku informatyka

semestr 4 i 5

dr inż. Michał Sajkowski

Instytut Informatyki PP

pok. 227G PON PAN, Wieniawskiego 17/19

Michal.Sajkowski@cs.put.poznan.pl

tel. +48 (61) 8 582 100

http://www.man.poznan.pl/~michal/

sieci komputerowe
wykład 3 - komutacja pakietów
część pierwsza

literatura podstawowa

wykład prawie w całości przygotowany na podstawie tekstu i rysunków
z rozdziału 4 w książce:

L.L. Peterson, B.S. Davie
„Sieci komputerowe. Podejście systemowe”
Wydawnictwo Nakom, Poznań 2000

problemy

- *istnieje górna granica ilości komputerów połączonych w sieci łączy bezpośrednich*:

  łącze dwupunktowe: 2 komputery

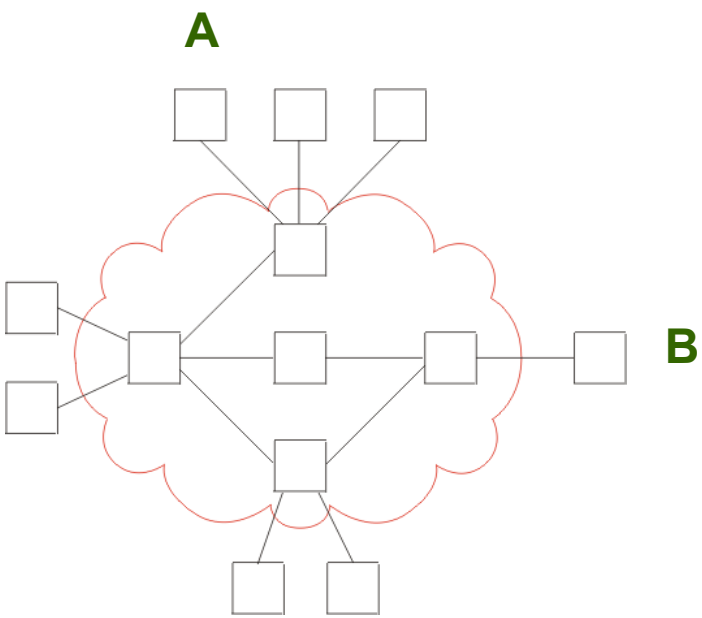
  Ethernet: 1024 komputery
- *istnieje górna granica wielkości obszaru geograficznego* obsługiwanego przez pojedynczą sieć:

  łącze dwupunktowe: nie w środku łącza

  Ethernet: zasięg 1500 m

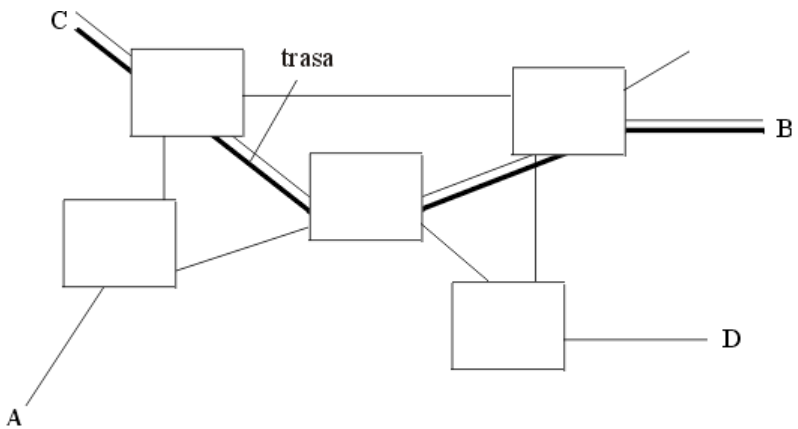
problemy

- *ponieważ celem jest budowanie sieci o zasięgu globalnym, kolejnym problemem staje się zapewnienie łączności między tymi komputerami, które nie są połączone bezpośrednio*



podobieństwo: klasyczna sieć telefoniczna

- *telefon nie jest bezpośrednio połączony z każdą osobą, do której chcemy zadzwonić*
- *jest dołączony do centrali zawierającej komutator*
- *sieci komputerowe stosują komutator pakietów, analogicznie do sieci telefonicznej, stosującej komutator kanałów*

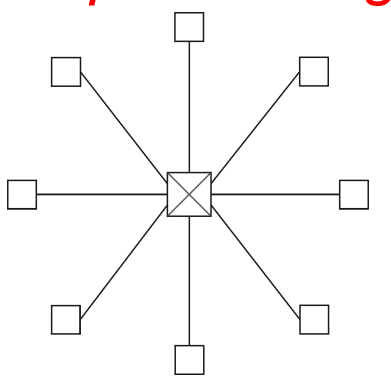


trzy problemy podejmowane przez komutator pakietów

- *pobieranie pakietów z wejść i kierowanie (forwarding) do właściwego wyjścia - znajomość, które wyjście jest właściwe wymaga od komutatora wiedzy o możliwych trasach do odbiorcy*
- *proces zbierania i dzielenia się wiedzą o trasach - wybór trasy (ang. routing)*
- *liczba pakietów przychodzących do komutatora przekracza przepustowość wyjścia, na które mają być kierowane - problem rywalizacji (ang. contention) pakietów. Komutator zmuszony do zbyt częstego odrzucania pakietów jest przeciążony (congested).*

kierowanie pakietów

- *komutator* : mechanizm, pozwalający łączyć łącza w celu tworzenia większych sieci
- dokonuje tego przez dodanie *topologii gwiazdy* do *topologii łącza dwupunktowego*:



oraz do topologii *szyny* (Ethernet) i *pierścienia* (FDDI)

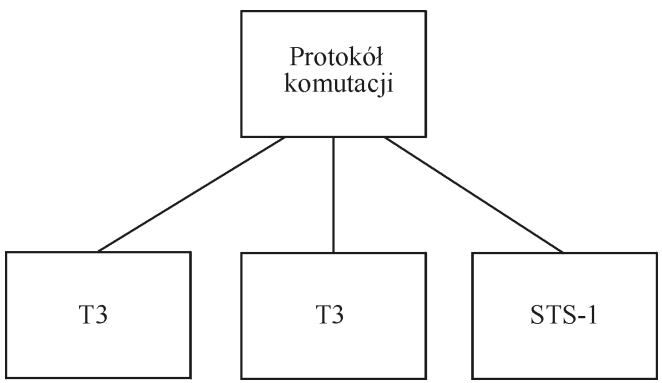
zalety topologii gwiazdy

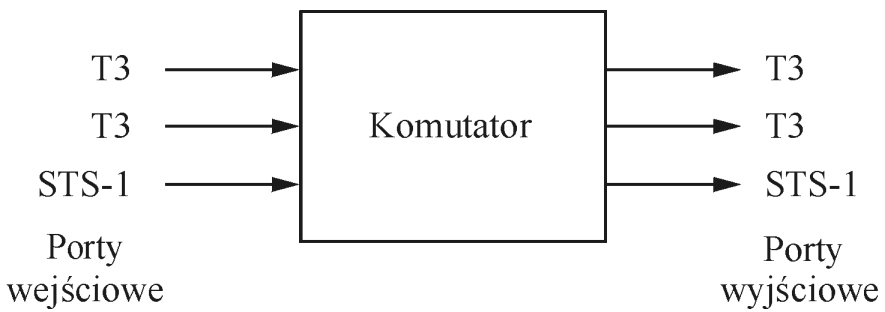
- budowa sieci *o dużym zasięgu geograficznym* (komputery dołączone do komutatora za pomocą łączy dwupunktowych)
- budowa *dużych sieci* (poprzez łączenie komutatorów)
- dodanie nowego komputera *nie musi oznaczać, że komputery dołączone wcześniej będą gorzej obsługiwane przez sieć*

kierowanie (komutacja)

- *podstawowa funkcja komutatora*:

  odbiór pakietów na jednym z łączy dołączonych do komutatora i nadawanie do jakiegoś innego łącza
- główna funkcja *warstwy sieci* (w architekturze OSI)
- *przykład*:graf protokołów uruchomiony w komutatorze sieci SONET (podział łącza na 2 *porty*: *we* i *wy*):





w jaki sposób komutator podejmuje decyzję do jakiego portu wyjściowego skierować pakiet?

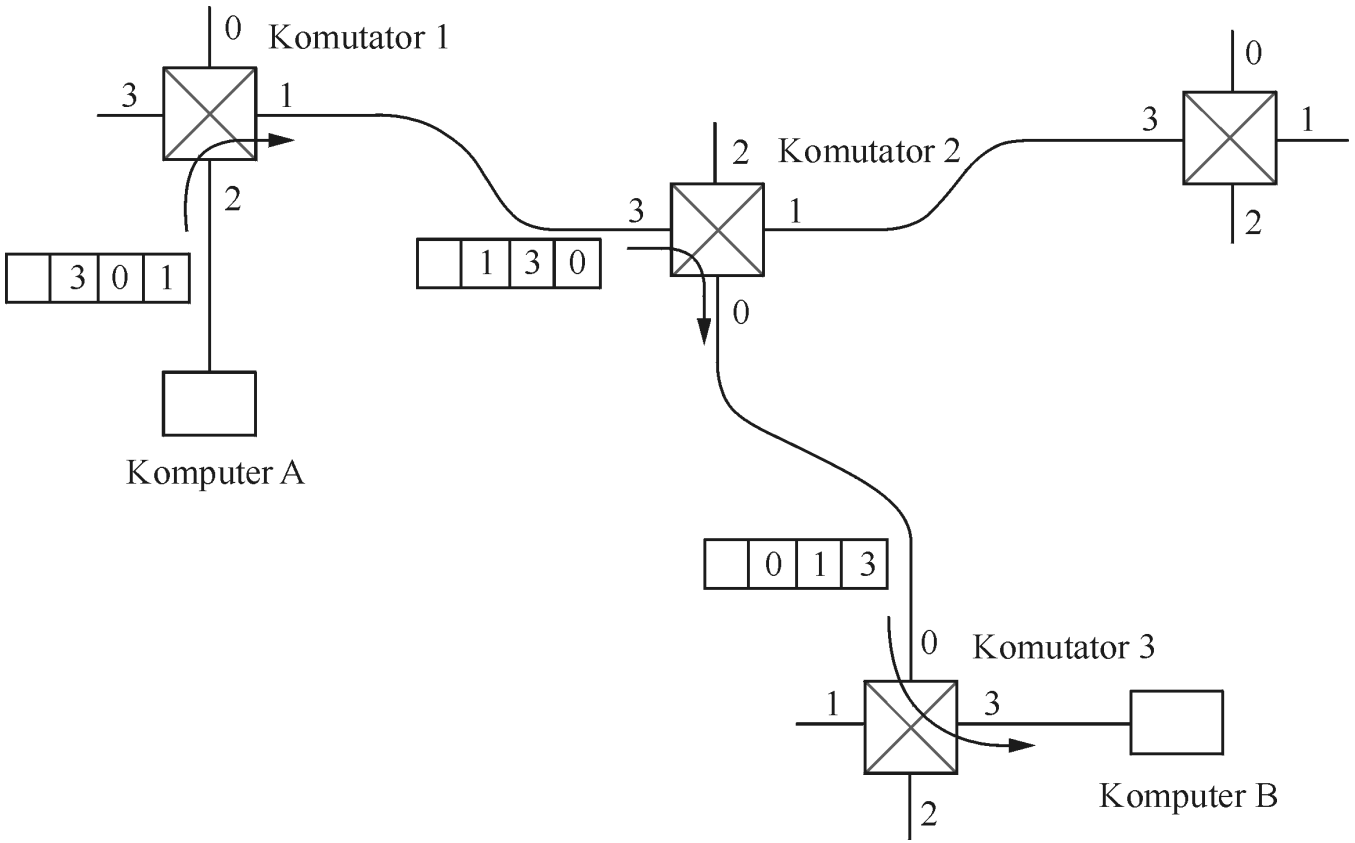
- sprawdza *identyfikator* w nagłówku pakietu
- są tu *trzy podejścia*:
- *datagramowe* (bezpołączeniowe)
- *kanał wirtualny* (połączeniowe)
- *wybór trasy przez nadawcę* (rzadko stosowane)

wybór trasy przez nadawcę

- *przydział numeru* każdemu wyjściu z komutatora i umieszczenie tego numeru w *nagłówku pakietu*
- *funkcja komutacji jest prosta*: *odczytanie* numeru portu w nagłówku pakietu na wejściu komutatora i *nadanie* pakietu do tego portu
- nagłówek pakietu zawiera *listę portów* w kolejnych komutatorach na ścieżce, którą pakiet *powinien przejść*
- lista ta jest poddana *rotacji*, tak aby pierwszy numer określał port w następnym komutatorze na ścieżce

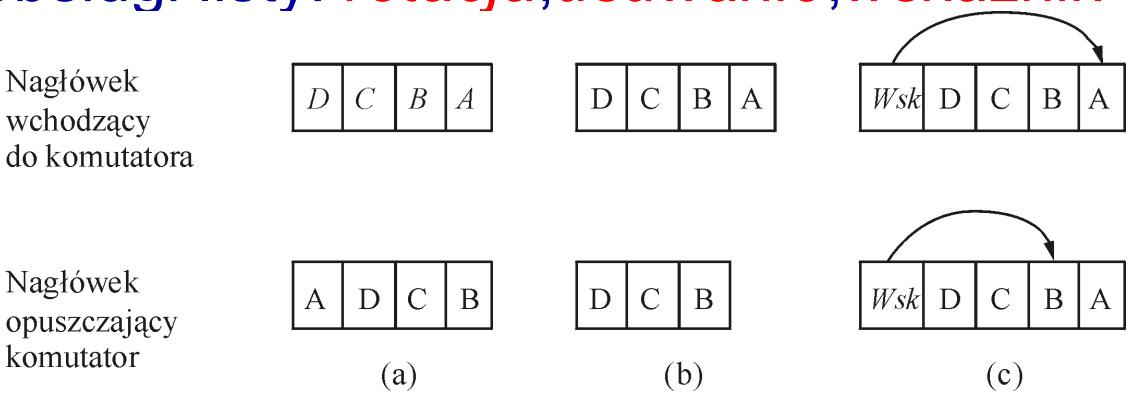
wybór trasy przez nadawcę w sieci komutowanej (komutator odczytuje pierwszy z prawej element listy)

- przykład:



wybór trasy przez nadawcę - analiza podejścia

- komputer *A zna topologię sieci na tyle*, aby utworzyć listę w nagłówku pakietu, zawierającą wszystkie poprawne kierunki dla każdego komutatora na ścieżce
- praktycznie, *nie można przewidzieć wielkości listy* w nagłówku pakietu - nagłówek ma zmienną długość
- trzy sposoby obsługi listy: *rotacja*,*usuwanie*,*wskaźnik*



- problem *skalowania* - brak danych dla dużej sieci

komutacja kanałów wirtualnych

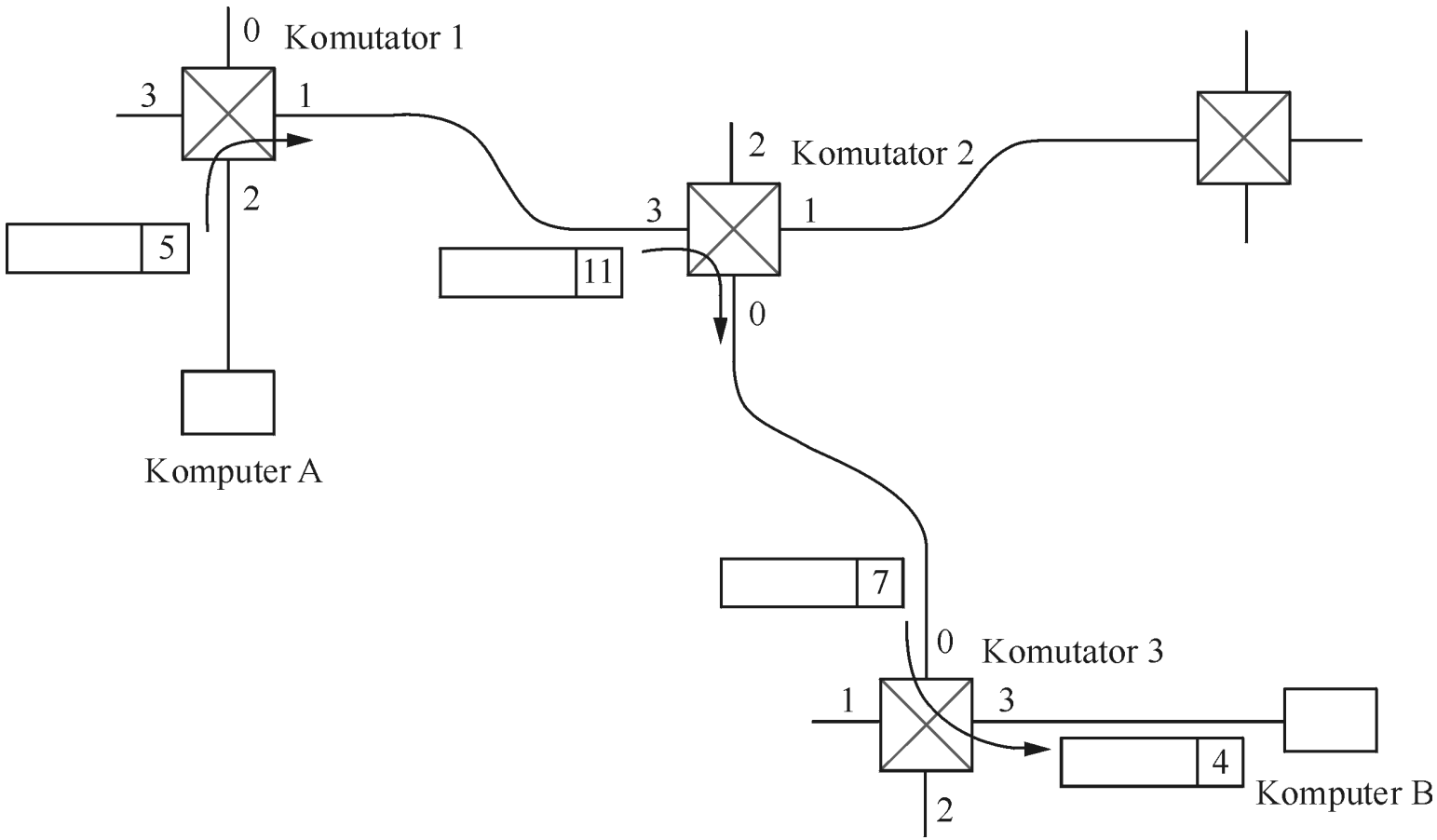
- *model połączeniowy*: wpierw ustanawia się *połączenie wirtualne* między komputerem nadawczym i odbiorczym
- komputer *A* nadaje komunikat z *żądaniem nawiązania połączenia* do komputera *B* - komunikat ten zawiera *adres* komputera *B* i unikalny *identyfikator kanału wirtualnego* (*VCI*)
- komutator wykorzystuje ten *VCI* do identyfikacji pakietów od komputera *A* do *B*
- *VCI* jest zmieniany na *inny* (unikalny) *w każdym komutatorze na ścieżce od A do B*

komutacja kanałów wirtualnych

- *model połączeniowy*: wpierw ustanawia się *połączenie wirtualne* między komputerem nadawczym i odbiorczym
- komputer *A* nadaje komunikat z *żądaniem nawiązania połączenia* do komputera *B* - komunikat ten zawiera *adres* komputera *B* i unikalny *identyfikator kanału wirtualnego* (*VCI*)
- komutator wykorzystuje ten *VCI* do identyfikacji pakietów od komputera *A* do *B*
- *VCI* jest zmieniany na *inny* (unikalny) *w każdym komutatorze na ścieżce od A do B*

komutacja kanałów wirtualnych

- przykład:



tablica kanałów wirtualnych dla komutatora 1

Port wejściowy	Identyfikator przychodzący	Port wyjściowy	Identyfikator wychodzący
2	1	2	4
2	4	0	3
2	5	1	11
2	6	0	4

- kiedy pakiet przychodzi do portu 2 z identyfikatorem 5, zastąp identyfikator 5 przez 11 i nadaj pakiet do portu 1

komutacja kanałów wirtualnych - analiza podejścia

- komputer *A* zanim nada pierwszy pakiet danych, musi czekać aż żądanie nawiązania połączenia osiągnie odległą stronę sieci i powróci - *opóźnienie jeden RTT*
- pełny adres komputera *B* w żądaniu nawiązania, w pakiecie danych jedynie identyfikator - *mały narzut*
- awaria łącza lub komutatora - stare połączenie *usuwane*,nowe *nawiązane* - *pielęgnacja tablic*
- na które łącze skierować samo żądanie nawiązania połączenia? (*o tym później*)

komutacja kanałów wirtualnych - analiza podejścia (c.d.)

- *sterowanie przepływem od węzła do węzła* (X25):

  przydział zasobów (buforów) do kanału wirtualnego

  protokół przesuwnego okna między każdą parą węzłów - w celu uniknięcia przepełnienia buforów w węźle odbiorczym

  odrzucanie kanału przez węzeł gdy nie ma gwarancji zapewnienia buforów
- *wyposażenie każdego kanału w odmienną jakość usługi* (QoS) - np. gwarancja określonej szerokości pasma, gwarancja maksymalnego opóźnienia

podejście datagramowe (bezpołączeniowe)

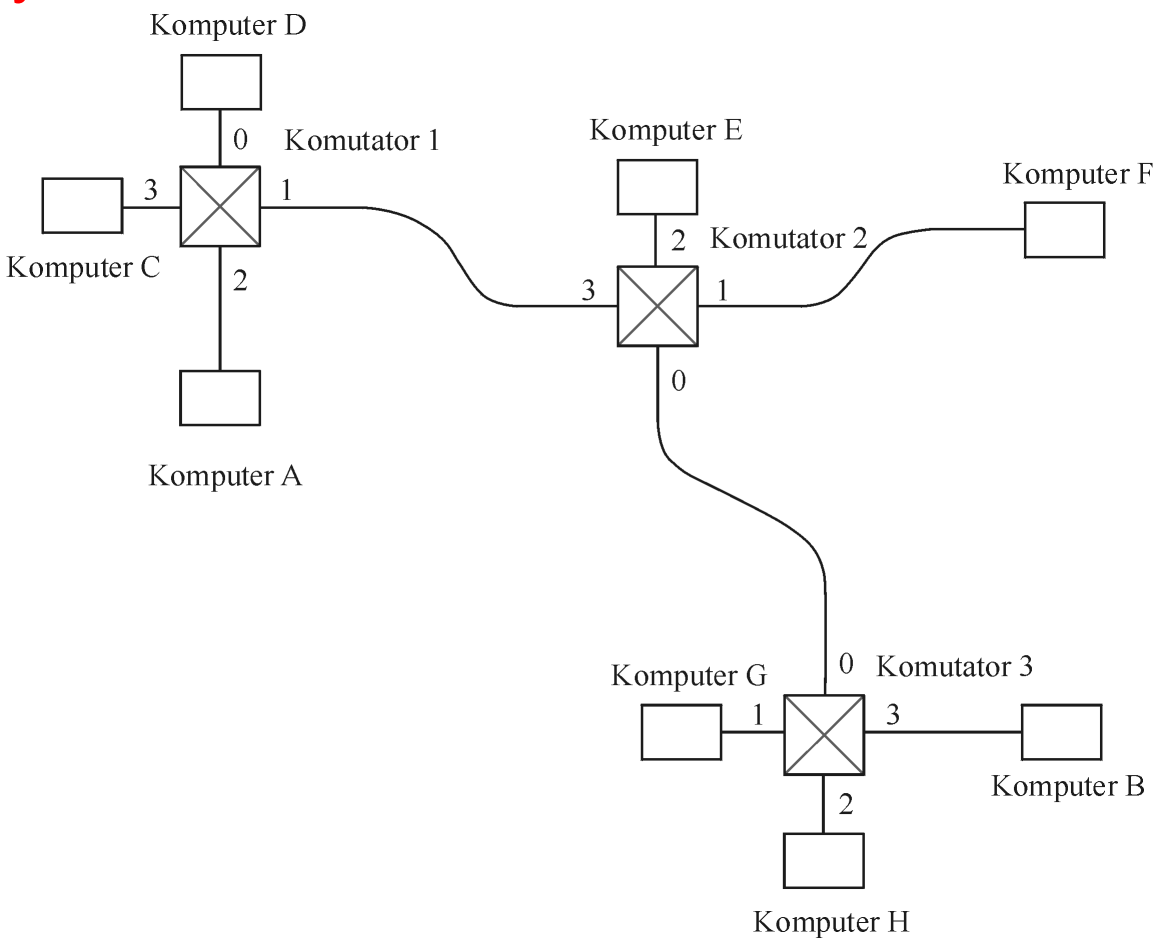
- *niewiarygodnie prosty pomysł*: pakiet musi mieć w nagłówku wystarczającą informację (kompletny adres odbiorcy) aby dostać się do odbiorcy

  - pakiet z takim nagłówkiem nazywa się *datagramem*
  - datagram jest jak *kret*, połączenie *niepotrzebne*

- aby zdecydować, gdzie kierować pakiet, komutator odwołuje się do *tablicy kierującej* (analogia do kierowania pakietu z żądaniem nawiązania połączenia w poprzednim podejściu)

datagramy

- przykład:

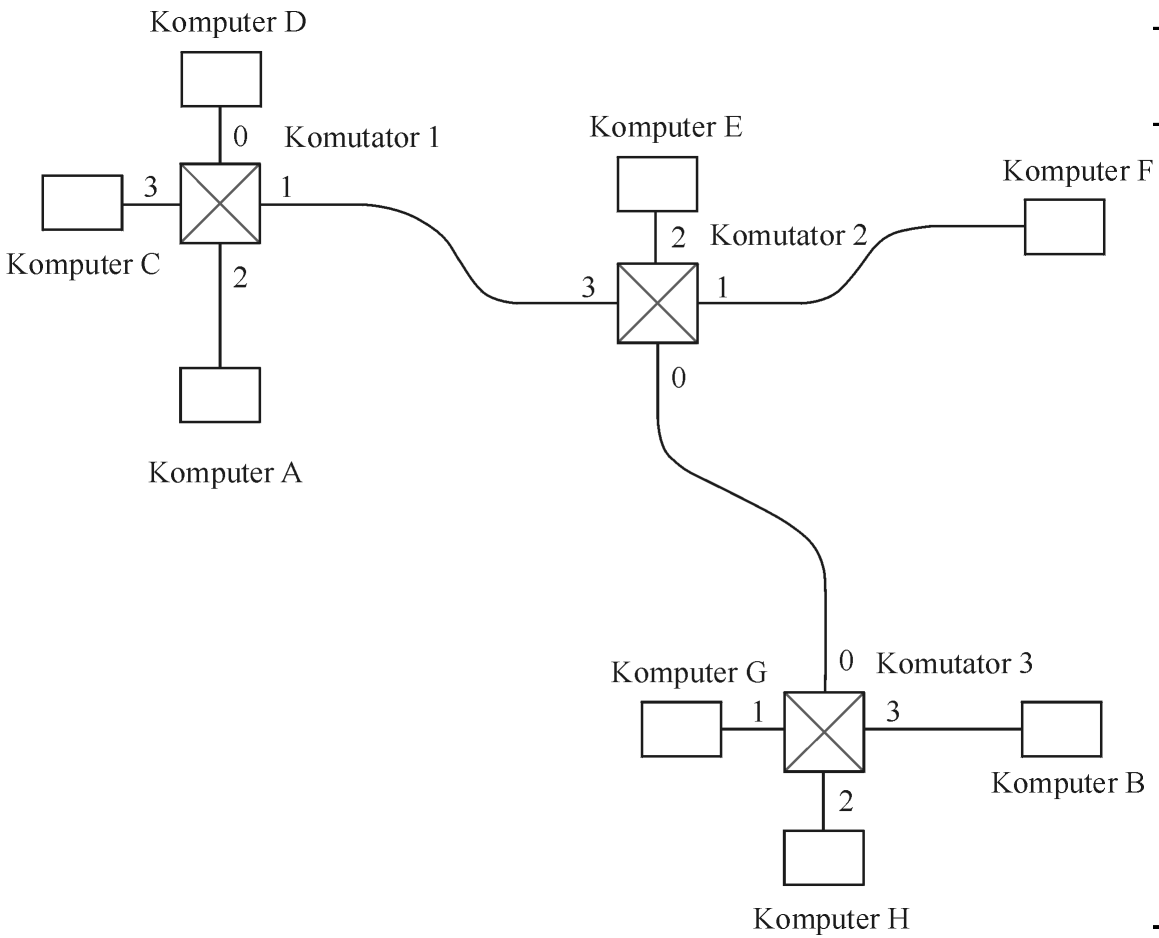


datagramy

- tablica kierująca dla komutatora 2

Odbiorca	Port komutatora
A	3
B	0
C	3
D	3
E	2
F	1
G	0
H	0

przykład - datagramy



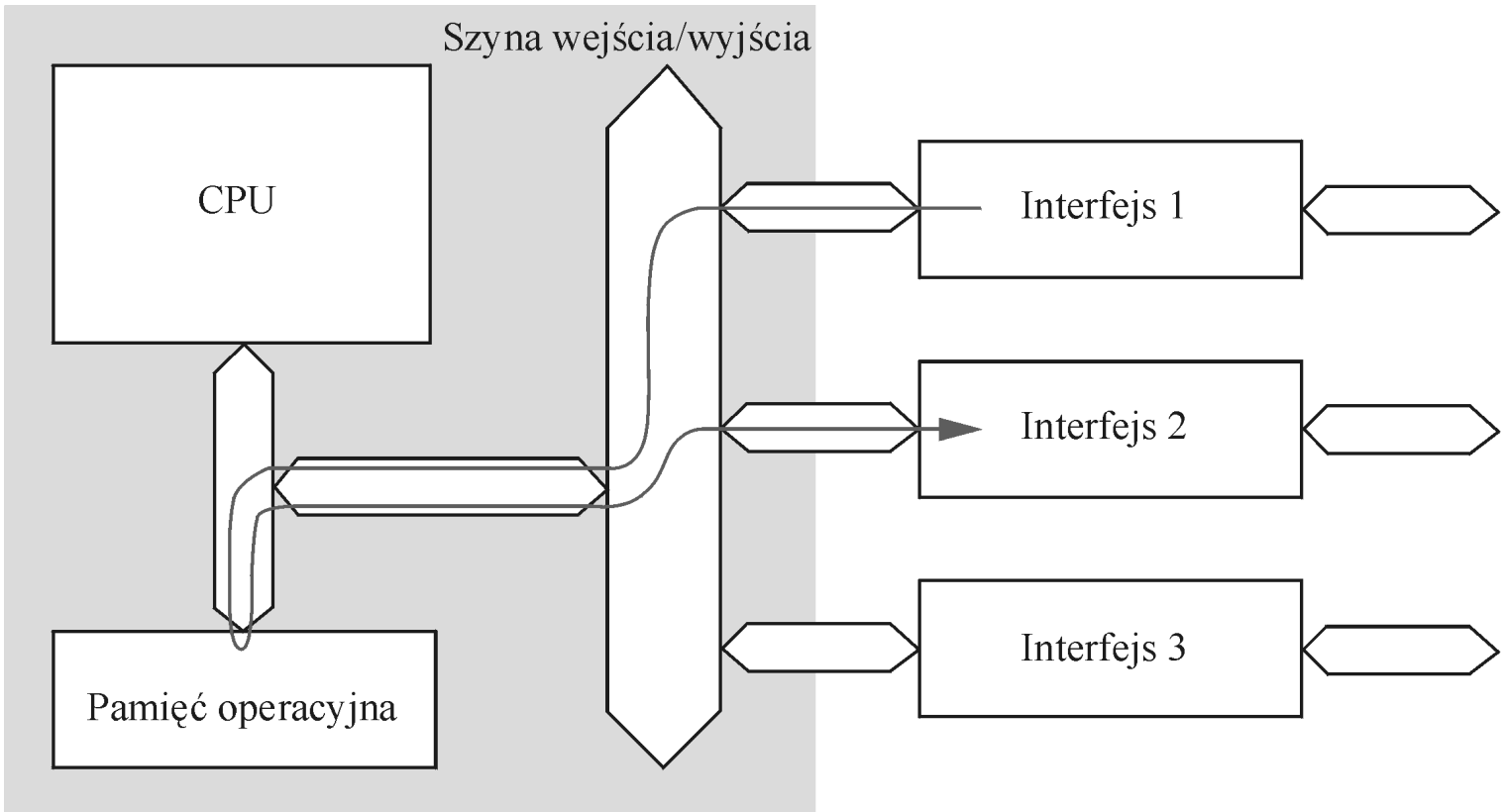
Odbiorca	Port komutatora
A	3
B	0
C	3
D	3
E	2
F	1
G	0
H	0

podejście datagramowe - analiza podejścia

- komputer nadaje dane, *gdy jest gotowy*
- komputer *nie wie*, czy sieć jest zdolna do dostarczenia pakietu
- każdy pakiet jest *nadawany niezależnie* - awaria komutatora wymaga jedynie aktualizacji tablic kierujących
- *narzut* na pakiet jest *większy* niż w modelu połączeniowym - (powód: pełny adres odbiorcy)

implementacja komutatora

- *stacja robocza* jako komutator



implementacja komutatora

- *stacja robocza z trzema interfejsami sieciowymi* jako komutator:
- *pakiet z interfejsu 1* (poprzez DMA) do pamięci i z pamięci (poprzez DMA) *na interfejs 2*
- gdy pakiet w pamięci, *CPU bada nagłówek pakietu*, aby określić interfejs do odbiorcy
- każdy pakiet przekracza szynę we/wy *dwukrotnie*
- *górna granica* na przepustowość urządzenia (suma szybkości danych utrzymywana na wejściach) to albo połowa szerokości pasma pamięci operacyjnej albo połowa szerokości pasma szyny we/wy

implementacja komutatora (c.d.)

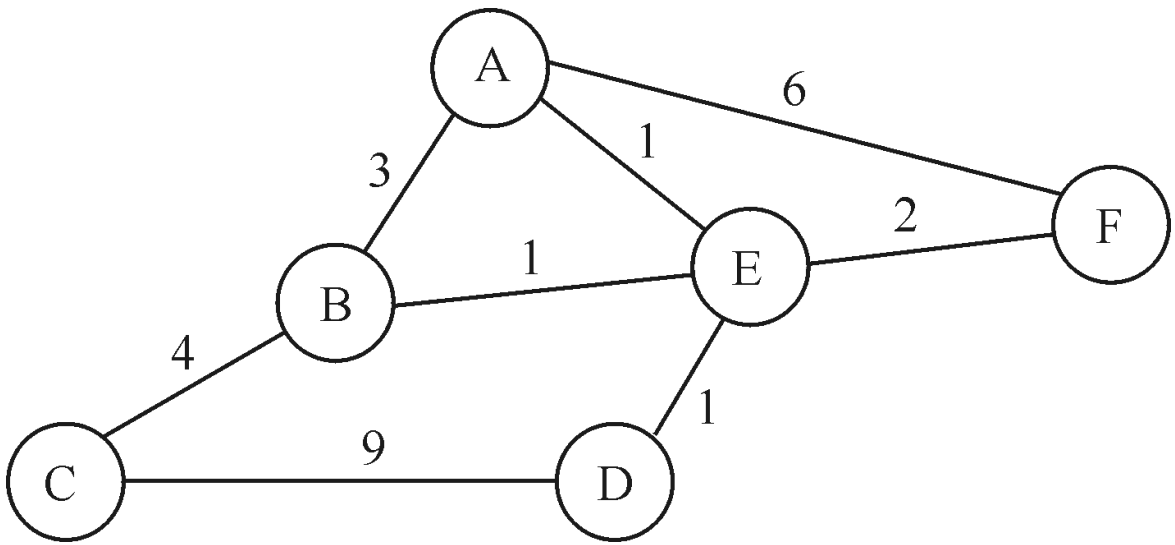
- *inne problemy:* obsługa krótkich pakietów (duży narzut), np. dla pakietów 64 bajtowych i 15 000 pakietów komutowanych na sek, przepustowość 7,68 Mb/s
- gdy wszystkie wejścia mają dane do jednego wyjścia, *rywalizacja jest nieunikniona*
- gdy dane do wielu wyjść, dobrze zaprojektowany komutator jest w stanie przekazywać pakiety *równolegle*

wprowadzenie do przeciążenia

- różnica między *przeciążeniem* (ang. *congestion*) a *rywalizacją* (ang. *contention*):
- *przeciążenie* oznacza, że komutator ma tak wiele pakietów w kolejce, że przekraczają przestrzeń buforów i są odrzucane
- *rywalizacja* występuje, gdy wiele pakietów musi czekać w kolejce w komutatorze, ponieważ współzawodniczą o to samo łącze wyjściowe
- *kanały wirtualne* - podejście konserwatywne - rezerwacja buforów (X25), niewykorzystanie komutatora
- *datagramy* - reakcja wtedy, gdy wystąpi przeciążenie

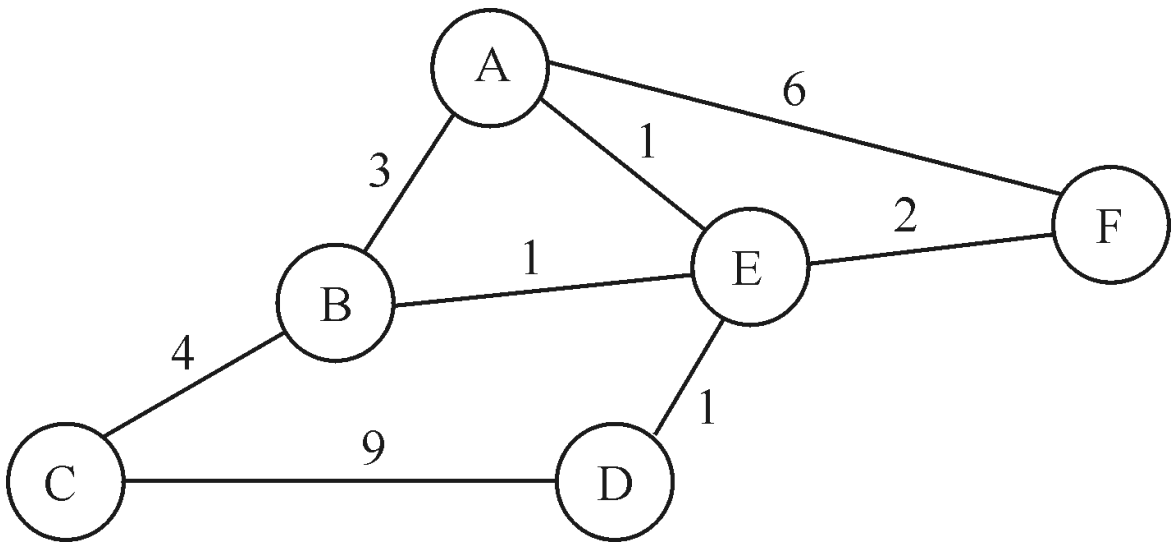
wybór trasy

- fundamentalny problem: *w jaki sposób komutatory pozyskują informację do własnych tablic kierujących?*
- wybór trasy jest problemem *teorii grafów*
- sieć reprezentowana przez *graf:*



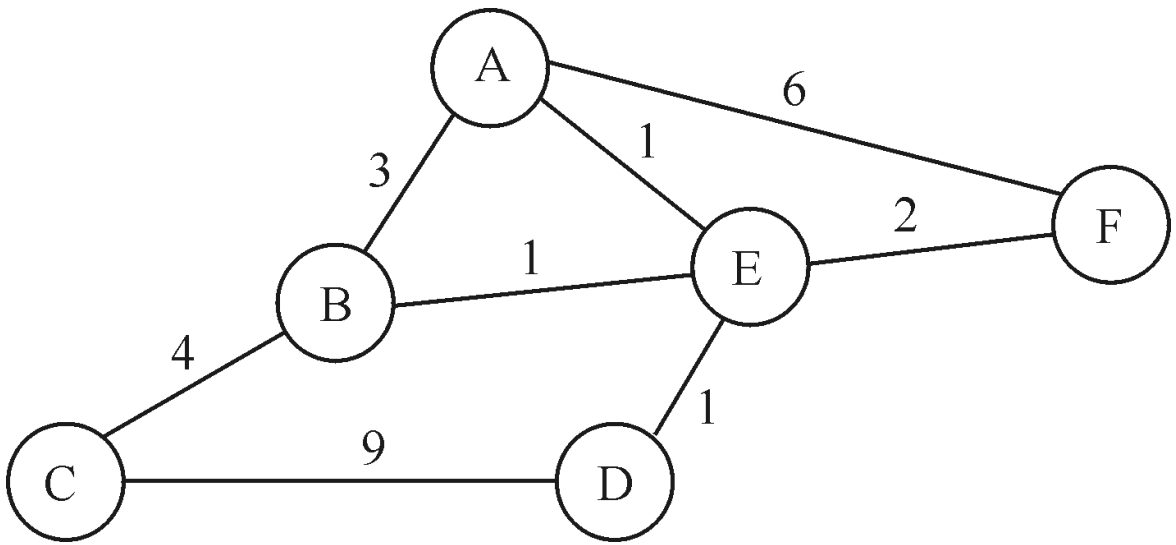
wybór trasy

- *węzeł*: komputer albo komutator
- *krawędź*: łącze
- *etykieta krawędzi*: koszt (czy warto wybrać to łącze?)
- założenie: każdy węzeł to komutator



podstawowy problem wybór trasy

- *wybór ścieżki między dwoma węzłami, charakteryzującej się najniższym kosztem*
- dla prostej sieci *podejście statyczne:* koszty wszystkich *najkrótszych* (*o najniższym koszcie*) ścieżek pamiętane w każdym węźle



wady podejścia statycznego

- nie radzi sobie z awariami węzłów i łączy (*brak odporności na błędy*)
- nie jest w stanie uporać się z dodawaniem nowych węzłów i łączy (problem *skalowalności*)
- zakłada, że koszty krawędzi są niezmienne, mimo że w praktyce jest inaczej (*jest nieelastyczne*)

konkluzja

- należy uruchomić *protokół wyboru trasy* między komutatorami, czyli zapewnić dynamiczny i rozproszony sposób znajdowania ścieżek o najniższych kosztach, przy istnieniu awarii łączy i węzłów, oraz zmieniających się kosztach krawędzi

algorytmy wyboru trasy

- na podstawie *wektora odległości*
- na podstawie *stanu łącza*

- założenie: znane są koszty krawędzi

wektor odległości

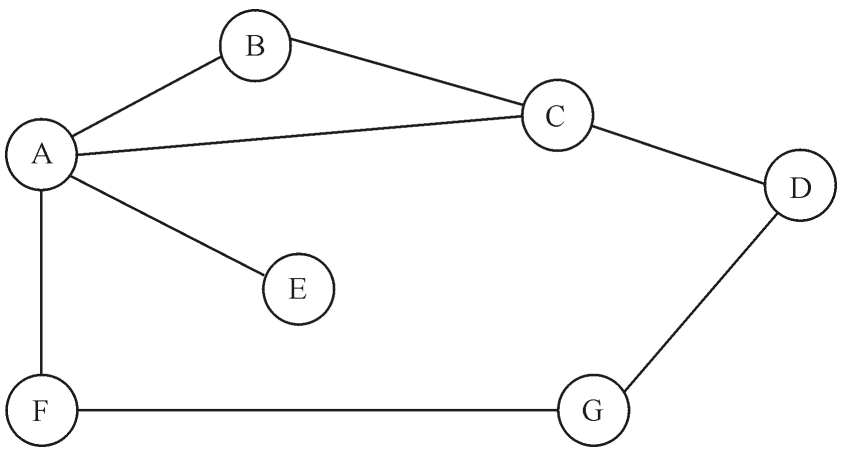
- *pomysł*: *odległość* jest tu miarą do zminimalizowania, *wektor* jest tu pierwszym kierunkiem na trasie
- *założenie początkowe*: każdy węzeł zna koszt łącza do każdego bezpośrednio przyłączonego sąsiada, łącze nie działające ma przydzielony koszt nieskończony (∞)
- spojrzenie *globalne* i spojrzenie *ze strony węzła*

wektor odległości - spojrzenie globalne

- w *przykładzie* koszt łącza ustawiony na jeden
- ścieżka o najniższym koszcie jest ścieżką o najniższej liczbie etapów
- każdy węzeł *zna* tylko informację *z jednego wiersza* stosowanej przy spojrzeniu globalnym tablicy początkowych odległości w każdym węźle
- początkowo każdy węzeł ustawia koszt na 1 dla węzłów bezpośrednio połączonych a ∞ dla pozostałych (*krok 1*)

wektor odległości - spojrzenie globalne

przykład sieci:



początkowe odległości zapamiętane w każdym węźle (spojrzenie globalne):

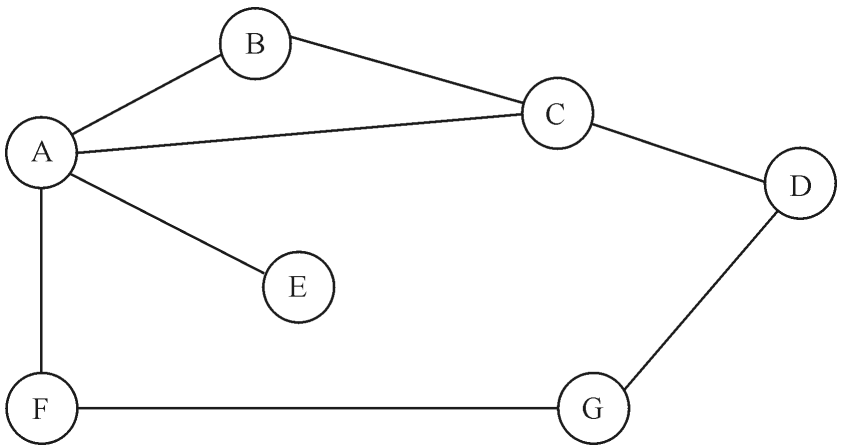
Informacja pamiętana w węźle	Odległość do osiągnięcia węzła						
	A	**B**	**C**	**D**	**E**	**F**	**G**
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

wektor odległości - spojrzenie globalne

- *krok 2* -każdy węzeł nadaje do swoich bezpośrednich sąsiadów swoją *prywatną listę odległości*, n.p. węzeł A nadaje ją do B,C,E i F
- *krok 3* - jeżeli sąsiad węzła A zorientuje się, że A poleca ścieżkę krótszą od tej, którą aktualnie zna, aktualizuje swoją własną listę, dodając nową długość ścieżki i *zaznacza*, że powinien nadawać pakiety do danego odbiorcy przez węzeł A (*tablica kierująca*) - stąd *wektor* w nazwie metody
- w ten sposób *po kilku aktualizacjach*, wszystkie węzły będą znały ścieżki o najmniejszym koszcie - uzyskano *końcowy zestaw odległości*

wektor odległości - spojrzenie globalne

przykład sieci:



końcowe odległości zapamiętane w każdym węźle (spojrzenie globalne):

Informacja pamiętana w węźle	Odległość do osiągnięcia węzła						
	A	**B**	**C**	**D**	**E**	**F**	**G**
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

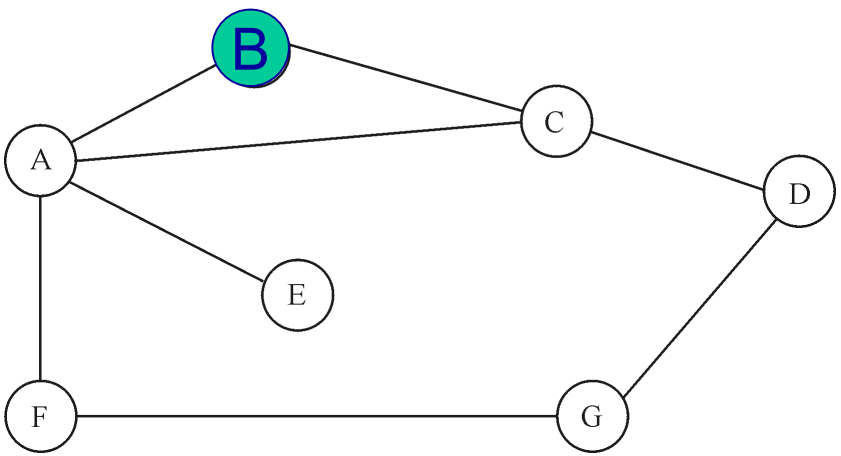
wektor odległości - spojrzenie ze strony węzła

- tablica kierująca w każdym węźle składa się ze zbioru trójek: (Odbiorca, Koszt, NastępnyEtap)
- każdy węzeł nadaje aktualizację:(Odbiorca, Koszt)

Odbiorca	Koszt	Następny węzeł
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	3	A

wektor odległości - spojrzenie ze strony węzła

przykład sieci:



tablica wyboru trasy utrzymywana w węźle B:

Odbiorca	Koszt	Następny węzeł
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	3	A

wektor odległości - spojrzenie ze strony węzła

- rodzaje aktualizacji nadawanej przez węzeł do sąsiadów:
  - *aktualizacja okresowa* (gdy się nic nie zmienia)
  - *aktualizacja wyzwalana* (gdy węzeł odbiera aktualizację od swojego sąsiada)
- działanie węzła po wykryciu awarii węzła albo łącza:
  - węzeł wysyła *nowe listy odległości* do swoich sąsiadów
  - system przechodzi do *nowego stanu*

wektor odległości - spojrzenie ze strony węzła

- *jak węzeł wykrywa awarię?*

  1. stale testując łącze do innego węzła, nadając pakiet kontrolny i czekając na potwierdzenie

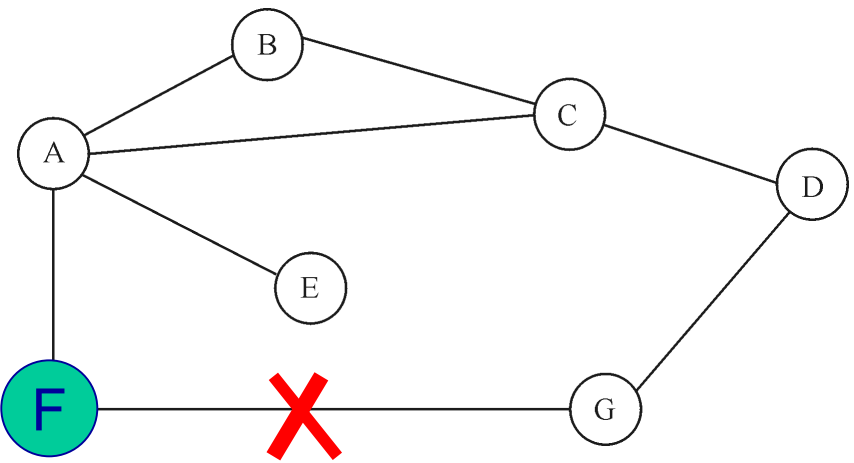
  2. węzeł nie odbiera oczekiwanej periodycznej aktualizacji

- *co się dzieje, kiedy węzeł wykrywa awarię łącza?*

  rozważmy przykład: węzeł F wykrywa, że łącze do węzła G uległo awarii

wektor odległości - spojrzenie ze strony węzła

węzeł F wykrywa awarię łącza do węzła G:



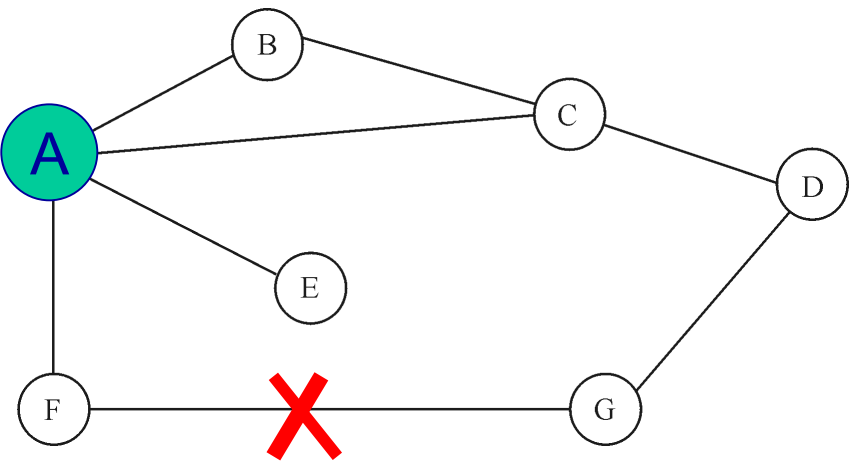
ustawia ∞, informuje A

tablica wyboru trasy utrzymywana w węźle F:

Odbiorca	Koszt	Następny węzeł
A	1	A
B	2	A
C	2	A
D	2	G
E	2	A
G	~~1~~ ∞	G

wektor odległości - spojrzenie ze strony węzła

węzeł A ustawia odległość do węzła G na ∞:



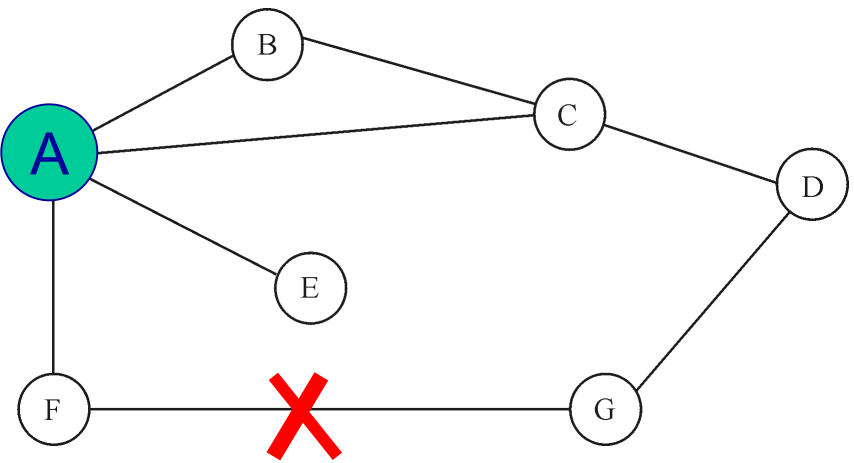
C informuje A o ścieżce równej 2 do G

tablica wyboru trasy utrzymywana w węźle A:

Odbiorca	Koszt	Następny węzeł
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	~~2~~ ∞	F

wektor odległości - spojrzenie ze strony węzła

węzeł A ustawia odległość do węzła G na 3, informuje F

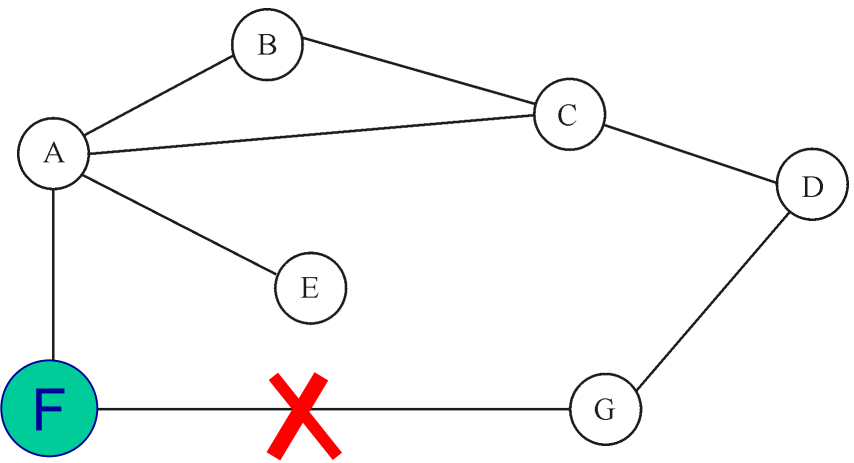


tablica wyboru trasy utrzymywana w węźle A:

Odbiorca	Koszt	Następny węzeł
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	~~∞~~ 3	~~F~~ C

wektor odległości - spojrzenie ze strony węzła

węzeł F ustawia odległość do węzła G na 4:



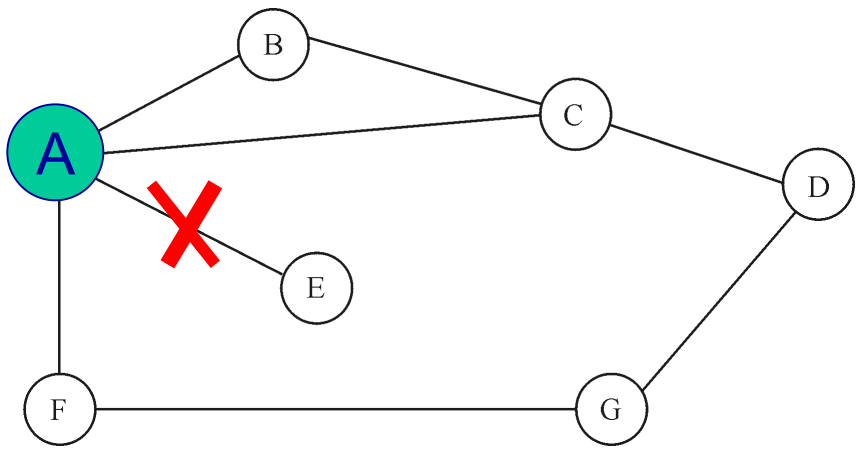
sieć się stabilizuje

tablica wyboru trasy utrzymywana w węźle F:

Odbiorca	Koszt	Następny węzeł
A	1	A
B	2	A
C	2	A
D	2	G
E	2	A
G	~~1~~ 4	~~G~~ A

wektor odległości - spojrzenie ze strony węzła

przykład 2: łącze od A do E ulega awarii



dwie różne aktualizacje:

A: odległość do E to ∞

B, C: odległość do E to 2.

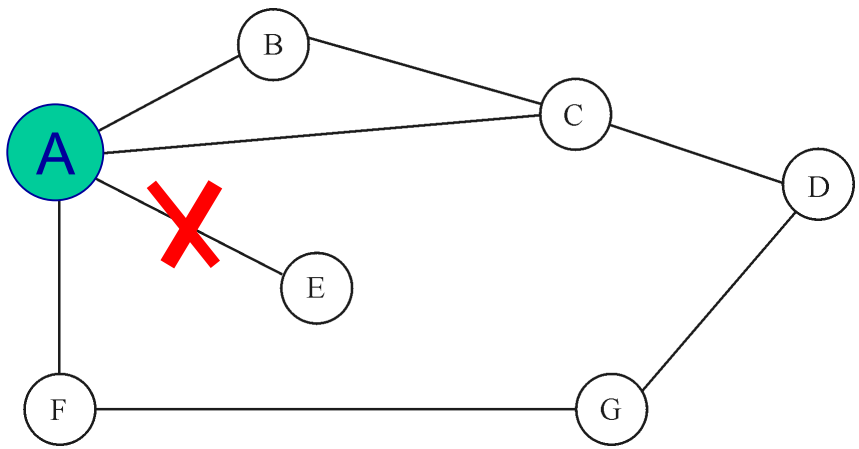
W zależności od relacji czasowych B wnioskuje:

jak od C do E w 2 etapach, to do E w 3 etapach i podaje to do A, A wnioskuje: do E w 4 etapach, podaje do C,

C wnioskuje: do E w 5 etapach, itd. żaden węzeł nie wie, że E jest nieosiągalny, tablice się nie stabilizują

wektor odległości - spojrzenie ze strony węzła

rozwiązanie problemu: przecięcie pętli w aktualizacji trasy:



1. podzielony horyzont:

gdy B nadaje aktualizację wyboru trasy do A to nie włącza do niej trasy (E, 2) gdyż o niej dowiedział się od A, bo ma (E,2,A) w swojej tablicy

2. podzielony horyzont z negatywną informacją zwrotną:

B nadaje aktualizację wyboru trasy do A w postaci informacji negatywnej (E, ∞) (zniechęcającej A) aby A nie korzystał z B do osiągnięcia E

wybór trasy na podstawie stanu łącza

- *założenia początkowe*:

  każdy węzeł jest zdolny do poznania stanu łączy do swoich sąsiadów i ustalenia kosztu każdego łącza
- *pomysł*: każdy węzeł wie jak osiągnąć swoich bezpośrednich sąsiadów
- ta informacja *jest rozpowszechniona* w każdym węźle
- każdy węzeł może zbudować *kompletną mapę sieci*
- protokół polega na dwóch mechanizmach:

- *niezawodne rozpowszechnianie informacji o łączu*
- *obliczenie tras z sumy wiedzy o stanach łączy*

niezawodny rozpływ

- proces upewnienia się, że każdy węzeł biorący udział w protokole wyboru trasy otrzymuje kopię informacji o stanie łącza od wszystkich pozostałych węzłów
- każdy węzeł tworzy pakiet aktualizujący, zwany *pakietem stanu łącza* (LSP), który zawiera:

  identyfikator węzła, lista sąsiadów+koszty łączy

  (*aby obliczyć trasę*) oraz

  numer sekwencyjny, czas życia pakietu TTL

  (*aby proces rozpływu był niezawodny* - zbadanie, czy dysponuje się *najnowszą* kopią informacji (LSP))

działanie niezawodnego rozpływu

- węzeł X odbiera kopię LSP pochodzącą od węzła Y
- jeżeli ma już kopię LSP od Y, porównuje numery sekwencyjne, gdy numer w nowej kopii wyższy od numeru w starej kopii to zapamiętuje nową kopię LSP
- X kieruje nową kopię LSP pochodzącą od Y do wszystkich sąsiadów X, za wyjątkiem sąsiada, który nadał tę kopię do X
- w ten sposób najnowsza kopia LSP osiąga wszystkie węzły

obliczenie trasy

- węzeł posiadający kopię LSP z każdego innego węzła jest w stanie *obliczyć kompletną mapę sieci* i *zdecydować o najlepszej trasie* do każdego odbiorcy
- trasę oblicza korzystając z *algorytmu Dijkstry obliczania najkrótszej ścieżki w grafie*
- *algorytm Dijkstry* - założenia i oznaczenia

węzeł $s \in N$ konstruuje graficzną reprezentację sieci na podstawie odebranych LSP (znajduje najkrótszą ścieżkę do wszystkich węzłów w zbiorze N), N zbiór węzłów grafu,

$l(i,j)$ nieujemny koszt krawędzi, $l(i,j) = \infty$ brak ścieżki od i do j

M zbiór węzłów wprowadzonych dotychczas do algorytmu

$C(n)$ koszt ścieżki od węzła s do węzła n

algorytm Dijkstry

$M = \{s\}$

for każdy n w $N - \{s\}$

$C(n) = l(s, n)$ /* inicjacja tablicy kosztów */

while $(N \neq M)$

$M = M \cup \{w\}$ taki, że $C(w)$ jest minimum dla wszystkich w w $(N-M)$

/* włącz węzeł osiągalny najmniejszym kosztem do M */

for każdy n w $(N-M)$

$C(n) = \mathbf{MIN}(C(n), C(w) + l(w, n))$

/* aktualizacja tablicy kosztów uwzględniając węzeł w */

algorytm Dijkstry

- w ostatnim wierszu algorytmu:

$$C(n) = \mathbf{MIN}(C(n), C(w) + l(w, n))$$

wybieramy nową trasę do węzła *n*, która przechodzi przez węzeł *w*, jeżeli całkowity koszt przejścia od źródła do węzła *w* i następnie od *w* do *n* jest mniejszy niż stara trasa, którą mieliśmy do *n*

algorytm Dijkstry w praktyce

- *w praktyce*, każdy komutator oblicza swoją tablicę wyboru trasy bezpośrednio z tych LSP, które uzbierał stosując realizację algorytmu Dijkstry, zwaną *algorytmem przeszukiwania w przód*
- *konkretnie*, każdy komutator utrzymuje dwie listy, znane jako `Wstępna` i `Potwierdzona`. Każda z list zawiera zbiór elementów w postaci (`Odbiorca`, `Koszt`, `NastępnyEtap`)
- *algorytm* działa następująco:

działanie algorytmu Dijkstry

1. Inicjuj listę `Potwierdzona` z elementem dla samego siebie; element ten ma koszt 0.
2. Węzeł właśnie dodany do listy `Potwierdzonej` w poprzednim kroku, nazwij `Następnym`, wybierz jego LSP.
3. Dla każdego `Sąsiada` węzła `Następnego`, oblicz `Koszt`, za cenę którego można dotrzeć do tego `Sąsiada`. Jest to suma kosztu ode mnie do węzła `Następnego` i od węzła `Następnego` do `Sąsiada`.

działanie algorytmu Dijkstry

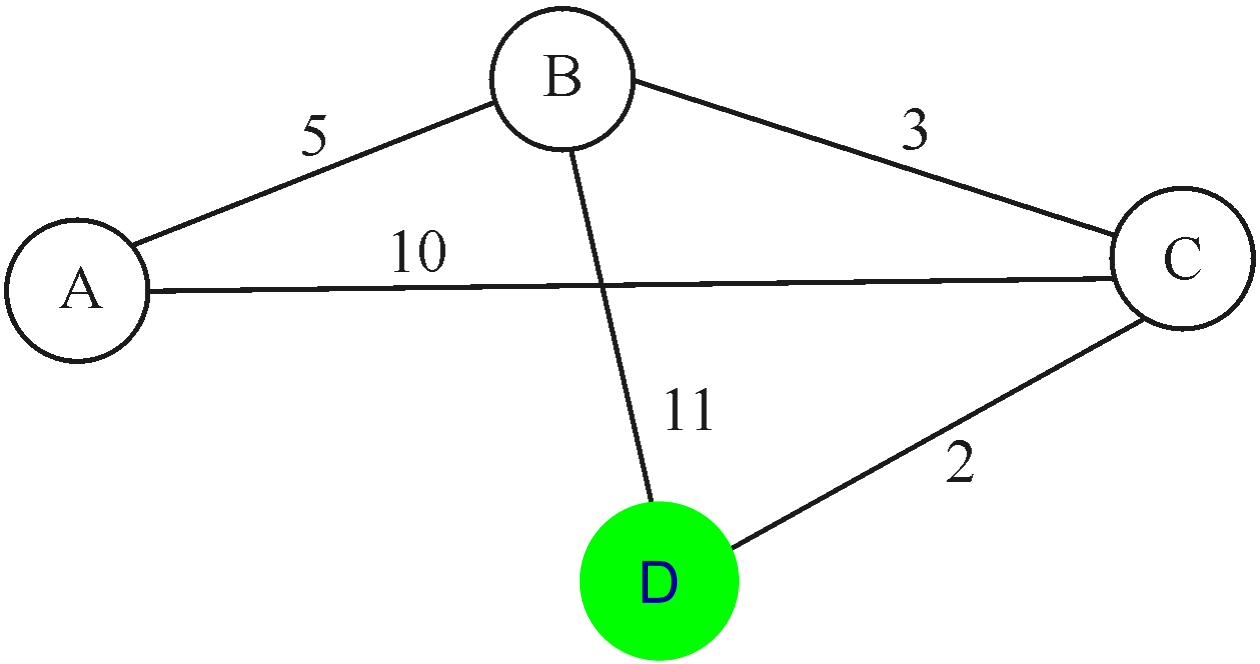
- (a) Jeżeli `Sąsiad` nie jest aktualnie ani na liście `Wstępnej` ani na liście `Potwierdzonej`, wtedy dodaj (`Sąsiad`, `Koszt`, `NastępnyEtap`) do listy `Wstępnej`, gdzie `NastępnyEtap` jest kierunkiem, w którym muszę się poruszać, aby dotrzeć do węzła `Następnego`.
- (b) Jeżeli `Sąsiad` jest aktualnie na liście `Wstępnej`, i `Koszt` jest mniejszy od aktualnie umieszczonego kosztu na liście dla `Sąsiada`, wtedy zastąp aktualny element przez (`Sąsiad`, `Koszt`, `NastępnyEtap`), gdzie `NastępnyEtap` jest kierunkiem, w którym powinienem się poruszać, aby dotrzeć do węzła `Następnego`.

działanie algorytmu Dijkstry

4. Jeżeli lista `Wstępna` jest pusta, zatrzymaj się.
   W przeciwnym przypadku, pobierz element z listy `Wstępnej` o najmniejszym koszcie, przesuń go na listę `Potwierdzoną` i wróć do kroku 2.

wybór trasy na podstawie stanu łącza

- *przykład sieci*

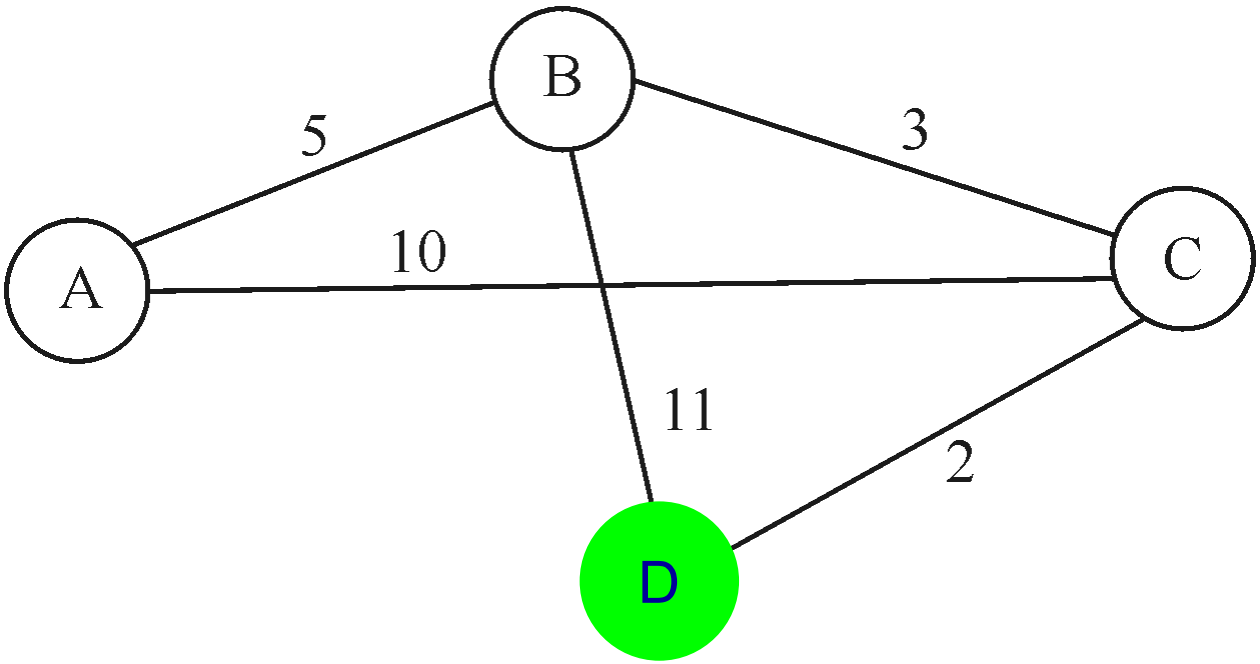


baza danych dla węzła D

	Lista pot-wierdzona	Lista wstępna	Komentarz
1	(D, 0, -)		Ponieważ węzeł D jest jedynym nowym elementem listy `Potwierdzonej`, zbadaj jego LSP
2	(D, 0, -)	(B, 11, B) (C, 2, C)	LSP węzła D podaje, że może osiągnąć węzeł B przez węzeł B kosztem 11 jednostek, co jest lepsze od czegokolwiek na innych listach, dlatego umieść go na liście `Wstępnej`. To samo dla węzła C.

wybór trasy na podstawie stanu łącza

- *przykład sieci*

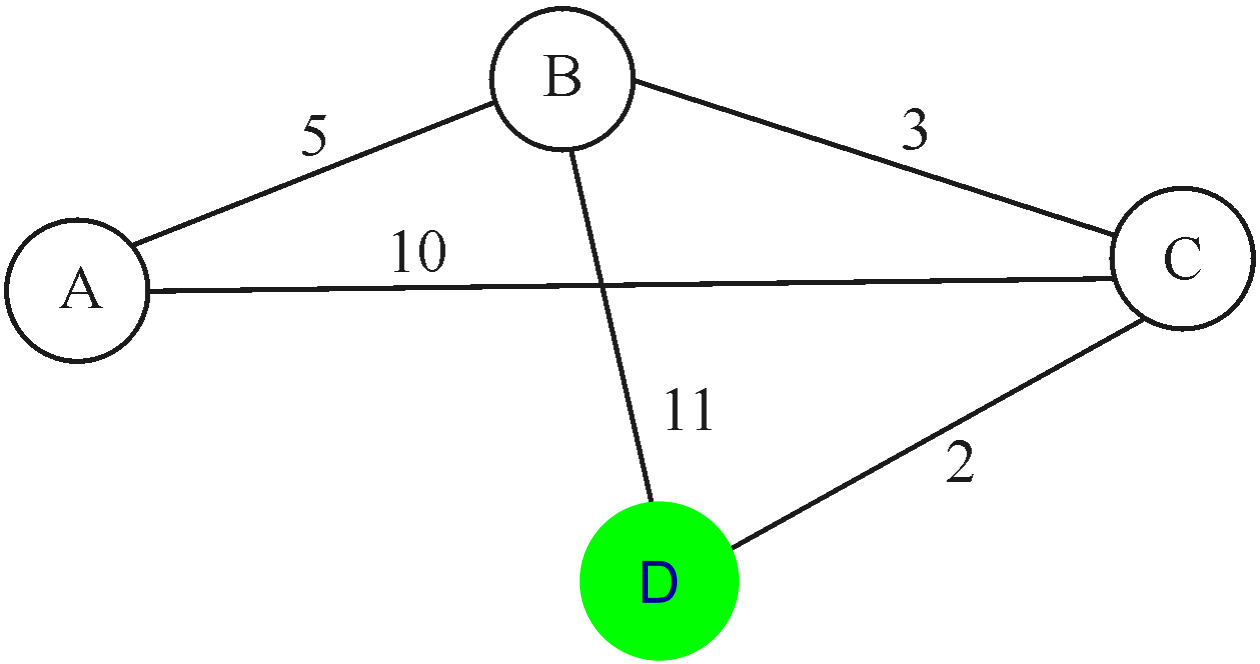


baza danych dla węzła D

	Lista pot-wierdzona	Lista wstępna	Komentarz
3	(D, 0, -) (C, 2, C)	(B, 11, B)	Umieść element o najniższym koszcie z listy `Wstępnej` (czyli węzeł C) na liście `Potwierdzonej`. Następnie, badaj LSP tego nowo potwierdzonego elementu (węzeł C)
4	(D, 0, -) (C, 2, C)	(B, 5, C) (A, 12, C)	Koszt osiągnięcia węzła B przy przejściu przez węzeł C jest równy 5, dlatego zastąp nim (B, 11, B). LSP węzła C podaje, że możemy osiągnąć węzeł A za cenę 12 jednostek

wybór trasy na podstawie stanu łącza

- *przykład sieci*

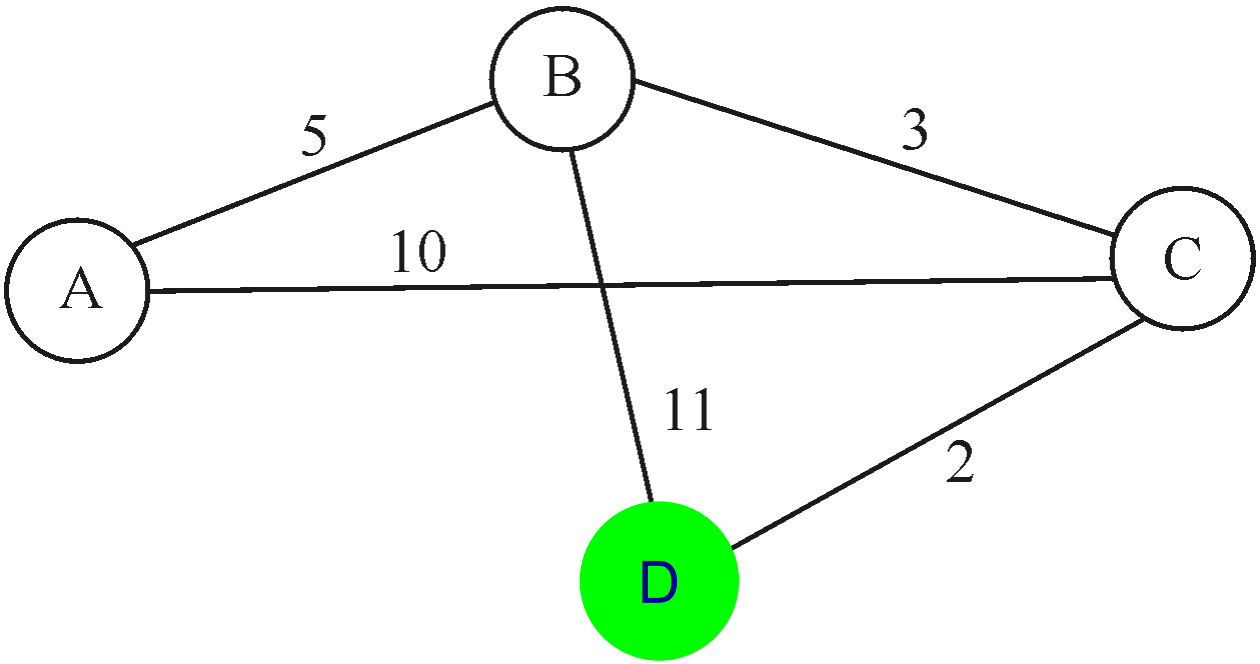


baza danych dla węzła D

	Lista pot-wierdzona	Lista wstępna	Komentarz
5	(D, 0, -) (C, 2, C) (B, 5, C)	(A, 12, C)	Przesuń element o najniższym koszcie z listy `Wstępnej` (czyli węzeł B) do listy `Potwierdzonej`. Sprawdź jego LSP.
6	(D, 0, -) (C, 2, C) (B, 5, C)	(A, 10, C)	Ponieważ można osiągnąć węzeł A kosztem 5 jednostek z węzła B, zastąp element na liście `Wstępnej`.
7	(D, 0, -) (C, 2, C) (B, 5, C) (A, 10, C)		Przesuń element o najniższym koszcie z listy `Wstępnej` (czyli węzeł A) na listę `Potwierdzoną`. Wszystko jest zrobione

wybór trasy na podstawie stanu łącza

- *przykład sieci*



analiza algorytmu wyboru trasy na podstawie stanu łącza

- *Zalety*
  - szybko powraca do stanu stabilnego
  - nie generuje dużo ruchu
  - szybko reaguje na zmiany topologii i awarię węzłów
- wady:
  - duża ilość informacji pamiętana w każdym węźle (problemy ze skalowalnością)

porównanie algorytmów wyboru trasy

- *wektor odległości*
  - każdy węzeł porozumiewa się jedynie ze swoimi bezpośrednio przyłączonymi sąsiadami
  - mówi im wszystko, czego się dowiedział (podaje im całą tablicę kierującą)
- *stan łącza*
  - każdy węzeł porozumiewa się ze wszystkimi innymi węzłami
  - mówi im tylko to, co wie na pewno (podaje im tylko stan bezpośrednio przyłączonych łączy)