Рачунарске мреже

Александар Картељ

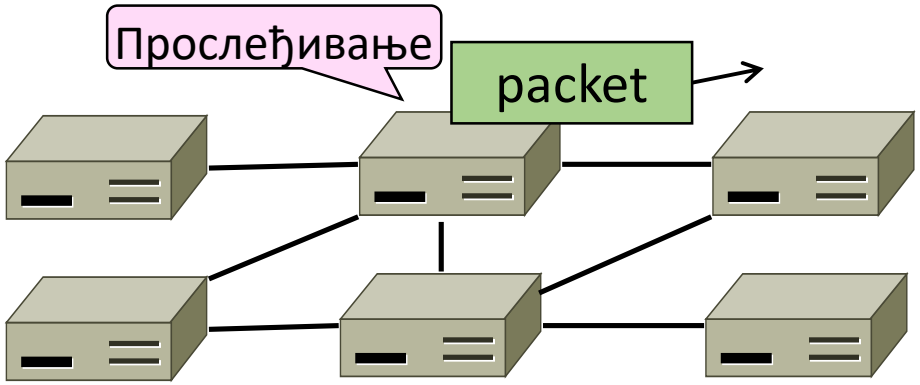
kartelj@matf.bg.ac.rs

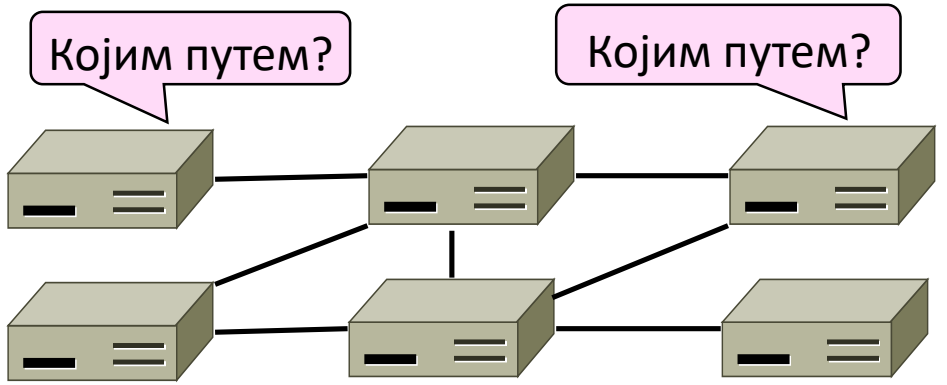
Рутирање

Преглед

Рутирање и прослеђивање

- Прослеђивање је процес слања пакета суседним чворовима
- Рутирање је процес одређивања путања којима ће се прослеђивање вршити





Из перспективе алокације протока

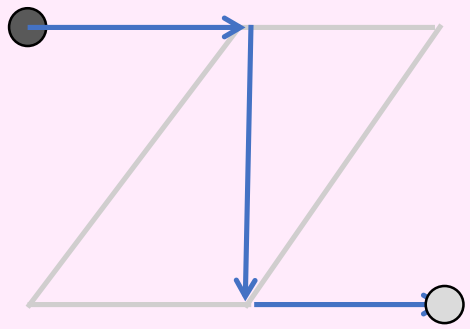
- Алокација протока је кључни аспект рутирања
- Рутирање алоцира проток тако да има на уму отказе чворова (постоје и други механизми алокације)

Механизам	**Време реакције/ Адаптација ка**
Рутирање осетљиво на оптерећење	Секунде / Критични чворови оптерећења
Рутирање	Минути / Откази чворова
Обликовање протока	Сати / Оптерећење мреже
Резервација протока	Месеци / Корисници мреже

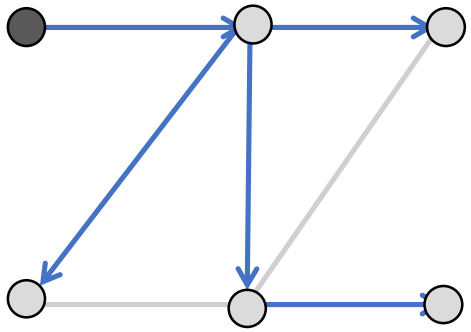
Модели испоруке

- Различити алгоритми рутирања за различите моделе

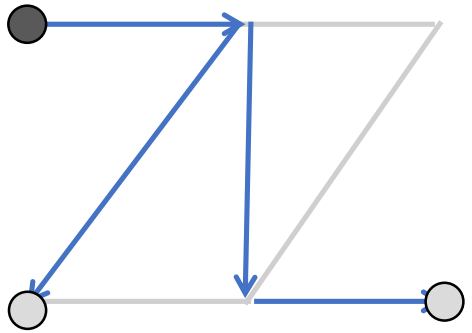
Unicast



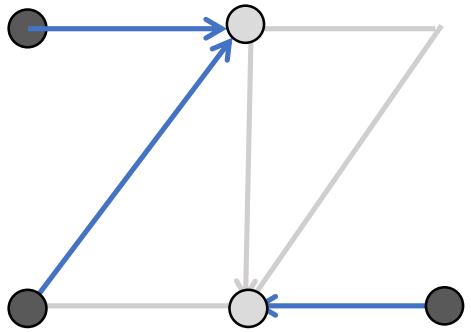
Broadcast



Multicast



Anycast



Циљеви рутирања

Својство	Значење
Тачност	Проналази путању која ради
Ефикасност	Рационално троши проток
Равноправност	Подједнака права чворова, не изгладњује неке чворове
Брза конвергенција	Брз опоравак након промена (отказа, нових чворова)
Скалабилност	Ради добро и када мрежа (број чворова, веза, ...) расте

Принципи за дизајн алгоритама рутирања

- Децентрализовани и дистрибуирани
  1. Чворови су рутери, не разматрамо корисничке рачунаре
     - Када податак дође до последњег рутера, прослеђивање ка корисничком рачунару разматра слој везе
  2. Сви чворови су равноправни, нема битнијих чворова
  3. Чворови сазнају укупно стање мреже тако што размењују поруке са суседима
  4. Чворови раде конкурентно
  5. Могу се десити откази чворова и веза или губљења порука

Теме

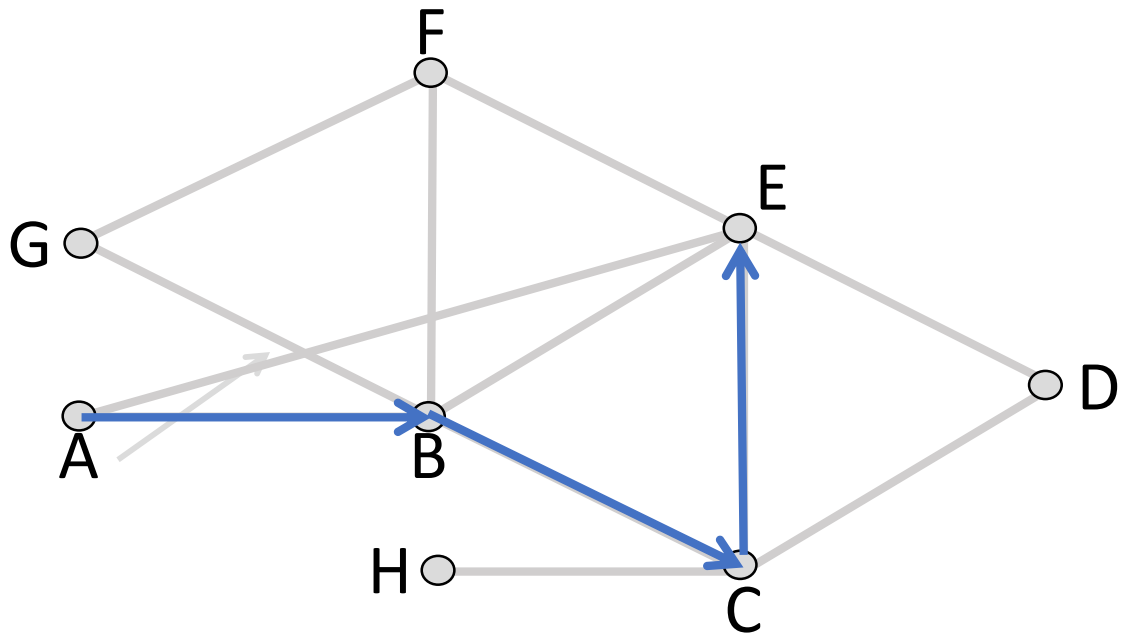
- Рутирање са најкраћим путевима (најмањим трошковима)
- Рутирање засновано на вектору раздаљине
- Плављење
- Рутирање засновано на стању веза
- Вишециљно рутирање са најкраћим путевима
- Хијерархијско рутирање и IP префикси

Рутирање

Рутирање са најкраћим путевима (најмањим трошком)

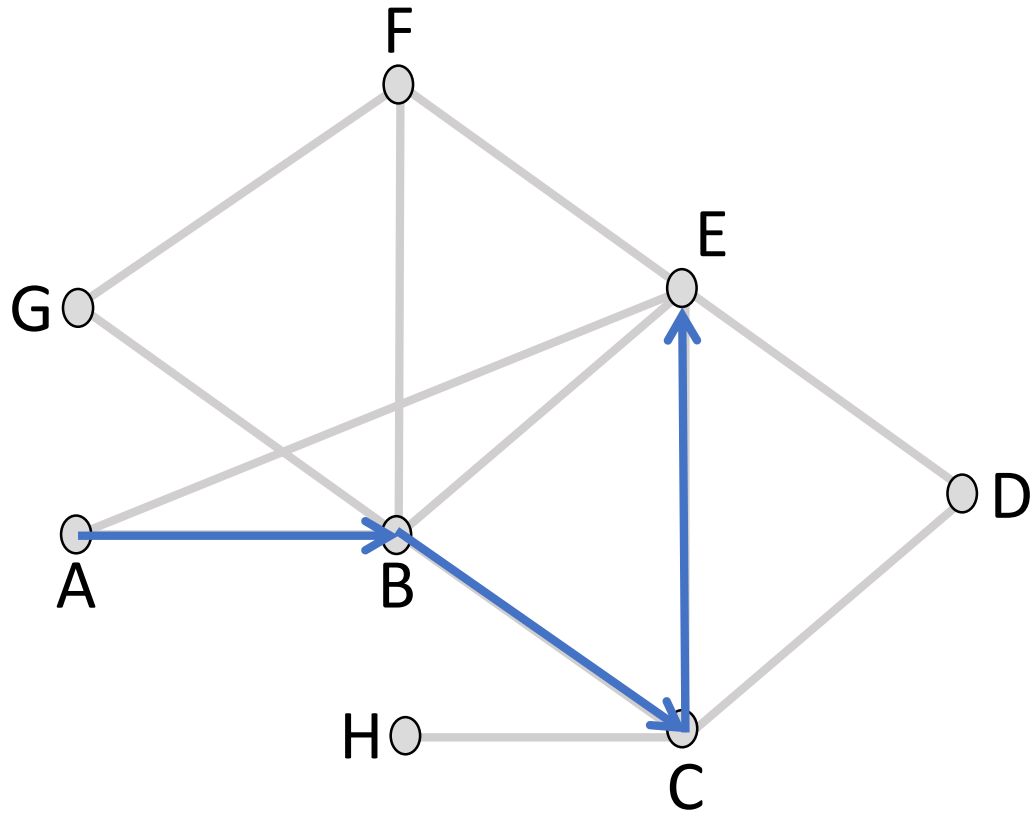
Тема

- Који пут је најбољи?
  - Мора се прво дефинисати према чему најбољи: дужини, цени, кашњењу или комбинацији истих...



Мере трошкова

- Велики број могућности:
  - Кашњење:
    избегава заобилазне путеве
  - Проток:
    избегава споре везе
  - Новац:
    избегава скупе везе
  - Број хопова:
    смањује искоришћеност комуникационе опреме



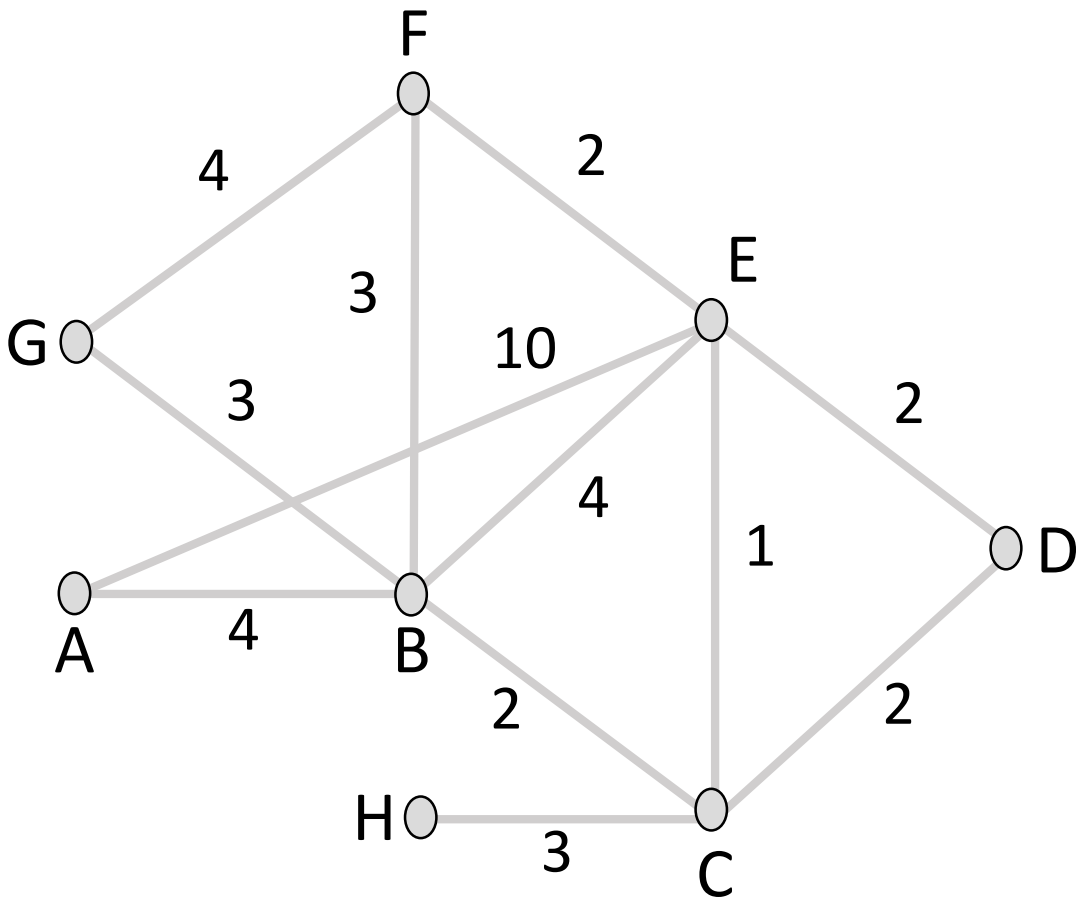
Најкраћи путеви

Пронаћи најкраћи пут за A → E

Претпоставимо да је граф:

1. Неусмерен (пакети у оба смера)
2. И да има симетричне трошкове

Могуће је моделовати алгоритме и за неусмерене, асиметричне графове.



Најкраћи путеви (2)

ABCE је најкраћи пут

dist(ABCE) = 4 + 2 + 1 = 7

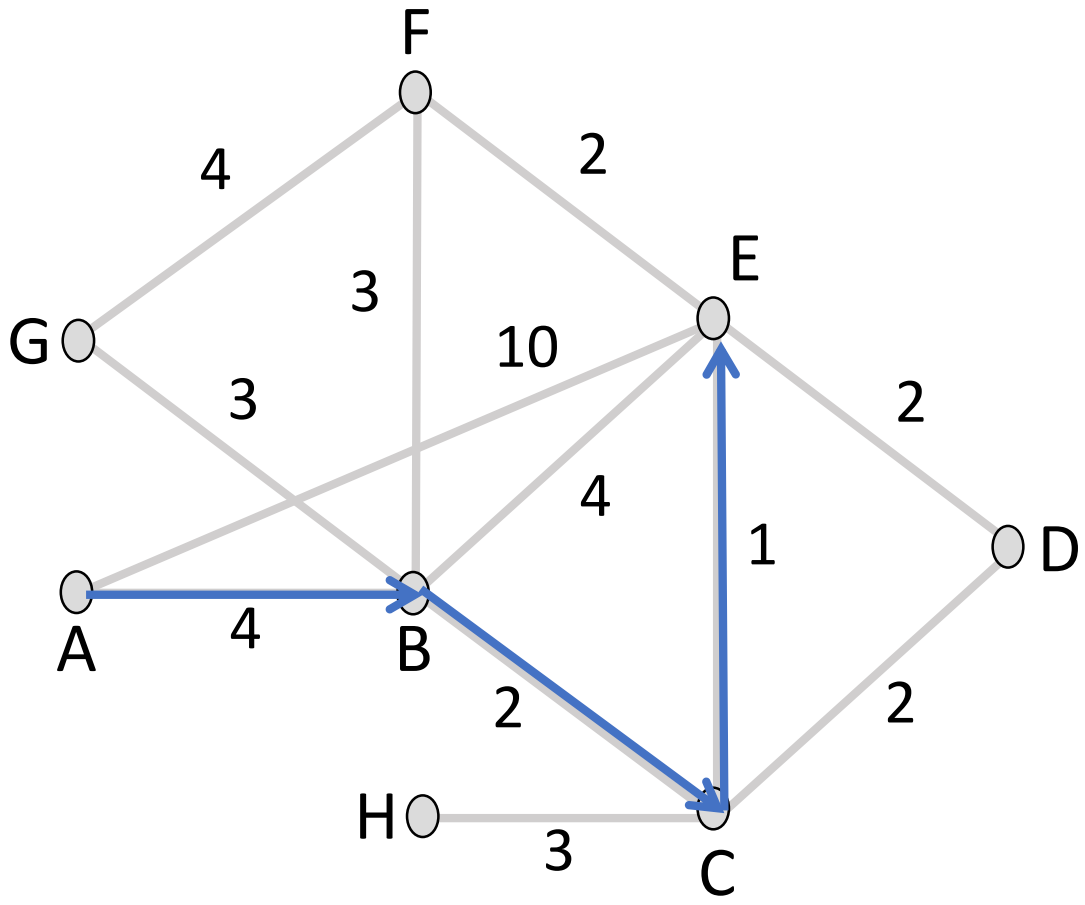
Она има трошак мањи од осталих путева:

dist(ABE) = 8

dist(ABFE) = 9

dist(AE) = 10

dist(ABCDE) = 10



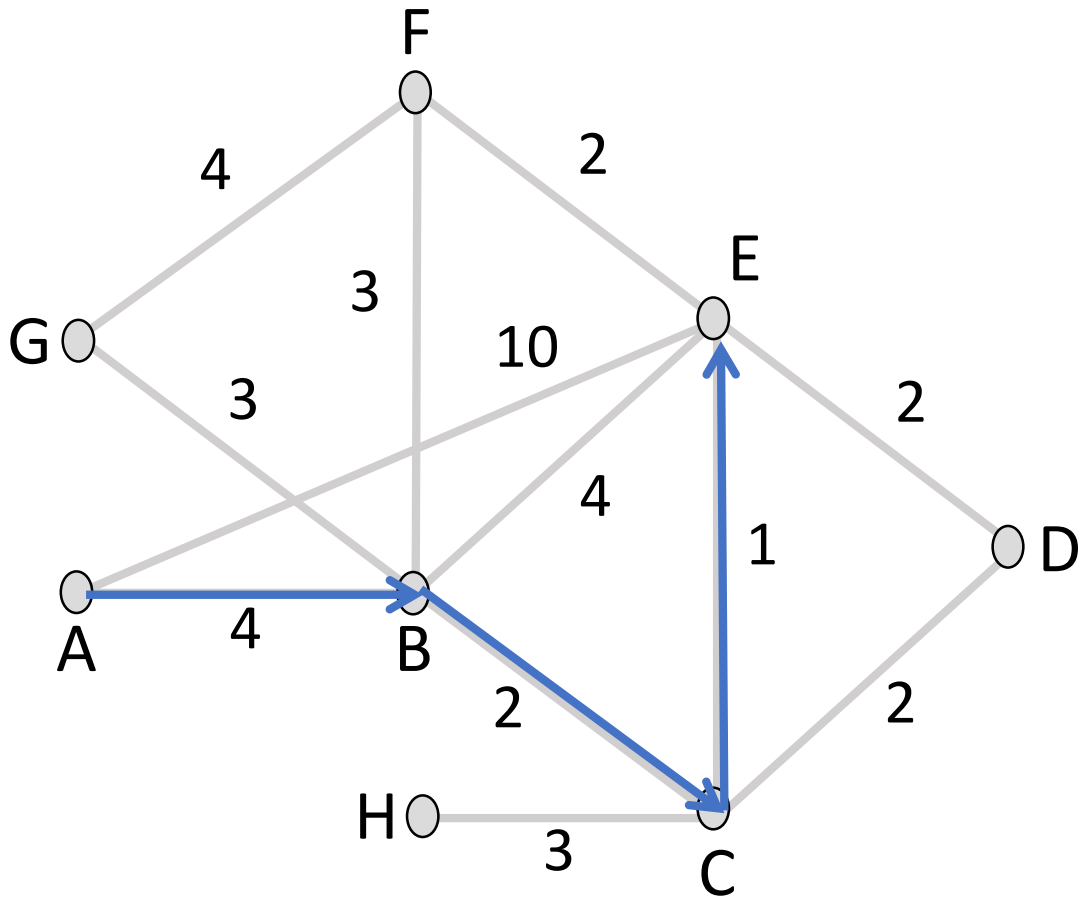
Најкраћи путеви (3)

<u>Принцип оптималности</u>:

Приметити да су сегменти оптималних (најкраћих путева) такође најкраћи путеви

ABCE је најкраћи пут између A и E

→Али су и ABC, AB, BCE, BC, CE најкраћи путеви између A и C, A и B, итд.

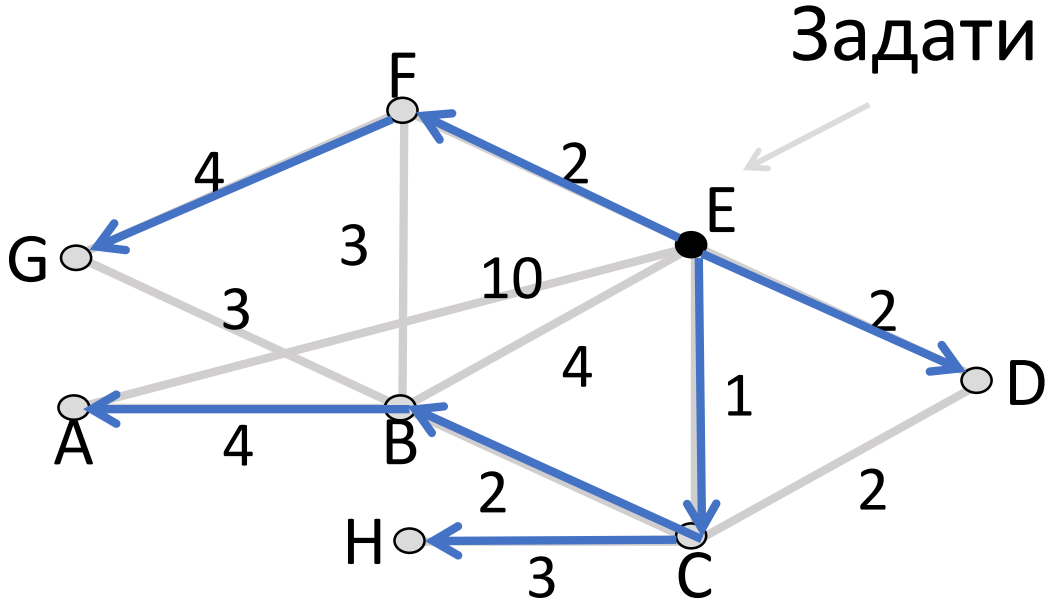


Рутирање

Дијкстрин алгоритам за најкраће путеве

Тема

- Дијкстрин алгоритам
  - Рачуна најкраће путеве између задатог чвора и свих осталих чворова
  - Резултат је дрво

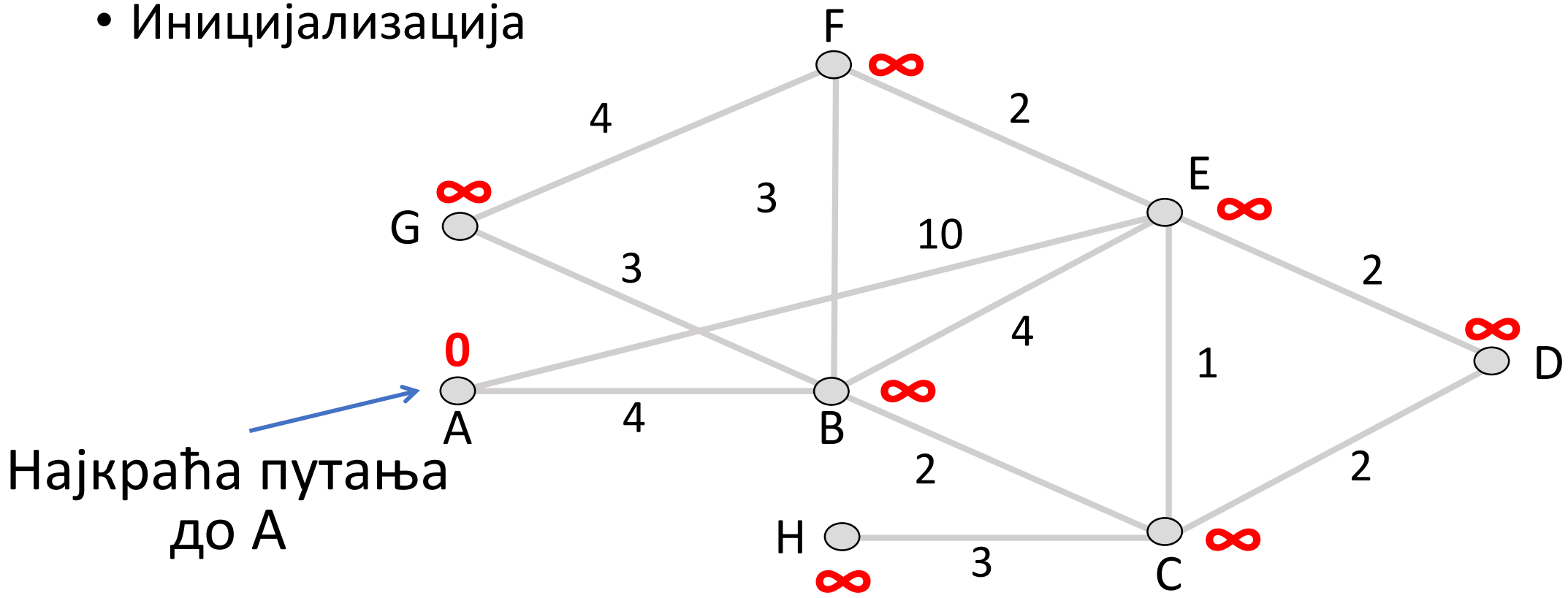


Дијкстрин алгоритам

- Поставимо све чворове као привремене
- Поставимо актуелне удаљености између задатог и свих чворова:
  - На вредност 0 ако је удаљеност до самог себе
  - На вредност ∞ (бесконачно) за све остале чворове
- Док има привремених чворова:
  - Узми привремени чвор X, који има најмању удаљеност од задатог чвора
  - Избаци X из скупа привремених чворова
    и додај одговарајућу везу ка њему у дрво
  - Умањи удаљености чворова суседних са X
    у складу са новододатом удаљеношћу

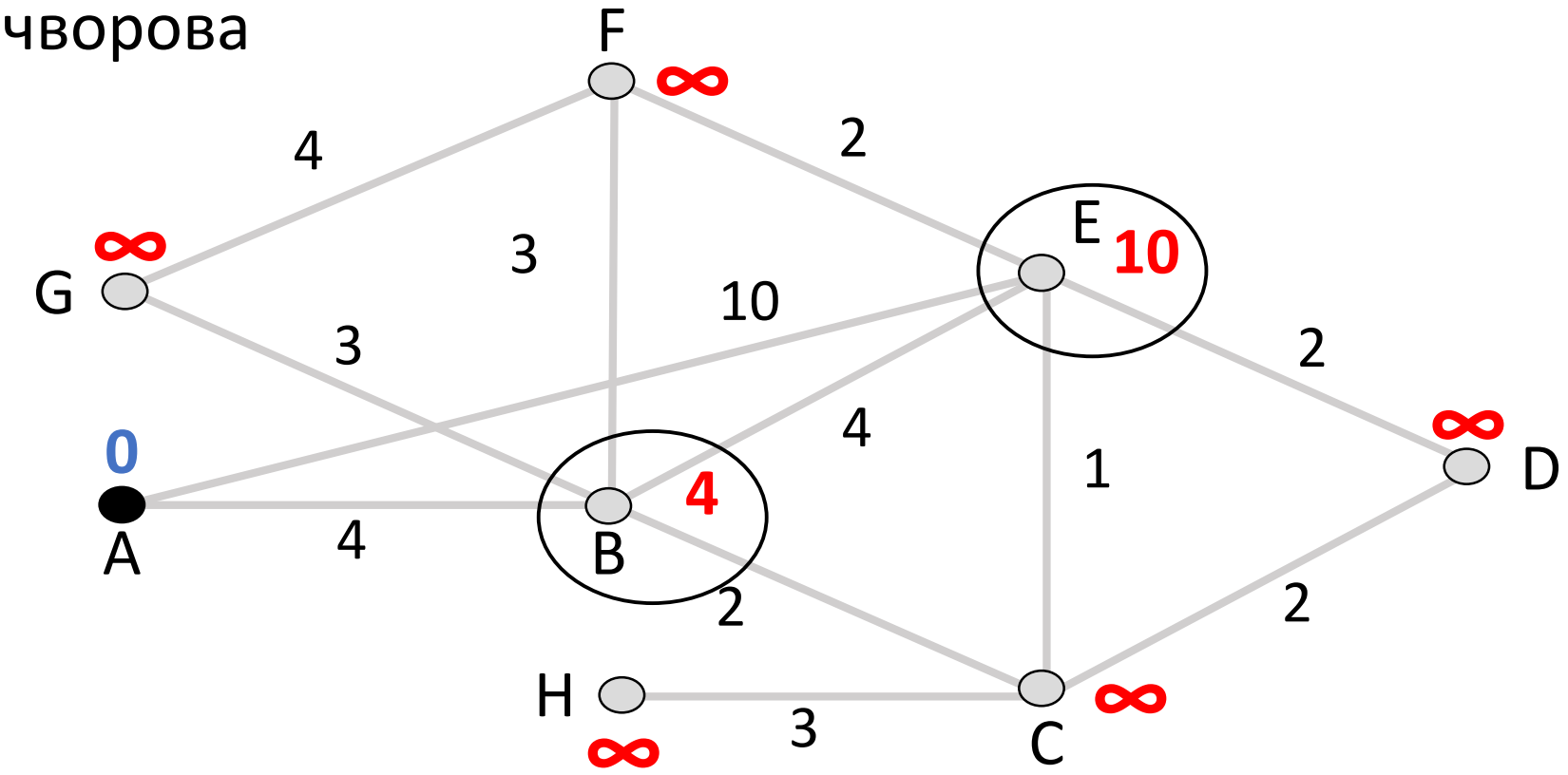
Дијкстрин алгоритам (2)

- Иницијализација



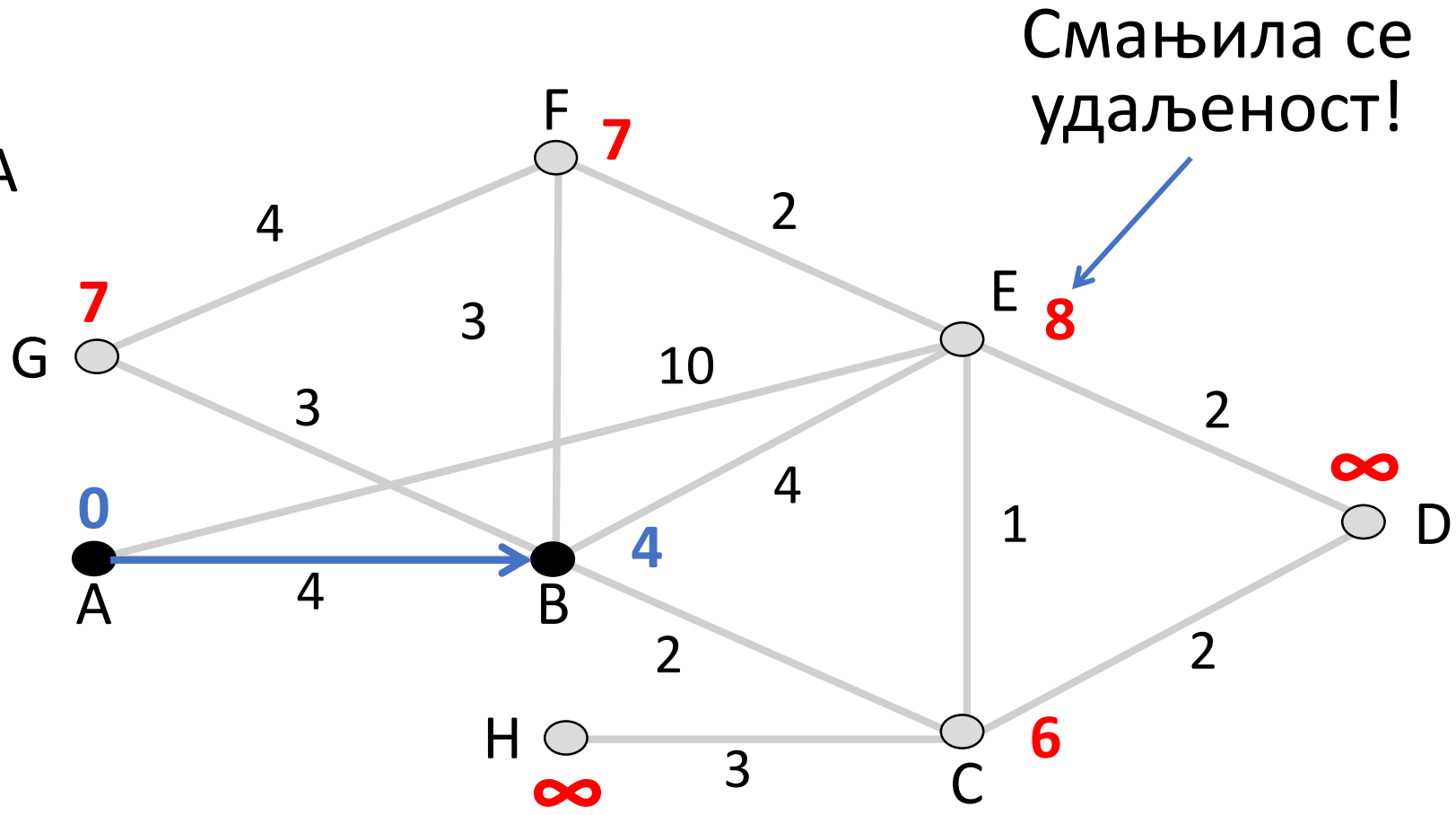
Дијкстрин алгоритам (3)

- Умањи удаљености чворова суседних са A



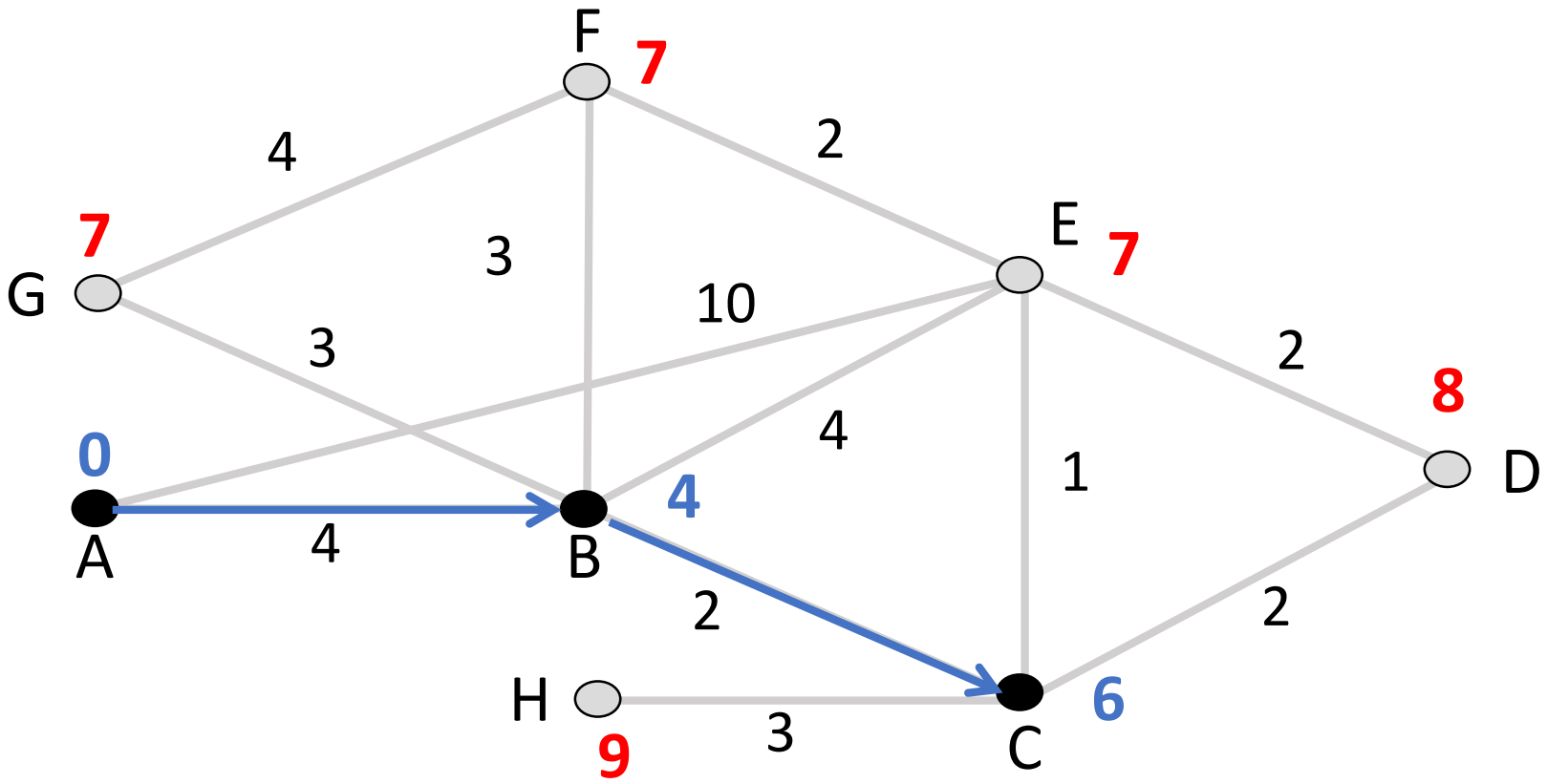
Дијкстрин алгоритам (4)

- Бирамо B,
  јер је он најближи A
- Такође ажурирамо чворове који су суседни са B
- У следећем кораку бирамо C, јер је он нови најближи



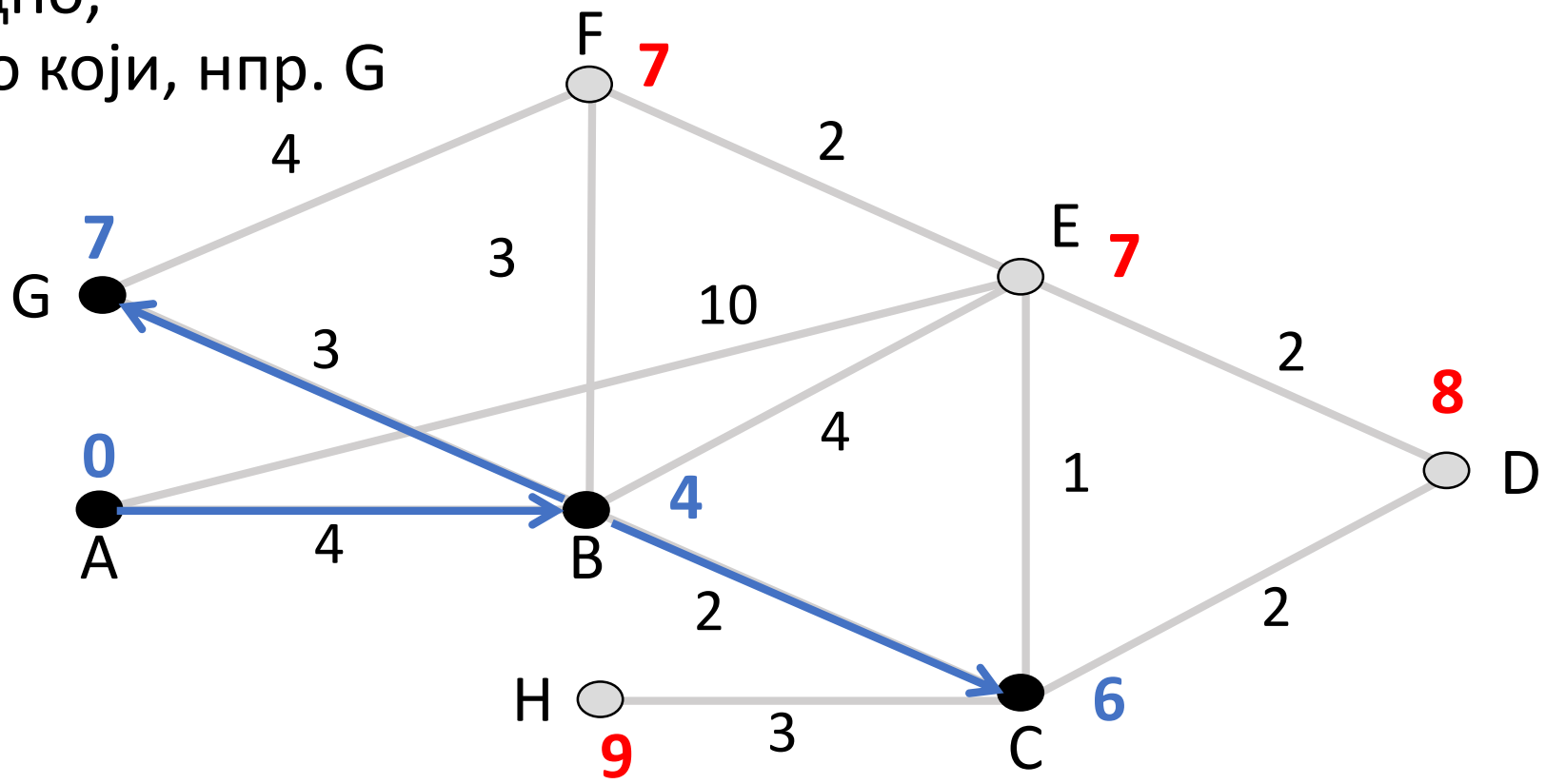
Дијкстрин алгоритам (5)

- Бирамо C



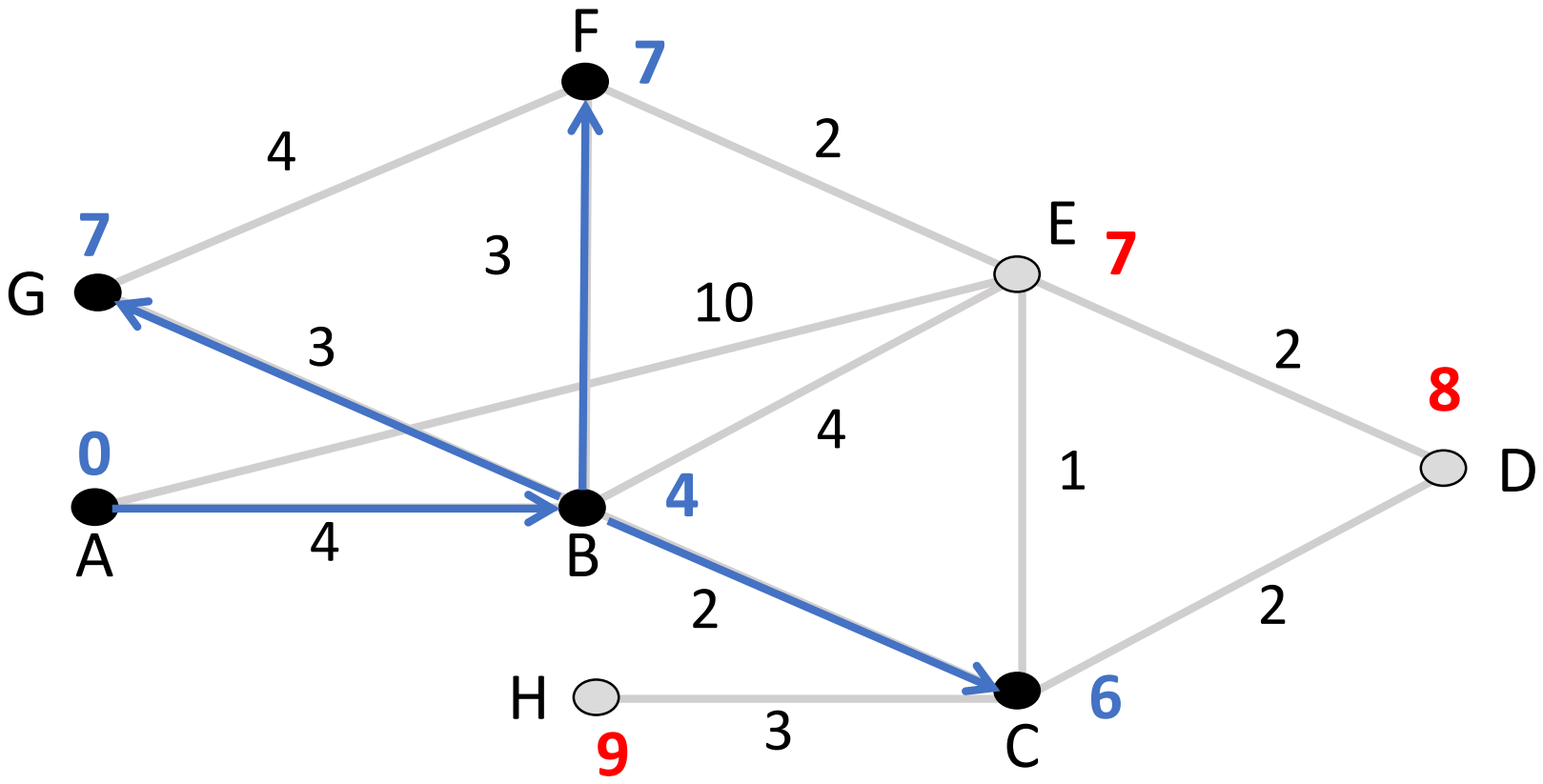
Дијкстрин алгоритам (6)

- Кад је свеједно, бирамо било који, нпр. G



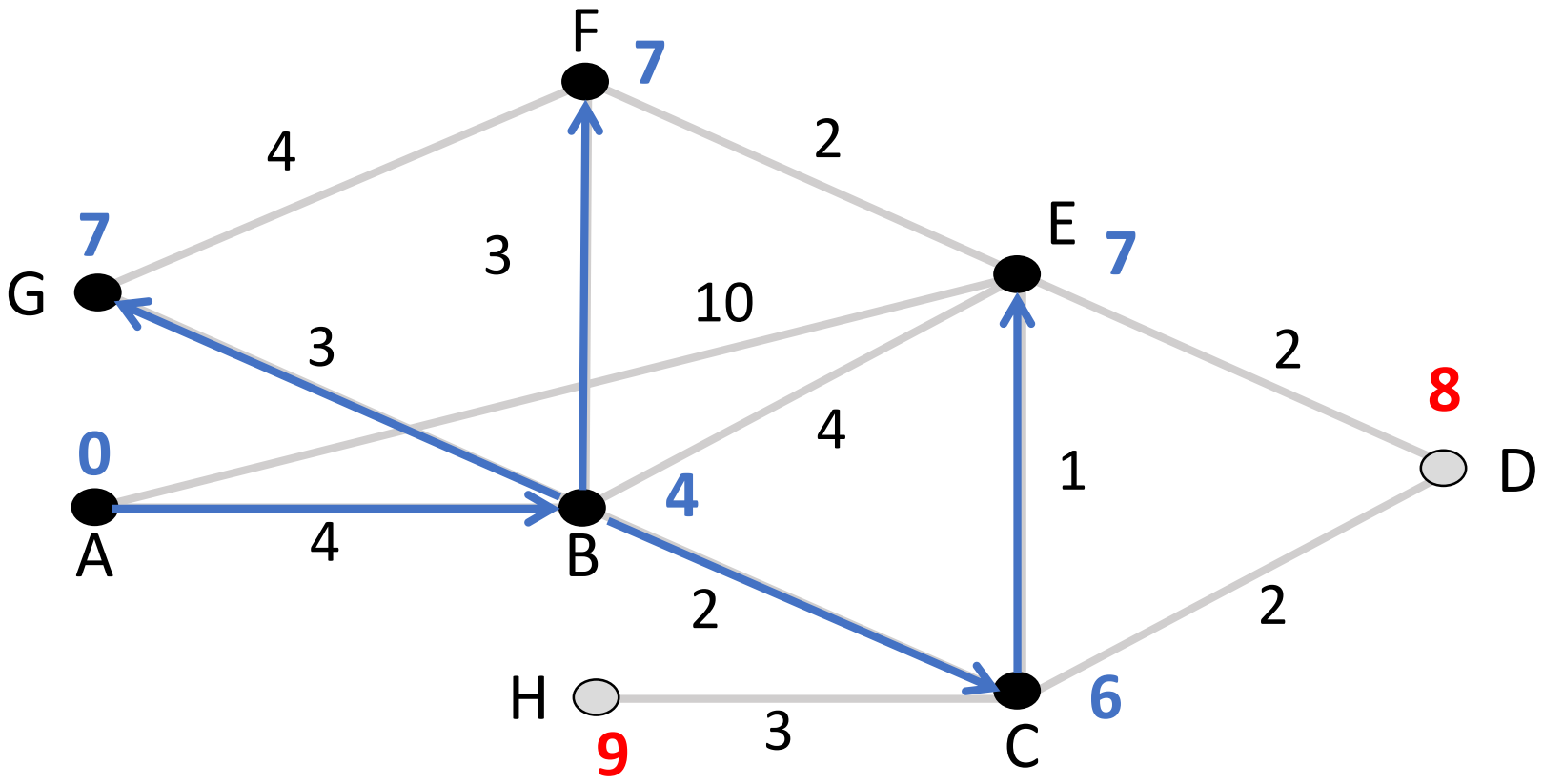
Дијкстрин алгоритам (7)

- Бирамо F



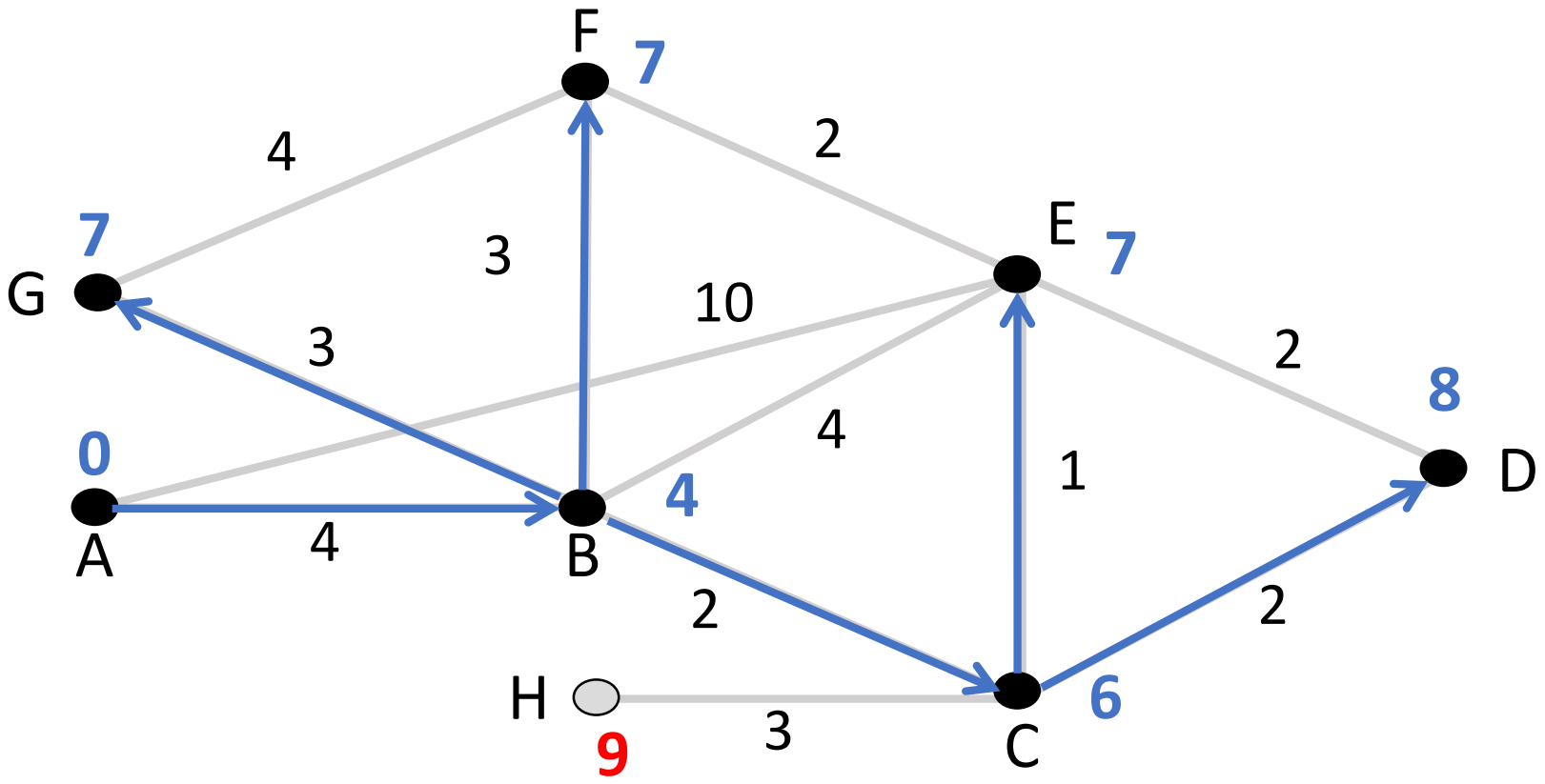
Дијкстрин алгоритам (8)

- Бирамо E



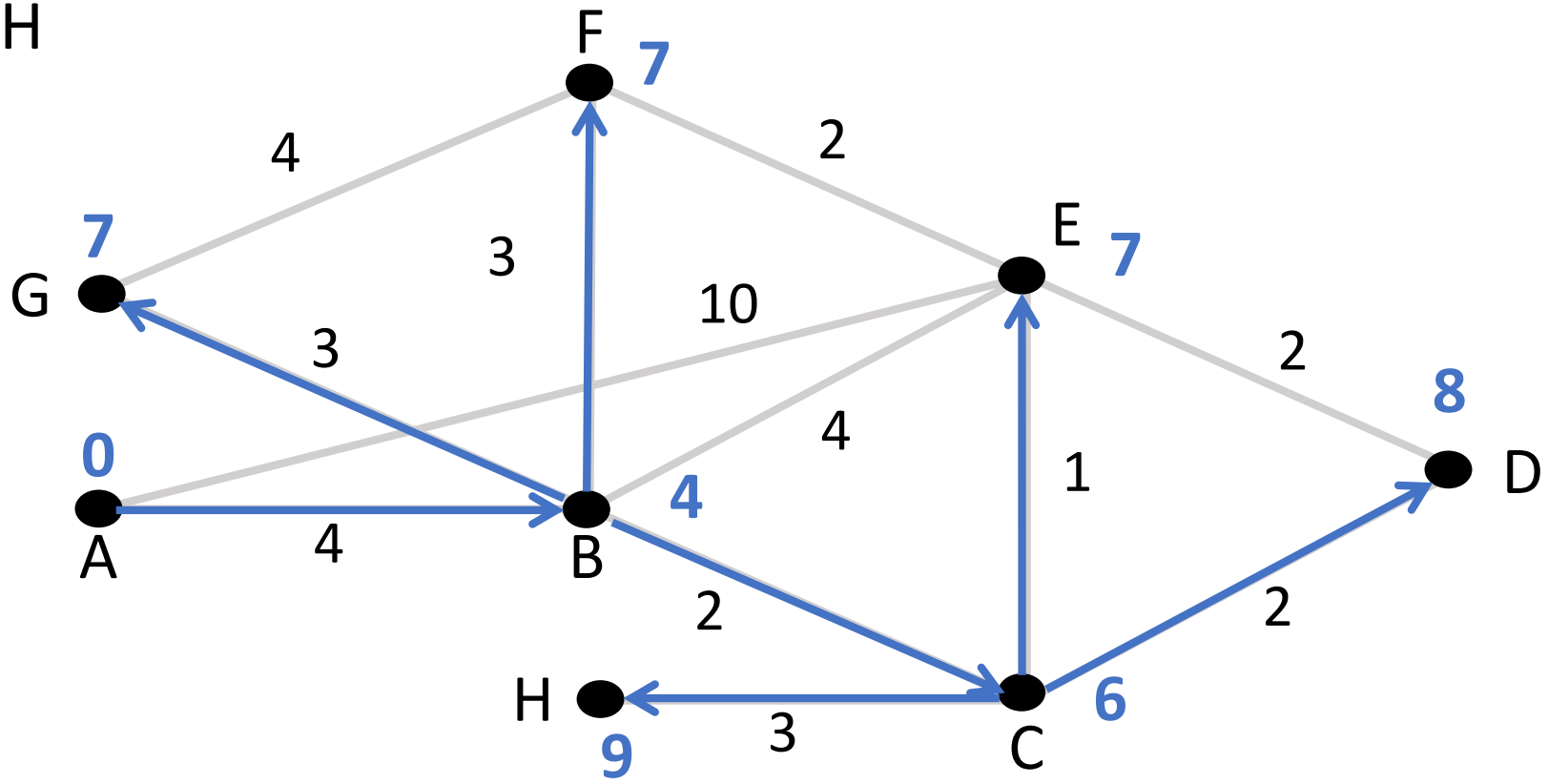
Дијкстрин алгоритам (9)

- Бирамо D



Дијкстрин алгоритам (10)

- И коначно H



Карактеристике алгоритма

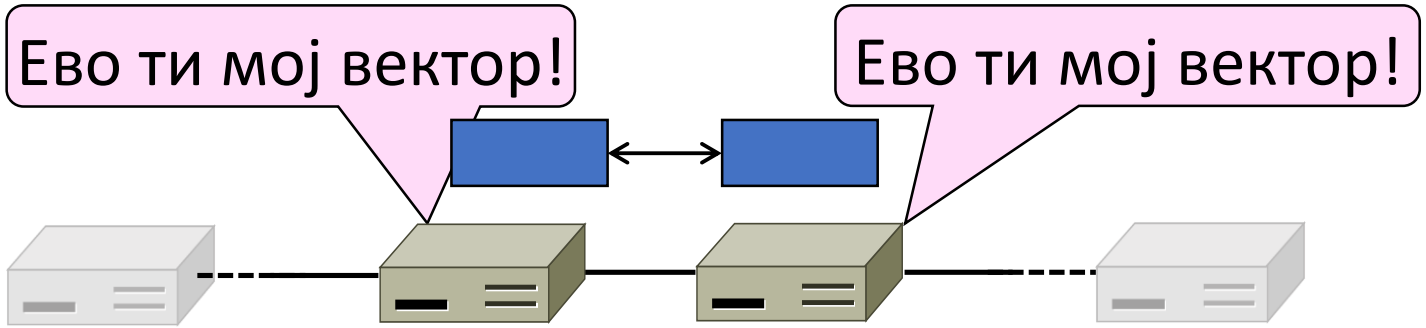
- Проналази путеве ка чворовима према растућем поретку дужина
  - Користи својство декомпозиције оптималности
  - Пошто у сваком кораку бирамо чвор X до којег је најкраћи пут,
    не може се десити да пут до X преко неког касније додатог чвора Y буде краћи!
- Време извршавања зависи од ефикасности проналажења најбољег привременог чвора (оног до кога постоји најкраћи пут)
  - Може се користити нпр. хип структура
  - Сложеност је већа од линеарне

Рутирање

Рутирање засновано на вектору раздаљине (Distance vector routing)

DV рутирање (Distance vector routing)

- Овај приступ се заснива на размени вектора (табела) раздаљине између суседних чворова
- Један од првих приступа (Arpanet)
  - У пракси се сада ретко користи



DV рутирање

Сваки чвор одржава вектор удаљености и следећих хопова
за све циљне чворове

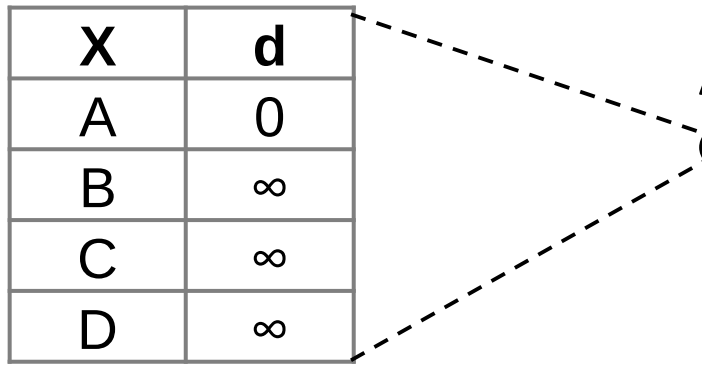
Сваки чвор ради следеће:

1. Иницијализује удаљеност до самог себе на 0, и удаљеност ка свим осталим циљним чворовима на ∞ (бесконачно)
2. Периодично шаље свој вектор ка суседима
3. Прерачунава свој вектор удаљености
   на основу вектора добијених од својих суседа

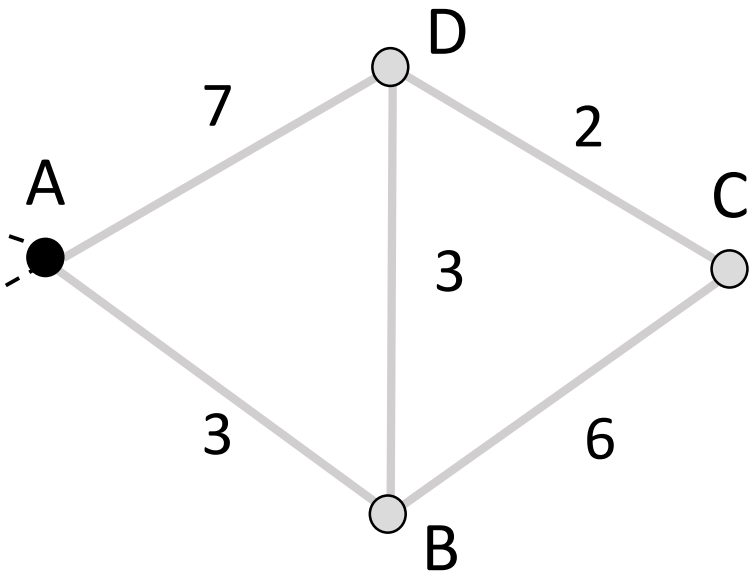
DV пример

Чвор A нпр. размењује векторе само са B и D

A
иницијално



X	d
A	0
B	∞
C	∞
D	∞



DV пример (2)

- Након првог корака, свако научи путеве дужине 1
- А увек ажурира вектор коришћењем вектора min(B+3, D+7)

X	B каже	D каже
A	∞	∞
B	0	∞
C	∞	∞
D	∞	0

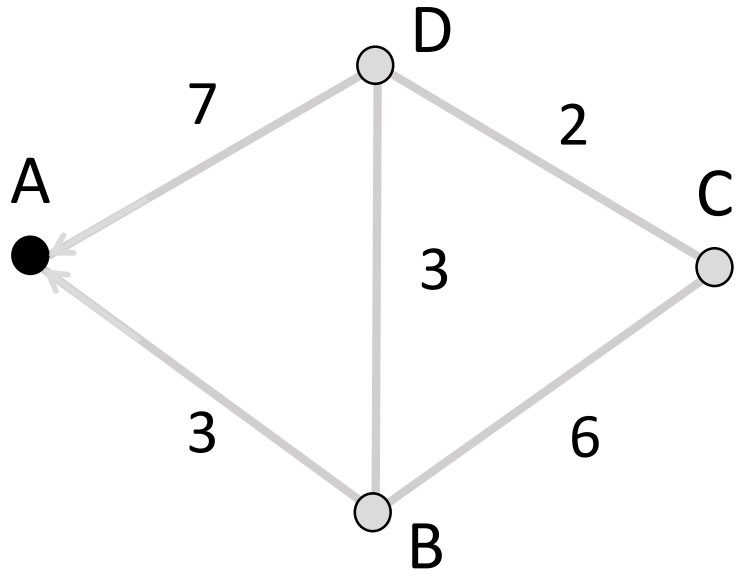
→

B +3	D +7
∞	∞
3	∞
∞	∞
∞	7

→

A научи	
d	Y
0	--
3	B
∞	--
7	D

= научен бољи пут



DV пример (3)

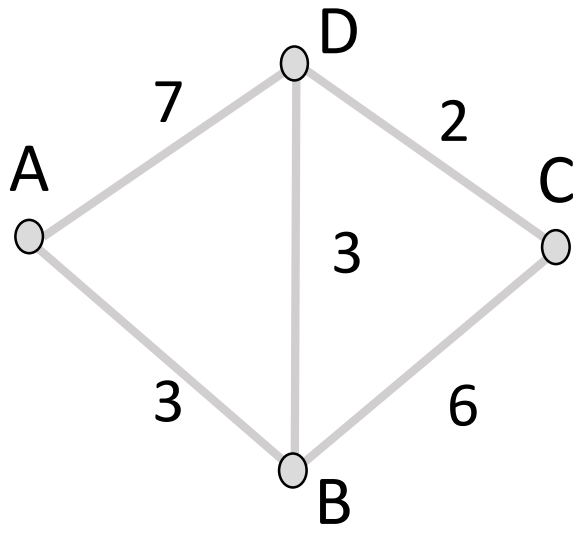
- Након свих првих размена, сви уче табелу 1-хоп путева
  - В нпр. учи на основу вектора min(A+3, C+6, D+3)

X	A каже	B каже	C каже	D каже
A	0	∞	∞	∞
B	∞	0	∞	∞
C	∞	∞	0	∞
D	∞	∞	∞	0

→

A научи		B научи		C научи		D научи	
d	Y	d	Y	d	Y	d	Y
0	--	3	A	∞	--	7	A
3	B	0	--	6	B	3	B
∞	--	6	C	0	--	2	C
7	D	3	D	2	D	0	--

= научен бољи пут



DV пример (4)

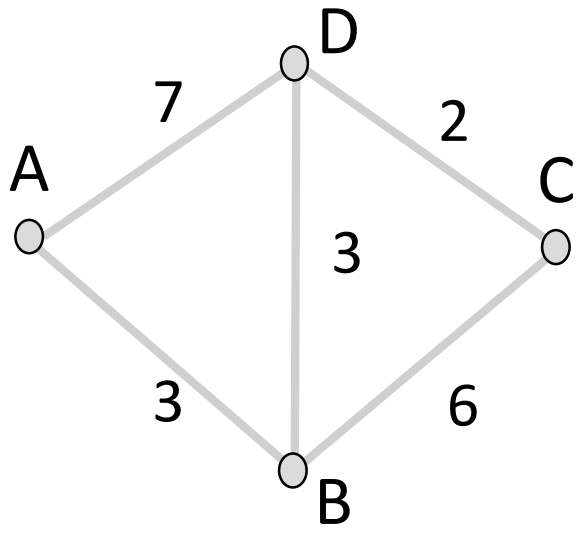
- Након другог круга размена, научени су сви најбољи путеви дужине 2 (2-хоп путеви)

X	A каже	B каже	C каже	D каже
A	0	3	∞	7
B	3	0	6	3
C	∞	6	0	2
D	7	3	2	0

→

A научи		B научи		C научи		D научи	
d	Y	d	Y	d	Y	d	Y
0	--	3	A	9	B	6	B
3	B	0	--	5	D	3	B
9	D	5	D	0	--	2	C
6	B	3	D	2	D	0	--

= научен бољи пут



DV пример (5)

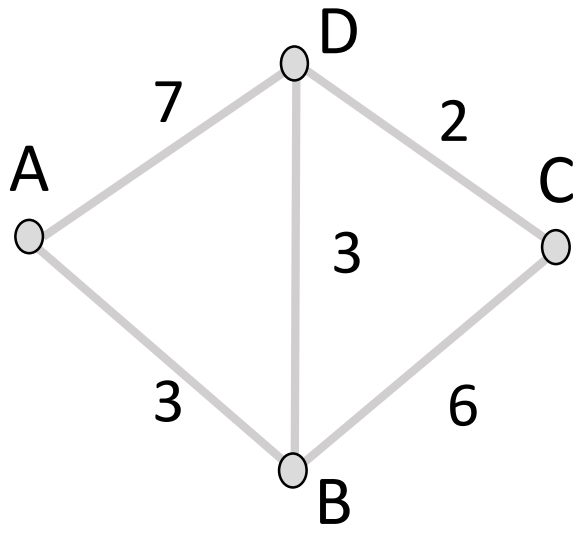
- Најбољи путеви до дужине 3

X	A каже	B каже	C каже	D каже
A	0	3	9	6
B	3	0	5	3
C	9	5	0	2
D	6	3	2	0

→

A научи		B научи		C научи		D научи	
d	Y	d	Y	d	Y	d	Y
0	--	3	A	8	D	6	B
3	B	0	--	5	D	3	B
8	B	5	D	0	--	2	C
6	B	3	D	2	D	0	--

= научен бољи пут



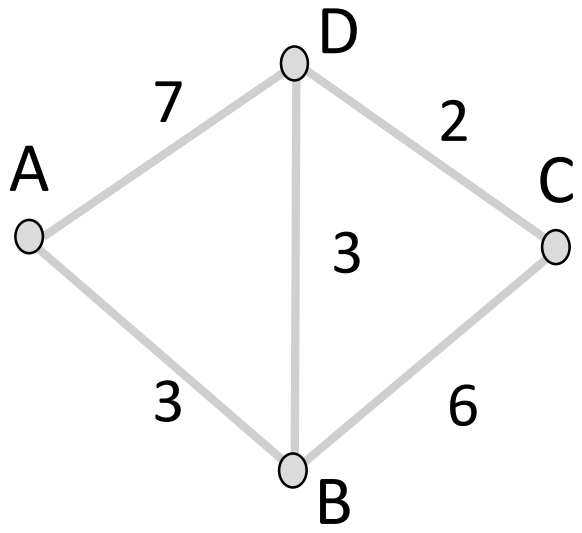
DV пример (5)

- Последњи круг, након овога табела конвергира (конвергенција наравно зависи од графа)

X	A каже	B каже	C каже	D каже
A	0	3	8	6
B	3	0	5	3
C	8	5	0	2
D	6	3	2	0

→

A научи		B научи		C научи		D научи	
d	Y	d	Y	d	Y	d	Y
0	--	3	A	8	D	6	B
3	B	0	--	5	D	3	B
8	B	5	D	0	--	2	C
6	B	3	D	2	D	0	--



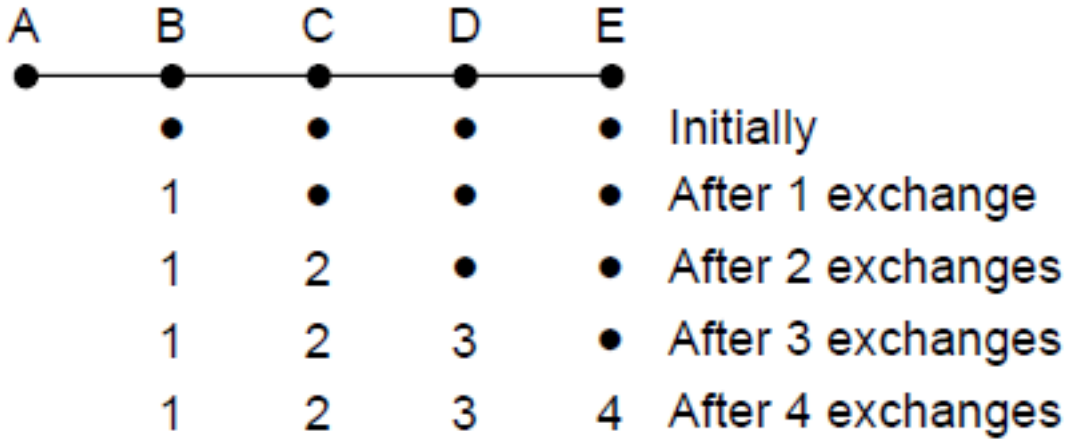
= научен бољи пут

DV рутирање - робусност

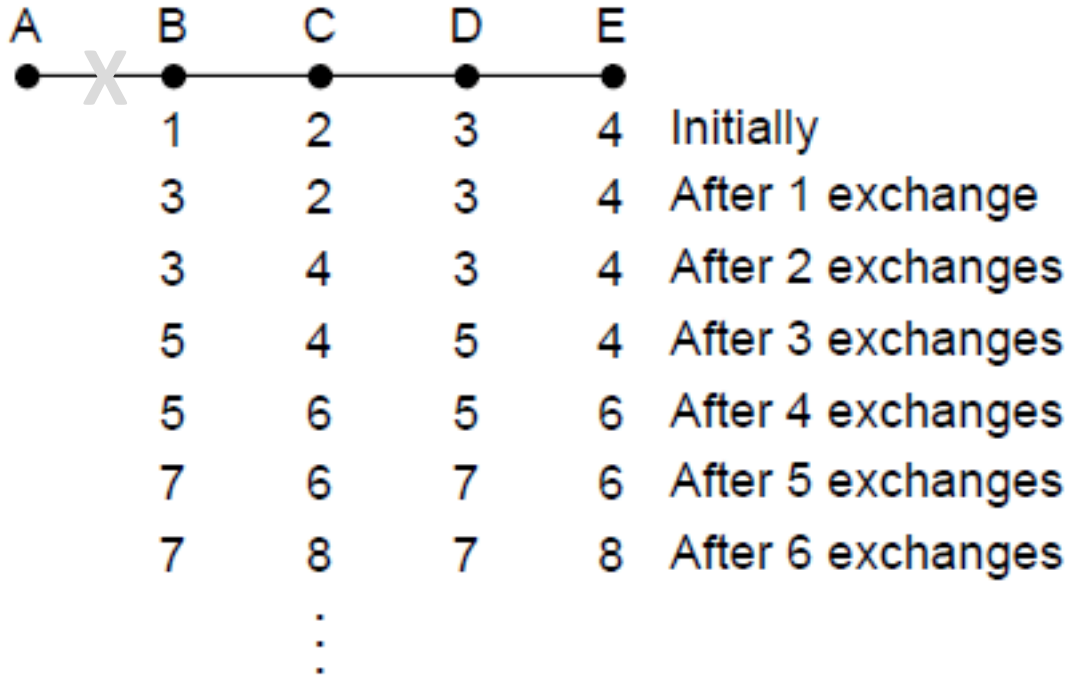
- Додавање веза или чворова:
  - Вест путује један хоп по размени
  - Ово није претерано брзо
- Уклањање веза или чворова:
  - Суседи не обављају више размену са њим па после неког времена забораве да уопште постоји
  - Може се увести максимална старост размењеног вектора
- Разбијања мреже на два дела су озбиљан проблем!
  - Проблем бројања до бесконачности
  - Постоје начини да се ово разреши, али ћемо прескочити тај део...

Проблем бројања до бесконачности (count to infinity scenario)

- Добра вест путује брзо, а лоша споро...



Све ок са мрежом



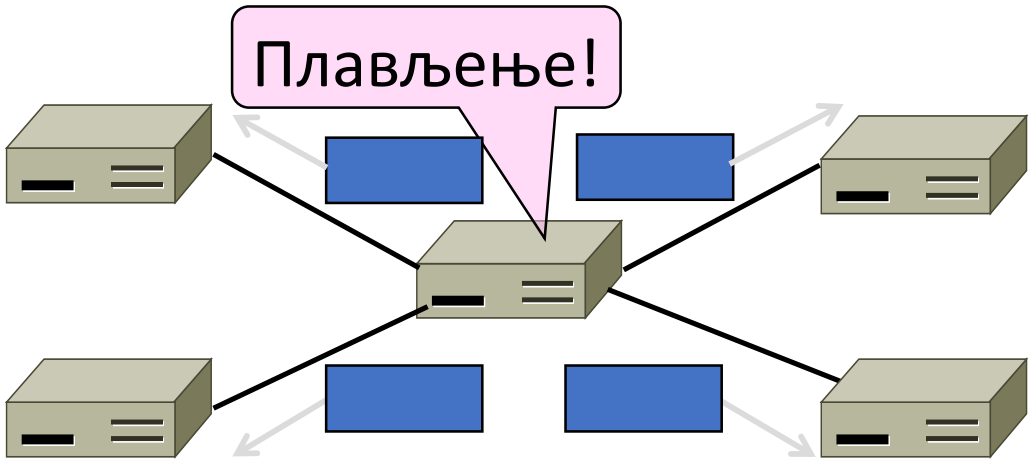
Мрежа разбијена на два дела
Бројање до бесконачно

Рутирање

Плављење

Тема

- Емитовање поруке свим чворовима помоћу технике плављења
  - Једноставан механизам, не претерано ефикасан

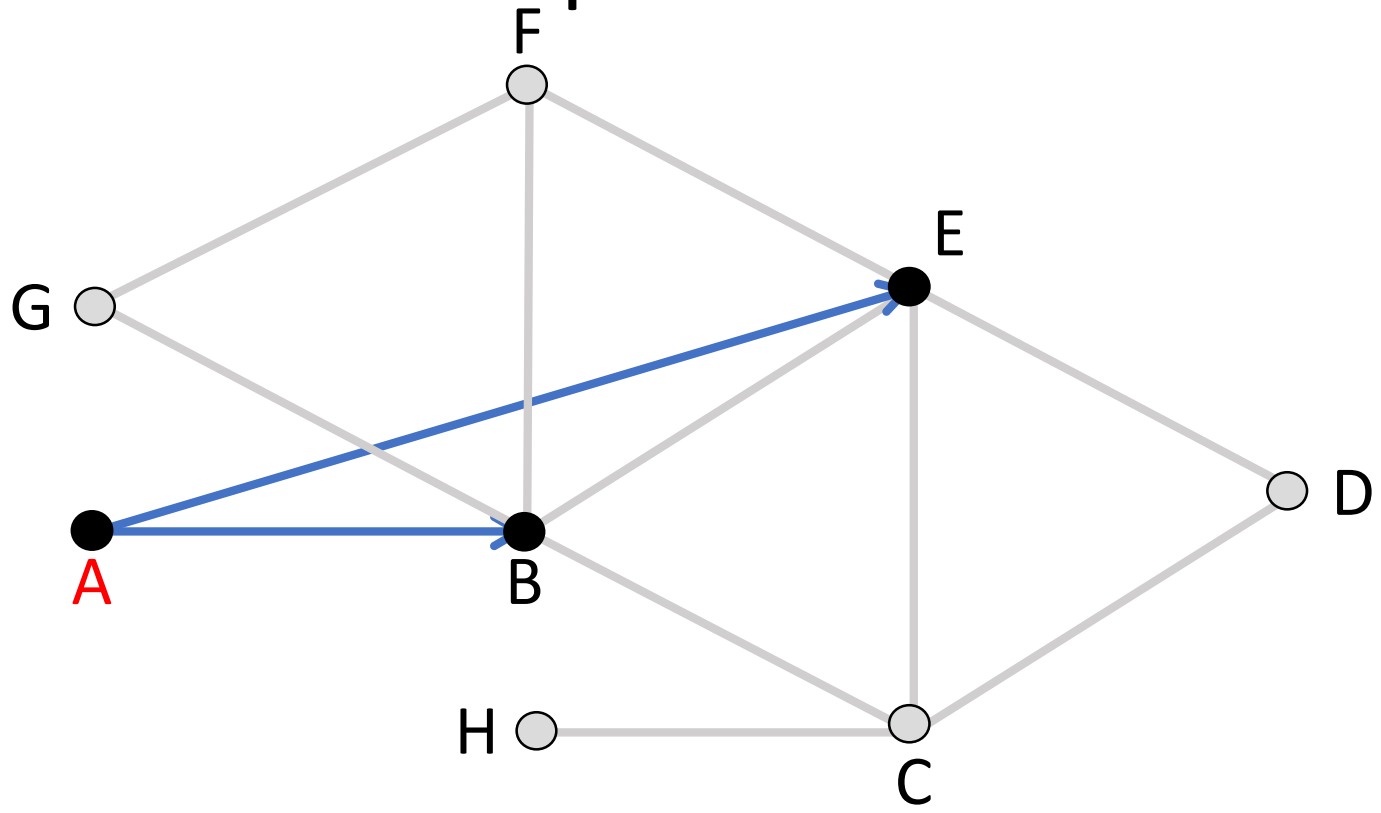


Плављење

- Правило на сваком чвору:
  - Након пристизања поруке, проследи је свим осталим суседима
  - Запамти некако поруку како је би поново прослеђивао ако ти опет стигне
- Неефикасно, јер један чвор може да добије вишеструке копије исте поруке

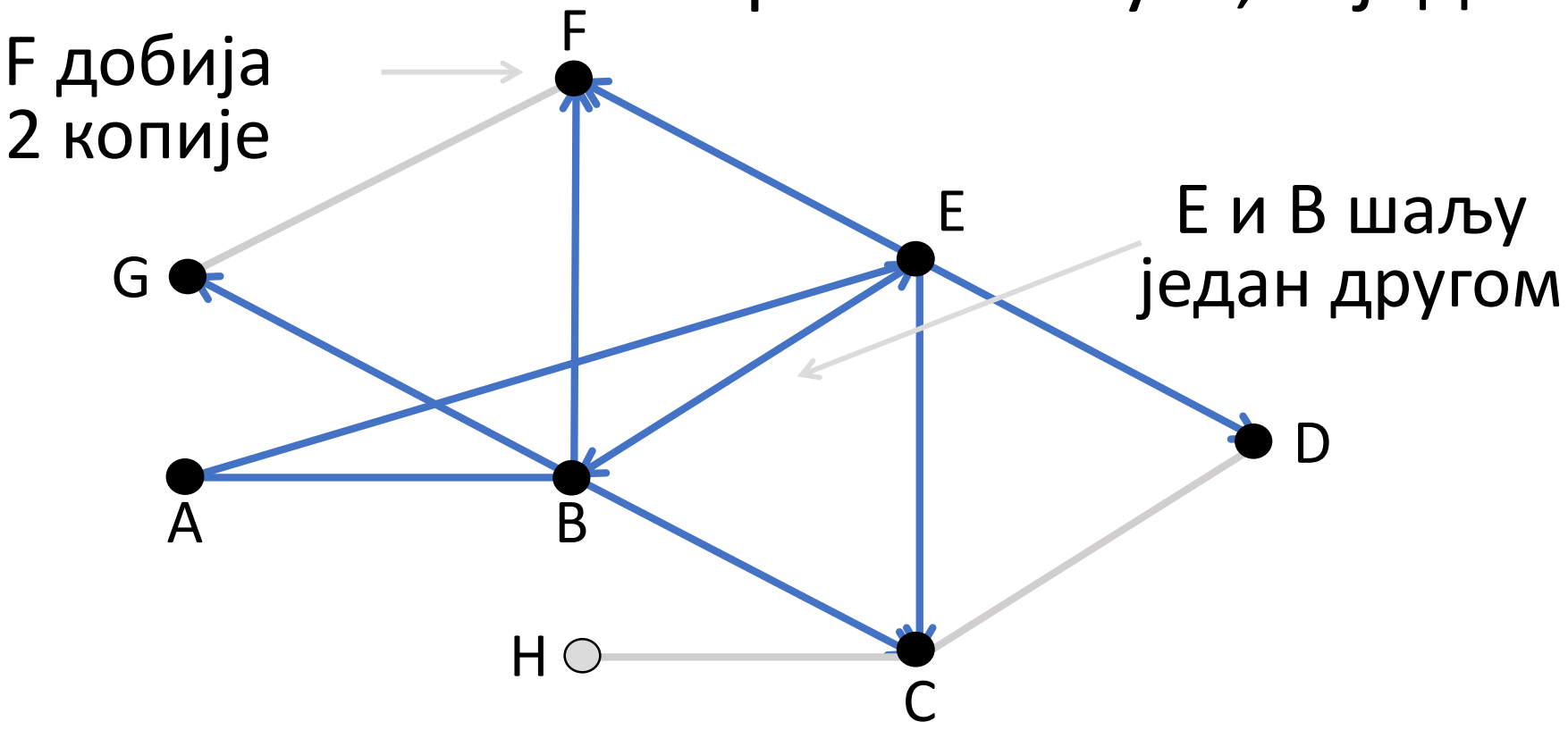
Плављење (2)

- Плављење полази нпр. из A



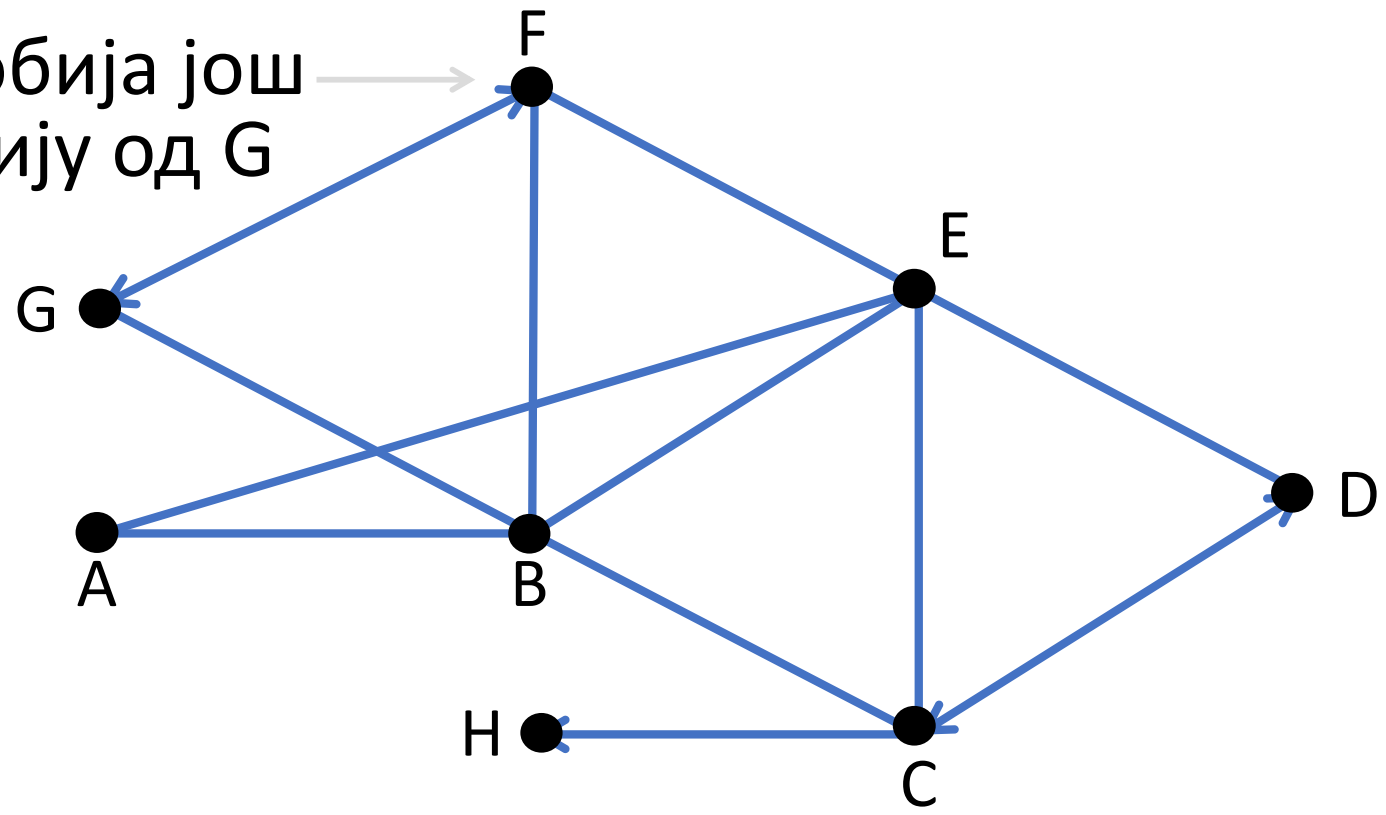
Плављење (3)

- Након тога B и E плаве чворове F 2 пута, G једном итд.



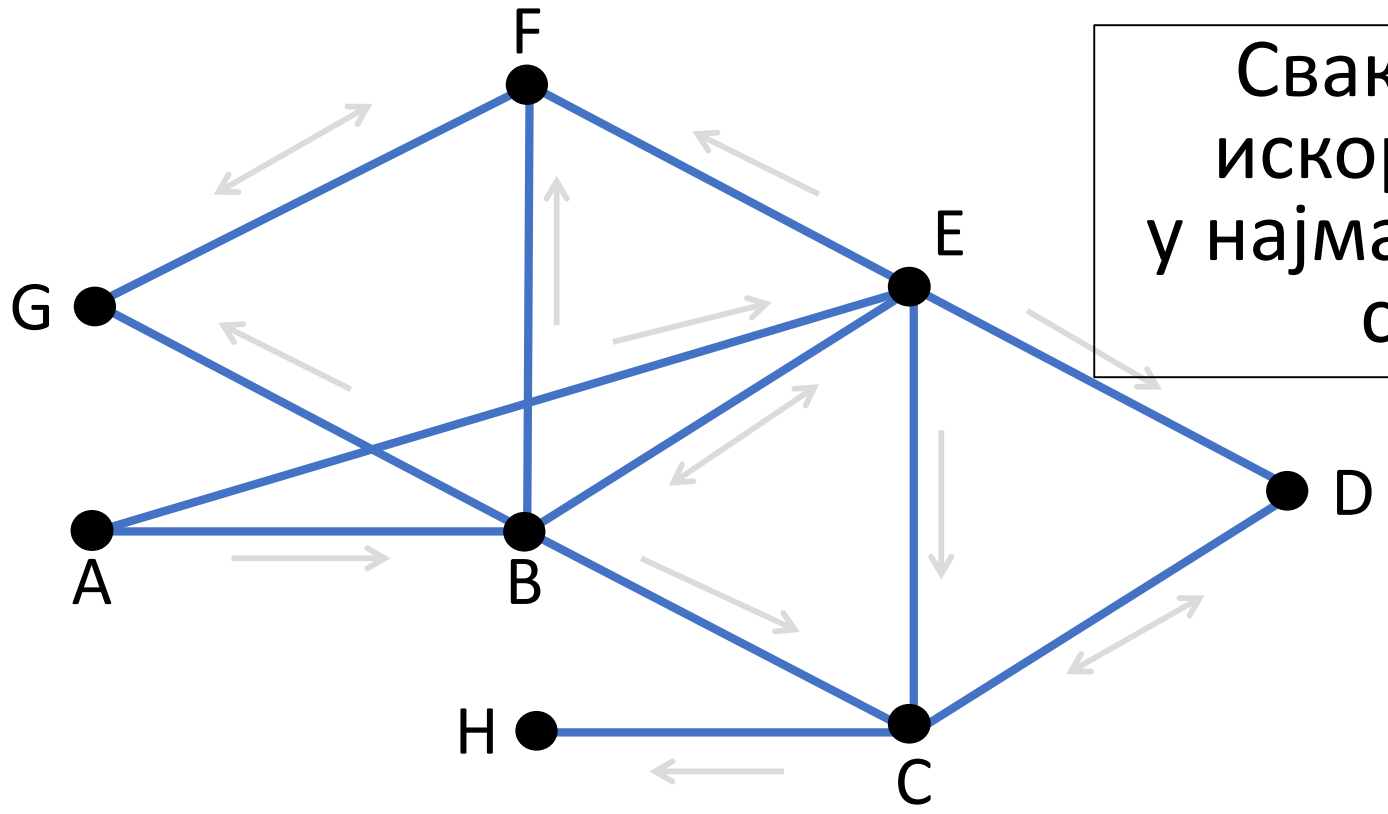
Плављење (4)

F потом добија још једну копију од G



Плављење (5)

- H не плави никога



Свака веза је искоришћена, у најмање једном смеру

Плављење – остали аспекти

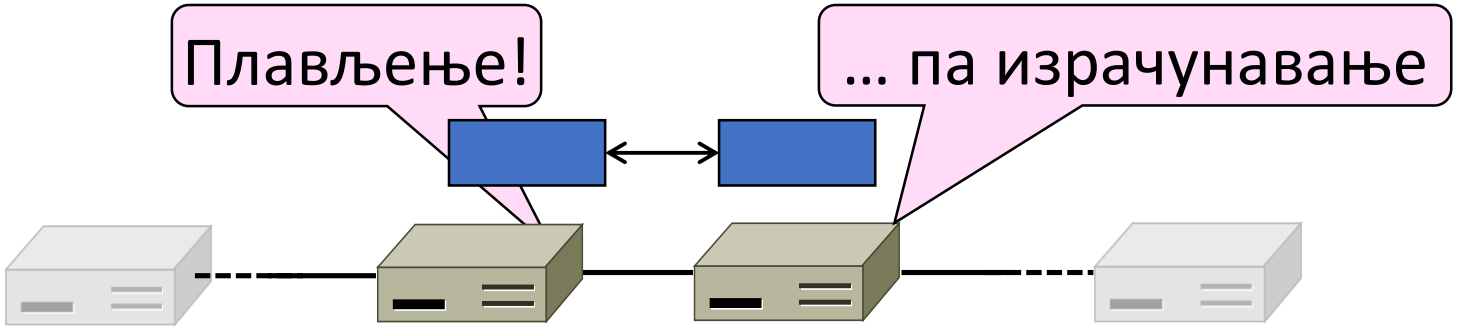
- Приликом памћења, довољно је запамтити само извор и редни број поруке
  - Следећа порука се прихвата само ако има виши редни број
- Могуће је омогућити и ARQ
  како би плављење било поузданије

Рутирање

Рутирање засновано на стању веза (Link state routing)

LS рутирање (Link state routing)

- Другачије од DV рутирања
- Користи претходно уведене концепте



LS рутирање

Две фазе:

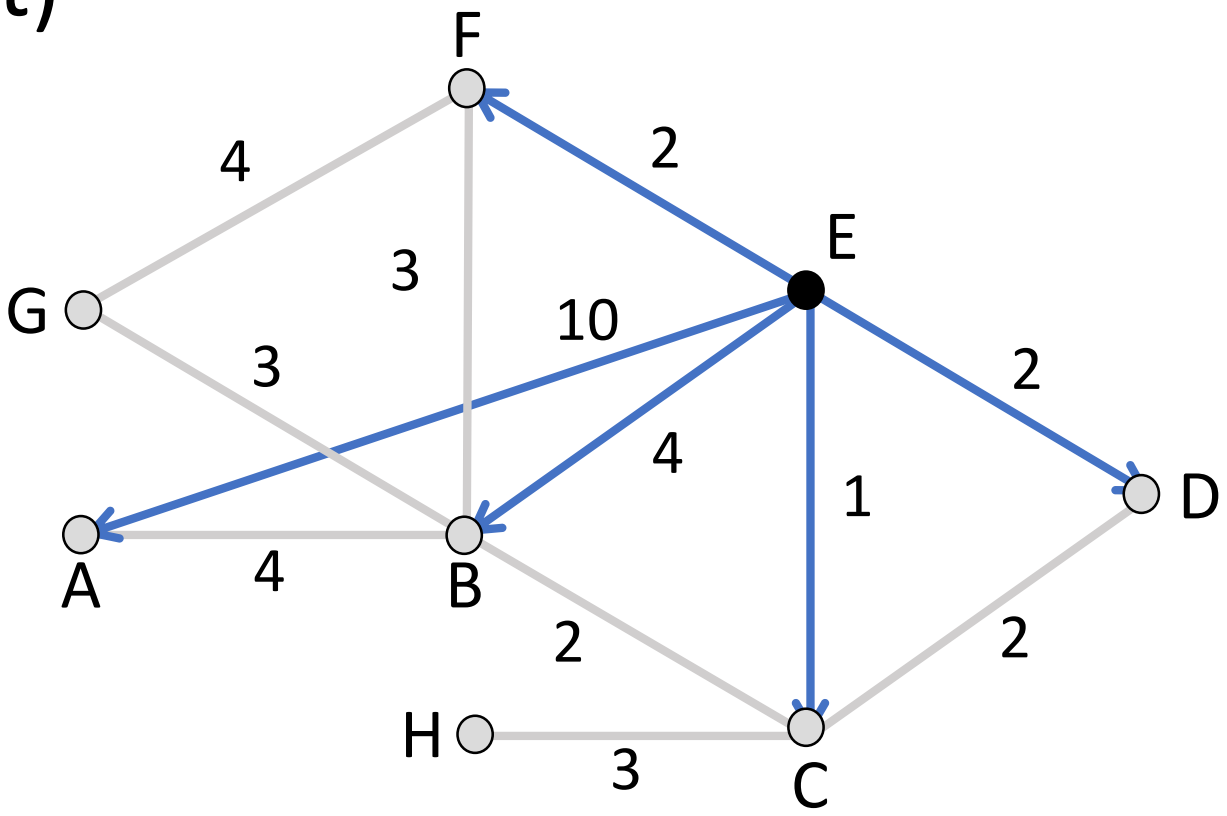
1. Чворови плаве мрежу информацијама о својој локалног топологији (суседима)
   - Сваки чвор је тако у стању да реконструише целокупну топологију
2. Сваки чвор рачуна своју табелу прослеђивања
   - Ово се постиже нпр. Дијкстриним алгоритмом

Фаза 1: Реконструкција топологије

- Сваки чвор плави мрежу својим LS пакетом (link state packet)

LSP чвора E

Seq. #	
A	10
B	4
C	1
D	2
F	2

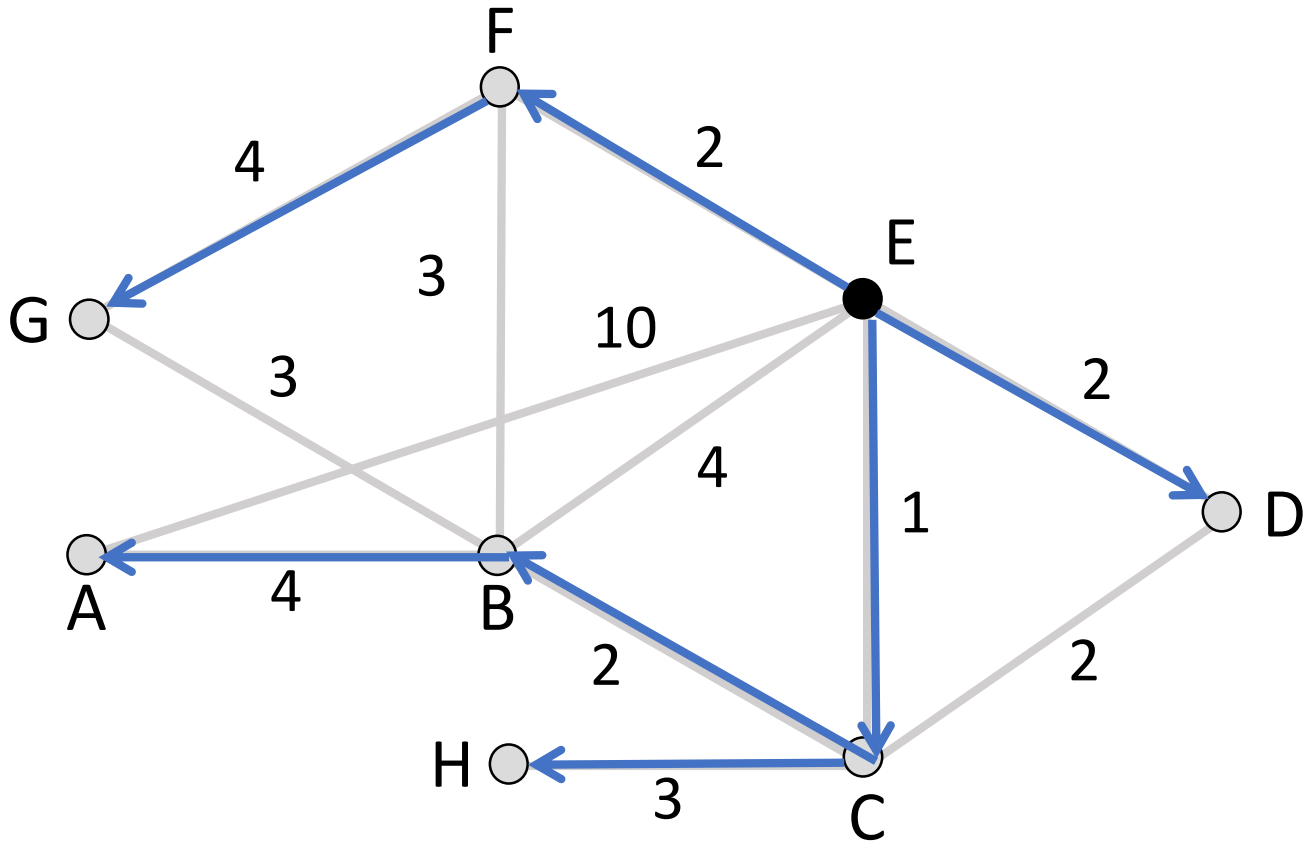


Фаза 2: Рачунање најкраћих путева

- Сваки чвор има пуну топологију
  - Добија је тако што комбинује све LSP
- Сваки чвор изврши Дијкстрин алгоритам
  - Делује да има сувишног израчунавања ако се гледа цела мрежа

Табела прослеђивања

Након Дијкстриног алгоритма



Табела чвора E

X	Y
A	C
B	C
C	C
D	D
E	--
F	F
G	F
H	C

Промене у мрежи

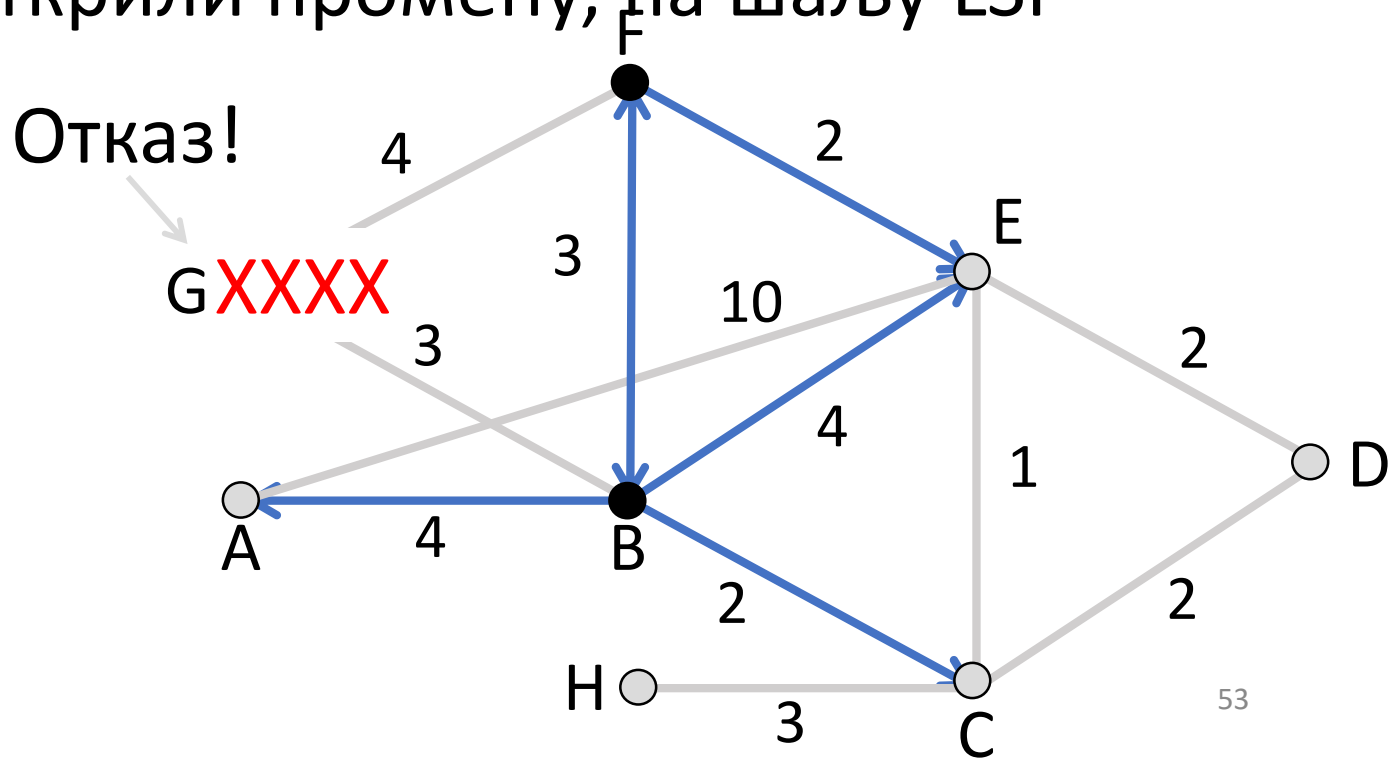
- Када неки чвор детектује локалну промену, он плави са новим LSP
  - Нпр. суседи од G су открили промену, па шаљу LSP

B LSP

Seq. #	
A	4
C	2
E	4
F	3
G	∞

F LSP

Seq. #	
B	3
E	2
G	∞



Промене у мрежи (2)

- Отказ везе
  - Чворови на крајима покварене везе врше плављење са ажурираним LSP
- Отказ чвора
  - Сви суседи примете да веза не ради
  - Не знају да ли је у питању отказ везе или чвора
    - Али није ни битно, раде исто као и случају отказа везе
  - Покварени чвор не може да ажурира сопствени LSP
    - Али то није ни битно, он свакако не ради!

Промене у мрежи (3)

- Додавање чвора или везе
  - Суседни чворови ће детектовати и ажурирати своје LSP
  - Након плављења, убрзо ће сви знати нову топологију

DV – LS поређење

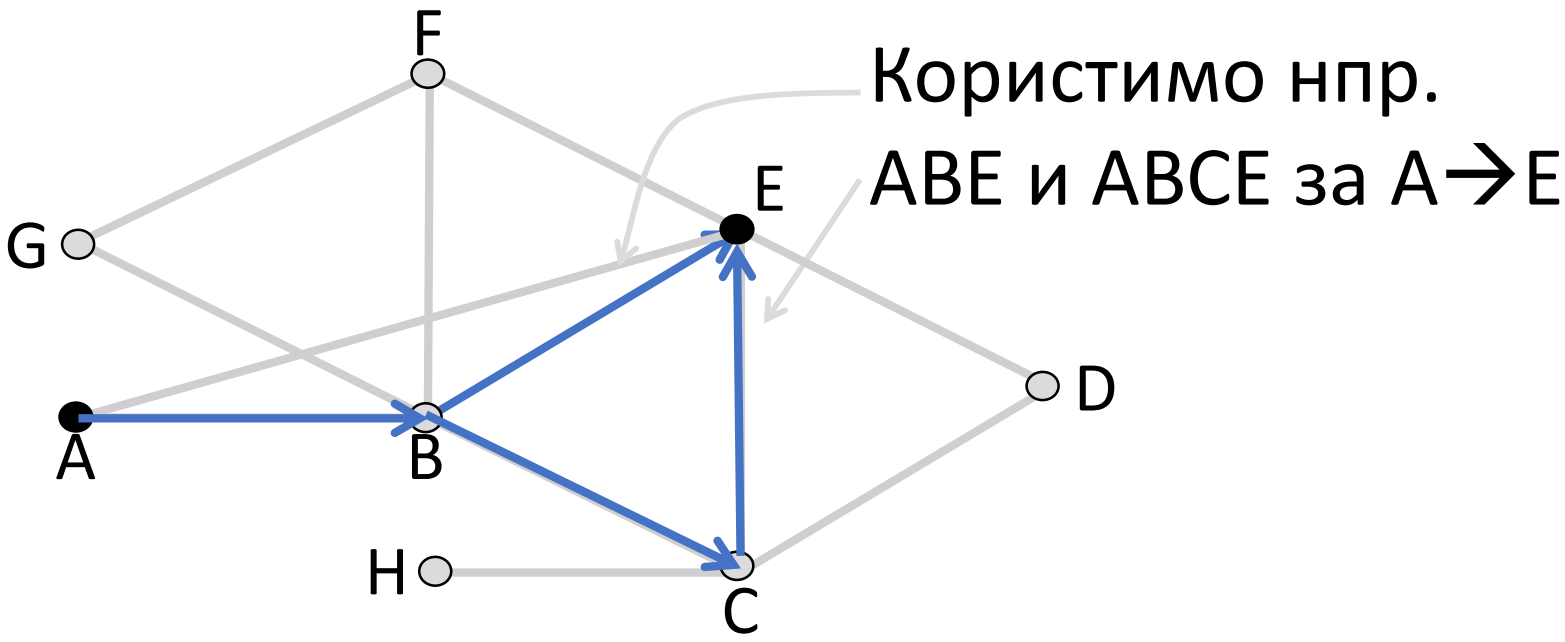
- Конвергенција:
  - Спора код DV, најбољи путеви на даљини k тек у k-тој размени
  - Брза код LS, јер је плављење брзо
- Скалабилност:
  - Одлична за DV, чворови рачунају решење потпроблема
  - Коректна код LS, сваки чвор рачуна решење целог проблема
    - Може се надоместити засад бољом опремом

Рутирање

Вишециљно рутирање са најкраћим путевима (Equal-cost multiple routing)

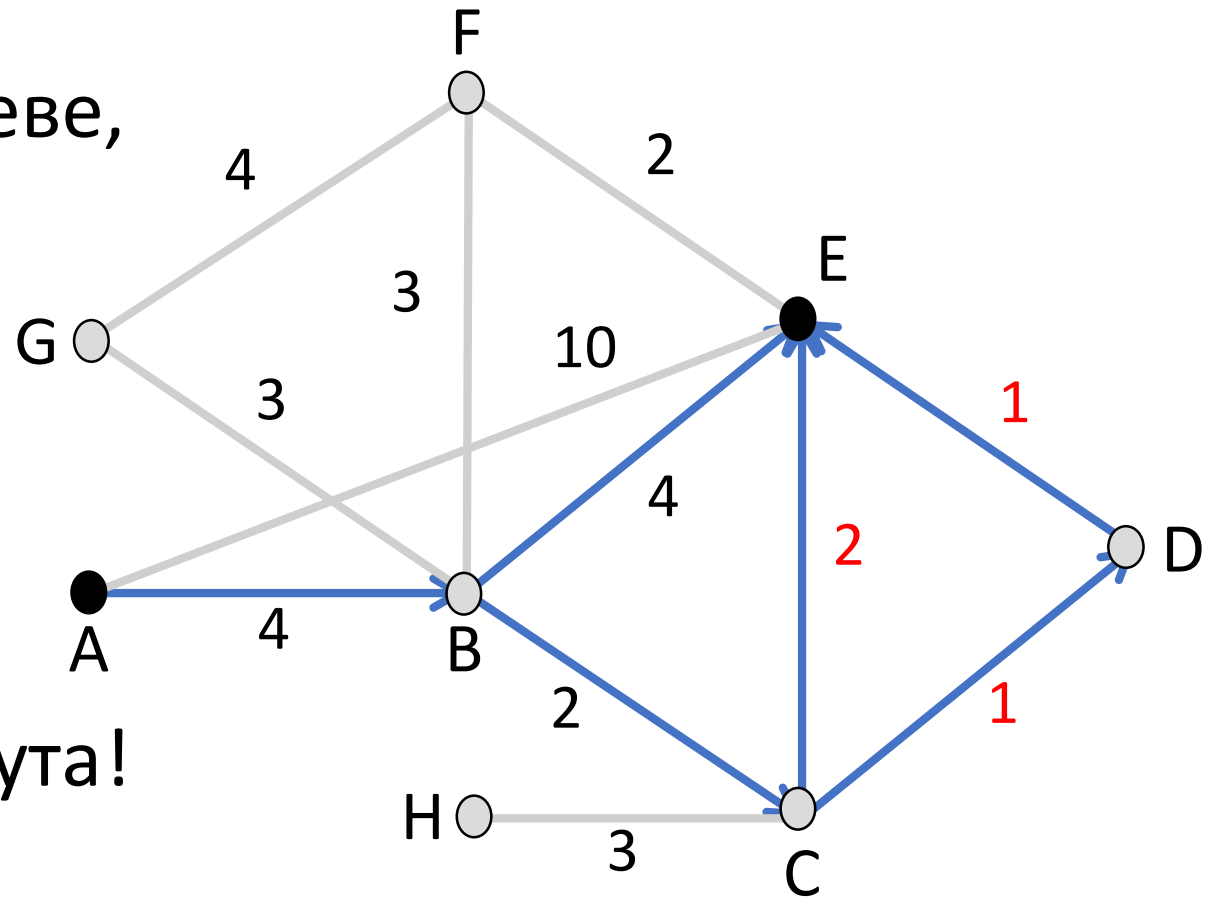
Тема

- Дозвољавамо вишеструке путеве
  - Редудантност побољшава поузданост
  - Али помаже и код смањења загушења



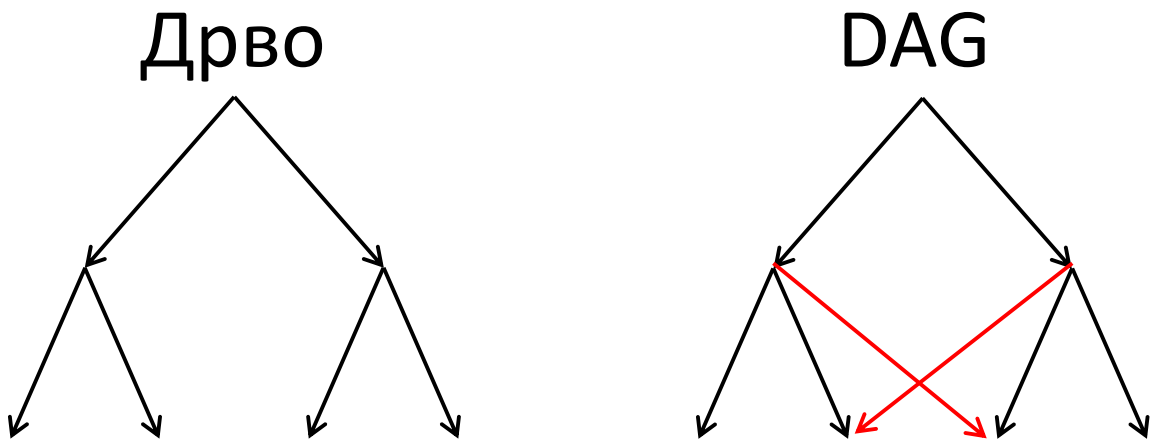
ECMP рутирање (Equal-cost multipath routing)

- Тип вишециљног рутирања
  - Задржавамо све најбоље путеве, а не само један
- Нпр. A→E
  - ABE = 4 + 4 = 8
  - ABCE = 4 + 2 + 2 = 8
  - ABCDE = 4 + 2 + 1 + 1 = 8
  - Користимо сва три најбоља пута!



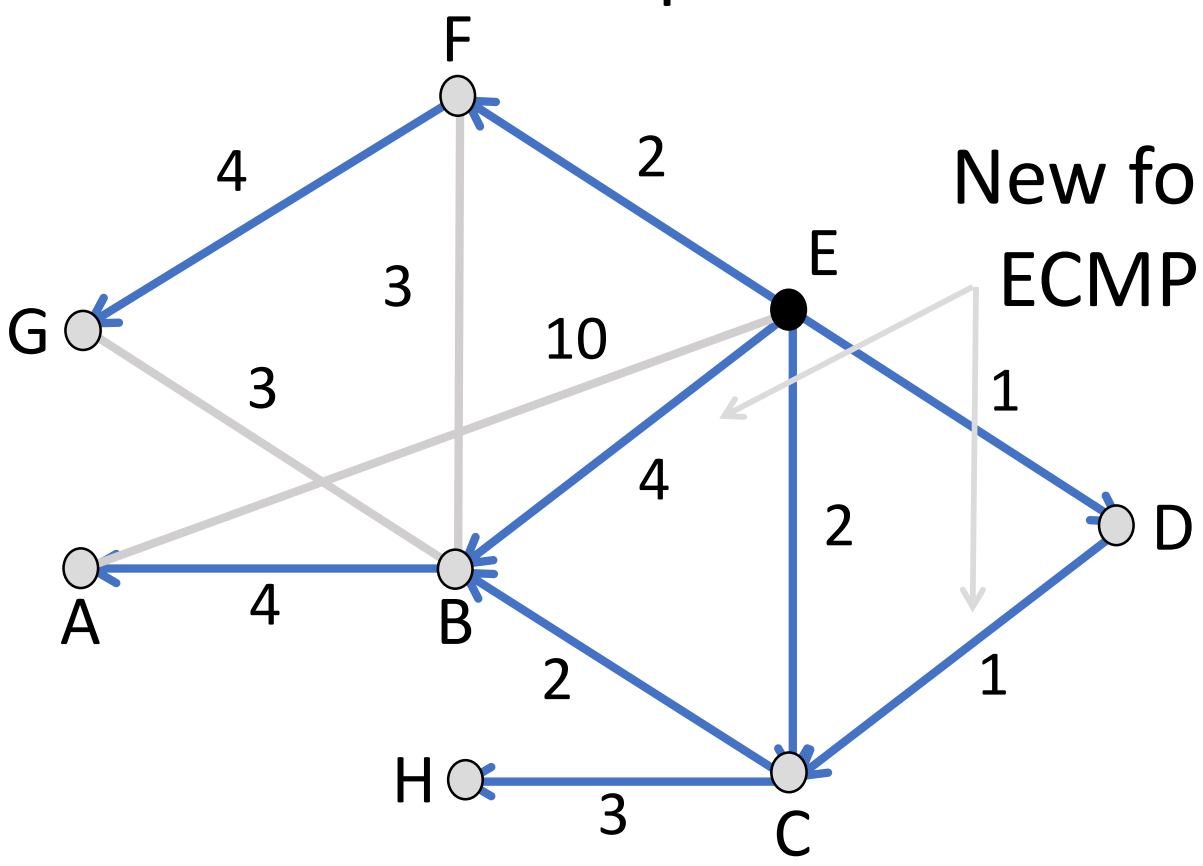
Топологија

- Код ECMP рутирања, најкраћи путеви од задатог чвора не формирају дрво као код једноциљног
- Формира се тзв. усмерени ациклички граф (DAG – directed acyclic graph)
- Дијкстрин алгоритам, само убацујемо све најбоље



ECMP

DAG за чвор E



New for ECMP

→

Табела за E

X	Y је листа
A	B, C, D
B	B, C, D
C	C, D
D	D
E	--
F	F
G	F
H	C, D

ECMP прослеђивање

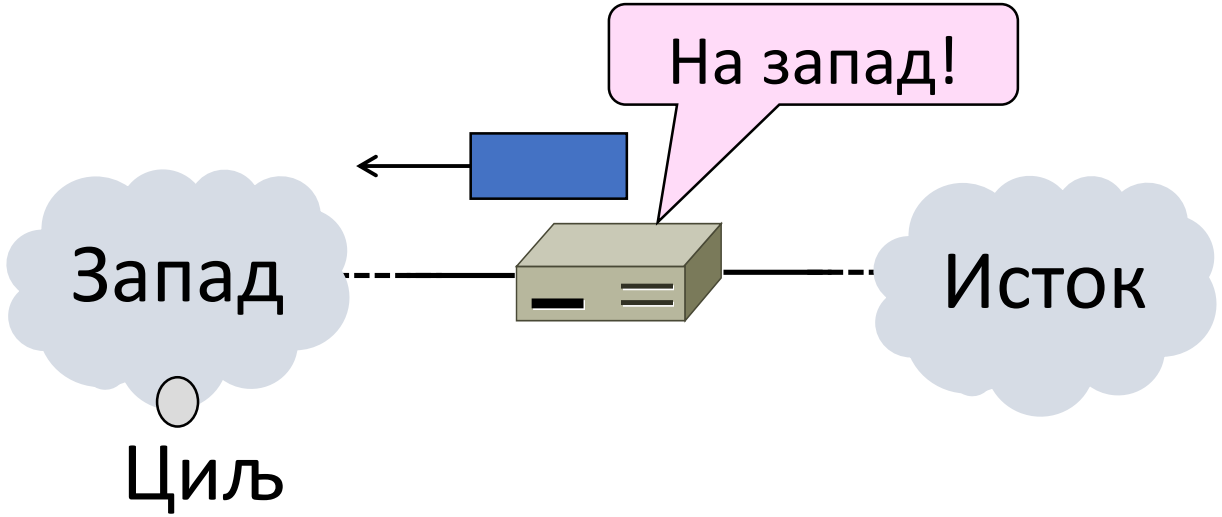
- Након што је одређена табела прослеђивања како се прослеђује:
  - Насумичним одабиром једне од путања?
    - Балансира оптерећење
    - Пакети могу имати различито кашњење, лоше за пренос у реалном времену
  - Боље је фиксирати избор на основу извора и циља
    - Дакле, контролисано насумично (насумично, али исто на нивоу извора и циља)
    - Оптерећење се и даље балансира на тај начин
    - Елиминише се употреба различитих путања за пакете из истих логичких целина, нпр. датотека, видеа, итд.

Рутирање

Хијерархијско рутирање

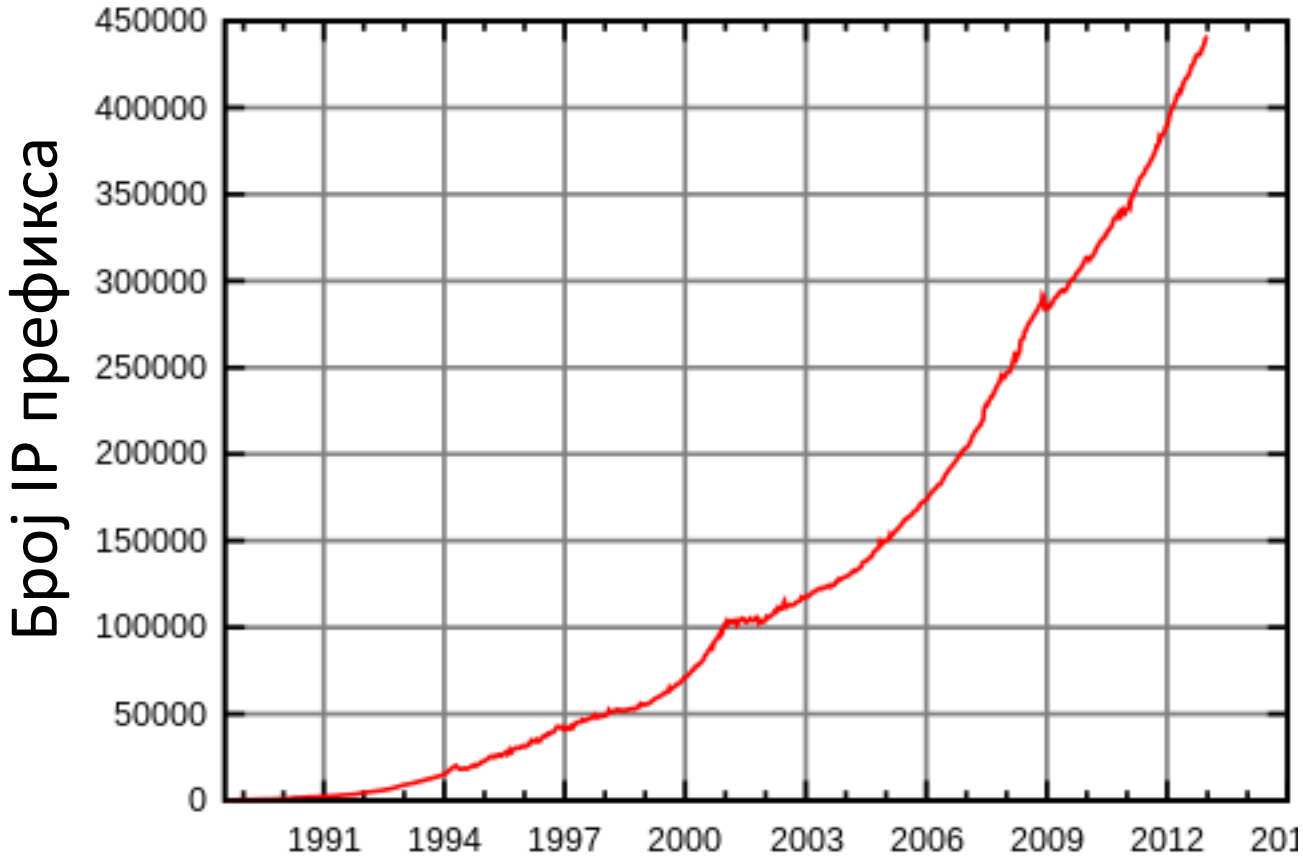
Тема

- Рутирање по сваком појединачном (чвору) рутеру се не скалира добро!
  - Зато чворове можемо да групишемо у регионе
  - Потом унутар региона вршимо специфичније рутирање



Раст броја рутера (IP префикса) на Интернету

- Нема смисла правити унос у табели прослеђивања за сваки рутер у свету!
- Напомена: овде се мисли на рутере који се појављују у табелама рутирања глобално, постоје и прикривени рутери за подмреже (последњи слајдови)

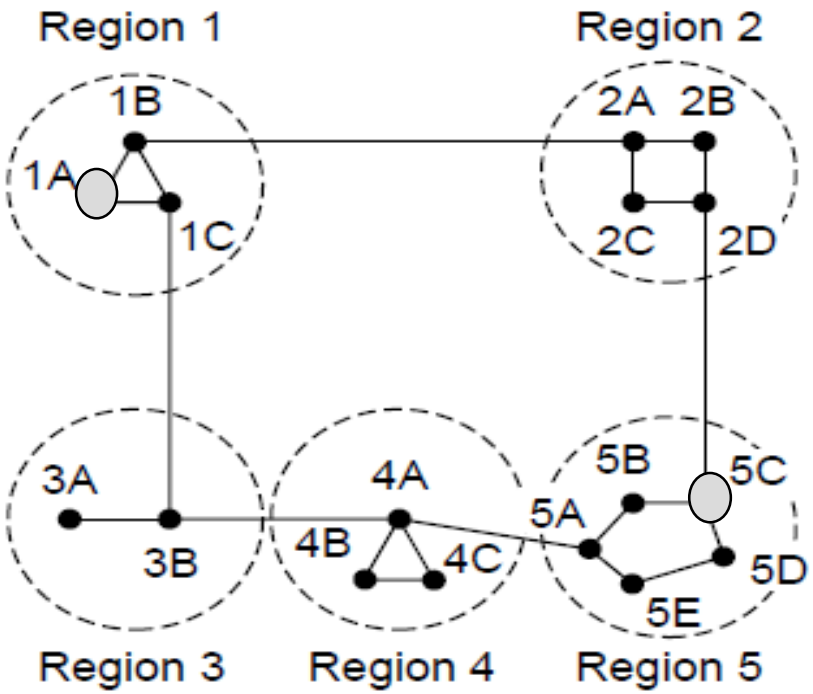


Source: By Mro (Own work), CC-BY-SA-3.0 , via Wikimedia Commons

Хијерархијско рутирање

- Уводимо веће јединице рутирања, нпр. регион неке државе
  - За те веће јединице вршимо рутирање,
    посматрајућих их као да су појединачни чворови
- Могу да постоје и подрегиони унутар региона
  - И за њих примењујемо исти приступ, нпр. ISP мрежа
- Коначну путању добијамо тако што:
  - Најпре рутирамо пакет у најмањем региону, нпр. у оквиру ISP мреже
  - Потом по изласку из ISP мреже, користи се рутирање на нивоу државе
  - Потом се поново улази у нпр. циљну ISP мрежи, итд.

Хијерархијско рутирање (2)



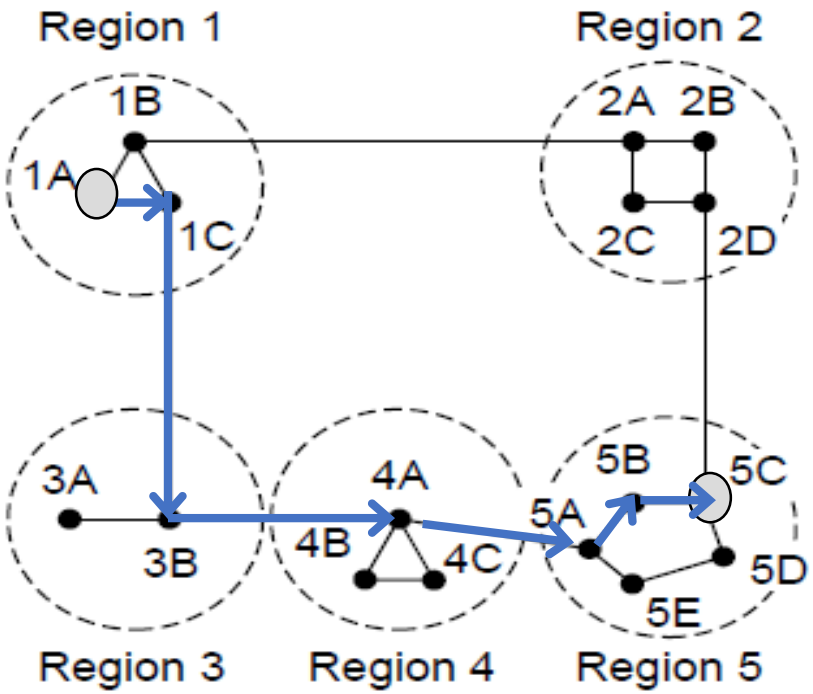
Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Хијеархијско рутирање (3)



Full table for 1A

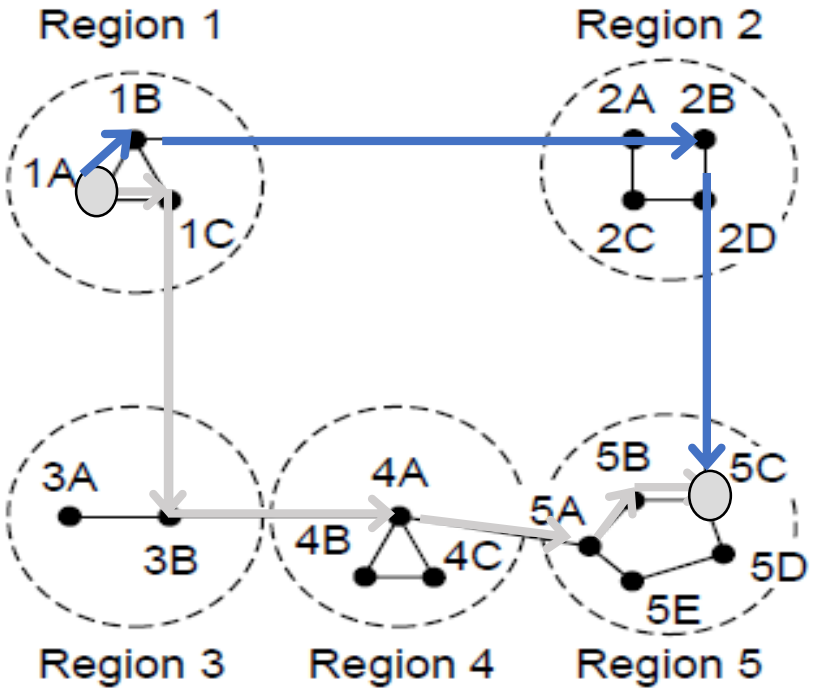
Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Хијерархијско рутирање (4)

- Не даје увек најкраћу путању, али је то компромис зарад ефикасности



Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

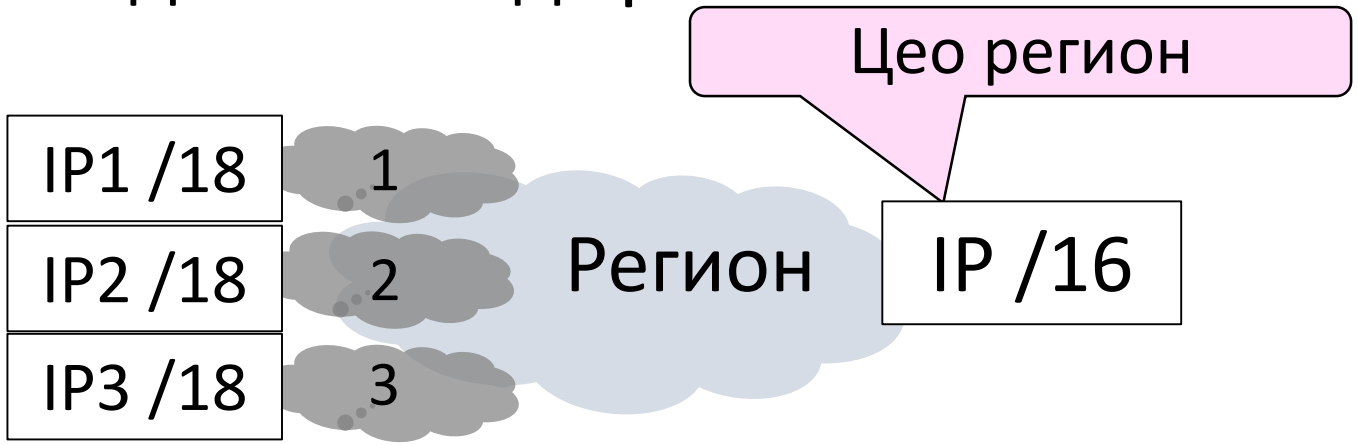
1C је најбоља генерално ако шаљемо у регион 5. Али конкретно, не мора бити најбоља за све чворове унутар њега, нпр. за 5C

Рутирање

Подела и сажимање IP префикса

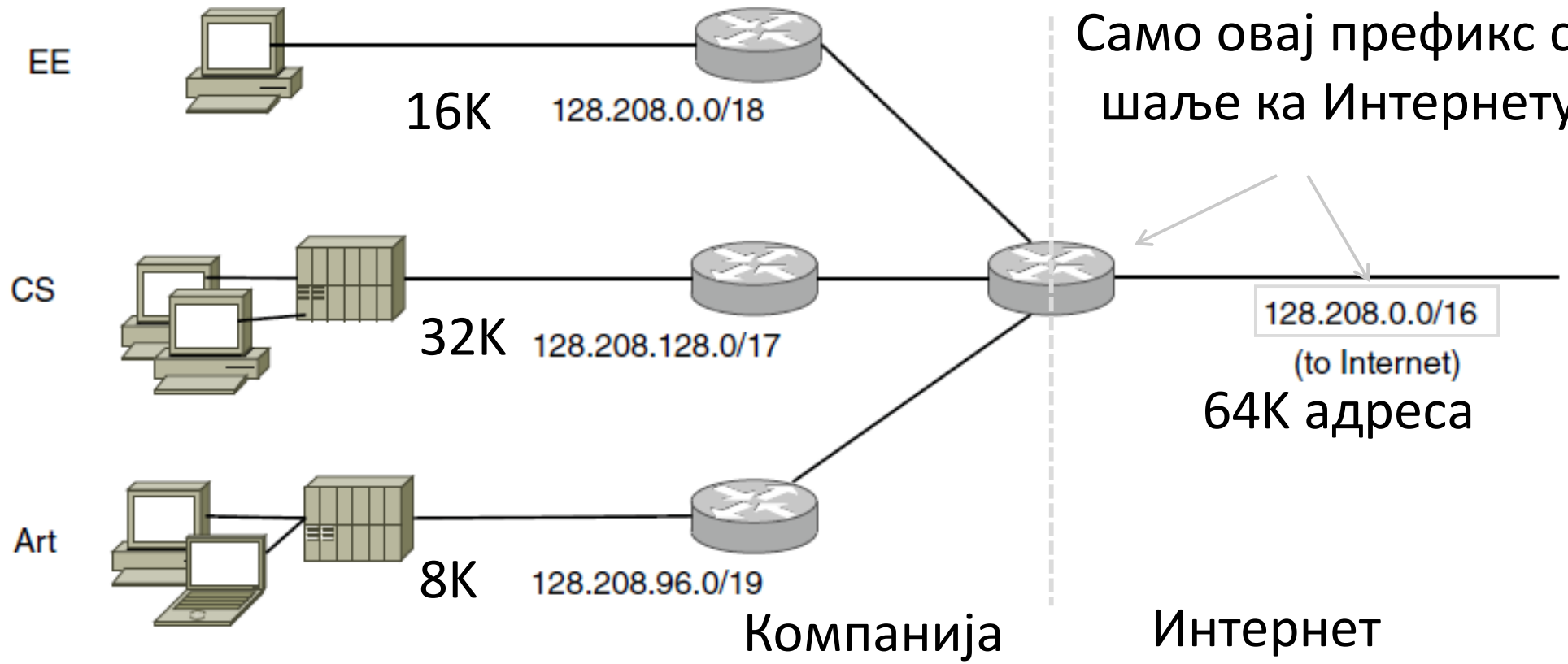
Тема

- Рутирање по регионима је и даље тешко за израчунавање
- Уводимо додатни подниво у виду IP префикса
  - Подела на подмреже и њихово касније сажимање



Подмреже

- Интерна подела IP префикса нпр. унутар компаније
  - IP префикси (EE, CS, Art) рутера нису видљиви споља



Сажимање

- Могуће је и спољне сажимање неповезаних установа
  - Ово обично раде ISP