Kompiuterių tinklai - Tinklo sluoksnis

Šešta paskaita (5.1 - 5.2 skyriai),
http://computernetworks5e.org/chap05.html

lekt. Vytautas Jančauskas

Tinklo sluoksnis

Tinklo sluoksnis

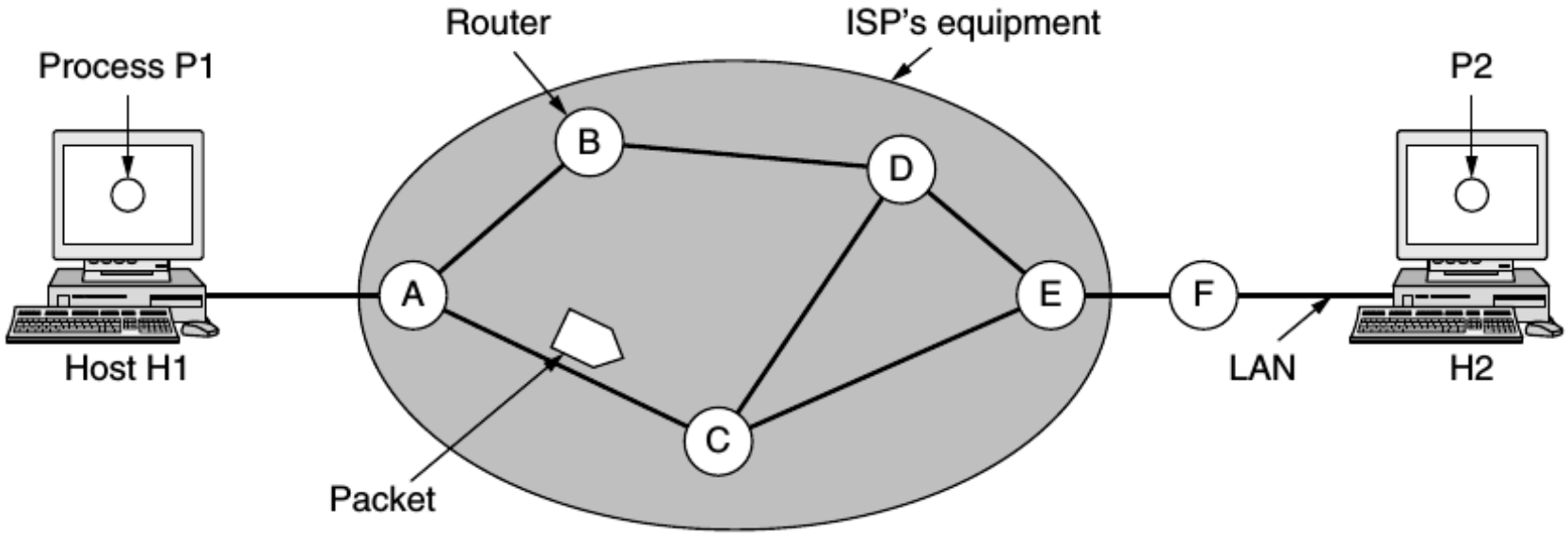


Figure 5-1. The environment of the network layer protocols.

Servisai teikiami transporto sluoksniui

Servisai kuriami atsižvelgiant į šiuos reikalavimus

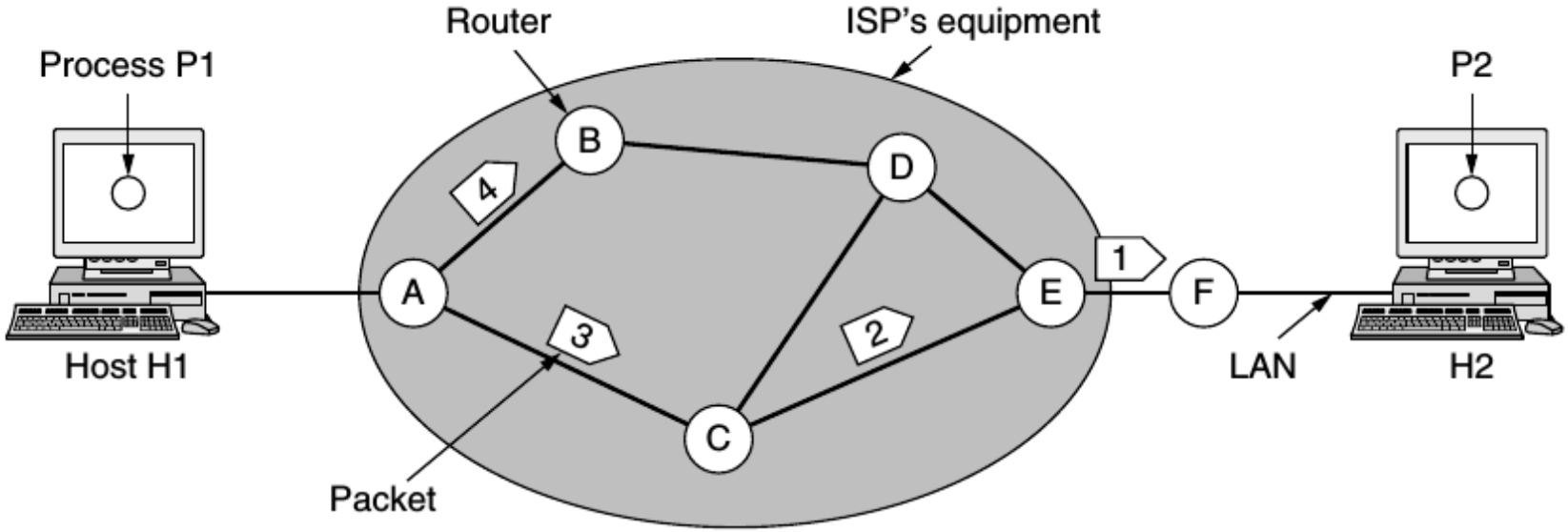
1. Jie neturėtų priklausyti nuo konkrečių maršrutizatorių technologijų.
2. Transporto sluoksnis neturėtų žinoti ir jam neturėtų rūpėti kiek, kokių ir kaip sujungtų maršrutizatorių yra tinkle.
3. Tinklo adresas pateikiamas transporto sluoksniui turėtų būti vienodai sudaromas ir nedviprasmis nepaisant LAN ir WAN ribų.

Yra dvi pagrindinės nuomonės dėl to, kokio tipo paslaugas turi teikti tinklo sluoksnis:

1. Tinklo sluoksnio maršrutizatoriai turėtų tiesiog perdavinėti gaunamus paketus, o duomenų perdavimo kokybę reikia užtikrinti kitais būdais.
2. Tinklas turėtų teikti patikimas paslaugas su sujungimu ir užtikrinti duomenų perdavimo kokybę.

Paslaugų be sujungimo įgyvendinimas

- Paketai (datagramos) paduodami tinklui ir keliauja nepriklausomai vienas nuo kito.
- Didesni duomenų kiekiai padalinami į paketus ir perduodami maršrutizatoriui kuris yra prieinamas siuntėjui.
- Maršrutizatorius dažniausiai turi maršrutizavimo lentelę. Joje įrašyta, kuriam tiesiogiai prijungtam kitam maršrutizatoriui perduoti paketą jeigu pakete nurodytas vienas ar kitas adresatas.
- Maršrutizavimo lentelių turinys gali keistis, pavyzdžiui atjungus kurį nors kitą maršrutizatorių arba dinamiškai atsižvelgiant į sąlygas tinkle.



A's table (initially)

Dest.	Line
A	–
B	B
C	C
D	B
E	C
F	C

A's table (later)

A	–
B	B
C	C
D	B
E	B
F	B

C's table

A	A
B	A
C	–
D	E
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	–
F	F

Figure 5-2. Routing within a datagram network.

Paslaugų su sujungimu įgyvendinimas

- Tinklai užtikrinantys paslaugas su sujungimu vadinami **Virtual Circuit** tinklais.
- Duomenys nepradedami siųsti, kol nesukuriamas sujungimas tarp siuntėjo ir gavėjo. Kitaip tariant kelias nėra parenkamas dinamiškai, bet galioja viso sujungimo metu.
- Maršrutizatoriai saugo duomenis apie sujungimus. Kitaip tariant kurio sujungimo duomenis nukreipti į kurį maršrutizatorių.
- Tokie tinklai geriau tinka dideliems duomenų kiekiams perduoti, kadangi sumažina klaidų ar prarastų paketų tikimybę.

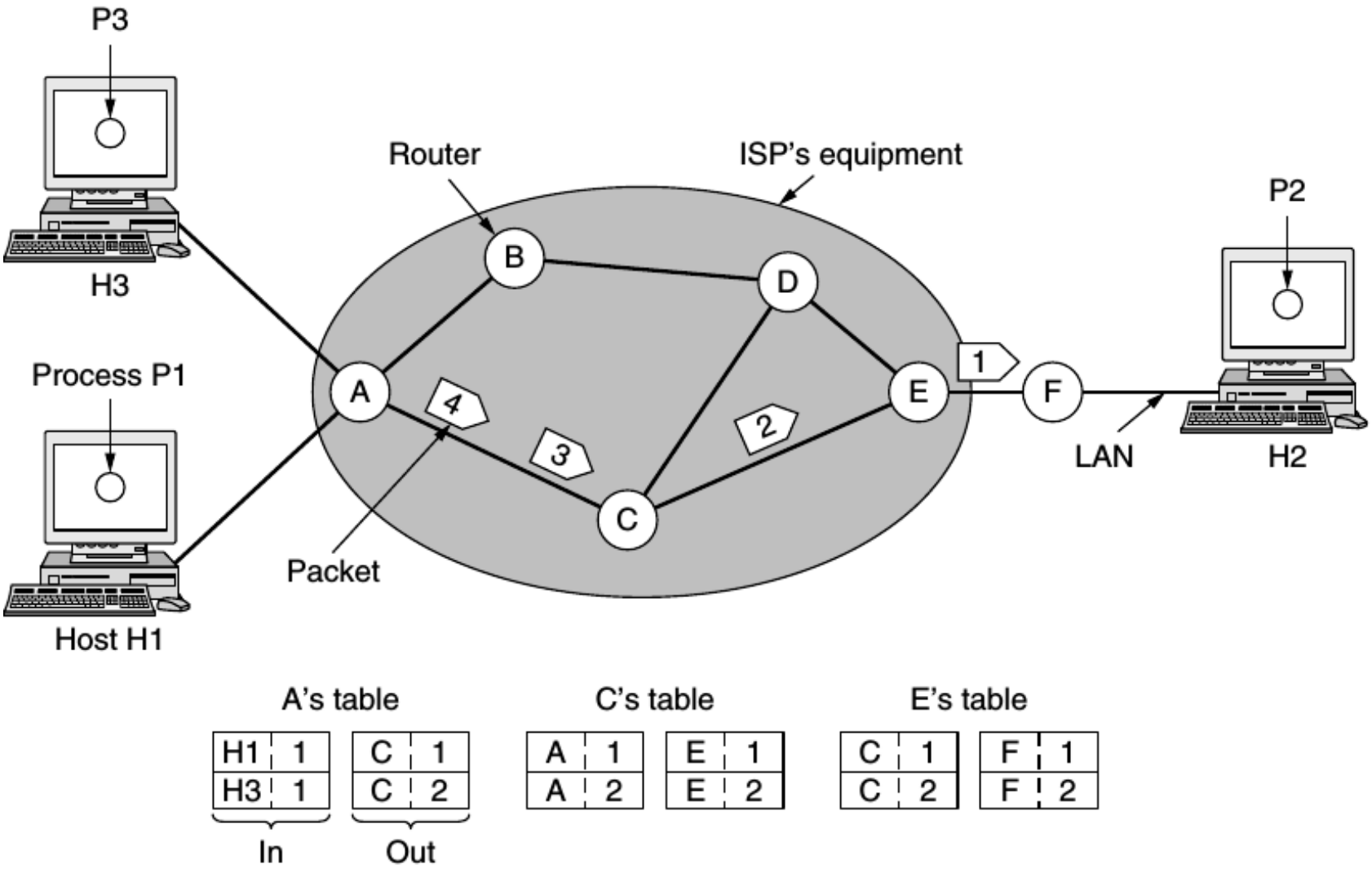


Figure 5-3. Routing within a virtual-circuit network.

Datagramų ir **Virtual Circuit** tinklų palyginimas

- Tiek datagramų tiek **virtual circuit** tinklai turi savo šalininkų ir priešininkų.
- **Virtual circuit** tinkluose užtrunka laiko ir resursų kol sukuriamas sujungimas, tačiau jį turint paketų perdavimas yra efektyvus.
- Datagramų tinkluose nereikia jokio išankstinio pasiruošimo siunčiant duomenis, tačiau pats perdavimas yra mažiau efektyvus.
- Adresai datagramose yra gerokai ilgesni, tačiau jie turi globalią prasmę. Jeigu paketas nedidelis tai gali turėti įtakos duomenų perdavimo efektyvumui.
- **Virtual circuit** geriau užtikrina paslaugos kokybę, nes resursai sujungimui gali būti alokuoti iš anksto.
- Datagramų tinklai atsparesni sutrikimams tinkle, pavyzdžiui jeigu atjungiamas maršrutizatorius.

Issue	Datagram network	Virtual-circuit network
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Figure 5-4. Comparison of datagram and virtual-circuit networks.

Maršrutizavimo algoritmai

Maršrutizavimo algoritmai

- Maršrutizavimo algoritmų tikslas yra parinkti geriausią (pagal kažkokius kriterijus) kelią iš siuntėjo iki gavėjo.
- Kuriant tokius algoritmus remiamasi šiais kriterijais: korektiškumas, paprastumas, atsparumas, stabilumas, sąžiningumas ir efektyvumas.
- Maršrutizavimo algoritmai gali būti skirstomi į tokias dvi pagrindines grupes - adaptyvūs ir neadaptyvūs.
- Neadaptyvūs algoritmai neatsižvelia į tinklo topologijos ar apkrovimo pakitimus. Maršrutai nustatomi iš anksto.
- Adaptyvūs algoritmai sudaro maršrutizavimo lenteles dinamiškai atsižvelgiant į pasikeitimus tinkle.

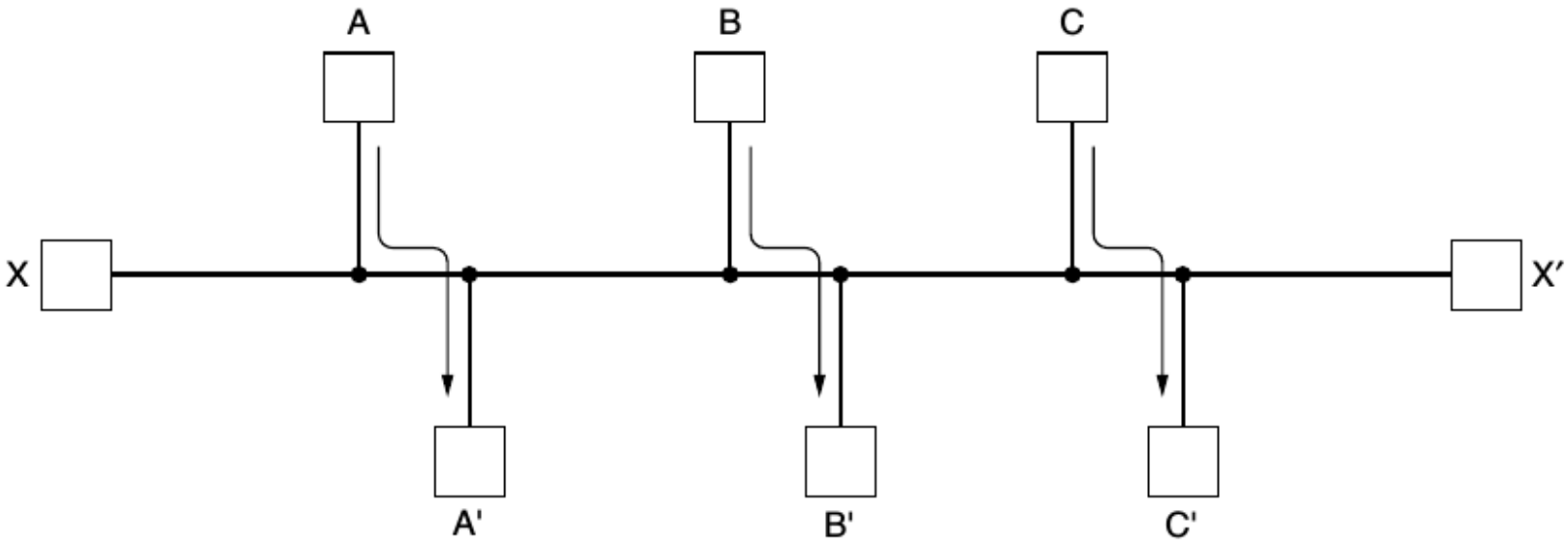


Figure 5-5. Network with a conflict between fairness and efficiency.

Optimalumo principas

- Optimalumo principas - jeigu maršrutizatorius J įeina į optimalų kelią iš maršrutizatoriaus I į maršrutizatorių K, tai tokiu atveju optimalus kelias iš J į K yra to paties kelio dalis.
- Šio principo pasekoje visų optimalių kelių aibė į kažkokį maršrutizatorių sudaro medį kurio šaknis yra tas maršrutizatorius. Tokį medį vadinsime **sink tree**.
- Maršrutizavimo algoritmų tikslas yra kiekvienam maršrutizatoriui surasti tokį medį.
- Tokių medžių gali būti ir keli, jeigu egzistuoja keli vienodo "gerumo" keliai.

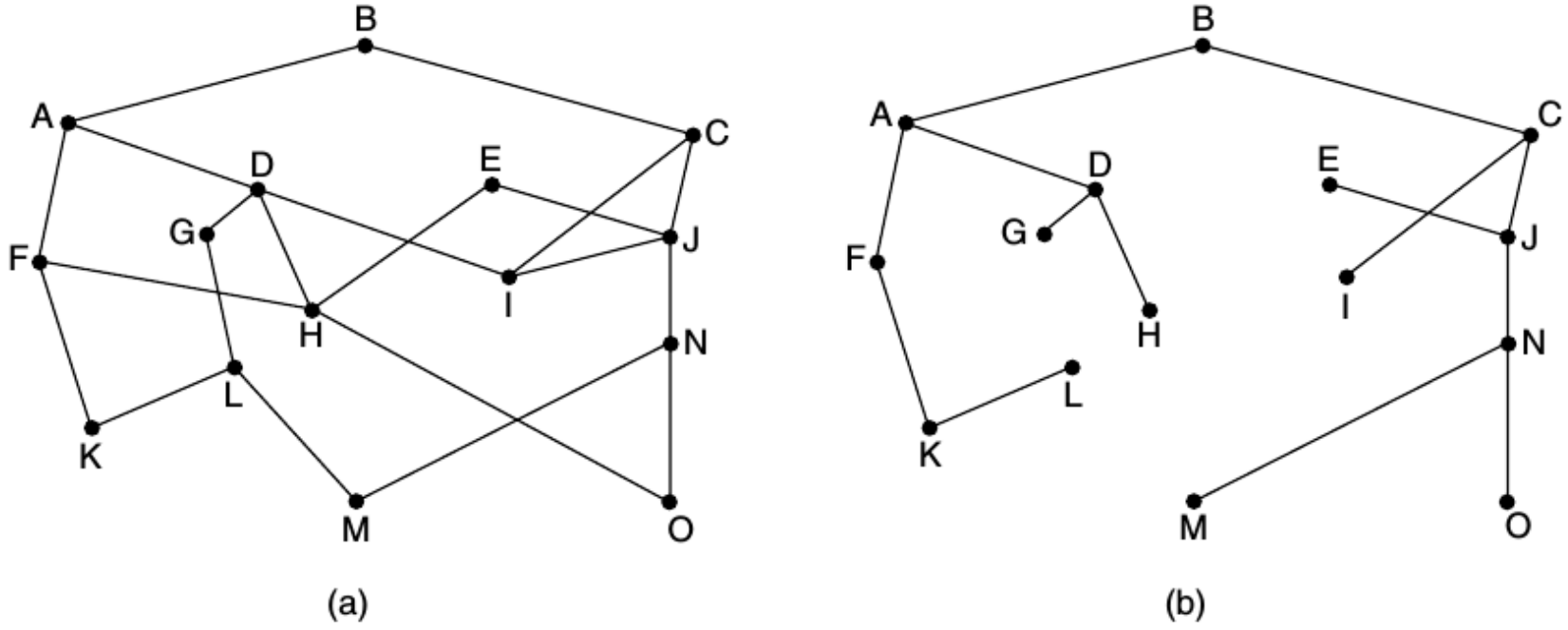


Figure 5-6. (a) A network. (b) A sink tree for router *B*.

Trumpiausio kelio algoritmas

- Trumpiausio kelio algoritmų tikslas yra rasti trumpiausią kelią (seką maršrutizatorių kurie tiesiogiai sujungti su maršrutizatoriumi einančiu sekoje iš kairės ir dešinės) pagal kažkokį kriterijų.
- Kriterijumi gali būti tiesiog maršrutizatorių skaičius kelyje, geografinis atstumas tarp maršrutizatorių, vėlinimas ar kiti kriterijai.
- Egzistuoja keleta tam skirtų algoritmų. Mes panagrinėsime pasiūlytą Dijktra (1959), kadangi jis yra populiarus, be to jo pagrindu dažnai kuriami kiti algoritmai.

```
 1  function Dijkstra(Graph, source):
 2
 3      dist[source] ← 0                          // Distance from source to source
 4      prev[source] ← undefined                  // Previous node in optimal path initialization
 5
 6      for each vertex v in Graph:  // Initialization
 7          if v ≠ source            // Where v has not yet been removed from Q (unvisited nodes)
 8              dist[v] ← infinity               // Unknown distance function from source to v
 9              prev[v] ← undefined              // Previous node in optimal path from source
10          end if
11          add v to Q                      // All nodes initially in Q (unvisited nodes)
12      end for
13
14      while Q is not empty:
15          u ← vertex in Q with min dist[u]  // Source node in first case
16          remove u from Q
17
18          for each neighbor v of u:           // where v is still in Q.
19              alt ← dist[u] + length(u, v)
20              if alt < dist[v]:               // A shorter path to v has been found
21                  dist[v] ← alt
22                  prev[v] ← u
23              end if
24          end for
25      end while
26
27      return dist[], prev[]
28
29  end function
```

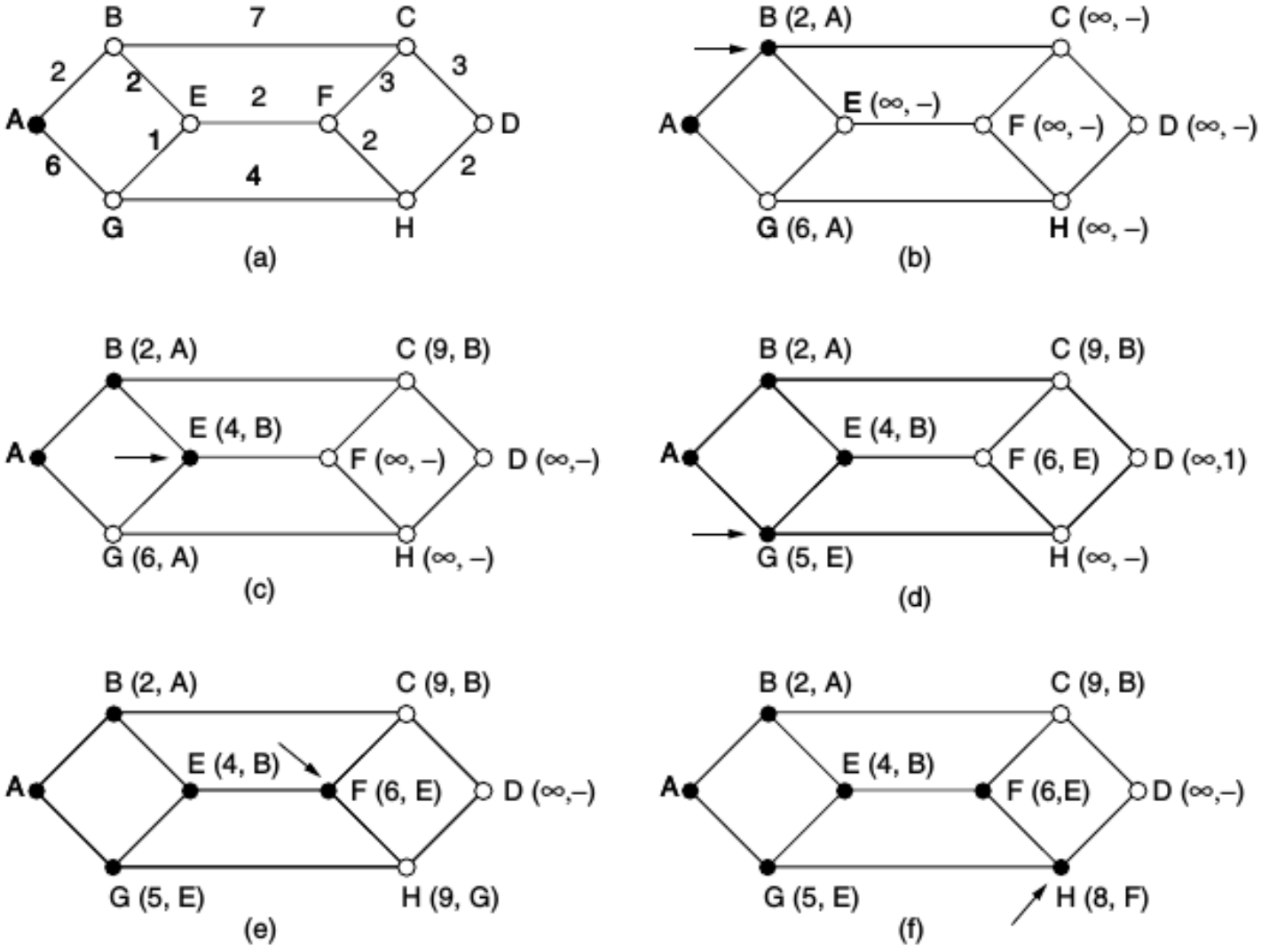


Figure 5-7. The first six steps used in computing the shortest path from *A* to *D*. The arrows indicate the working node.

Flooding (I)

- **Flooding** atveju kiekvienas maršrutizatorius kiekvieną gautą paketą išsiunčia visiems prie jo prijungtiems maršrutizatoriams išskyrus tą iš kurio jį gavo.
- Jeigu proceso nesustabdyti tai paketai keliautų amžinai. Tam dažnai naudojamas peršokimų skaičiavimas, kai kiekvieną kartą persiuntus paketą sumažinamas jo headeryje esantis skaitliukas.
- Kitas būdas yra neleisti maršrutizatoriams antrą kartą siųsti paketo jeigu jis jau buvo gautas ir išsiųstas. Tai galima padaryti jeigu, pavyzdžiui, siuntėjas numeruoja išsiunčiamus paketus.
- **Flooding** labai apkrauna tinklą tačiau turi dvi geras savybes - paketas pasiekia visus prie tinklo prijungtus maršrutizatorius (ko gali ir nereikėti jei gavėjas tik vienas) be to šis būdas itin atsparus tinklo gedimams.

Flooding (II)

- Naudojant šį būda nereikia jokių išankstinių kelių skaičiavimų.
- Šis metodas gali būti naudojamas įvertinti naujus maršrutizavimo algoritmus - jeigu jie veikia ne žymiai geriau nei **flooding** jie ne daug ko verti.
- Šis būdas visada ras patį trumpiausią kelią.
- Joks kitas algoritmas (ignoruojant pačio **flooding** proceso sukurtą tinklo apkrovimą) neduoda geresnių vėlinimo reikšmių.

Atstumų vektoriaus maršrutizavimas (I)

- Kiekvienas maršrutizatoriu turi lentelę (vektorių) kurioje saugomas mažiausias žinomas atstumas iki kiekvieno kito maršrutizatoriaus tinkle ir per kurį tiesiogiai prijungtą maršrutizatorių ten patekti.
- Atsumas gali būti matuojamas pagal tai kiek maršrutizatorių reikės pereiti, kad pasiekti gavėją arba kitais būdais (pvz. pagal vėlinimo reikšmę).
- Naudojamas RIP tipo protokoluose.
- Dabar praktikoje pakeistas **Link state** maršrutizavimo algoritmais.

Atstumų vektoriaus maršrutizavimas (II)

- Kas kažkokį laiko tarpą kiekvienas maršrutizatorius išsiunčia savo lentelė visiems savo kaimynams.
- Tuo pačiu, kas kažkokį laiko tarpą, kiekvienas maršrutizatorius gauna savo kaimynų lenteles.
- Maršrutizatorius žino, koks atstumas yra tarp jo ir kaimynų.
- Pradžioje atstumas iki savęs yra nulis, iki kitų maršrutizatorių atstumas yra begalybė.
- Gavęs lentelę, norėdamas apskaičiuoti trumpiausią atstuma tarp savęs ir kito maršrutizatoriaus iš kaimynų lentelių išrenka tą atstumą, kuris yra mažiausias iki norimo maršrutizatoriaus, padaro tą kaimyną kitu žingsniu norint pasiekti tą maršrutizatorių ir prie atstumo prideda atstumą tarp savęs ir kaimyno.

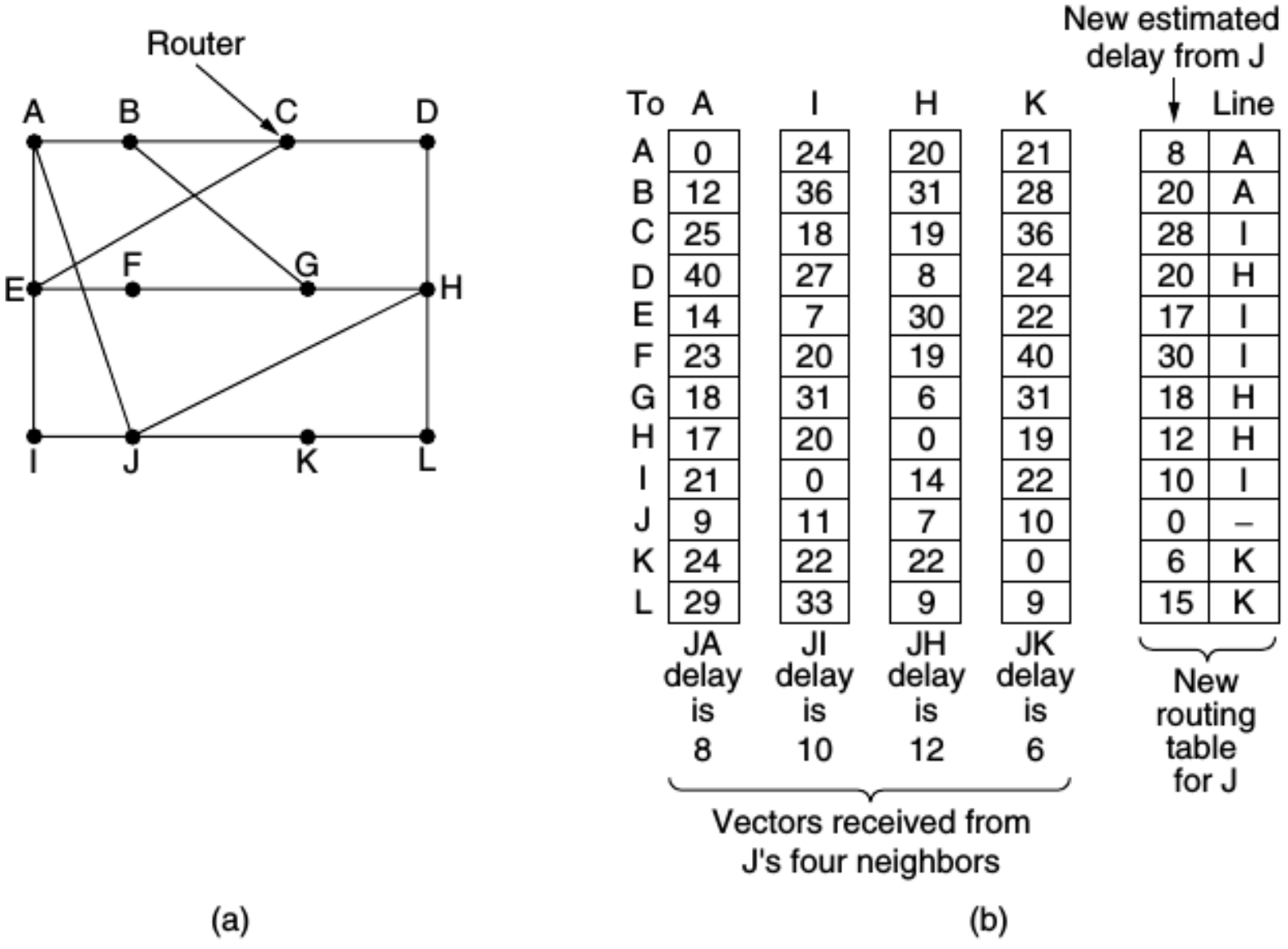


Figure 5-9. (a) A network. (b) Input from *A*, *I*, *H*, *K*, and the new routing table for *J*.

Atstumų vektoriaus maršrutizavimas - **Count-to-infinity** problema

- Trumpiausių kelių nusistovėjimas vadinamas tinklo konvergavimu.
- Tinkluose naudojančiuose atstumų vektorių maršrutizavimą prijungus naują maršrutizatorių atstumai iki jo atnaujinami greitai.
- Jeigu maršrutizatoriu išimamas iš tinklo informacija apie tai sklinda labai lėtai.
- Atnaujinant lenteles kiekvienas maršrutizatorius išsirinks trumpiausią kelią iki dingusio maršrutizatoriaus tarp savo kaimynų ir pridės prie jo atstumą iki to kaimyno.
- Taip, pamažu, trumpiausi keliai iki dingusio maršrutizatoriaus augs iki begalybės.
- Yra sukurta euristinių metodų padedančių kovoti su šia problema, tačiau praktikoje jie veikia prastai.

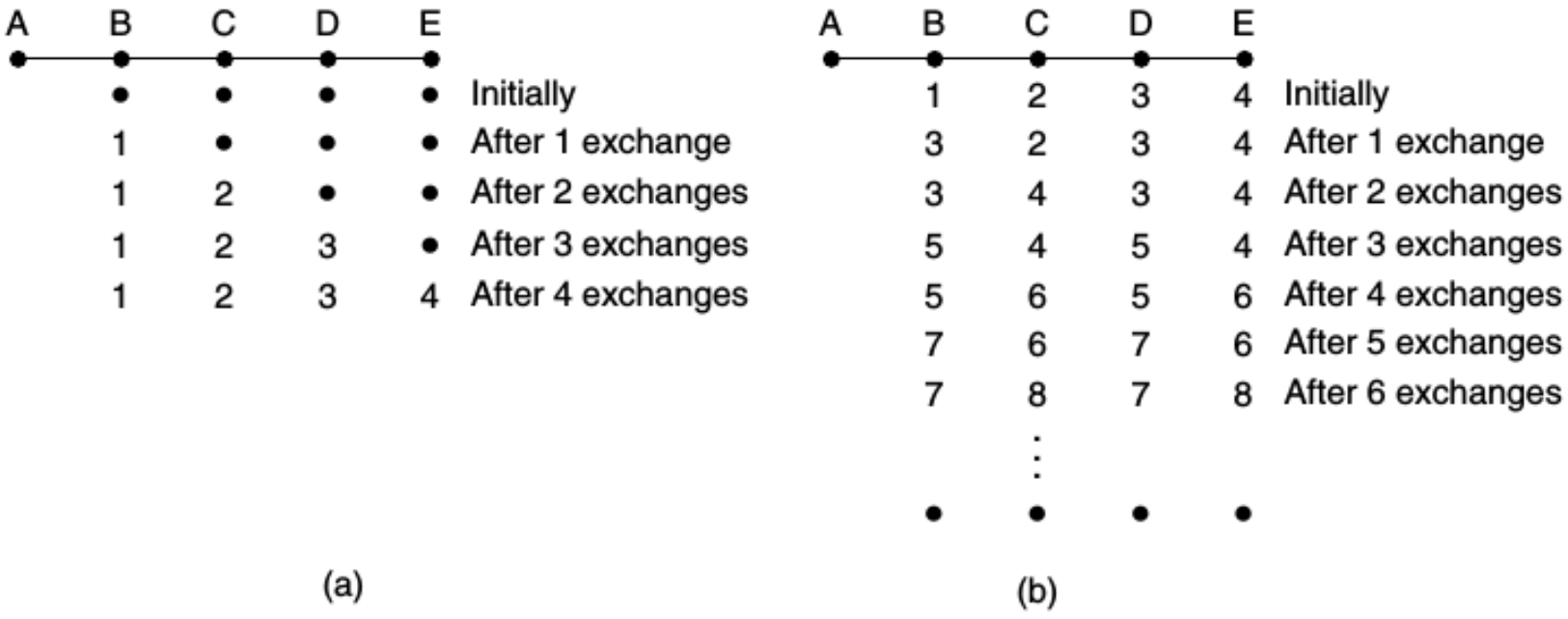


Figure 5-10. The count-to-infinity problem.

Link state maršrutizavimas

Šis algoritmas praktikoje pakeitė atstumų vektoriaus algoritmus. Kiekvienas maršrutizatorius atlieka tokius veiksmus:

1. Nustatomi kaimynai ir jų adresai.
2. Nustatomi atstumai ar kitų metrikų reikšmės iki kaimynų.
3. Sukuriamas paketas su šia informacija.
4. Ši informacija išsiunčiama visiems maršrutizatoriams ir gaunama iš visų kitų maršrutizatorių.
5. Lokaliai apskaičiuojamas trumpiausias kelias iki kiekvieno kito maršrutizatoriaus.

Tai reiškia, kad kiekvienas maršrutizatorius informuojamas apie viso tinklo topologiją. Tada jis gali naudoti pvz. Dijkstra algoritmą rasti trumpiausiam keliui.

Link state - kaimynų radimas

- Įsijungus maršrutizatoriui pirmas jo darbas yra nustatyti, kas yra jo kaimynai.
- Tam jis išsiunčia specialų HELLO paketą kiekvienam prie jo prijungtam maršrutizatoriui.
- Problema yra sunkesnė jeigu maršrutizatoriai sujungti per **broadcast** LAN.
- Tokiu atveju sukuriamas “dirbtinis” maršrutizatorius kuris simbolizuoja tinklo **broadcast** dalį.

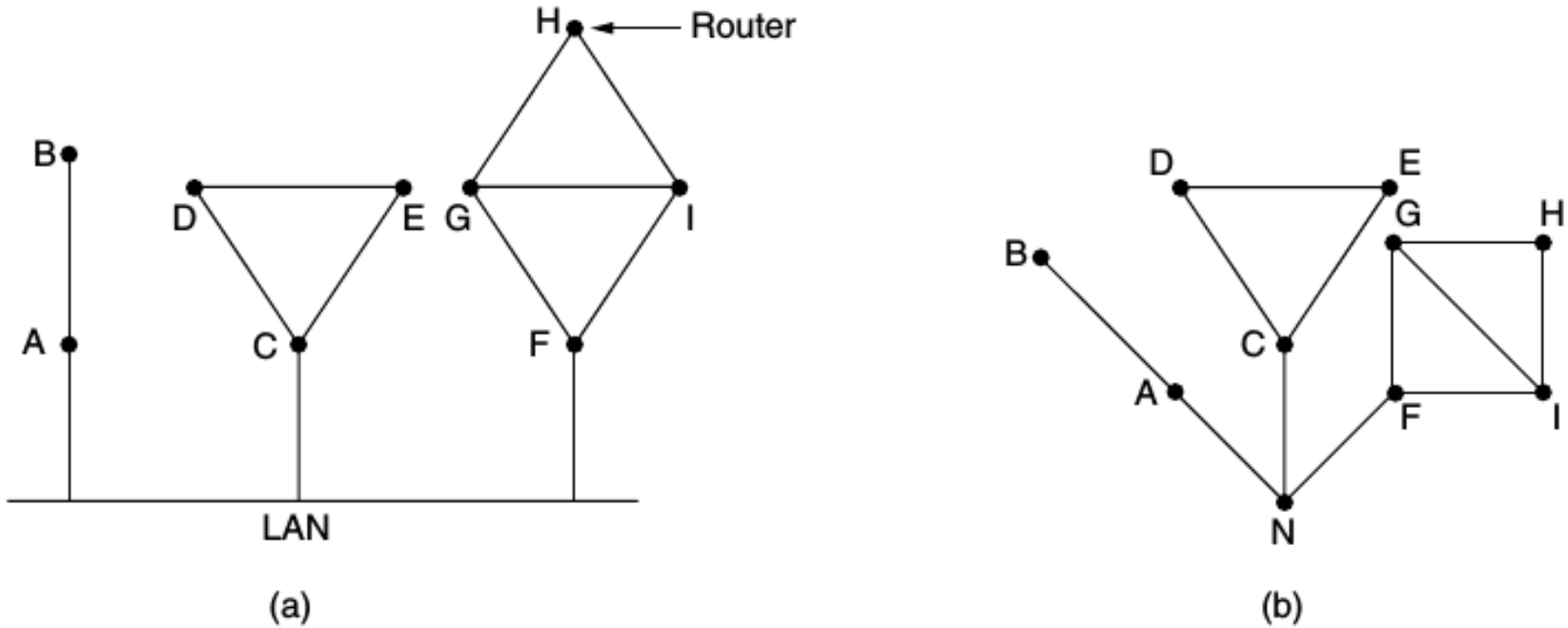


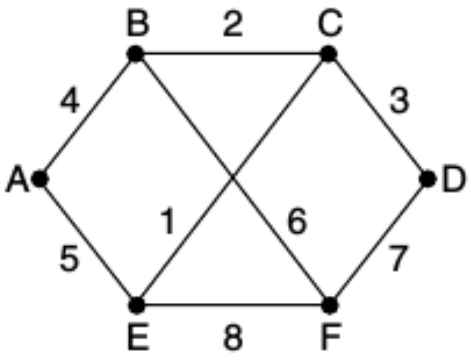
Figure 5-11. (a) Nine routers and a broadcast LAN. (b) A graph model of (a).

Link state - jungčių kainos nustatymas

- Paprastas būdas jungties kainai nustatyti yra padaryti ją atvirkščiai proporcingą kanalo pločiui. Pavyzdžiui 1-Gbps Ethernet gautų kainą 1 o 100-Mbps Ethernet gautų kainą 10.
- Jeigu tinklas pasklidęs plačioje geografiškai vietovėje gali tekti atsižvelgti į vėlinimą.
- Paprasčiausias būdas nustatyti vėlinimą yra išsiųsti ECHO paketą ir skaičiuoti kiek užtrunka per kiek laiko jis bus gautas atgal.

Link state - paketų sudarymas

- Kiekvienas maršrutizatorius sudaro savo paketą su duomenimis apie jo kaimynus.
- Paketą sudaro:
  1. Siuntėjo identifikatorius.
  2. Eilės numeris ir amžius.
  3. Kaimynų sąrašas ir kaina pasiekti tą kaimyną.
- Tokie paketai gali būti siunčiami periodiškai arba įvykus kokiam nors įvykiui, pavyzdžiui atsijungus ar prisijungus kaimynui.



(a)

Link State Packets

A		B		C		D		E		F	
Seq.		Seq.		Seq.		Seq.		Seq.		Seq.	
Age		Age		Age		Age		Age		Age	
B	4	A	4	B	2	C	3	A	5	B	6
E	5	C	2	D	3	F	7	C	1	D	7
		F	6	E	1			F	8	E	8

(b)

Figure 5-12. (a) A network. (b) The link state packets for this network.

Link state - paketų paskirstymas

- Visi maršrutizatoriai **link state** paketus turi gauti greitai ir patikimai.
- Pagrindinė idėja - naudoti **flooding** kad užtikrinti, jog visi maršrutizatoriai gaus visą informaciją apie tinklą.
- Gavus tokį paketą tikrinama ar jis anksčiau nebuvo gautas pagal eilės numerį ir jeigu nebuvo išsiunčiamas visiems maršrutizatoriams išskyrus tą iš kurio jis gautas.
- Jeigu eilės numeris bus kokiu nors būdu sugadintas (pvz. iš 4 pasidarys 60000) visi žemesnės eilės paketai bus ignoruojami.
- Kad išvengti auksčiau aprašytos problemos paketas pažymimas amžiumi kuris kas sekundę mažinamas ir kai jis lygus nuliui paketas nebeperduodamas.

Source	Seq.	Age	Send flags A	Send flags C	Send flags F	ACK flags A	ACK flags C	ACK flags F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

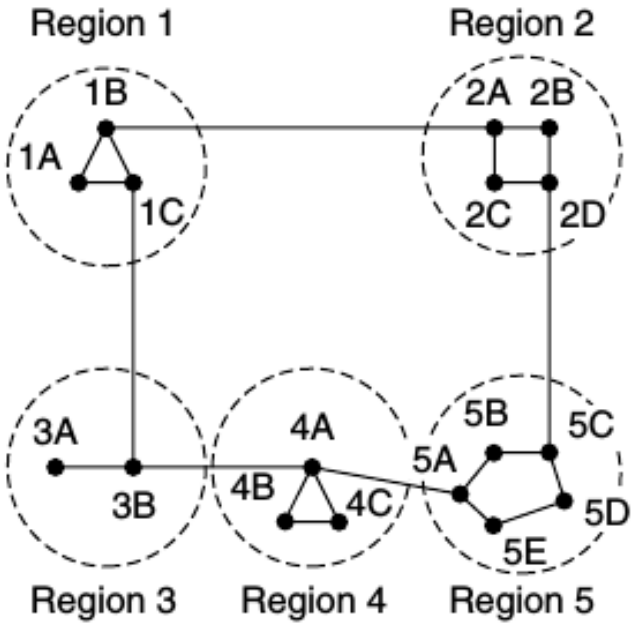
Figure 5-13. The packet buffer for router *B* in Fig. 5-12(a).

Link state - naujų kelių skaičiavimas

- Gavus informaciją apie visą tinklą maršrutizatorius gali suskaičiuoti trumpiausią kelią iki bet kurio maršrutizatoriaus.
- Akivaizdu, kad **link state** naudos žymiai daugiau atminties negu atstumų vektoriaus algoritmai.
- **Link state** naudojamas **IS-IS (Intermediate System - Intermediate System)** ir **OSPF (Open Shorted Path First)** protokoluose.
- Viena iš **link state** problemų yra ta, kad jeigu maršrutizatoriai praneš neteisingą informaciją apie tinklą arba sugadins specialius paketus likę maršrutizatoriai turės klaidingą tinklo topologiją.

Hierarchinis maršrutizavimas

- Jeigu maršrutizatorių skaičius tinkle yra labai didelis maršrutizavimo lentelės labai išauga ir gali atsirasti poreikis įvesti hierarchinį maršrutizavimą.
- Maršrutizatoriai dalinami į grupes, o grupės jungiamos tarpusavyje.
- Maršrutizatoriai žino tik savo grupės topologiją ir per kurį savo grupės maršrutizatorių pasiekti kitas grupes.
- Galimi keli hierarchijos lygiai.
- Nustatyta, kad optimalus hirearchijos lygių skaičius yra lnN, kur N yra maršrutizatorių skaičius.



(a)

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

(b)

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

(c)

Figure 5-14. Hierarchical routing.

Transliavimo maršrutizavimas (I)

- Norint transliuoti paketus daugeliui kompiuterių tinkle yra keli būdai.
- Vienas būdas yra tiesiog siuntėjui išsiųsti paketus kiekvienam adresatui atskirai - reikia žinoti kas tie adresatai ir apkraunamas tinklas.
- Kitas būdas saugoti paketo antraštėje visus gavėjus. Kiekvienas maršrutizatorius darys paketo kopijas kuriose bus tų adresų poaibis ir nukreips reikiama kruptimi. Vistiek reikia žinoti kas tie adresatai.
- Trečias būdas yra **flooding**. Jis šiuo atveju labai tinka tačiau yra dar efektyvesnis metodas.

Transliavimo maršrutizavimas (II)

- Gavęs **broadcast** paketą maršrutizatorius tikrina ar jis atėjo duomenų perdavimo kanalu kuris paprastai naudojamas siunčiant duomenis **link** paketo siuntėjo.
- Jeigu taip, vadinasi jo kelias greičiausiai buvo optimalus ir yra pirma paketo kopija atėjusi į šį konkretų maršrutizatorių.
- Maršrutizatorius tokiu atveju išsiųs šį paketą visais kanalais išskyrus tą iš kurio jis atėjo.
- Jeigu žinomas tinklą aprėpiantis medis šį algoritmą galima padaryti dar efektyvesniu, galima tiesiog siųsti broadcast paketus aprėpiančio medžio šakomis.

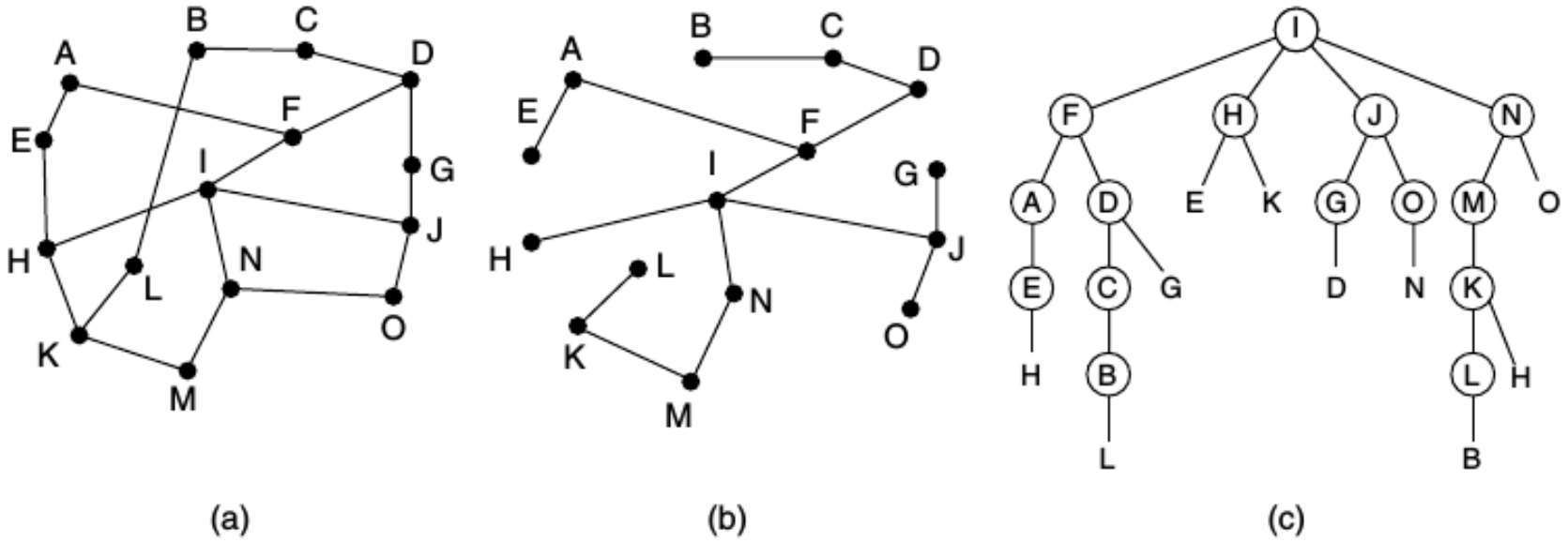


Figure 5-15. Reverse path forwarding. (a) A network. (b) A sink tree. (c) The tree built by reverse path forwarding.

Multicast maršrutizavimas (I)

- Multicast maršrutizavimas naudingas kai paketus reikia siųsti gavėjų grupėmis kurios yra didelės skaitiškai tačiau mažos viso tinklo kontekste (kai netinka transliavimo maršrutizavimas).
- Naudojami panašūs metodai kaip ir transliavimo atveju.
- Paprasčiausias būdas yra sukonstruoti tinklo aprėpiantį medį ir iš jo pašalinti tas šakas kurios neveda į grupei priklausančius maršrutizatorius.
- Naudojant **link state** maršrutizavimą tokį medį sukonstruoti yra nesunku, nes kiekvienas maršrutizatorius žino viso tinklo topologiją. **MOSPF (Multicast OSPF)** yra tokio protokolo pavyzdys.
- Naudojant atstumų vektorių algoritmus tai daroma specialiomis žinutėmis informuojančiomis, kad maršrutizatorius nesuinteresuotas multicast paketais. **DVMRP (Distance Vector Multicast Routing Protocol)** yra tokio protokolo pavyzdys.

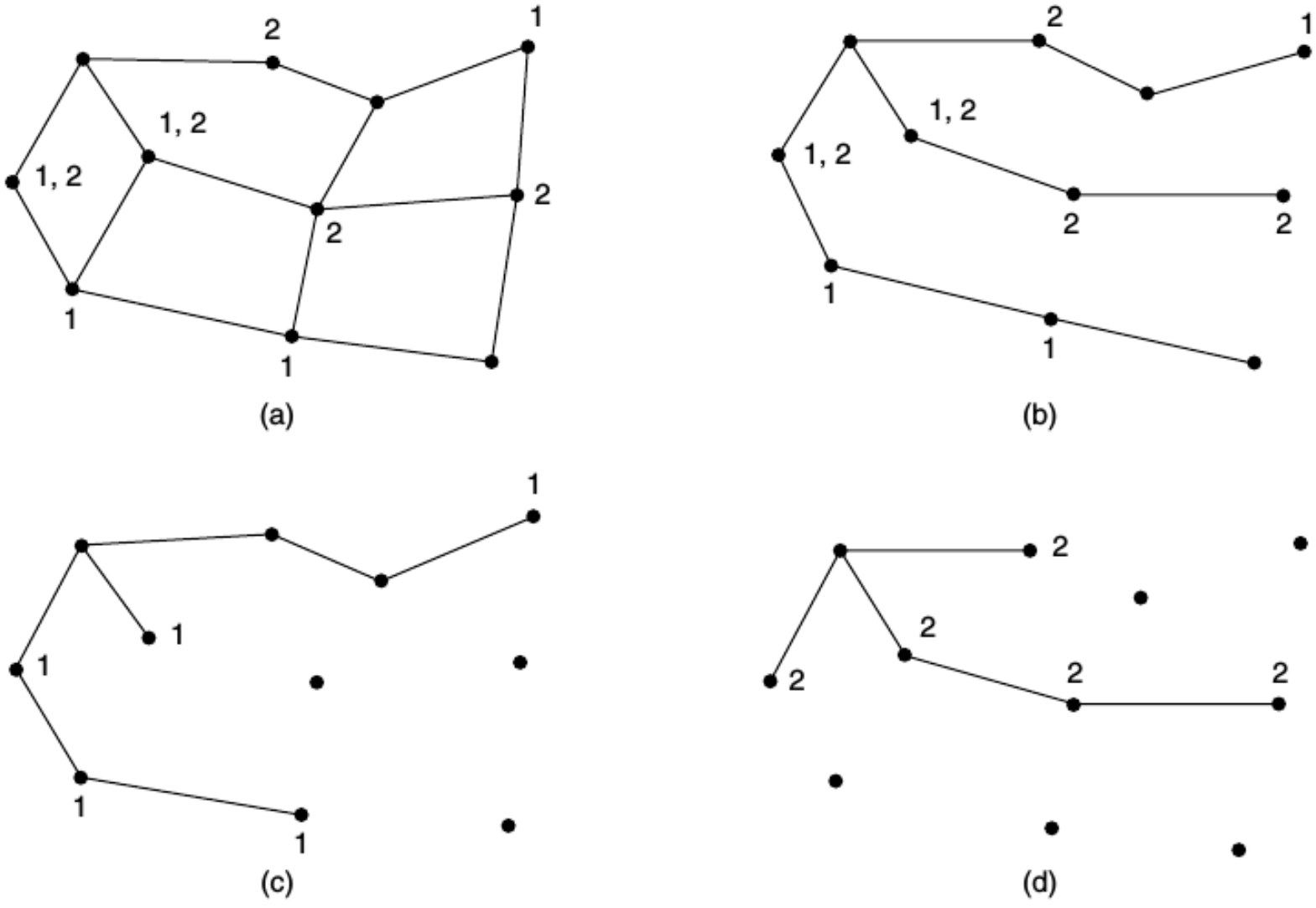


Figure 5-16. (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

Multicast maršrutizavimas (II)

- Nenorint skaičiuoti atskiro aprėpiančio medžio kiekvienam grupės maršrutizatoriui naudojami **core-based** medžiai.
- Tokiu atveju skaičiuojamas tik vienas medis grupei.
- Susitariama dėl vieno pagrindinio mazgo (**core** arba **rendezvous** taško) ir siunčiami jam pranešimai. Keliai kuriuos nueina šie pranešimai ir sudaro grupės aprėpiantį medį.
- Kai noriama išsiųsti paketą grupei jis siunčiamas **core** mazgui ir jis jį persiunčia visiems grupę sudarantiems maršrutizatoriams.
- Toks metodas naudojamas **PIM (Protocol Independent Multicast)** protokole.

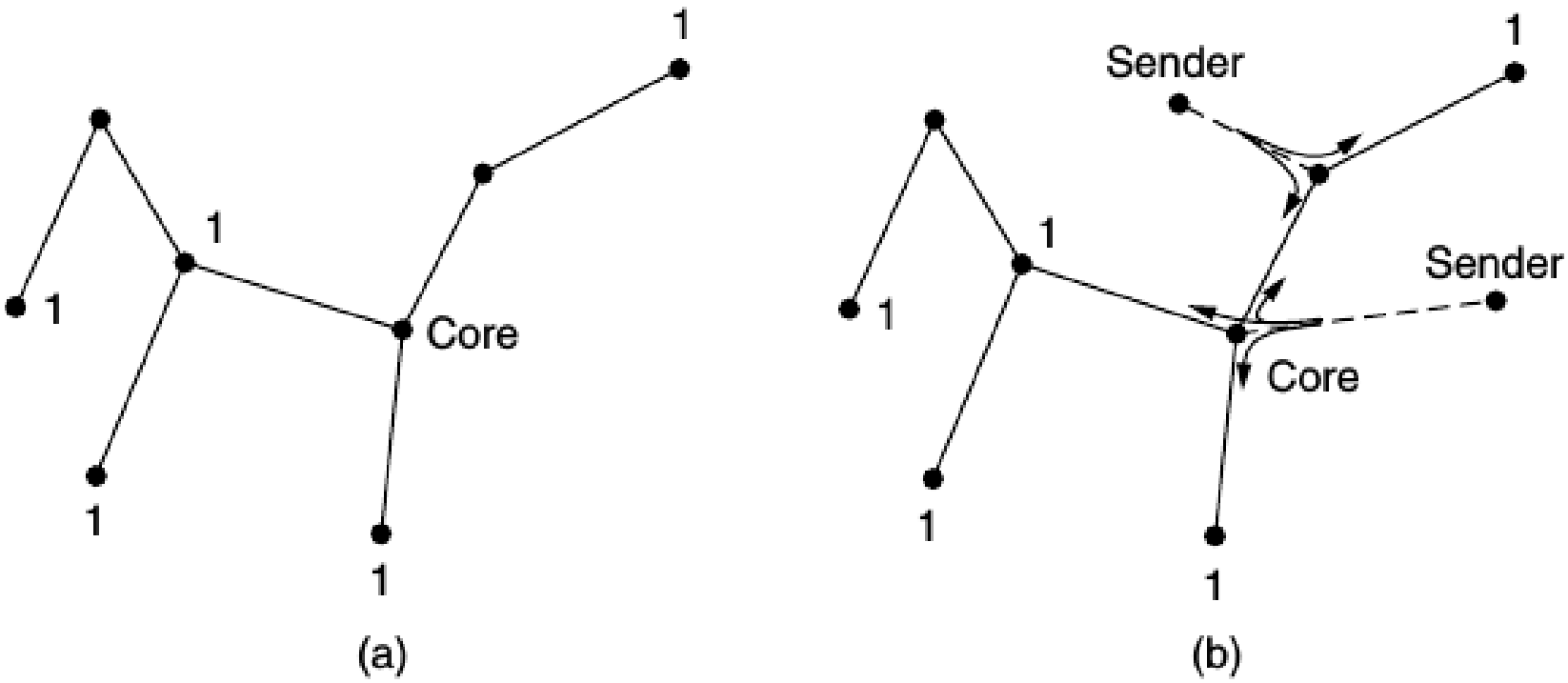


Figure 5-17. (a) Core-based tree for group 1. (b) Sending to group 1.

Anycast maršrutizavimas

- Anycast tai toks duomenų perdavimo būdas kai paketas perduodamas artimiausiam grupės nariui.
- Tai naudojama, kai užtenka gauti reikiamą informaciją, nesvarbu iš ko. Tokio tipo paslaugos naudojamos, pavyzdžiui, DNS.
- Galima naudoti standartinius maršrutizavimo algoritmus.

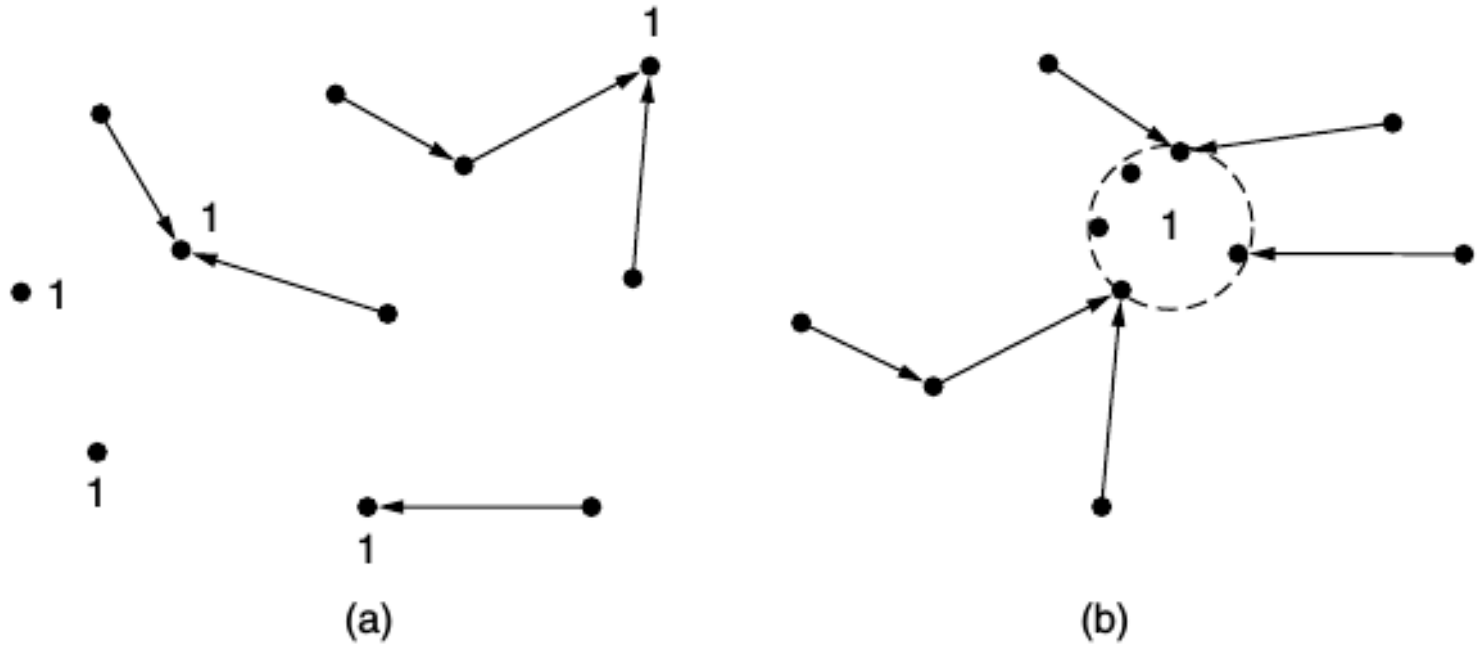


Figure 5-18. (a) Anycast routes to group 1. (b) Topology seen by the routing protocol.

Maršrutizavimas mobiliems kompiuteriams (I)

- Maršrutizuojant įrenginiams kurie yra mobilūs (arba judantys arba bent jau laikas nuo laiko kažkur perkeliami) iškyla papildomų problemų.
- Pagrindinis uždavinys yra kaip juos surasti.
- Jei tai nešiojamas kompiuteris, jį prijungus naujoje vietoje jis tiesiog gauną naują tinklo adresą. Niekas tinkle nežino, kad tai tas pats kompiuteris.

Maršrutizavimas mobiliems kompiuteriams (II)

- Kitas variantas yra naudoti **namų agentą - home agent**. Kompiuteris praneša namų agentui kur jis yra. Bendravimas su įrenginiu vyksta per namų agentą, jis perduoda paketus gavėjui.
- Namų agentas visus paketus siuntus mobiliam gavėjui priima ir perduoda jam. Siuntėjui atrodo, kad jis bendrauja tiesiogiai su gavėju. Toks metodas vadinamas tuneliavimu - **tunneling**.
- Vėliau mobilus kompiuteris gali keistis duomenimis su siuntėju tiesiogiai.
- Taip galima pasiekti, kad tinklas rastų konkretų kompiuterį net jam keičiant vietą.

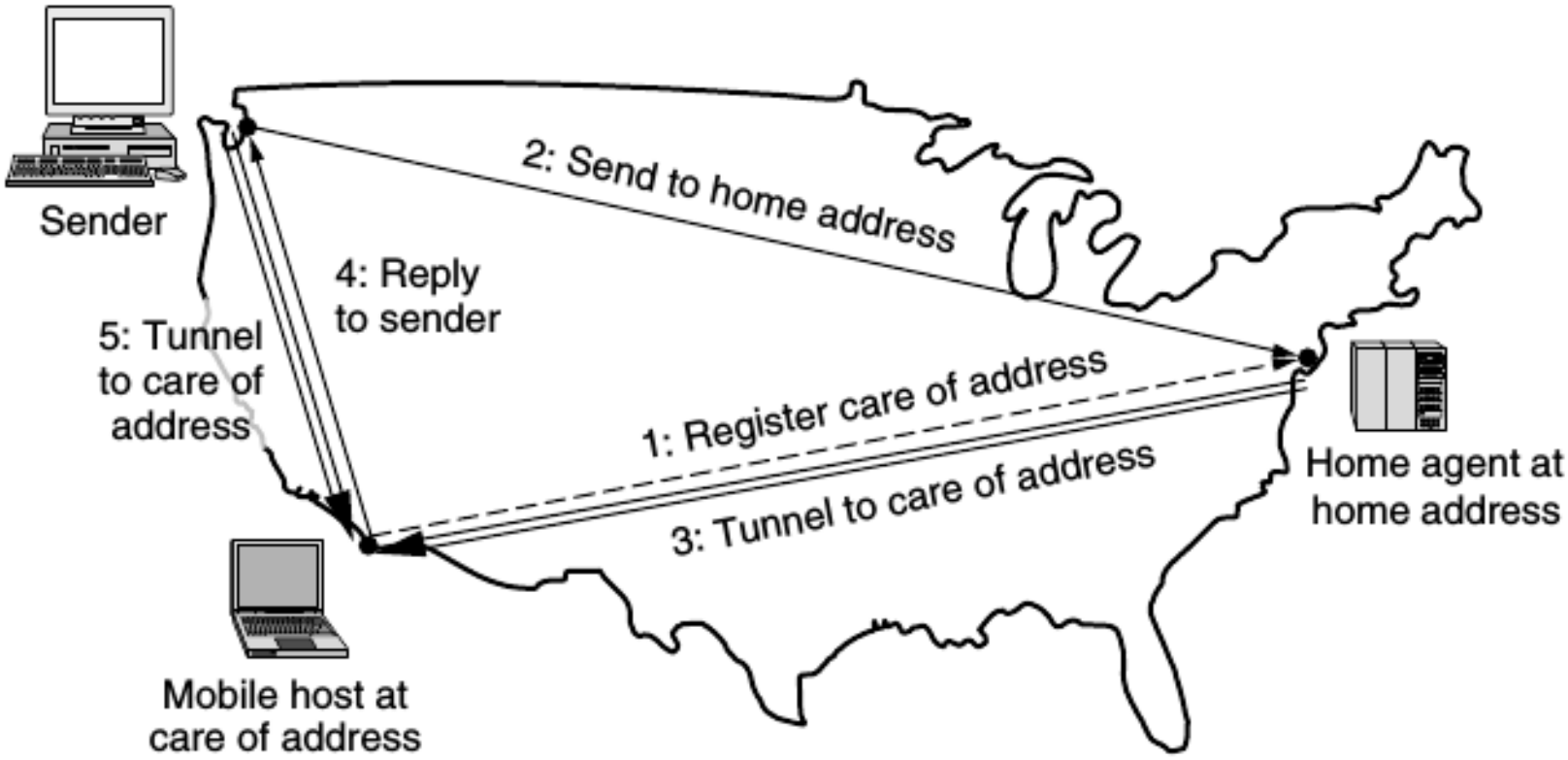


Figure 5-19. Packet routing for mobile hosts.

Maršrutizavimas Ad Hoc tinkluose

- Ką daryti kai patys maršrutizatoriai gali būti mobilūs? Pavyzdžiai - laivų flotilė, mašinų tinklai, palydovų tinklai, armija.
- Kiekvienas kompiuteris čia veikia ir kaip maršrutizatorius, duomenimis keičiamasi naudojant radijo bangas. Tokie tinklai vadinami **ad hoc** tinklais arba **MANET - Mobile Ad hoc NETworks**.
- Problema čia ta, kad nėra stabilios topologijos, ji nuolat keičiasi įrenginiams įeinant ir išeinant iš vienas kito ryšio zonos.
- Pavyzdys protokolo skirto tokiems tinklams yra **AODV - Ad hoc On-demand Distance Vector** protokolas.

Ad Hoc tinklai - kelio radimas

- AODV protokole keliai randami atsiradus poreikiui nusiųsti paketą kažkokiam gavėjui.
- Mazgas transliuoja ROUTE REQUEST paketą naudodamas **flooding**. Jis nukeliauja mazgams kurie tiesiogiai pasiekiami tam mazgui ir perduodami kitiems.
- Kai paketas pasiekia gavėją su kuriam norima perduoti duomenis jis atsako ROUTE REPLY paketu, kuris perduodamas atrastu keliu.
- Taip nustatomas kelias nuo siuntėjo iki gavėjo.

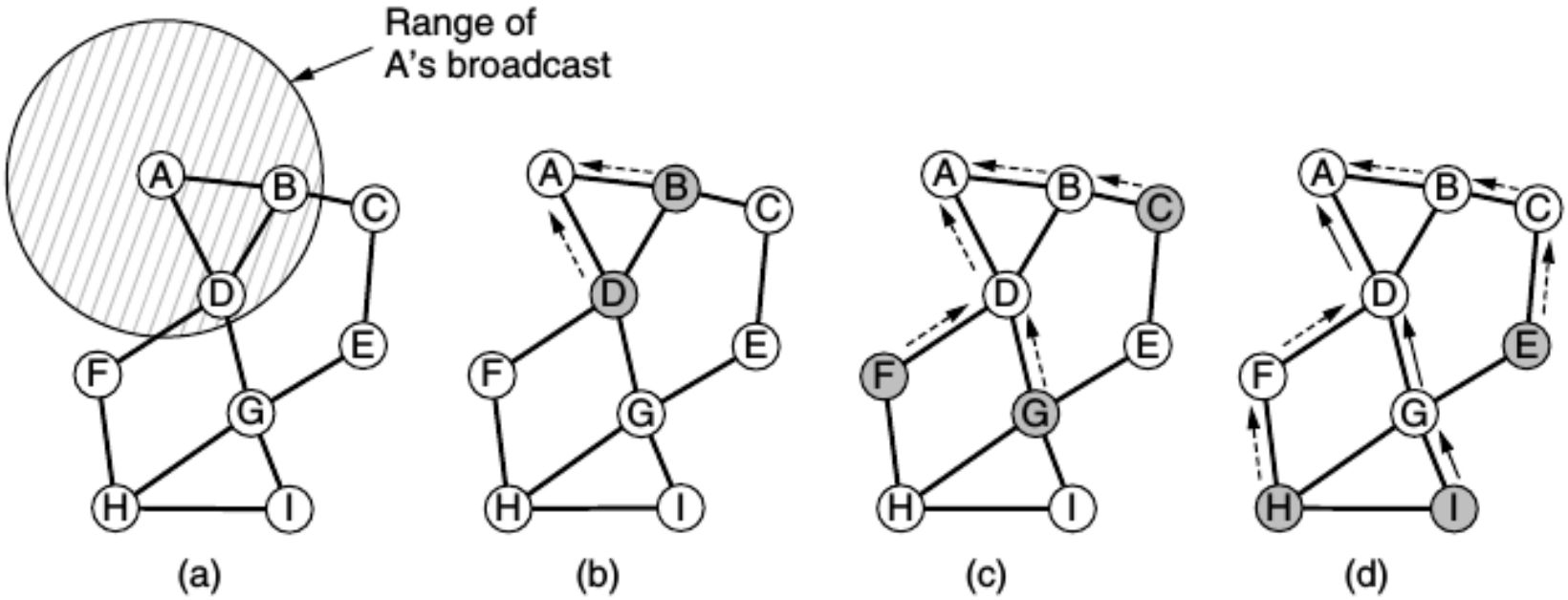


Figure 5-20. (a) Range of *A*'s broadcast. (b) After *B* and *D* receive it. (c) After *C*, *F*, and *G* receive it. (d) After *E*, *H*, and *I* receive it. The shaded nodes are new recipients. The dashed lines show possible reverse routes. The solid lines show the discovered route.

Ad Hoc tinklai - kelio priežiūra

- Kadangi mazgai gali būti atjungti arba persikelti į kitą vietą - tinklo topologija gali greitai pasikeisti.
- Mazgai siunčia HELLO paketus visiems kaimynams į kuriuos laukia atsakymo. Sulaukus atsakymo žinoma, kad kaimynas dar gyvas.
- Jeigu dingsta vienas iš kelio mazgų, kelias iš naujo perskaičiuojamas.
- Yra sukurta daug ad hoc maršrutizavimo schemų populiaresnės - **DSR - Dynamic Source Routing**, **GPSR - Greedy Perimeter Stateless Routing**.

Uždaviniai

Uždaviniai (I)

42. Duotas tinklas (bus nupaišyta) naudojantis atstumų vektorių maršrutizavimą, maršrutizatorius A gauna maršrutizavimo lenteles iš kaimynų, atnaujinti maršrutizatoriaus A lentelę.
43. Duotas tinklas, apskaičiuoti trumpiausią kelią jame naudojant Dijkstra algoritmą.
44. Jeigu jungties kainos saugomos 8 bitų skaičiuose, tinkle yra 50 maršrutizatorių o atstumų vektoriais keičiamasi du kartus per sekundę, kiek juostos pločio kiekviename duomenų kanale sunaudoja maršrutizavimo algoritmas. Kiekvienas maršrutizatorius prijungtas prie trijų duomenų kanalų.
45. Aprašykite efektyviausią nagrinėtą maršrutizavimo metodą kai duomenis reikia perduoti transliavimo būdu.

Uždaviniai (II)

46. Suskaičiuokite aprėpiantį medį konkrečiam maršrutizatoriui kai duomenis norima perduoti multicast būdu o į multicast grupę įeina nurodyti maršrutizatoriai. Tinklas bus nupaišytas.
47. Nupaišytas tinklas, kiek paketų jame bus sugeneruojama transliuojant iš nurodyto maršrutizatoriaus naudojant: a) **sink tree** b) **reverse path forwarding** metodus.
48. Išvardinkite skirtumus tarp datagramų ir **virtual circuit** tinklų. Įvertinkite šių tinklų privalumus ir trūkumus.
49. Savais žodžiais aprašykite **flooding** metodą trumpiausių kelių radimui. Kokie jo trūkumai ir kaip jų bandoma išvengti?

Uždaviniai (III)

50. Savais žodžiais aprašykite atstumų vektorių maršrutizavimą.
51. Savais žodžiais aprašykite **link state** maršrutizavimą.
52. Aprašykite vieną kokį nors Ad Hoc tinkluose naudojamą maršrutizavimo algoritmą.
53. Kuo skiriasi **broadcast**, **multicast** ir **anycast** naudojami maršrutizavimo algoritmai? Kokie jiems keliami reikalavimai? Kam reikalingi šie skirtingi duomenų perdavimo tipai?