Mrežni sloj
(Network layer)

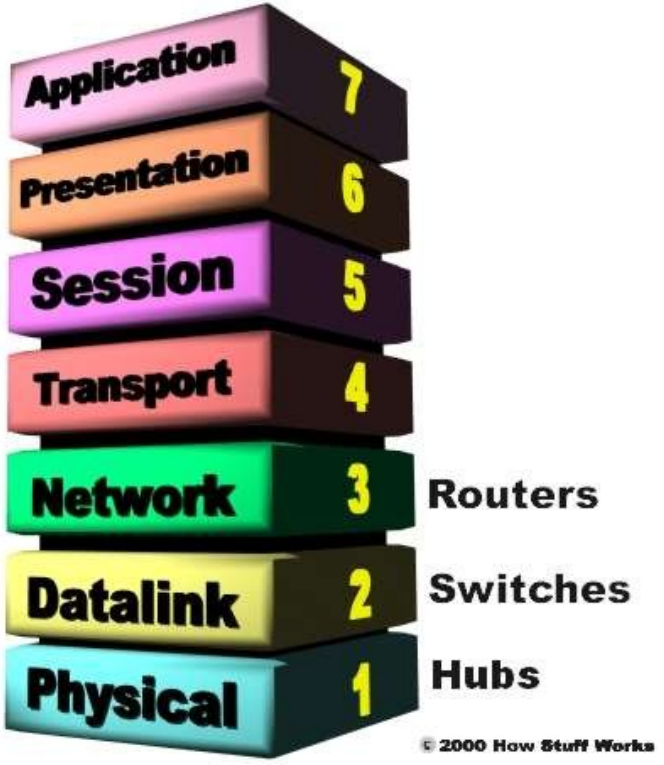
Računarske mreže i mrežne tehnologije
Maj 2013. god.

Sadržaj predavanja

- Projektovanje mrežnog sloja, usluge
- Realizacija sa uspostavljanjem direktne veze i datagramska usluga
- Rutiranje najkraćom putanjom (*shortest-path*)
- Plavljenje (*flooding*)
- *Distance Vector Routing*
- *Link State Routing*
- *Neusmereno emitovanje (Broadcast)*
- Višesmerno emitovanje *(Multicast)*
- Hijerarhijsko rutiranje
- Kontrola zagušenja *(Congestion Control)*
- Kvalitet usluga *(Quality of Service)*

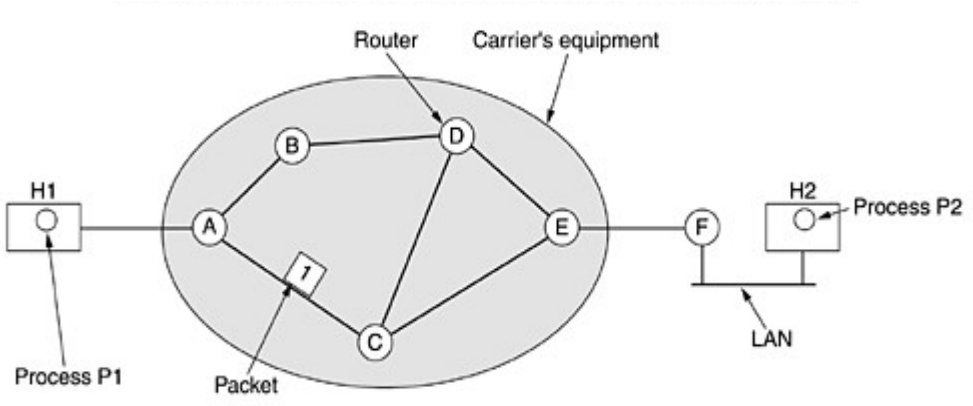
Zadatak mrežnog sloja

- Da pakete od izvorišta do odredišta **sprovede celim putem**, što znači da treba da ih provuče kroz sve usputne rutere (usmerivače)
- Mrežni sloj mora da poznaje **topologiju komunikacione podmreže**, tj. skupa rutera



Komutiranje “čuvaj i prosledi”

- Oprema za prenos podataka telekom. Kompanije
- Korisnička oprema
- Paket se u svakom ruteru čuva dok ne stigne u potpunosti kako bi mogao da mu se proveri kontrolni zbir
- Zatim se šalje dalje

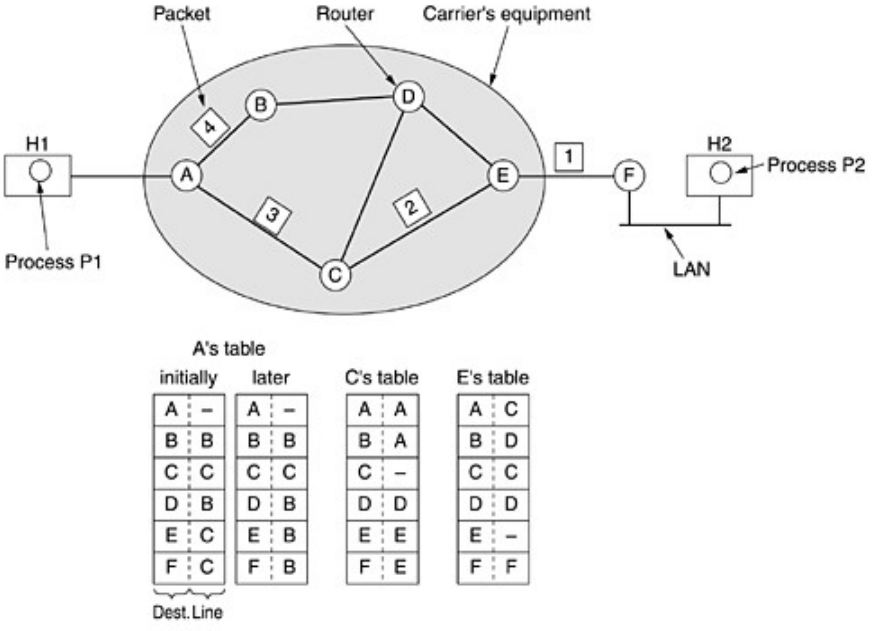


Usluge mrežnog sloja transportnom

- Usluge treba da budu **nezavisne od tehnologije rutera**
- Transportni sloj ne sme da zna ništa o broju, vrstama i topologiji rutera
- **Mrežne adrese moraju da budu uniformno označene**, čak i u lokalnim mrežama
- **Jedan tabor (Internet kompanije i korisnici)** tvrdi da ruteri treba samo da usmeravaju pakete i ništa više. Kažu da je podmreža po prirodi nepouzdana, a kontrolu redosleda pristizanja paketa i kontrolu toka ionako rade sami računari, tako da nema potrebe da se ruteri time bave
- **Drugi tabor (telefonske kompanije)** tvrdi da podmreža treba da obezbedi pouzdanu uslugu sa uspostavljanjem direktne veze, što ima smisla ako je glavnica saobraćaja govor ili video materijal

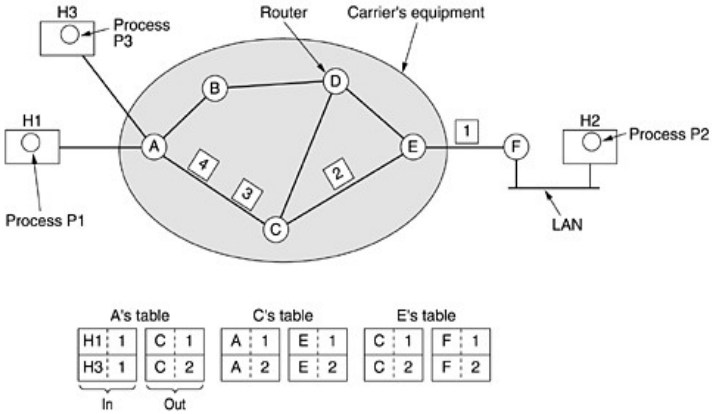
Usluga datagrama

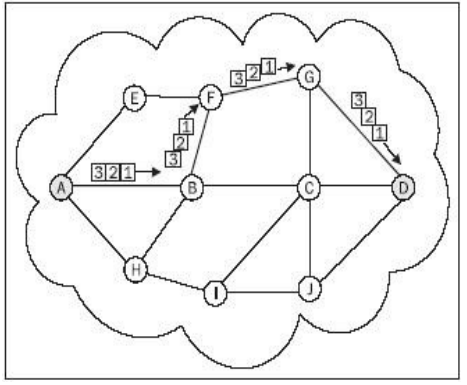
- Paketi se u podmrežu upućuju nezavisno jedan od drugog
- Paket broj 4 iz nekog razloga (možda prekid linije ACE) prati drugu putanju, jer ga ruter A usmerava prema ažuriranoj tabeli
- Algoritam koji radi sa tabelama i donosi odluke o usmeravanju naziva se **algoritam za rutiranje**



Usluga sa uspostavljanjem direktne veze

- Za uslugu sa uspostavljanjem direktne veze potrebna su tzv. **virtuelna kola** (*Virtual Circuit* – VC)
- Za svaki paket se sada **ne izmišlja nova putanja**. Umesto toga, kada se veza uspostavi, putanja između izvorišta i odredišta se upisuje u tabele rutera
- **Kao i u sistemu telefonije**, sav saobraćaj se odigrava preko tog virtuelnog kola, dok svaki paket nosi identifikator virtuelnog kola kojem pripada
- Proces P3 pokušava da isporuči paket kome je on dodelio identifikator virtuelnog kola 1 (jer nije deo podmreže). Ruter A mu automatski menja identifikator u 2, na taj način izbegavajući sukob



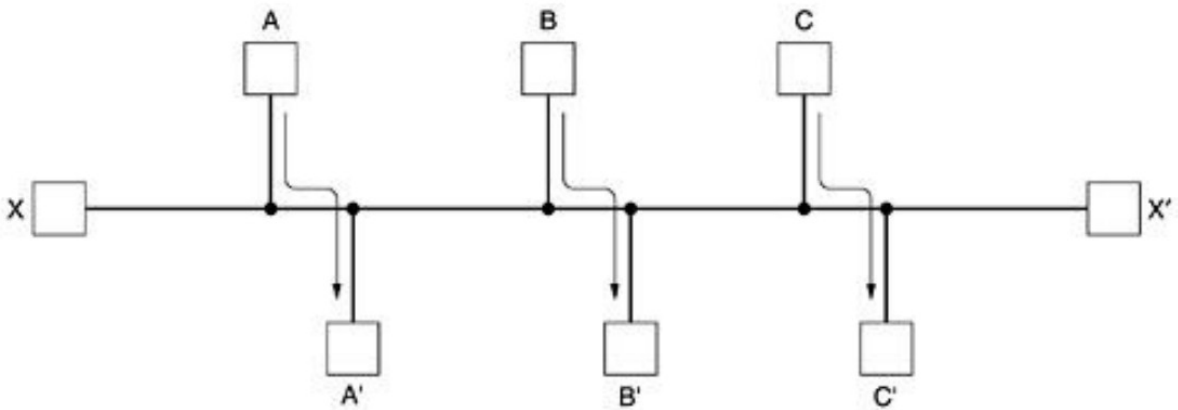


Poređenje usluga datagrama i VC

Issue	Datagram subnet	Virtual-circuit subnet
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

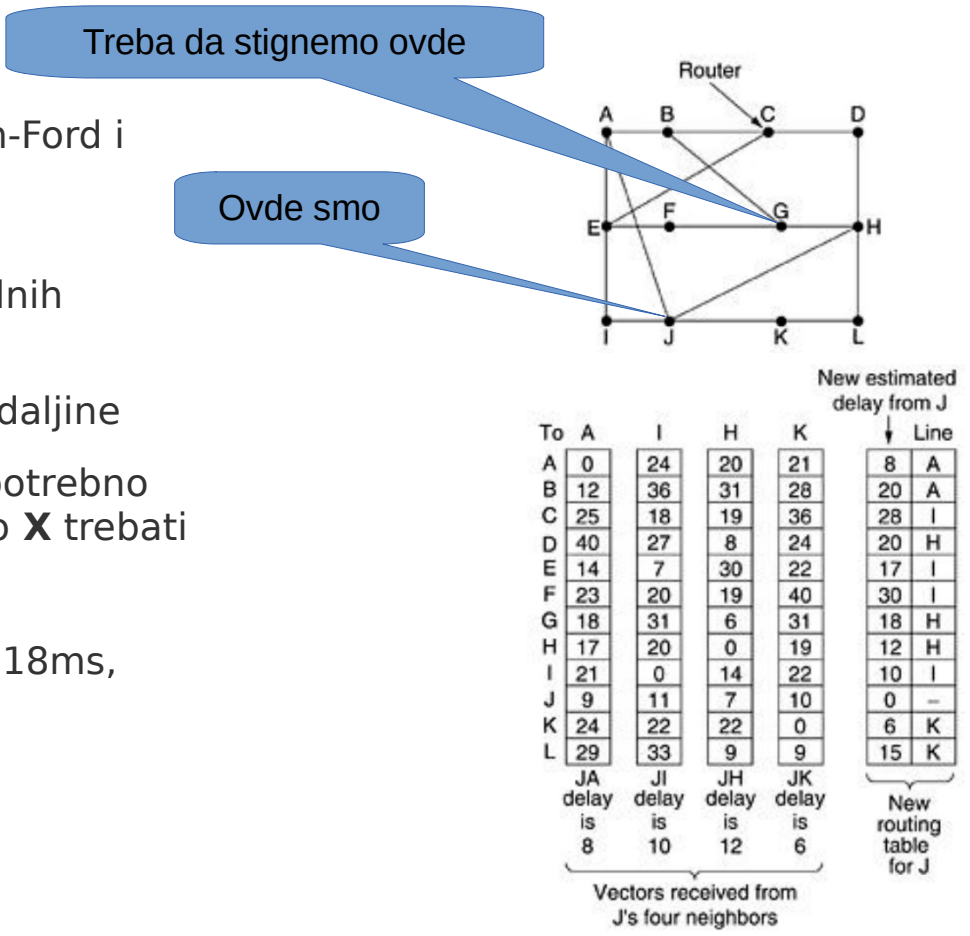
Algoritmi za rutiranje

- **Usmeravanje** i **prosleđivanje (*Routing* *vs* *Forwarding*)**. U svakom ruteru se odigravaju dva paralelna procesa
- Kod usluge sa uspostavljanjem direktne veze (VC), koristi se **usmeravanje za sesiju**
- Svojstva: **tačnost, jednostavnost, robusnost, stabilnost, pravičnost i optimalnost**
- Pravičnost je u sukobu sa optimalnošću
- Neprilagodljivi i prilagodljivi algoritmi (***static/dynamic routing***)



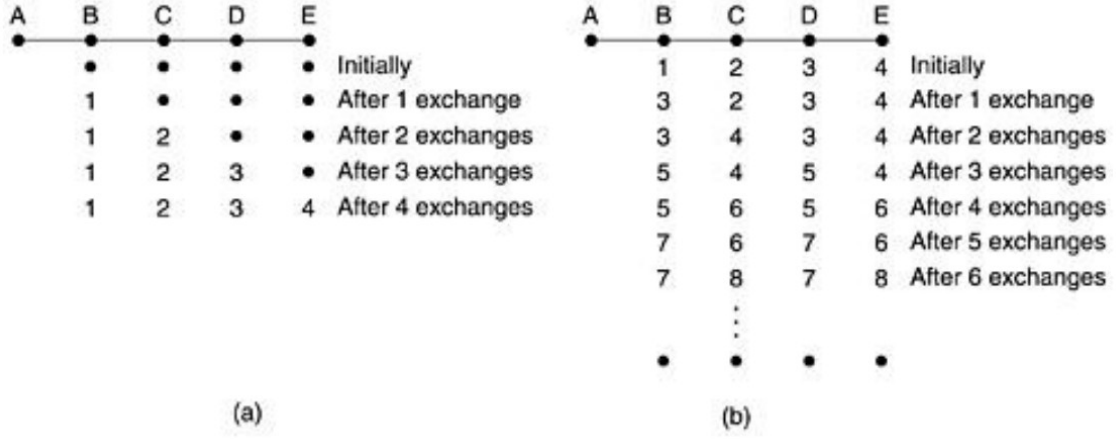
Rutiranje na osnovu vektora razdaljine (RIP)

- Dinamički algoritam, poznat još kao Belman-Ford i Ford-Fulkerson, na ARPANET-u RIP (*Routing Information Protocol*)
- Svaki ruter određuje rastojanje do neposrednih suseda pomoću ECHO paketa
- Svakih *T* sekundi, razmenjuju se vektori razdaljine
- Ako ruter zna da je paketu do susednog **X** potrebno *m* milisekundi, onda će mu do rutera i preko **X** trebati X_i+m milisekundi
- Do G: preko A 26ms, preko I 41ms, preko H 18ms, preko K 37ms



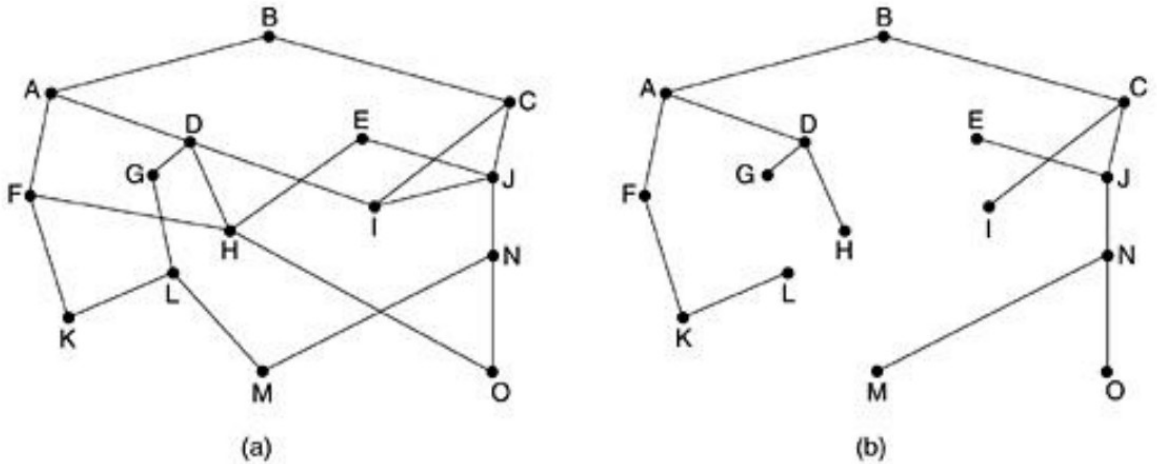
Problem približavanja beskonačnosti

- Nedostatak usmeravanja zasnovanog na vektoru razdaljine – **brzo reaguje na povoljne informacije, a odbija da upiše nepovoljne**
- **Uzrok**: U jednoj iteraciji rastojanja se poemraju najviše za 1. Teško je dostići beskonačnost.
- **Rešenje**: definisati beskonačnost kao vrednost za 1 veću od najduže putanje. A šta ako se meri vreme, a ne broj skokova?



Princip optimalnosti

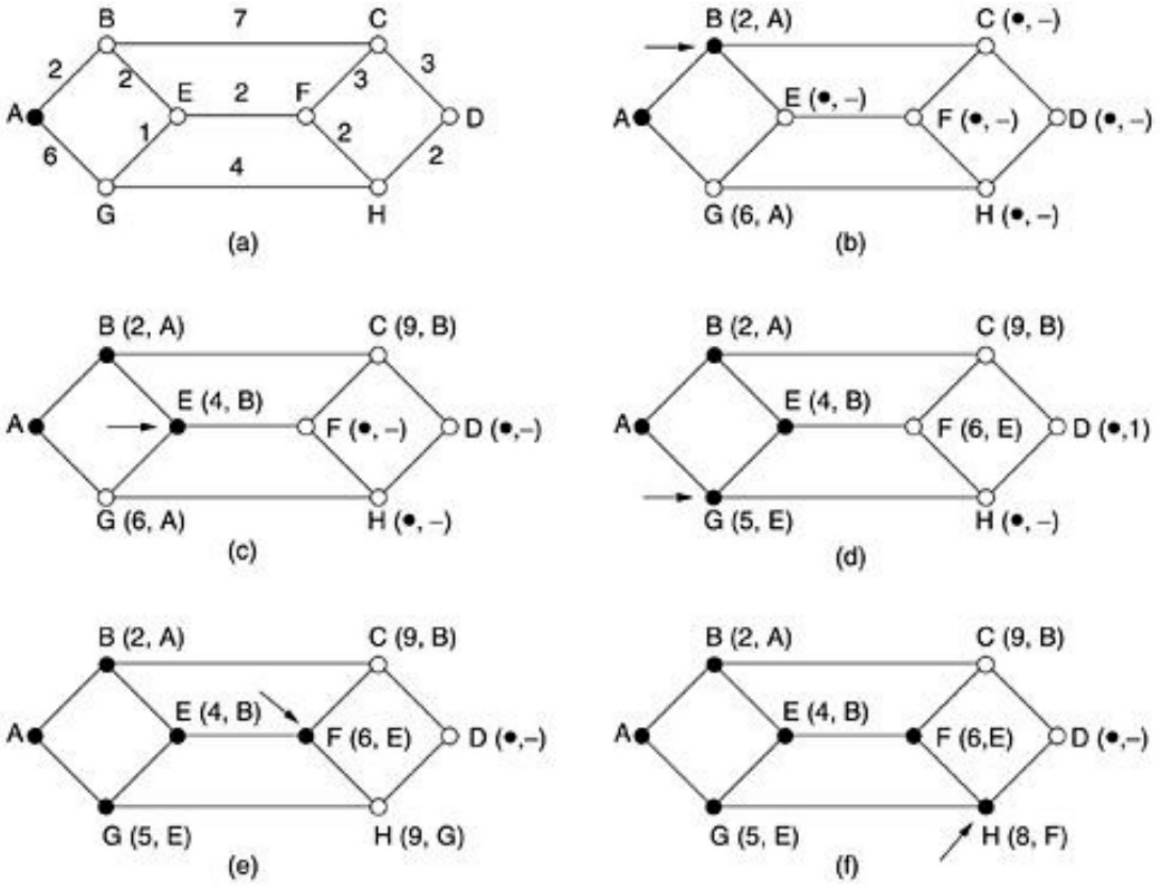
- Ako se ruter ***J*** nalazi na optimalnoj putanji između rutera ***I*** i rutera ***K***, tada se optimalna putanja između ***J*** i ***K*** takođe nalazi na toj istoj putanji.
- Posledica: Skup optimalnih putanja iz svih izvora ka jednom odredištu obrazuje **stablo ukorenjeno na odredištu**
- **Stablo optimalnih putanja – *sink tree*** ne sadrži petlje, pa se svaki paket isporučuje nakon **konačnog** broja skokova
- U opštem slučaju može biti više optimalnih stabala



Usmeravanje najkraćom putanjom

- **Pitanje metrike**: broj skokova, geografski, prosečno kašnjenje, propusni opseg, prosečni saobraćaj, cena, ...
- http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
- Privremene (*tentative*) i trajne (*permanent*) oznake
- Označi se početni čvor kao trajni i ispitaju se rastojanja do njegovih suseda
- Kada čvoru pridružimo rastojanje, naznačava se i čvor od koga je mereno kako bi se putanja na kraju rekonstruisala
- Zatim se ispitaju svi privremeni čvorovi i onaj sa najmanjim rastojanjem proglasi novim trajnim.
- Postupak se onda ponavlja
- Zašto algoritam radi? Pretpostavimo da postoji AXYZE kraće od ABE (slika *c*). Ako je Z već postao trajan, znači da je E već ispitan, što dalje znači da nismo mogli da zaobiđemo AXYZE. Ako je pak Z još uvek privremen, AXYZE ne može biti kraće od ABE jer bi onda Z morao prvo postati trajan pa tek onda ispitati E.

Usmeravanje najkraćom putanjom (2)



Usmeravanje najkraćom putanjom (3)

```
#define MAX_NODES 1024                  /* maximum number of nodes */
#define INFINITY 1000000000             /* a number larger than every maximum path
int n, dist[MAX_NODES][MAX_NODES];/* dist[i][j] is the distance from i to j */

void shortest_path(int s, int t, int path[])
{ struct state {                        /* the path being worked on */
      int predecessor;                  /* previous node */
      int length;                       /* length from source to this node */
      enum {permanent, tentative} label; /* label state */
  } state[MAX_NODES];

  int i, k, min;
  struct state *p;

  for (p = &state[0]; p < &state[n]; p++) { /* initialize state */
      p->predecessor = -1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0;  state[t].label = permanent;
  k = t;                                /* k is the initial working node */
  do {                                  /* Is there a better path from k? */
      for (i = 0; i < n; i++)           /* this graph has n nodes */
          if (dist[k][i] != 0 && state[i].label == tentative) {
              if (state[k].length + dist[k][i] < state[i].length) {
                  state[i].predecessor = k;
                  state[i].length = state[k].length + dist[k][i];
              }
          }
```

Usmeravanje najkraćom putanjom (4)

```
    /* Find the tentatively labeled node with the smallest label. */
    k = 0; min = INFINITY;
    for (i = 0; i < n; i++)
        if (state[i].label == tentative && state[i].length < min) {
            min = state[i].length;
            k = i;
        }
    state[k].label = permanent;
} while (k != s);

/* Copy the path into the output array. */
i = 0;  k = s;
do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

- Pretpostavka algoritma je da je graf simetričan, tj. da smer ne utiče na rastojanje
- Podaci *n* i *dist[][]* opisuju graf

Plavljenje (*flooding*)

- Svaki dolazni paket se **šalje na sve linije osim na onu s koje je došao**. Šta je problem ovog pristupa? Kako se može rešiti:

  (1) Poplava se može ograničiti **brojanjem skokova**. Idealno rastojanjem između *src-dest,* ali u praksi prečnikom podmreže

  (2) **Vođenje liste paketa**. Ako je takav paket već stigao, ne emituje se dalje. Da lista ne bi neograničeno rasla, uvodi se brojač. Ako ima vrednost *k*, to znači da su svi paketi do *k* već stigli.

- **Selektivno plavljenje** – paket se šalje samo na linije koje se pružaju u približno istom smeru
- **Primena**: U ratnim uslovima, ažuriranje distribuiranih baza, uvek se pronalazi optimalna putanja

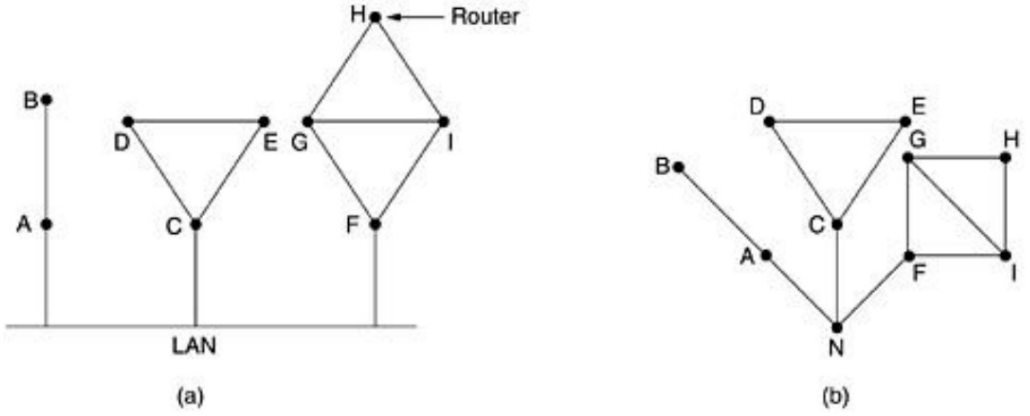
Usmeravanje zasnovano na stanju veze

Link State Routing. Problemi prethodnog algoritma je sporo dostizanje ravnoteže i metrika koja nije u obzir uzimala propusni opseg. Svaki ruter treba:

1. Da otkrije svoje susede i sazna njihove mrežne adrese
2. Da izmeri vreme i troškove slanja do njih
3. Da napravi paket sa podacima do kojih je došao
4. Da taj paket pošalje svim ruterima
5. Da izračuna najkraću putanju do svakog drugog rutera

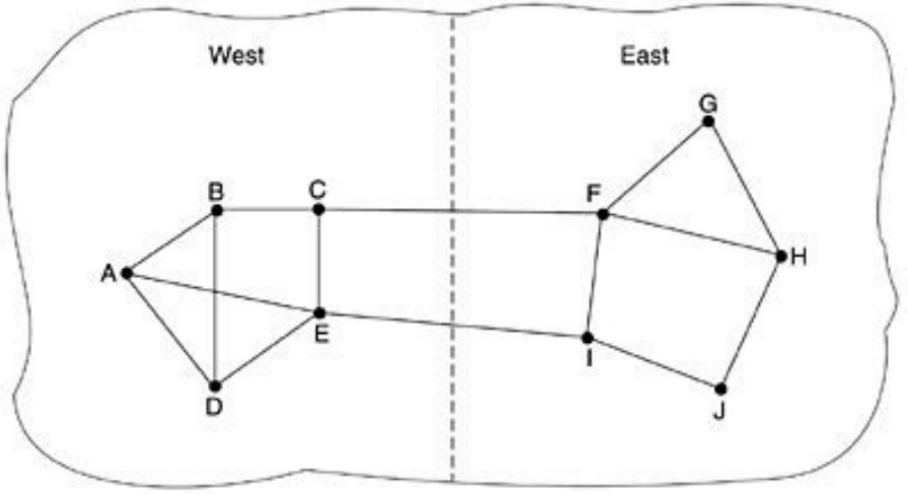
Upoznavanje suseda

- Na svaku P2P liniju šalje paket HELLO. Ruter s druge strane se predstavlja svojom adresom.
- Na slici je prikazana situacija kada se dva ili više rutera nalaze na istoj lokalnoj mreži.



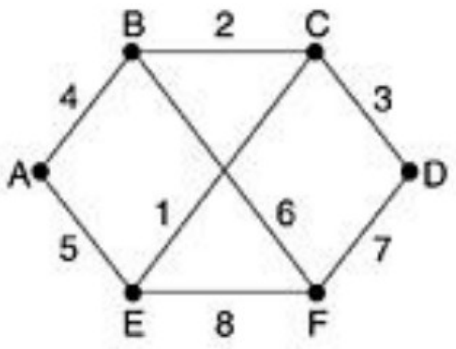
Merenje troškova slanja linijom

- Na svaku liniju se šalje paket ECHO, izvlači se prosek iz više slanja
- **Pitanje**: Da li opterećenje linije treba uzeti u obzir ili samo propusni opseg?

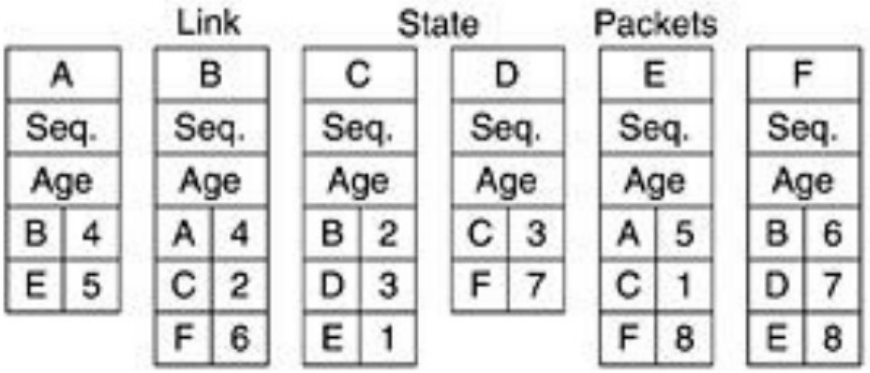


Kreiranje paketa sa stanjem veze

- Kada praviti pakete sa stanjem veze?
- Paketi se prave ili periodično ili kada se desi neki događaj, kao što je otkaz ili oporavak suseda



(a)

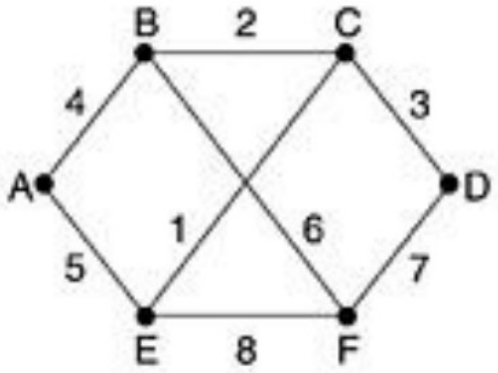


(b)

Distribuiranje paketa sa stanjem

- Najproblematičniji deo algoritma jer je potrebno rešiti sinhronizaciju (pojava petlji itd.)
- Svaki ruter evidentira parove (ruter, r.br. paketa)
- Ako je paket zaista nov, prosleđuje se na sve linije (plavljenje)
- Za pakete koji nose stanje veze **traži se potvrda**
- **PROBLEMI**:
    1) Ako bi se redni brojevi ciklično ponavljali, nastala bi zabuna. Rešenje je da r.br. ima bar 32 bita
    2) Ako ruter otkaže, gubi evidenciju r.br. Kad ponovo počne od 0, njegov paket se odbacuje kao duplikat
    3) Ako se r.br. ošteti, npr. umesto 5 stiže 65540, svi <65540 se odbaciju kao zastareli
- Sva 3 problema se mogu rešiti uvođenjem “**starosti**” paketa, brojača koji će se svake sekunde smanjivati za 1

Distribuiranje paketa sa stanjem (2)



- Paketi sa stanjem veze se moraju potvrditi
- Struktura podataka koju koristi ruter B sa slike, data je u tabeli
- Treći paket od rutera E se šalje samo C jer stižu dva takva paketa putanjama EAB i EFB, a potvrđuju se ruterima A i F
- Flegovi se dinamički mogu menjati kada stigne novi paket sa stanjem veze

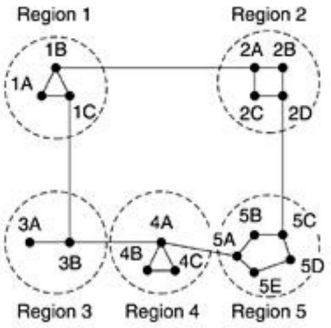
			Send flags			ACK flags			
Source	Seq.	Age	A	C	F	A	C	F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Izračunavanje novih putanja

- Kada se dobiju potpuni podaci, može se upotrebiti Dijkstrin algoritam da se konstruišu putanje do svih mogućih odredišta
- *n* rutera, svaki po *k* suseda, memorija *~kn*
- Problem može da bude i trajanje računa
- **Protokol OSPF** (*Open Shortest Path First*) se često koristi na Internetu
- **Protokol IS-IS** (*IntermediateSystem-IntermediateSystem*) se takođe koristi u nekim okosnicama. Njegova prednost je što sarađuje kako sa IP, tako i sa IPX, AppleTalk, itd.

Hijerarhijsko rutiranje

- **Raste mreža**=>rastu tabele, memorija, CPU vreme, troši se propusni opseg za razmenu informacija
- **Hijerarhijsko rutiranje** – ruteri se dele u oblasti, i svaki zna sve o rutiranju samo u svojoj oblasti
- Pitanje optimalnog broja stepeni – 720 rutera (1 stepen:720 odrednica, 2 stepena:24x30:30+23=53 odrednice, 3 stepena-8x9x10:10+8+7=25)
- Optimalan broj rutera za mrežu veličine *N* je *lnN*, kada tabele sadrže po *e lnN* odrednica



(a)

Full table for 1A

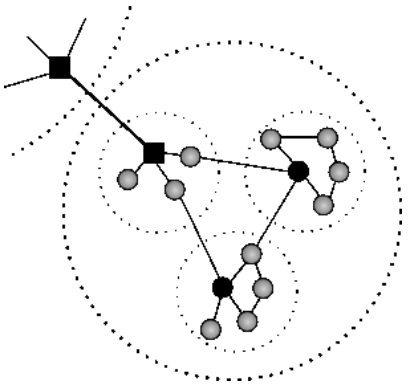
Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(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)

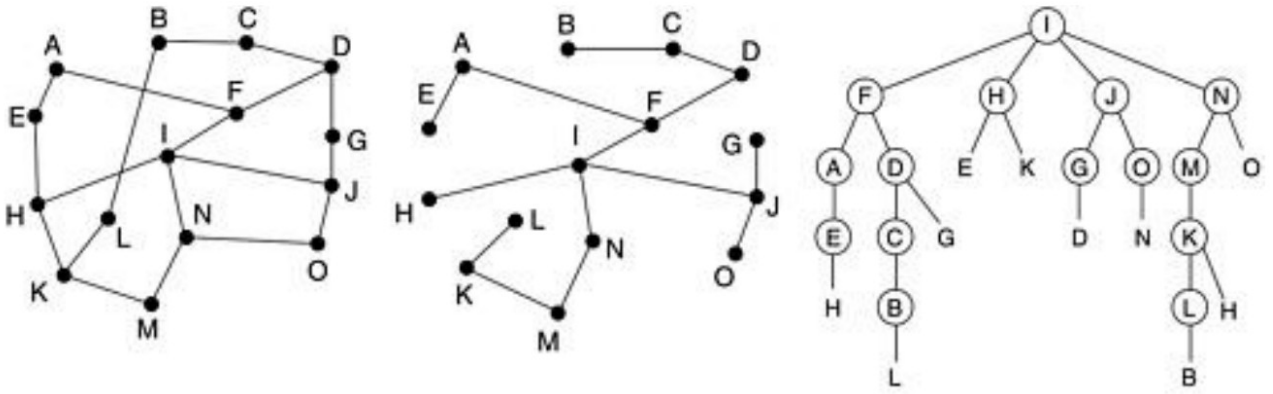


Neusmereno emitovanje (*broadcast*)

- **Metod 1**: Po jedan paket na svaku destinaciju. Neefikasno, a i zahteva da se zna adresa svake destinacije
- **Metod 2**: Plavljenje. Očigledna mogućnost. Ali, zauzima se veliki propusni opseg.
- **Metod 3**: Usmeravanje na više odredišta (*multidestination routing*). Ruter generiše po paket za svaku izlaznu liniju za destinacije koje se dostižu preko te linije. Na kraju ostane običan paket sa jedinstvenim adresama
- **Metod 4**: Razgranato stablo (*spanning tree*). Svaki ruter zna koja od njegovih linija pripada stablu.

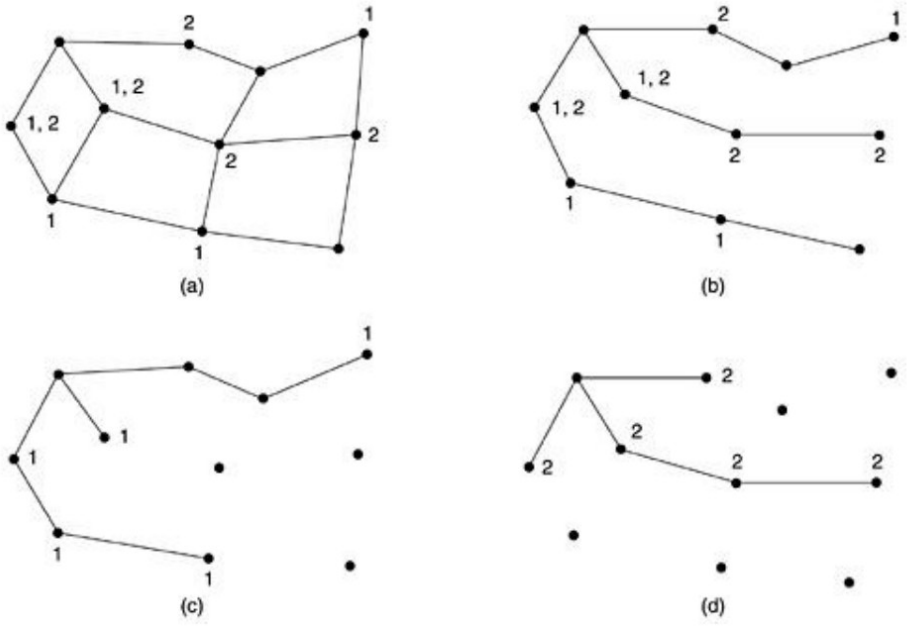
Neusmereno emitovanje (*broadcast*)

- **Metod 5**: Prosleđivanje **ispitivanjem izvorišta (*reverse path forwarding*)**. Kada neusmereno emitovan paket stigne, ruter proverava da li je stigao linijom koja se obično koristi za slanje *izvorištu*. Ako je odgovor potvrdan, verovatno je paket slučajno našao najbolju putanju, tj. da je stigla njegova prva kopija. U tom slučaju paket se šalje na sve linije. U suprotnom se odbacije.
- Lako se realizuje i nije potreban mehanizam prekida kao kod plavljenja



Višesmerno usmeravanje (*multicast*)

- Ponekada se može koristiti *broadcast*
- Zahteva rad sa **grupom**. Mora postojati mehanizam za konstruisanje i raspuštanje grupa. Obaveštavanje o grupisanju se širi na neki način do svakog rutera u podmreži.
- Svaki ruter generiše razgranato stablo do svih ostalih
- Vrši se “**proređivanje stabla”**

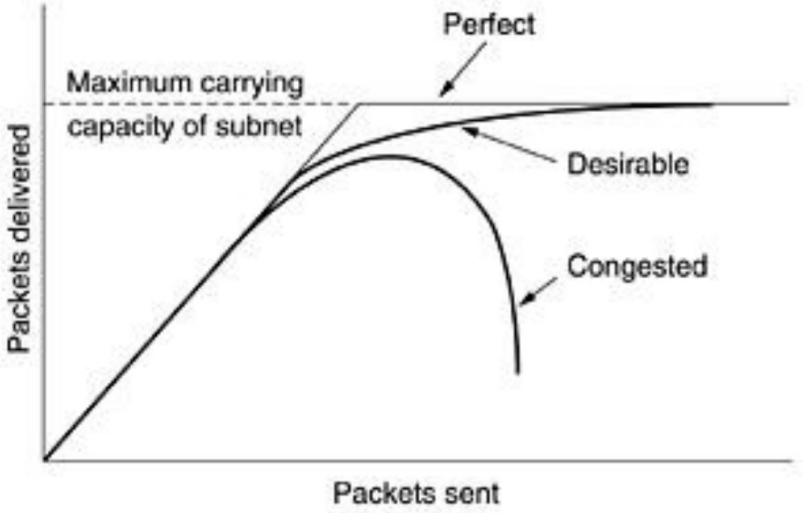


Višesmerno usmeravanje (*multicast*)

- Proređivanje stabla može se izvršiti na više načina:
  - Ako se izvršava **algoritam usmeravanja na osnovu stanja veze**. Zna se kompletna topologija, pa se uklanjaju ruteri koji ne spadaju u datu grupu.
  - Ako se izvršava **rutiranje na osnovu vektora razdaljine**, rutira se **ispitivanjem izvorišta**. Takođe, kad god ruter primi poruku za grupu s kojom nema veze, vraća PRUNE (okreši)
- Algoritam ima manu sa skaliranjem. *n* grupa od *m* računara vodi do *nm* stabala koja moraju da se pamte
- Alternativa je **stablo zasnovano na korenu**, kada je definisani koren grupe zadužen da pošalje poruku ostalim članovima.

Upravljanje zagušenjem

- ***Congestion control***
- Ruteri ne mogu da održe korak i počinju da gube pakete. Performanse se degradiraju,a paketi skoro i ne isporučuju
- Postoji dokaz da povećanje memorije rutera samo dovodi do još većeg zagušenja. Zašto?
- Zagušenje izazivaju i spori procesori rutera
- **Kontrola toka** i **kontrola zagušenja** su različiti pojmovi. Na šta se koji pojam odnosi?



Osnovni principi kontrole zagušenja

- Dobro projektovanje kako do zagušenja ne bi uopšte moglo da dođe
- Korišćenje povratnih informacija:
  - **Stalni nadzor** (procenat odbačenih paketa, prosečno čekanje u redu, broj ponovljenih,...)
  - **Prosleđivanje informacija** do mesta gde se nešto može preduzeti (slanje info paketa ili samo bita, probni paketi,...)
  - **Podešavanje parametara** da bi se problem otklonio (problem oscilovanja)
- **Preventivni algoritmi** (na izvorištu i na odredištu)
- **Kurativni algoritmi** (eksplicitno i implicitno upozoravanje)
- Proširiti resurse ili smanjiti opterećenje

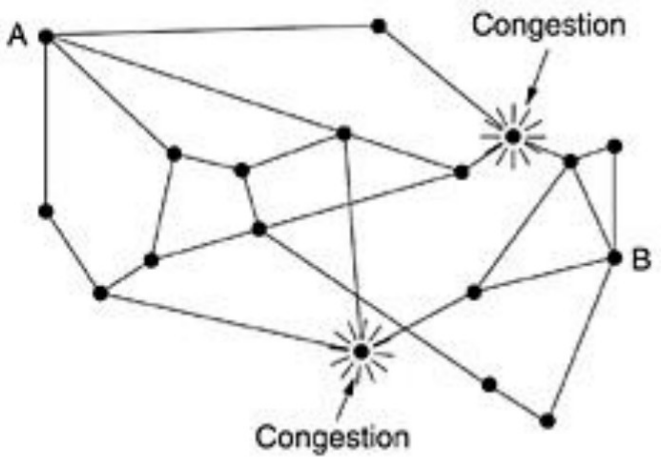
Pravila sprečavanja zagušenja

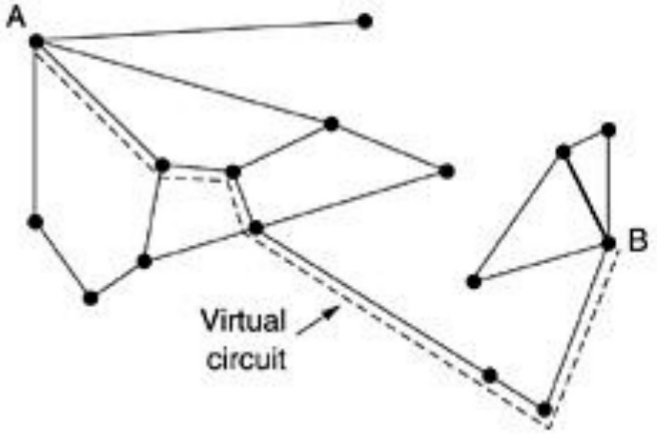
- Sprečavanje zagušenja unapred, razvrstano po tehnikama koje se koriste u različitim slojevima

Layer	Policies
Transport	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy • Timeout determination
Network	• Virtual circuits versus datagram inside the subnet • Packet queueing and service policy • Packet discard policy • Routing algorithm • Packet lifetime management
Data link	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy

Kontrola zagušenja u VC mrežama

- Kontrola pristupa (*admission control*) – ne uspostavljati nove veze dok se zagušenje ne otkloni
- Zaobilaženje tačaka zagušenja





Kontrola zagušenja u datagramskim podmrežama

- Svaki ruter lako može da prati opterećenje svojih izlaznih linija
- Kada u pređe prag, preduzima se akcija
- ***u*** - opterećenje
- ***f*** - trenutno stanje (0,1)
- ***a*** - koeficijent koji određuje brzinu kojom ruter zaboravlja prethodno stanje

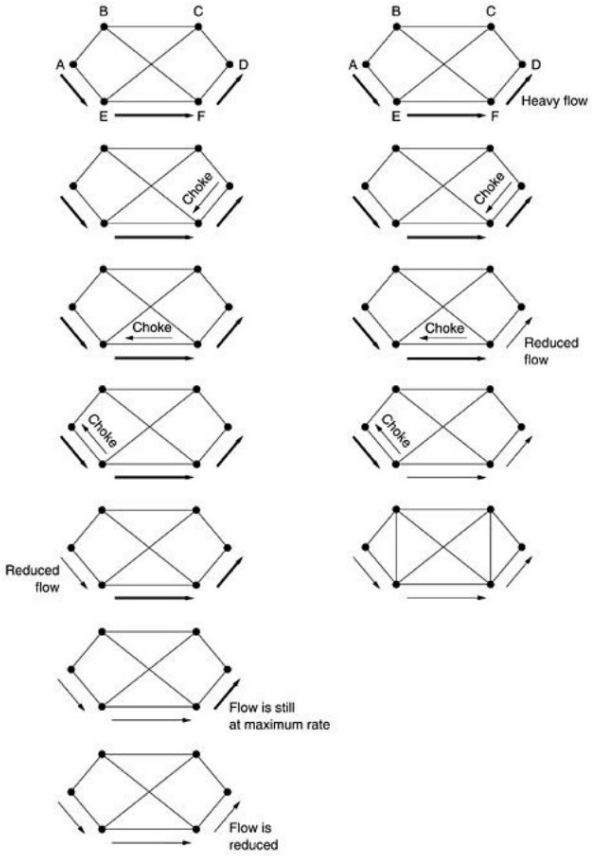
$$u_{novo} = a \cdot u_{staro} + (1 - a) \cdot f$$

Bit upozorenja i prigušni paketi

- Transportni proces na destinaciji stavlja **bit upozorenja** u sledeću potvrdu ka izvorištu. Svaki ruter na putanji može da postavi bit, tako da se saobraćaj obnavlja tek kad nijedan ruter nije više u kritičnom stanju
- **Prigušni paket** (*choke packet*) - šalje se ceo paket na izvorište. Nakon prijema paketa neko vreme se zanemaruju. Podešavanje npr. može da se izvede smanjenjem prozora za slanje na 50%, 25%, ...

Prigušni paketi za svaki skok

- Delimično rešava problem spore reakcije pri velikim brzinama i/ili velikim razdaljinama
- NY->SF na 155Mbps, prigušni paket stiže tek posle 30ms (novih 4.6Mb je već otišlo)

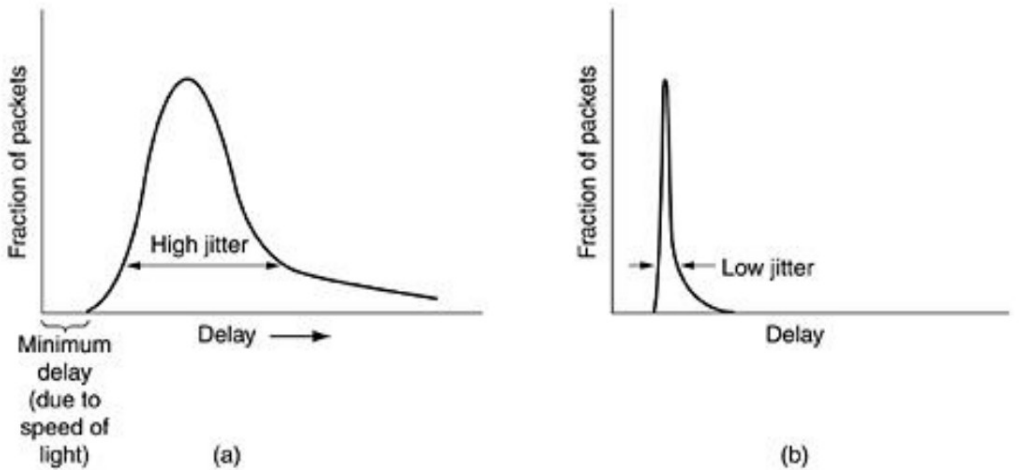


Odbacivanje paketa, rano otkrivanje

- **Odbacivanje paketa**
  - ***Load shedding*** - ako nijedna druga metoda ne urodi plodom
  - Da li odbacivati slučajnim izborom, stare ili nove pakete? Primer transfera fajla i videa
  - Mogućnost uvođenja klasa prioriteta. Naplata?
- Rano otkrivanje zagušenja
  - ***Random Early Detection (RED)***
  - Praćenje prosečne dužine redova čekanja
  - Nasumično odbacivanje paketa iz prepunog reda?
  - Da li slati prigušni paket ili čekati isticanje tajmera i odluku da se uspori slanje kad se paketi već gube?

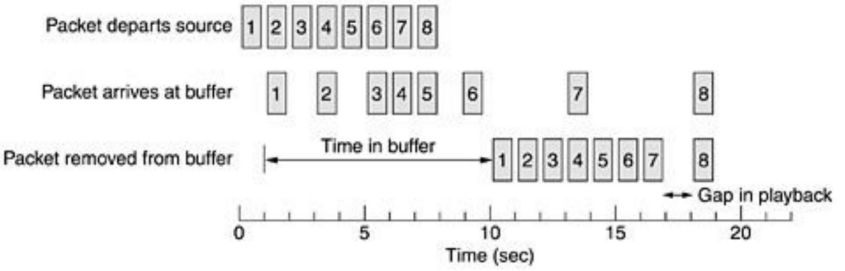
Kontrola neravnomernosti

- ***Jitter***
- Za audio/video i ostale *real-time* primene nije bitno kašnjenje, već konstantnost kašnjenja
- Pristup: Meriti srednje vreme pa ako paket stigne kasnije, odmah ga proslediti, a ako stigne pre zadržati ga.
- Alg. rutiranja bira paket koji najviše kasni



Kvalitet usluga (QoS)

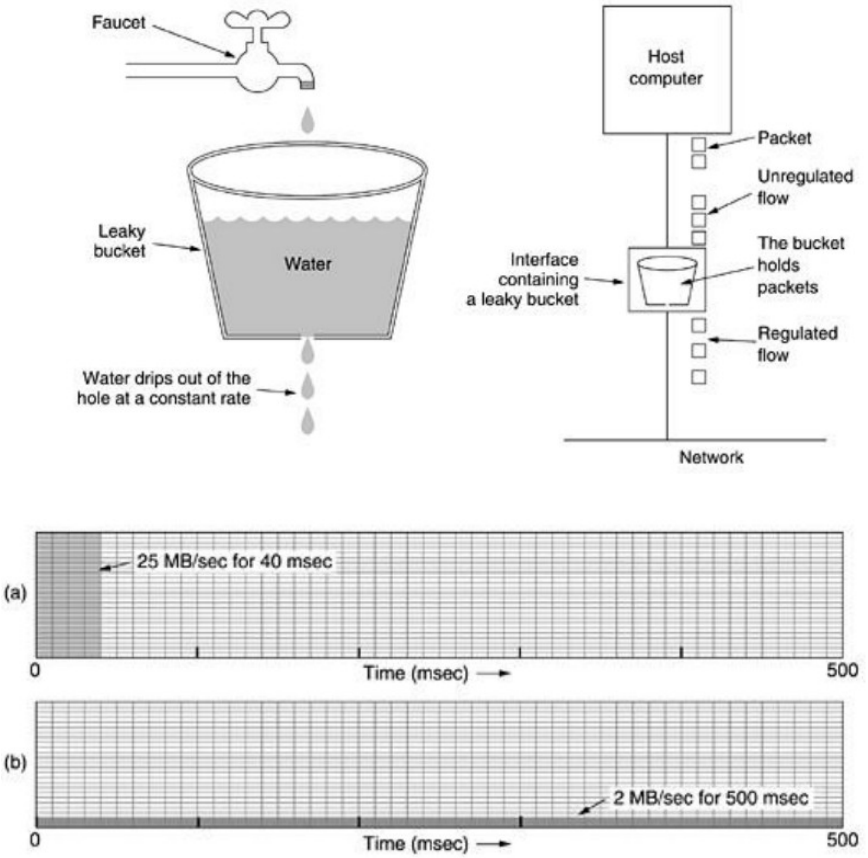
1) Obezbeđivanje viška resursa. Skupo
2) Privremeno skladištenje. Veličina bafera?
3) Ujednačavanje saobraćaja – SLA, lakše sa VC mrežama



Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

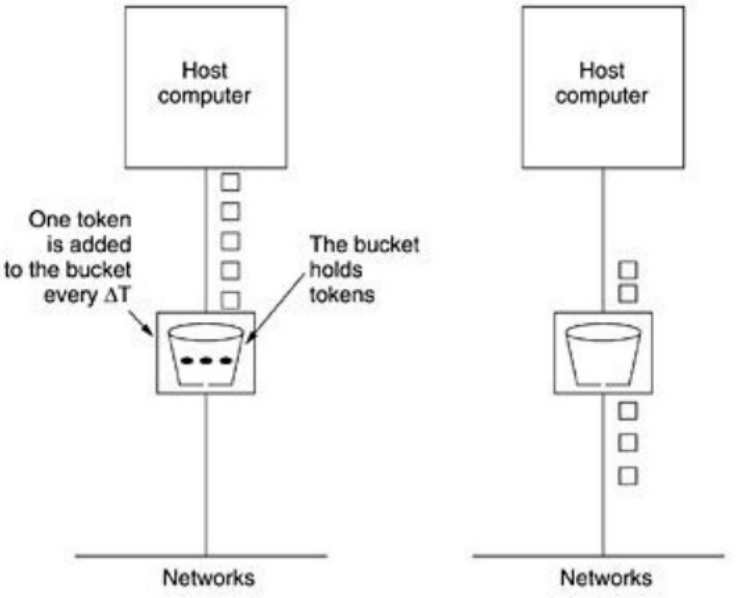
Algoritam bušne kofe (*leaky bucket*)

- Red čekanja sa konstantnom brzinom opsluživanja
- Konst. broj paketa vs. konst. broj bajtova po jednom otkucaju sata
- Jednostavan za implementaicju, hardverski ili OS
- Primer: 25MB/s stiže u rafalima od 40ms, kapacitet kofe **C**=1MB i brzina isticanja **ρ**=2MB/s. Znači da se rafal može obraditi bez gubitaka i razložiti na 500ms



Algoritam kofe sa žetonima (*token bucket*)

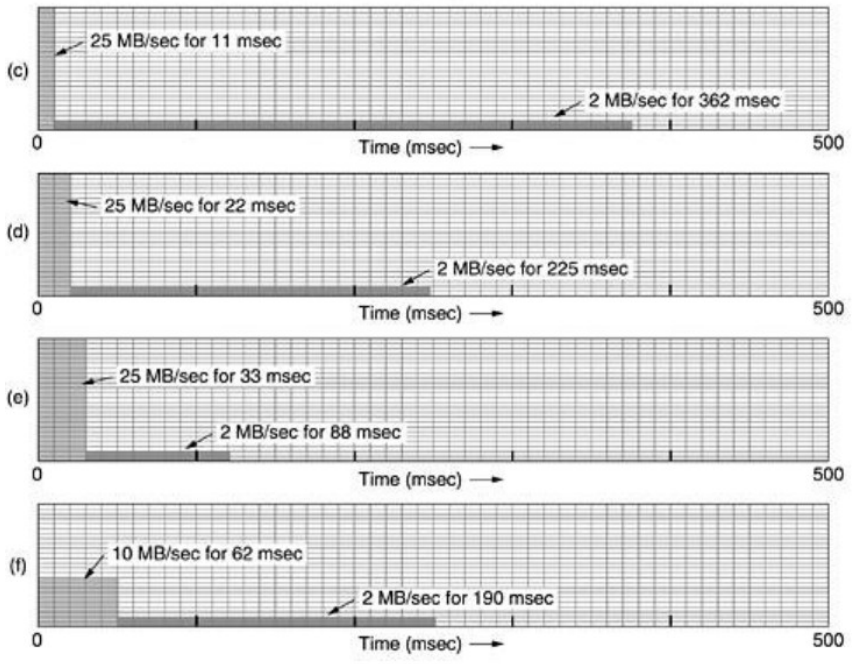
- Nekad je korisno malo ubrzati saobraćaj kada naiđe rafal podataka, kako bi se smanjila verovatnoća “prelivanja” - gubitka podataka
- Dozvoliti računarima koji trenutno nemaju ništa da emituju da rezervišu pravo
- Sistemski sat generiše po jedan žeton svakih **Δt**. Paket uzima po jedan žeton iz kofe kad se šalje
- Kofa sa žetonima uobličava neravnomerno poslate podatke u rafale određene maksimalne dužine. Koje dužine?
- ***S*** – trajanje rafala
- ***C*** – kapacitet kofe
- ***M*** – maksimalna brzina slanja iz kofe
- ***ρ*** – brzina generisanja žetona



$$C + \rho S = MS$$

Algoritam kofe sa žetonima (*token bucket*)

- Na prve tri slike je prikazan rad algoritma kofe sa žetonima za **C**=250KB, 500KB i 750KB.
- Propuštanje snažnih rafala može da bude problem. Rešava se tako što se bušna kofa postavi iza kofe sa žetonima. Na poslednjoj slici dat je slučaj kofe sa žetonima kapaciteta 500KB ispred bušne kofe brzine curenja 10MB/s



Rezervisanje resursa

- **Propusni opseg**
- **Prostor u baferima** – ako se želi bolji QoS, neki baferi se mogu rezervisati za određeni tok podataka
- **Procesorsko vreme**
  - Ako je vreme obrade jednog paketa 1μs, da li to znači da može da se obradi 10^6 paketa/s?
  - **λ** – prosečna brzina pristizanja paketa [paketa/s]
  - **μ** – prosečna brzina obrade paketa [paketa/s]
  - 1/**μ** – trajanje usluge u odsustvu konkurentnosti
  - **ρ=λ/μ** – iskorišćenje procesora
  - Koristi se Poasonova raspodela iz teorije redova

$$T = \frac{1}{\mu} \cdot \frac{1}{1 - \frac{\lambda}{\mu}} = \frac{1}{\mu} \cdot \frac{1}{1 - \rho}$$

- Primer: **λ**=950000 paketa/s, **μ**=1000000 paketa/s, tada je **ρ**=0.95, a paketi umesto 1μs kasne 20μs.

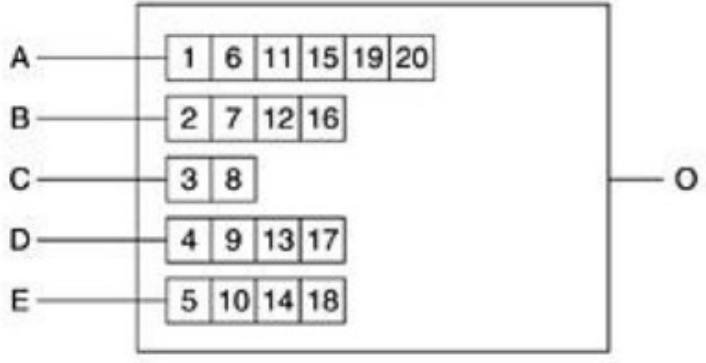
Kontrola pristupa i proporcionalno rutiranje

- **Odluka o prihvatanju ili odbijanju toka koji je opisan određenim karakteristikama**
- Odluka o prihvatanju ili odbijanju toka ne može se precizno doneti prema zahtevima za propusnim opegom, baferima i procesorskim vremenom (nepoznavanje bafera i CPU vremena, cenjkanje)
- Zbog toga se uvodi **specifikacija toka koja se sastoji iz 5 stavki**
- Preslikavanje specifikacija toka u rezervaciju određenih resursa nije standardizovano
- Jedno od pravila je da se nikada ne prelazi **50% zauzeća procesora**
- Što je strožija specifikacija, ruteri će je pre prihvatiti. Recimo brzina generisanja žetona 5MB/s, a veličina paketa varira od 50 do 1500B, brzina prenosa paketa varira između 3500 i 105000 paketa/s. Ruter će vrlo verovatno odbiti takav neodređeni tok.
- **Proporcionalno rutiranje** je upućivanje saobraćaja na više izlaznih linija

Parametar	Jedinica
Brzina generisanja žetona	Bajtovi/s
Kapacitet kofe sa žetonima	Bajtovi
Max.brzina slanja podataka	Bajtovi/s
Minimalna veličina paketa	Bajtovi
Maksimalna veličina paketa	Bajtovi

Raspoređivanje paketa

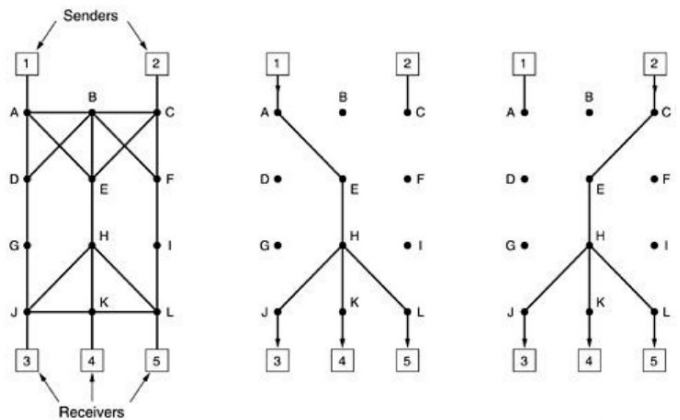
- Obrada paketa redom stizanja može da znači da **agresivan pošiljalac prigrabi njegov kapacitet!**
- **Algoritam za ravnopravnu obradu redova čekanja**. Ideja je da se formira red čekanja za svaki tok i da se redom obrađuju. Ova šema ima problem da veći propusni opseg dobijaju oni računari koji šalju veće pakete.
- Rešenje ovog problema je predložio Demers. Na slici je dato 5 paketa dužine od 2 do 6 bajtova.

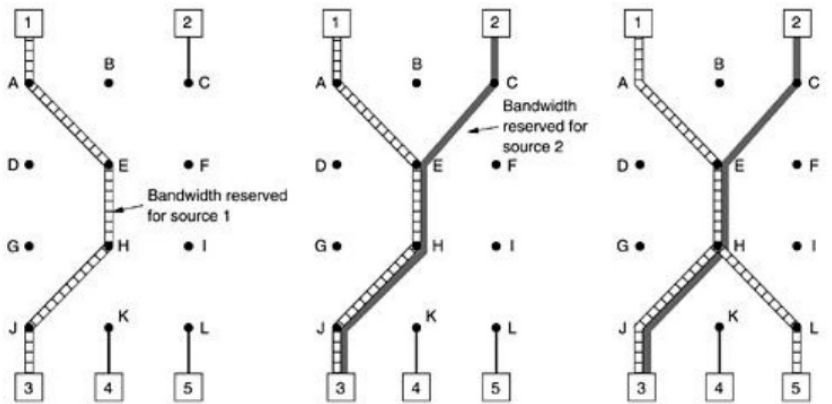


Packet	Finishing time
C	8
B	16
D	17
E	18
A	20

Integrisane usluge

- Multimedijalne aplikacije koje rade kako sa jednosmernim, tako i sa višesmernim slanjem
- Algoritmi zasnovani na toku podataka
- **RSVP – ReSource reserVation Protocol**
- Radi sa grupama i razgranatim stablima
- Svaki primalac iz grupe može da pošalje poruku izvorištu. Poruka se prosleđuje algoritmom za ispitivanje izvorišta, a svaki ruter na putu vidi zahtev i rezerviše traženi propusni opseg.





Diferencirane usluge

- **Ekspresno prosleđivanje** – npr. ako se na ekspresnoj liniji očekuje saobraćaj od 10%, rezerviše se 20% propusnog opsega
- **Garantovano prosleđivanje** – 4 klase usluga sa 3 prioriteta za odbacivanje kod zagušenja:
  1. **Svrstavanje paketa u jednu od 4 klase**
  2. **Obeležavanje paketa prema klasama** (IP zaglavlje ima 8-bitno polje *Service Type*)
  3. **Prolazak kroz filer** za oblikovanje/odbacivanje, zasnovan na bušnoj kofi ili kofi sa žetonima