DISTRIBUIRANI ALGORITMI I SISTEMI

Iz kursa CSCE 668
Proleće 2014

Autor izvorne prezentacije:
Prof. Jennifer Welch

Specifikacije usluga slanja svima (1)

- Osnovna specifikacija usluge slanja svima obuhvata njene ulaze i izlaze:
- Ulazi: bc-send$_i$(m)
  - Ulaz u uslugu slanja svima
  - p_i želi da pošalje m svim procesima (uključujući sebe)
- Izlazi: bc-recv$_i$(m,j)
  - Izlaz iz usluge slanja svima
  - Usluga isporučuje poruku m, poslatu od p_j, svim p_i

Specifikacije usluga slanja svima (2)

- Sekvenca ulaza i izlaza (bc-sends i bc-recvs) je prihvatljiva akko postoji preslikavanje κ od svakog bc-recv$_i$(*m,j*) događaja do ranijeg bc-send$_j$(*m*) događaja tako da:
  - κ je „dobro definisana“: svaka por. iz bc-recv je predhodno bila poslata sa bc-sent (Integritet)
  - κ ograničeno na bc-recv$_i$ događaje, za svako *i*, je „jedan-na-jedan“: ni jedna por. iz bc-recv se ne prima više od jedanput na svakom proc. (Nema duplikata)
  - κ ograničeno na bc-recv$_i$ događaje, za svako *i*, je „na“: svaka bc-sent por. se prima na svakom proc. (Životnost)

Svojstva redosleda

- Ponekad je potrebno da usluga slanja svima takođe obezbedi neku vrstu garancije za *redosled* u kom se poruke isporučuju.
- Mogu se dodati dopunska ograničenja na κ:
  - **FIFO iz istog izvora (SSF)** ili
  - **Potpuno uređenje** ili
  - **Uzročno uređenje**

FIFO iz istog izvora

- Za sve poruke m_1 i m_2 i sve p_i i p_j, ako p_i šalje m_1 pre nego šalje m_2, i ako p_j prima m_1 i m_2, onda p_j prima m_1 pre nego primi m_2.

- Pažljivo sročeno da se izbegne zahtev da obe poruke budu primljene.
  - To je u nadležnosti svojstva životnosti

Potpuno uređenje

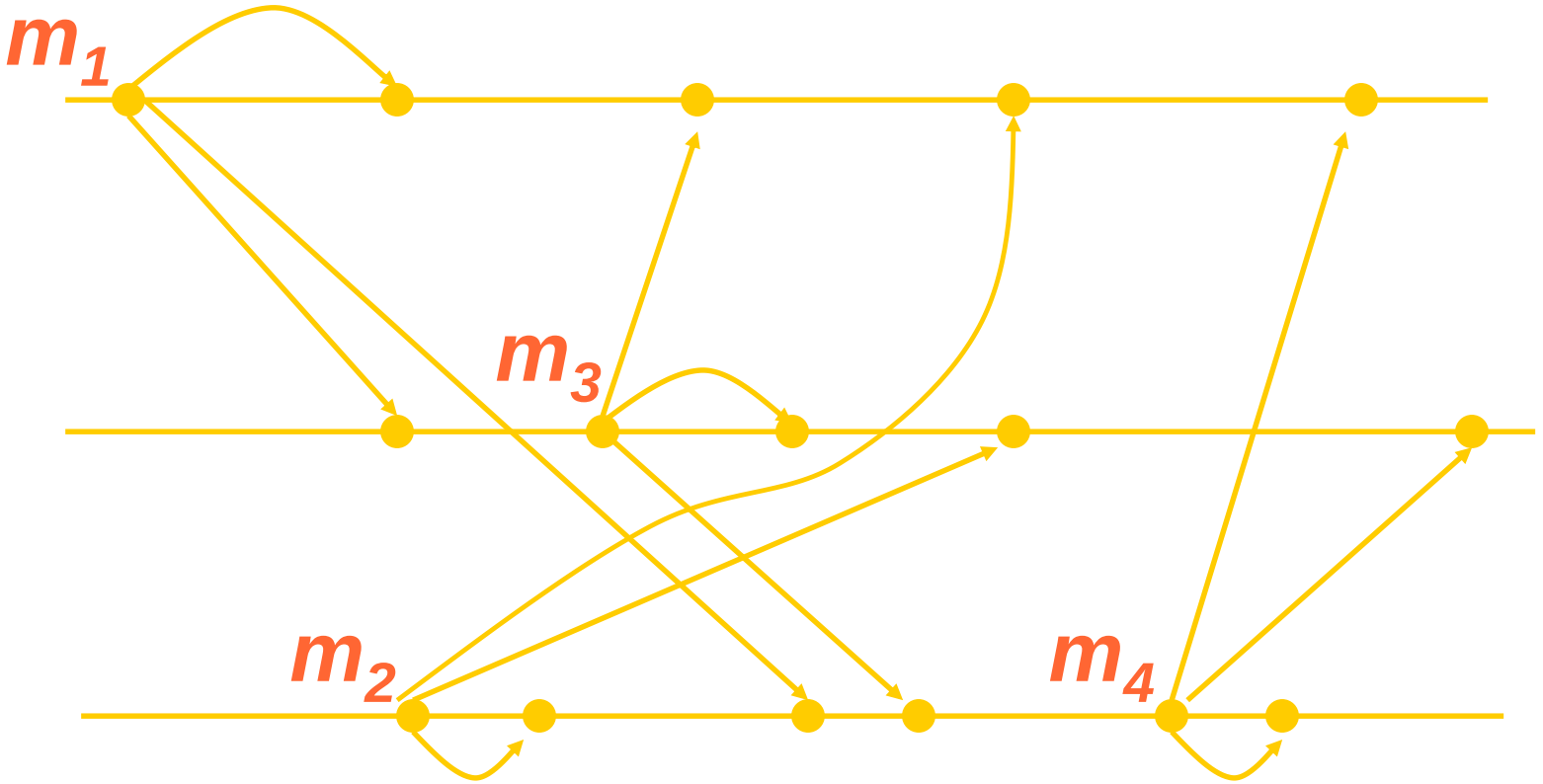
- Za sve poruke m_1 i m_2 i sve p_i i p_j, ako p_i i p_j prime obe poruke, onda ih oni primaju u istom redosledu.

- Pažljivo sročeno da se izbegne zahtev da obe poruke budu primljene na oba procesora.
  - To je u nadležnosti svojstva životnosti

„Desila se pre“ za por. upućene svima

- Ranije smo definisali relaciju „desio se pre" za događaje.
- Sad proširujemo tu def. na por. upućene svima.
- Predpostavimo da je sva komunikacija kroz uslugu slanja svima.
- Poruka m_1 se desila pre poruke m_2 ako:
  - bc-recv događaj za m_1 se desio pre (u starom smislu) od bc-send događaja za m_2, ili
  - m_1 i m_2 je poslao isti procesor i m_1 je poslata pre m_2.

Primeri relacije „desila se pre“ za poruke upućene svima



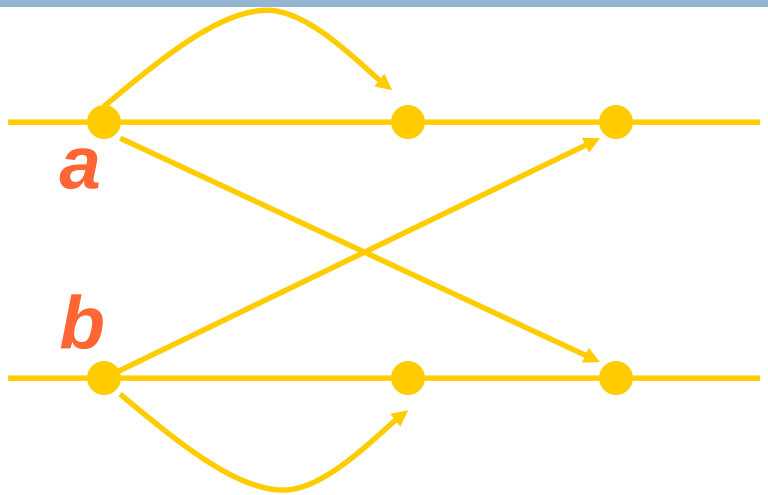
m_1 se desila pre m_3 i m_4
m_2 se desila pre m_4
m_3 se desila pre m_4

Uzročno uređenje

- Za sve poruke m_1 i m_2 i sve p_i, ako se m_1 desila pre m_2, i ako p_i primi i m_1 i m_2, onda p_i prima m_1 pre nego primi m_2.

- Pažljivo sročeno da se izbegne zahtev da obe poruke budu primljene.
  - To je u nadležnosti svojstva životnosti

Primer 1: Uzročno uređenje

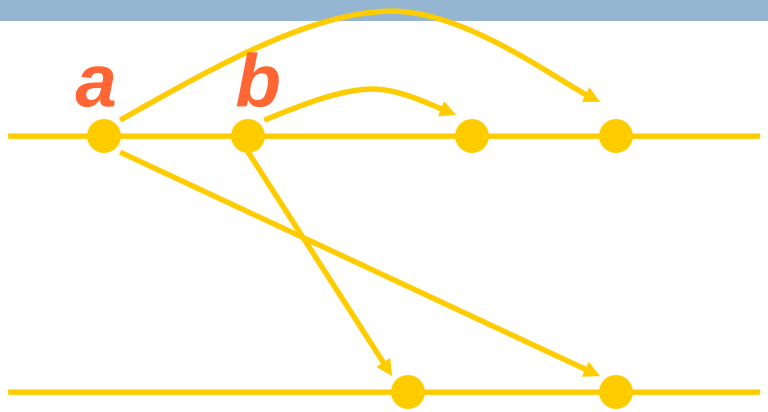


FIFO iz istog izvora? Da.

potpuno uređenje? Ne.

uzročno uređenje? Da.

Primer 2: Potpuno uređenje

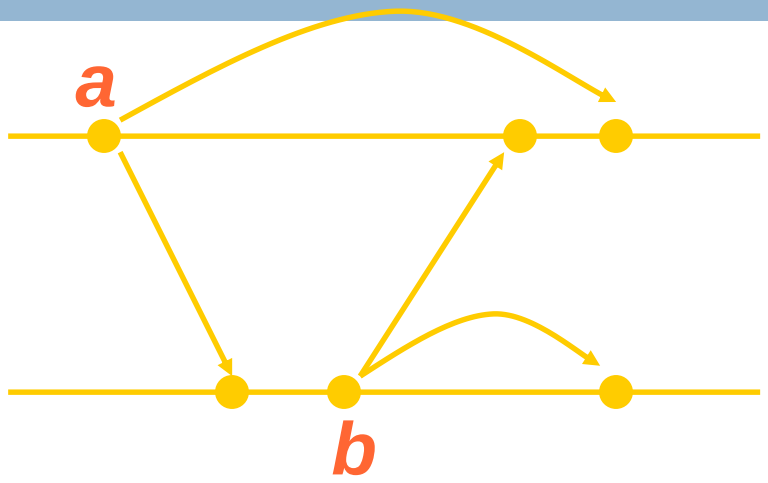


FIFO iz istog izvora? Ne.

potpuno uređenje? Da.

uzročno uređenje? Ne.

Primer 3: FIFO iz istog izvora



FIFO iz istog izvora? Da.

potpuno uređenje? Ne.

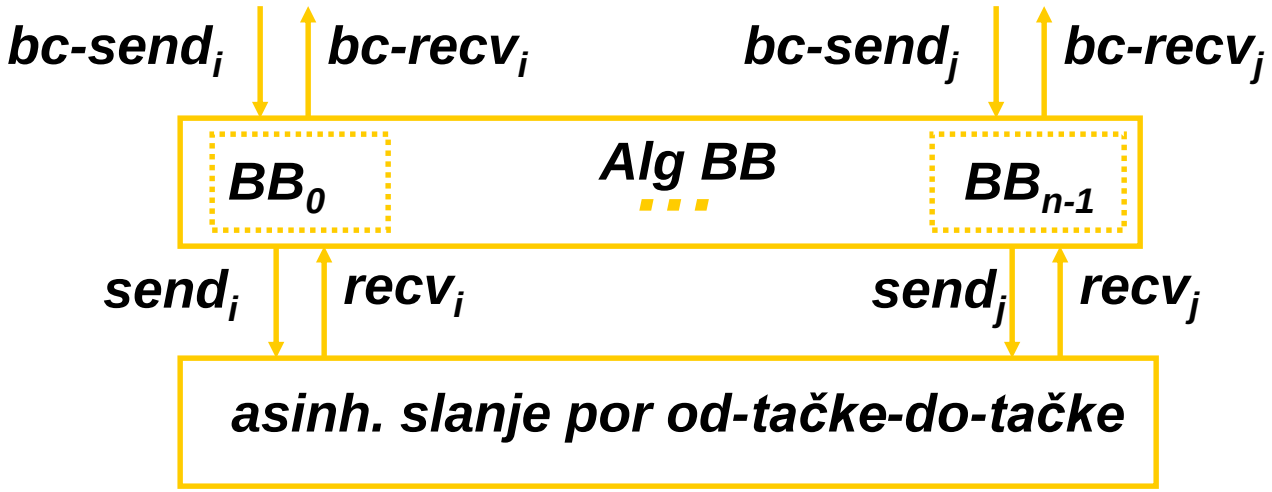
uzročno uređenje? Ne.

Algoritam BB za simulaciju Osnovne usluge na vrhu komuni. sis. od-tačke-do-tačke

- Kad se desi bc-send$_i(m)$:
  - p_i šalje posebnu kopiju m svakom procesoru (i sebi) koristeći komunikacioni sistem za slanje poruka od-tačke-do-tačke

- Kada p_i može da izvede bc-recv$_i(m)$?
  - Onda kad primi m od komunikacionog sistema za slanje poruka od-tačke-do-tačke

Simulacija Osnovne usluge slanja svima (BB)



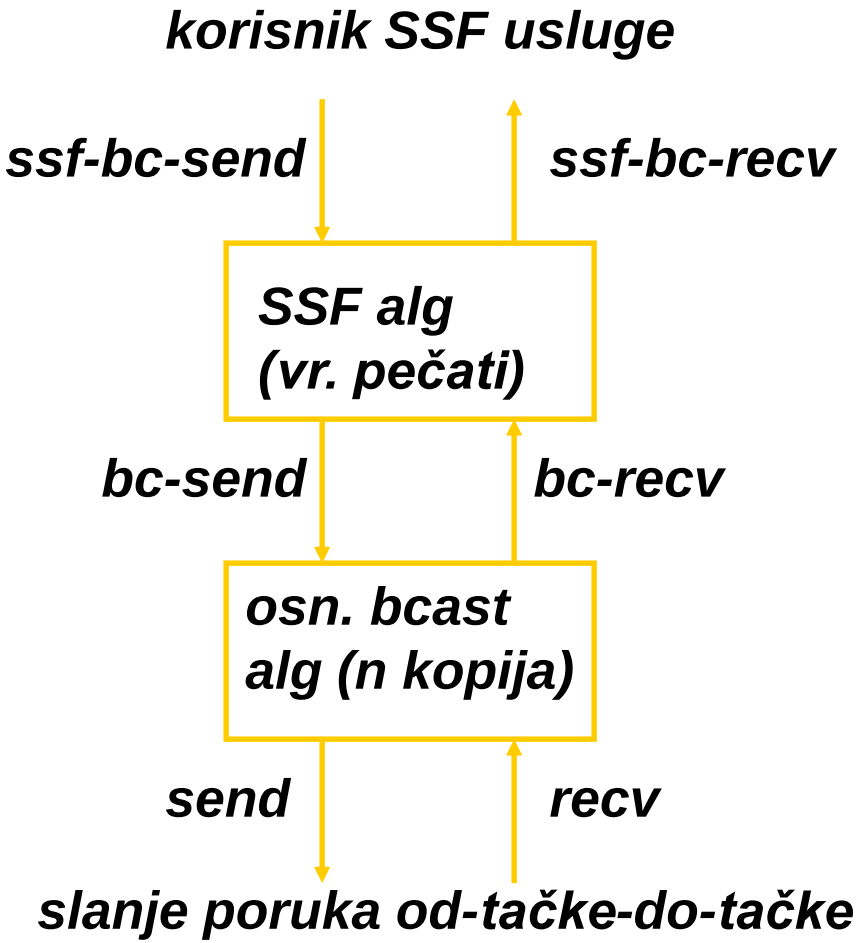
Korektnost Algoritma BB

- Predpost. da je sistem za slanje poruka od-tačke-do-tačke korektan (tj. da zadovoljava svoju specifikaciju).
- Treba proveriti da li algoritam BB zadovoljava:
  - Integritet
  - Nema duplikata
  - Životnost

Algoritam FIFO iz istog izvora

- Predpost. da sistem ispod obezbeđuje osnovnu uslugu slanja svima.
- Kad se desi ssf-bc-send$_i(m)$:
  - p_i koristi postojeću osnovnu uslugu da pošalje m zajedno sa njenim rednim brojem
  - p_i povećava redni broj za 1 za svaku sledeću poruku
- Kada p_i može da izvede ssf-bc-recv$_i(m)$?
  - Onda kad od p_j primi m sa rednim brojem T i kad je primio sve poruke od p_j sa manjim rednim brojevima

Algoritam FIFO iz istog izvora



Asimetričan algoritam za Potpuni redosled (1/2)

- Predpost. da sistem ispod obezbeđuje osnovnu uslugu slanja svima.
- Jedan proc. je centralni koordinator p_c
- Kad se desi to-bcast$_i$(m):
  - p_i šalje m ka p_c (ili sistem ispod poseduje mehanizam od-tačke-do-tačke, ili svi osim p_c ignorišu tu poruku)
- Kad p_c iz sistema ispod primi poruku m od p_i :
  - Dodeli poruci m redni broj i pošalje je svima

Asimetričan algoritam za Potpuni redosled (2/2)

- Kada p_i može da izvede to-bc-recv(*m*)?
  - Onda kad p_i primi *m* sa rednim brojem *T* i kad je primio sve poruke sa manjim rednim brojevima

Osobine asimetričnog algoritma

- Jednostavan
- Samo zahteva osnovnu uslugu za slanje svima
- Ali, p_c je usko grlo
- Sledi alternativan pristup…

Simetričan algoritam za Potpuni redosled (1/2)

- Predpost. da sistem ispod obezbeđuje uslugu slanja svima FIFO iz istog izvora.
- Svaki proc. označava svaku por. koju šalje sa vremenskim pečatom (u rastućem redosledu).
  - U slučaju istih vrednosti pečata, koriste se id-ovi procesora
- Svaki proc. održava vektor procena vremenskih pečata drugih procesora:
  - Ako procena od p_i za p_j iznosi k, onda p_i kasnije nikad neće primiti poruku od p_j sa pečatom k.
  - Procene se ažuriraju na osnovu primljenih poruka (pored redovnih postoje i specijalne por. za ažuriranje pečata)

Asimetričan algoritam za Potpuni redosled (2/2)

- Svaki proc. održava svoje pečate da budu ≥ od svih svojih procena:
  - Onda kad p_i treba da poveća svoj pečat zbog prijema poruke, on šalje specijalnu por. za ažuriranje pečata
- Procesor može da isporuči por. sa pečatom *T* ako je svaki elem. u vektoru procena barem jednak *T*.

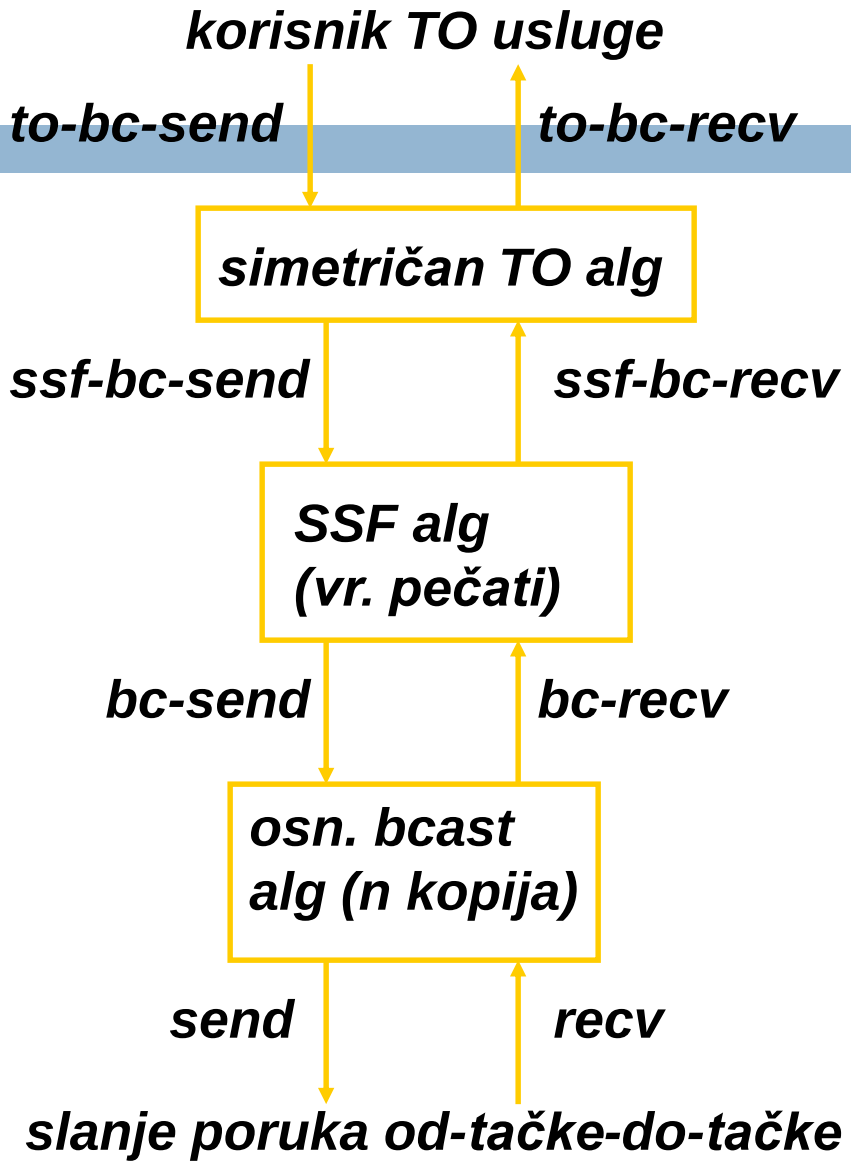
Simetričan algoritam

kad se desi to-bc-send$_i$*(m)*:
  ts[i]++
  dodaj (*m,ts[i],i)* u *pending*
  pozovi ssf-bc-send$_i$*((m,ts[i]))*

kad se desi ssf-bc-recv$_i$*((m,T))*
od p_j :
  ts[j] := T
  dodaj *(m,T,j)* u *pending*
  if *T* > *ts[i]* then
    ts[i] := T
    pozovi ssf-bc-send$_i$*("ts-up",T)*

pozovi to-bc-recv$_i$*(m,j)* kad je:
  (*m,T,j*) elem. u *pending* sa
    najmanjim (*T,j)*
  T ≤ *ts[k]* for all *k*
rezultat: ukloni (*m,T,j*) iz
    pending

kad se desi ssf-bc-recv$_i$*("ts-up",T)*
od p_j :
  ts[j] := T


korisnik TO usluge
to-bc-send
to-bc-recv
simetričan TO alg
ssf-bc-send
ssf-bc-recv
SSF alg
(vr. pečati)
bc-send
bc-recv
osn. bcast
alg (n kopija)
send
recv
slanje poruka od-tačke-do-tačke

Korektnost simetričnog algoritma (1/3)

Lema (5.2): Vremenski pečati dodeljeni porukama čine potpuno uređenje (isti pečati se uređuju po id proc).

Teorema (5.3): Simetričan algoritam simulira uslugu slanja svima sa potpunim redosledom.

Dokaz: Moramo pokazati da izlazi iz simetričnog algoritma zadovoljavaju 4 svojstva u svakom prihvatljivom izvršenju (predpostavlja se da je korišćena SSF usluga korektna).

Korektnost simetričnog algoritma (2/3)

Integritet: sledi iz istog svojstva za ssf-bcast.

Nema duplikata: sledi iz istog svojstva za ssf-bcast.

Životnost:

- Predpost. radi kontradikcije da p_i ima neki elem. (m,T,j) zauvek zaglavljen u svom skupu *pending*, gde je (T,j) najmanji pečat od svih zaglavljenih elemenata.
- Zašto je (m,T,j) zaglavljen na p_i? Zato što je procena od p_i za pečat nekog p_k zaglavljena na nekoj vrednosti $T' < T$.
- Ali to bi značilo da ili p_k nikada ne prima (m,T,j) ili da p_i nikad ne prima poruka za ažuriranje pečata od p_k, koju p_k šalje nakon prijema (m,T,j), što je u kontradikciji sa korektnošću ssf-bcast.

Korektnost simetričnog algoritma (3/3)

Potpuno uređenje: Neka p_i izvede to-bc-recv za por *m* sa pečatom (T,j), i kasnije izvede to-bc-recv za por m' sa pečatom (T',j'). Onda treba pokazati: $(T,j) < (T',j')$.

- Prema kodu, ako je (m',T',j') u skupu *pending* od p_i kad p_i izvede to-bc-recv za *m*, onda $(T,j) < (T',j')$.
- Neka (m',T',j') tada još nije u skupu *pending* od p_i.
- Kad p_i izvede to-bc-recv za *m*, preduslov je da: $T \leq ts[j']$. Zato je p_i primio por. od $p_{j'}$ sa pečatom $\geq T$.
- Prema svojstvu SSF, svaka sledeća por. koju p_i primi od $p_{j'}$ ima pečat $> T$, pa T' mora biti $> T$.

Algoritam uzročnog redosleda (1/2)

- Simetričan algoritam potpunog redosleda garantuje uzročan redosled:
  - redosled vremenskih pečata proširuje redosled „desila se pre“ nad porukama.
- Uzročno uređenje se može napraviti i bez viška poruka za potpuno uređenje, pomoću algoritma na bazi vektorskih satova…

Algoritam uzročnog redosleda (2/2)

Kod za p_i :

kad se desi co-bc-send$_i$*(m)*:
  vt[i]++
  pozovi co-bc-recv$_i$*(m)*
  pozovi bc-send$_i$*((m,vt))*

kad se desi bc-recv$_i$*((m,w))* od p_j :
  dodaj *(m,w,j)* u *pending*

pozovi co-bc-recv$_i$*(m,j)* kad:
  (*m,w,j*) je u *pending*
  w[j] = vt[j] + 1
  w[k] ≤ vt[k] for all *k ≠ j*
rezultat:
  ukloni (*m,w,j*) iz *pending*
  vt[j]++

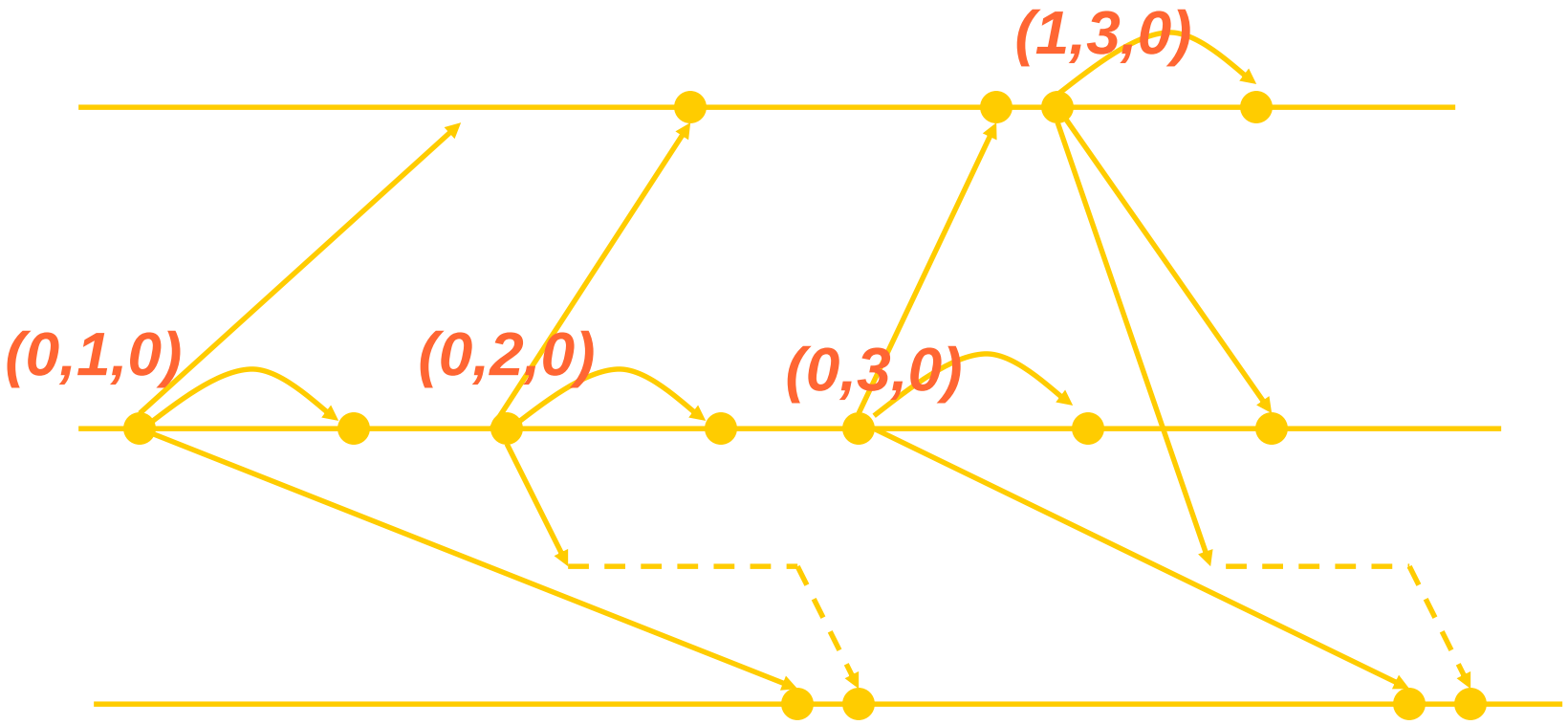
Napomena: vt[j] pamti koliko por. od p_j je p_i primio sa co-bc-recv

Osobine algoritma uzročog redosleda

- Vektorski satovi su izvedeni malo drugačije nego u slučaju od-tačke-do-tačke.
- Tamo smo koristili indirektnu (tranzitivnu) informaciju o porukama primljenim od drugih procesora.
- Ovde nam to ne treba, pošto svaki proc. nekad konačno prima svaku poruku direktno.

Primer za algoritam uzročnog redosleda

- Algoritam zakašnjava isporuku nekih por. da svojstvo uzročnog redosleda ne bi bilo narušeno.



Korektnost algoritma uzročnog redosleda (Skica)

Lema (5.6): Promenljiva *vt* je vektorski sat.

Teorema (5.7): Algoritam simulira uslugu slanja svima sa uzročnim redosledom, ako sistem ispod zadovoljava specifikaciju osnovne usluge.

Dokaz: Integritet i Nema duplikata slede iz istih svojstava osnovne usluge. Životnost zahteva dodatnu argumentaciju. Uzročni redosled sledi iz gornje leme.

Pouzdano slanje svima

- Šta se zahteva od usluge slanja kad neki od proc. mogu biti neispravni?
- Specifikacije se razlikuju od onih za slučaj sa ispravni proc. na dva načina:
  1. Indeksi procesora se dele na „neispravne" i „ispravne"
  2. Svojstvo životnosti se menja…

Specifikacija pouzdanog slanja svima

- Životnost bez otkaza: Svaka por. koju je poslao *ispravan* proc. se nakada konačno prima na svim *ispravnim* procesorima.
- Životnost sa otkazima: Svaka por koju je poslao *neispravan* proc se prima ili na svim *ispravnim* proc ili ni na jednom od njih.

Diskusija Specifikacije pouzdanog slanja svima

- Specifikacija je nezavisna od konkretnog modela otkaza.
- Razmotrićemo samo implementaciju za otkaze tipa ispada (crash).
- Nema garancija o tome koje poruke se primaju na procesorima u otkazu.
- Ova spec. se može proširiti za razne varijante redosleda :
  - Poruke koje se primaju na ispravnim proc moraju poštivati relevantno svojstvo redosleda.

Spec. Sistema za slanje por. od-tačke-do-tačke koji je podložan greškama (1/2)

- Da bi mogli da projektujemo algoritam za pouzdano slanje svima, moramo znati šta možemo očekivati od korišćenog komunikacionog sistema.
- Modifikujemo predhodnu spec. Sis. za slanje por. od-tačke-do-tačke za slučaj sa greškama:
  1. Delimo indekse proc. na „ispravne" i „neispravne"
  2. Menjamo svojstvo životnosti...

Spec. Sistema za slanje por. od-tačke-do-tačke koji je podložan greškama (2/2)

- Životnost bez otkaza: svaka por. poslata od *ispravnog* proc. bilo kom *ispravnom* proc. se nekad konačno prima.

Primetimo da ovo ne nameće nikakve uslove o isporuci poruka neispravnim procesorima.

Algoritam pouzdanog slanja svima

kad se desi rel-bc-send$_i$(*m*) :
  pozovi send$_i$(*m*) za sve proc.

Kad se desi recv$_i$(*m*) od p_j :
  if *m* nije već primljena then
    pozovi send$_i$(*m*) za sve proc.
    pozovi rel-bc-recv$_i$(*m*)

Korektnost Algoritma pouzdanog slanja svima

- Integritet: sledi iz istog svojstva sistema ispod.
- Nema duplikata: sledi iz istog svojstva sistema ispod i provere da ova por. već nije primljena.
- Životnost bez otkaza: sledi iz istog svojstva sistema ispod.
- Životnost sa otkazima: sledi iz istog svojstva sistema ispod.