Diplomski studij

Informacijska i komunikacijska tehnologija:

Telekomunikacije i informatika

Računarstvo:

Programsko inženjerstvo i informacijski sustavi

Računarska znanost

Ak.g. 2009./20010.

Raspodijeljeni sustavi

7.

Sinkronizacija procesa u vremenu

4.11.2009.

Sadržaj predavanja

- **slobodno smijete:**
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **remiksirati** — prerađivati djelo
- **pod sljedećim uvjetima:**
  - **imenovanje**. Morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno**. Ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima**. Ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela. Najbolji način da to učinite je linkom na ovu internetsku stranicu.
Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.
Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

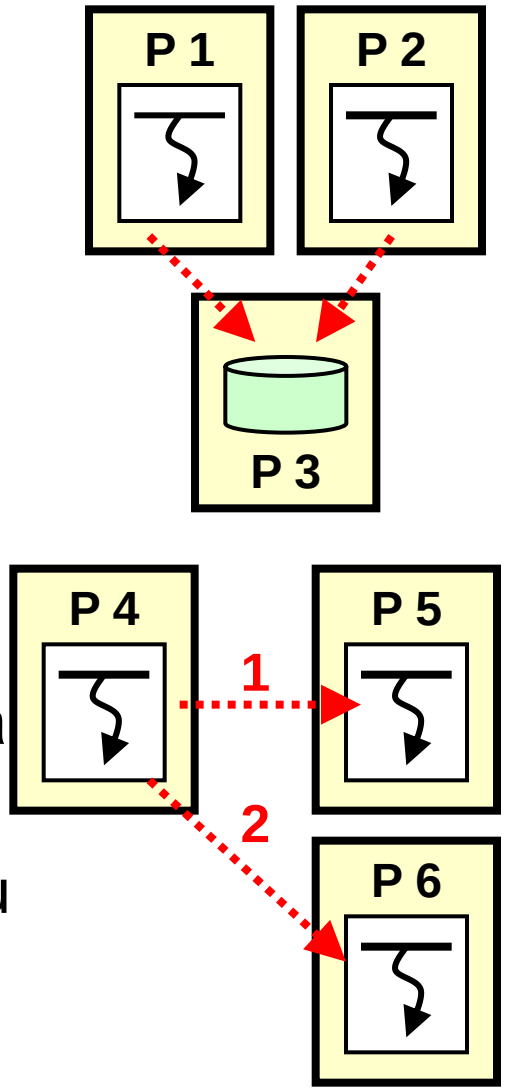
Tekst licencije preuzet je s http://creativecommons.org/.

Sadržaj predavanja

- Potreba za sinkronizacijom procesa
- Primjena sata u jednoprocesorskoj okolini
- Primjena sata raspodijeljenoj okolini
- Sinkronizacija tijeka izvođenja procesa
- Međusobno isključivanje procesa

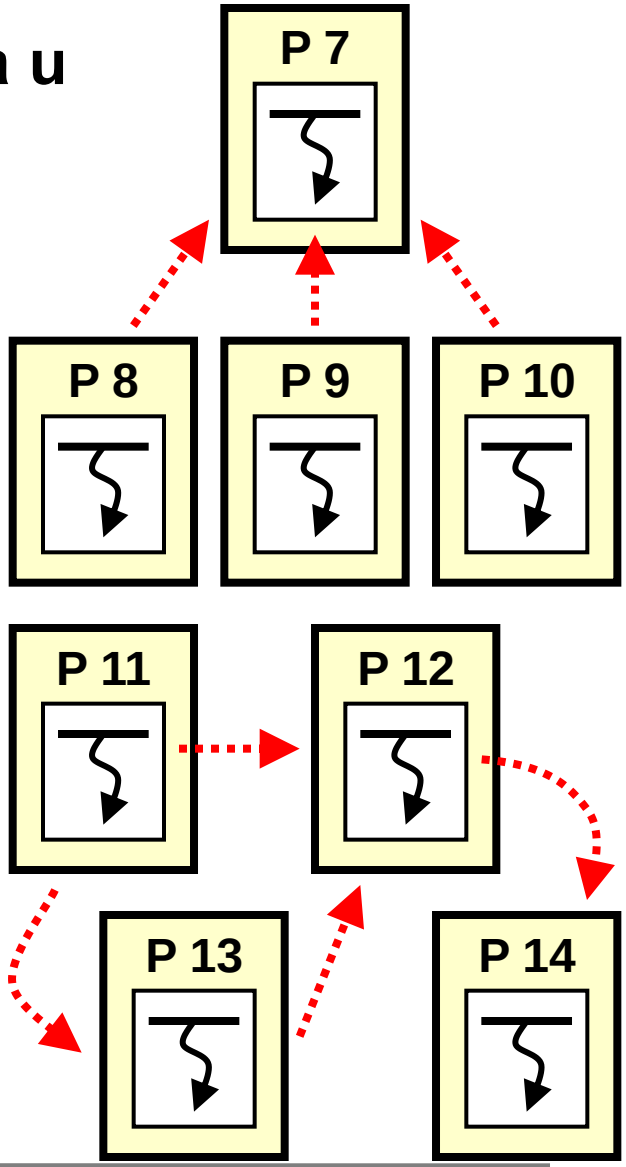
Potreba za sinkronizacijom procesa

- **Uporaba dijeljenih sredstava u raspodijeljenoj okolini**
  - Procesi istodobno ostvaruju pristup dijeljenim sredstvima
  - Potrebno je ostvariti pristup dijeljenom sredstvu na međusobno isključiv način
  - Dogovor o redoslijedu ostvarivanja pristupa
- **Usaglašavanje vremenskog redoslijeda akcija**
  - Vremenski redoslijed izvođenja akcija u raspodijeljenoj okolini



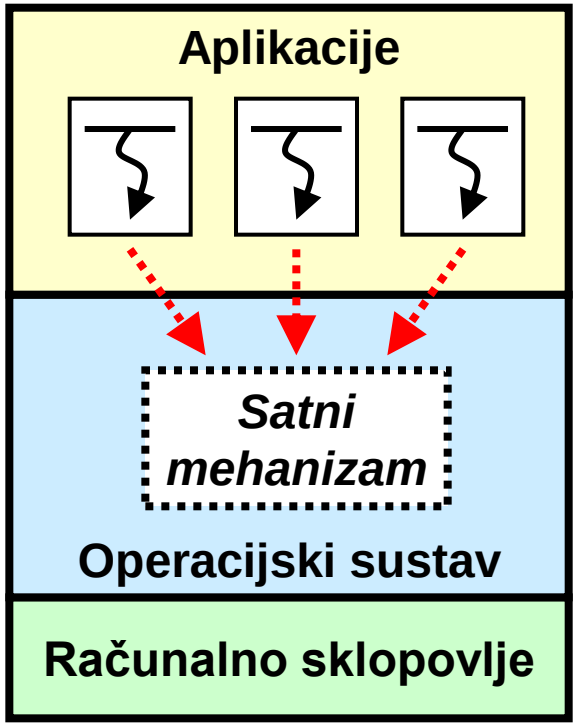
Potreba za sinkronizacijom procesa

- **Nadgledanje i upravljanje zadaćama u raspodijeljenoj okolini**
  - Odabir upravljačkog procesa
  - Upravljački proces nadzire i određuje aktivnosti radnih procesa u raspodijeljenoj okolini
- **Uspostava suradnje skupa procesa raspodijeljenoj okolini**
  - Ostvarivanje vremenski i prostorno usklađenog raspodijeljenog tijeka izvođenja



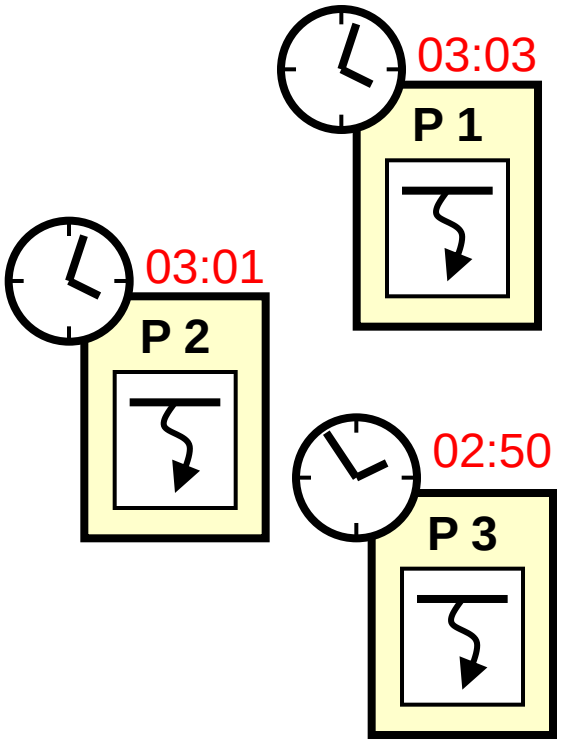
Primjena sata u jednoprocesorskoj okolini

- **Satni mehanizam operacijskog sustava**
- **Aplikacije**
  - Procesi koriste i upravljaju mehanizmom sata
  - Primjena programskih knjižnica za uporabu satnog mehanizma
- Zatvorena okolina
  - Predvidiva vremenima izvođenja procesa
  - Jednostavnija sinkronizacija procesa u vremenu



Primjena sata u raspodijeljenoj okolini

- **Svako računalo ima vlastiti satni mehanizam**
  - Satovi nisu usklađeni
  - Satovi imaju različiti takt
  - Satovi imaju različita odstupanja
- **Usuglašavanje vremena**
  - Fizički sat u raspodijeljenoj okolini
  - Logički sat u raspodijeljenoj okolini



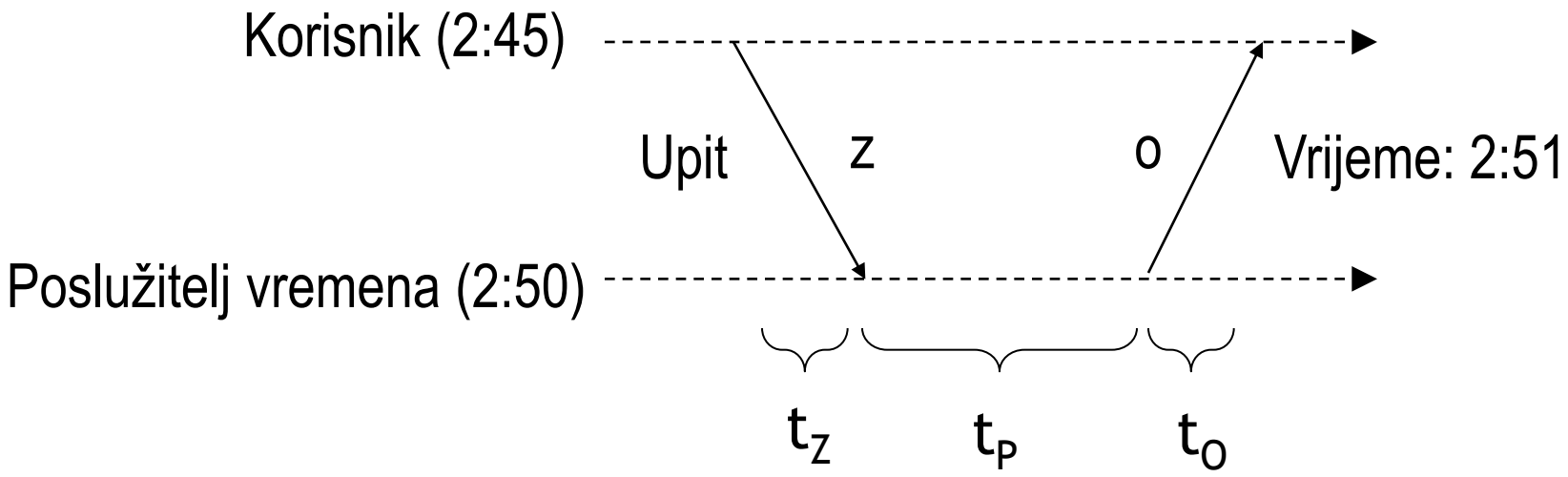
Fizički sat u raspodijeljenoj okolini

- **Cristian algoritam**
  - Primjena poslužitelja s točnim vremenom
  - Dohvaćanje informacije o vremenu prema potrebi

- **Berkeley algoritam**
  - Primjena upravitelja vremena
  - Periodičko odašiljanje informacije o vremenu

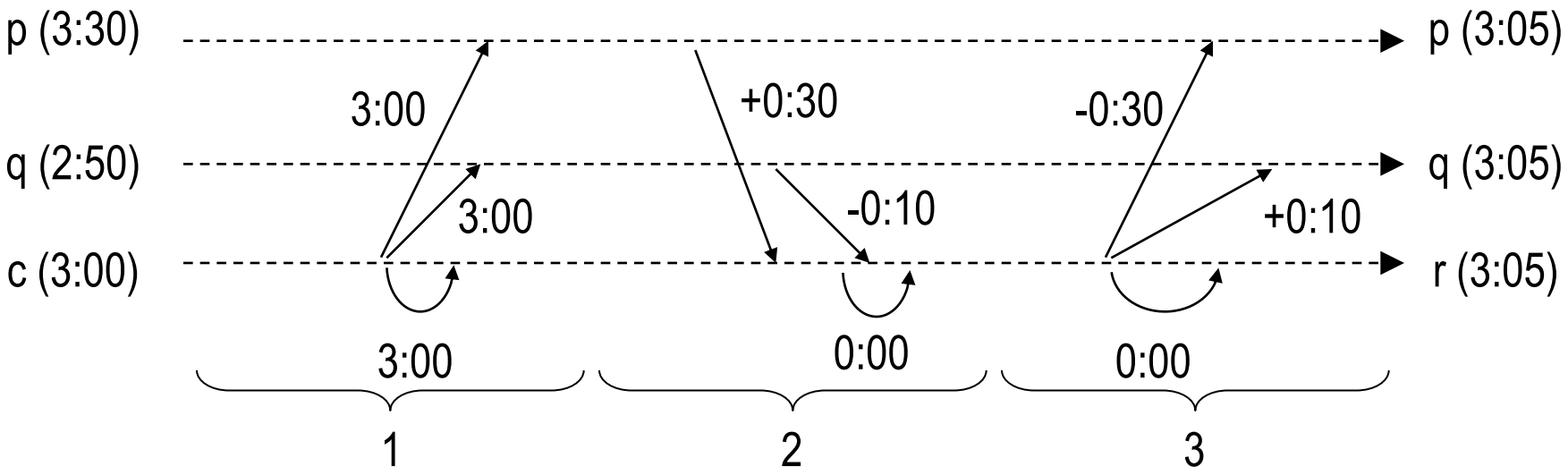
Christian algoritam

- **Primjena poslužitelja vremena**
- **Koraci algoritma**
  1) Korisnički proces upućuje zahtjev za dohvat vremena (z)
  2) Proces poslužitelj šalje trenutno vrijeme (o)



Berkeley algoritam

- **Primjena upravitelja vremena**
- **Koraci algoritma**
  1) Upravljački proces *c* šalje vrijeme procesima *p*, *q*, *c*
  2) Procesi *p*, *q*, *c* šalju razliku vremena upravljačkom procesu *c*
  3) Upravljački proces *c* šalje pomak procesima *p*, *q*, *c*



Logički sat u raspodijeljenoj okolini

- **Skup fizičkih satnih mehanizama**
  - Satni mehanizmi su potpuno nezavisni
- **Usklađivanje globalnog tijeka vremena**
  - Primjena logičkih oznaka vremena
- **Vrste logičkih oznaka**
  - Skalarne oznake vremena
  - Vektorske oznake vremena

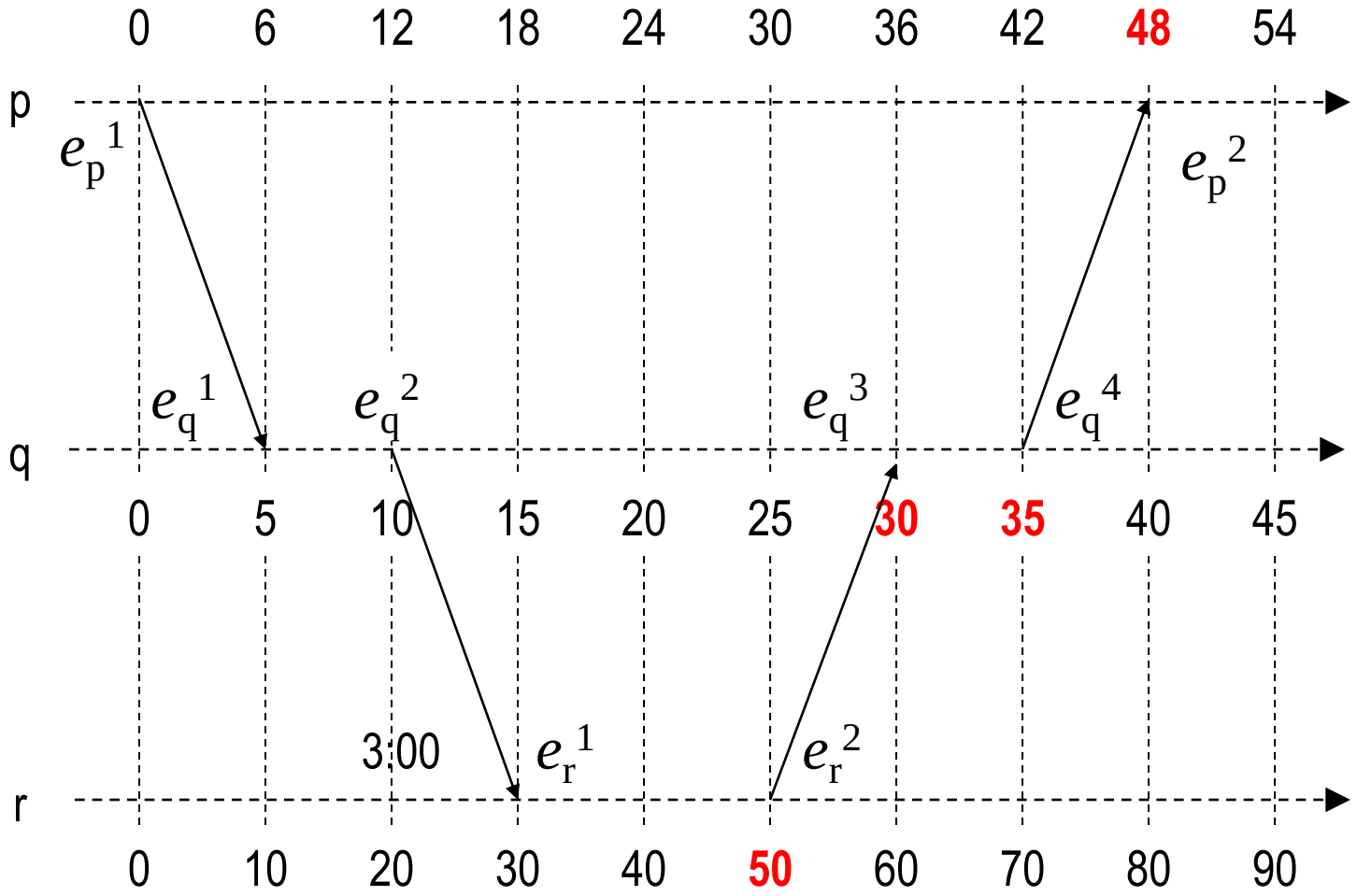
Primjena fizičkih satnih mehanizama

$e_p^1 \rightarrow e_q^1 \quad e_q^2 \rightarrow e_r^1$

$e_r^2 \rightarrow e_q^3 \quad e_q^4 \rightarrow e_p^2$

$T(e_r^2) > T(e_q^3)$

$T(e_q^4) > T(e_p^2)$



Skalarne oznake vremena

- **Globalno logičko vrijeme**
  - Sva računala na jednak način bilježe tijek globalnog logičkog vremena

- **Oznake logičkog vremena**
  - Svakoj akciji *a* koju provode procesi u raspodijeljenoj okolini pridružena je jedinstvena oznaka vremena T(*a*)
  - Ako za događaj *a* i *b* vrijedi uzročna relacija $a \rightarrow b$ tada vrijedi da je akcija *a* ostvarena u vremenu prije akcije *b* [T(*a*) < T(*b*)]

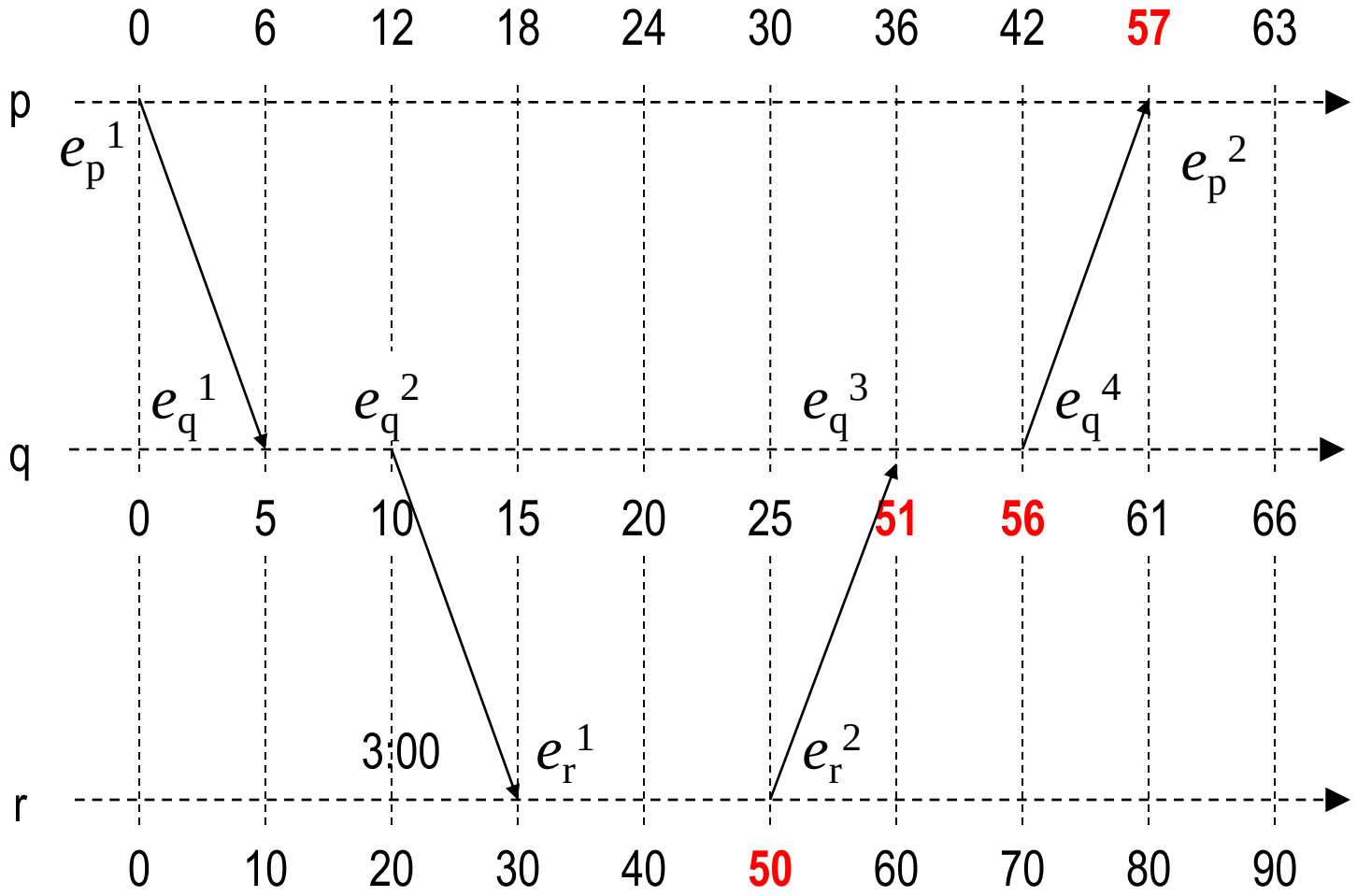
Primjer uporabe skalarnih oznaka vremena

$e_p^1 \rightarrow e_q^1$ $e_q^2 \rightarrow e_r^1$

$e_r^2 \rightarrow e_q^3$ $e_q^4 \rightarrow e_p^2$

$T(e_r^2) < T(e_q^3)$

$T(e_q^4) < T(e_p^2)$

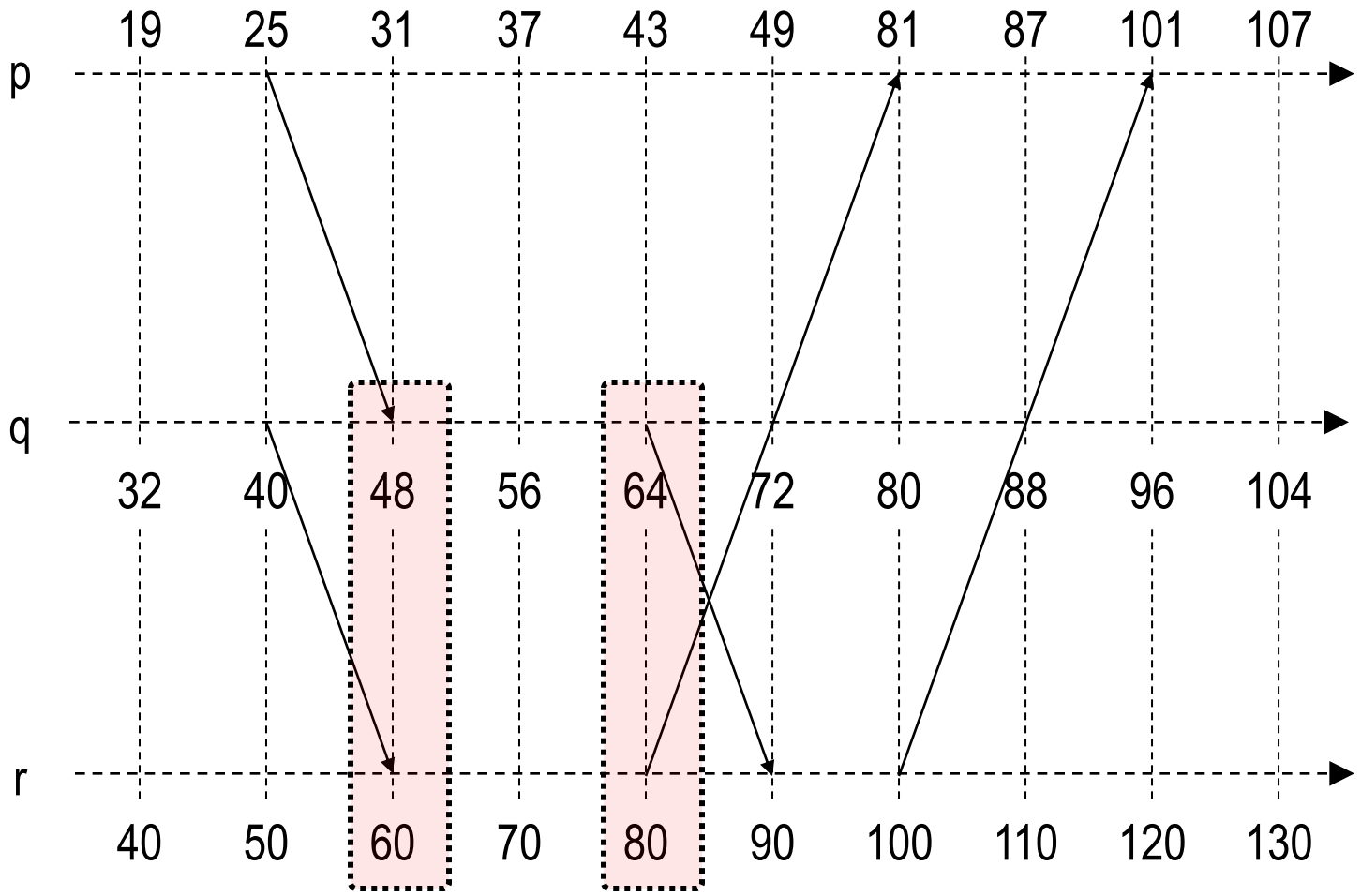


Skalarne oznake vremena

- **Prednosti primjene skalarnih oznaka**
  - Tijek vremena zasnovan je na jednostavnom modelu
  - Svi procesi usklađeni su s globalnim tijekom vremena
  - Usuglašeni su vremenski trenutci nastupanja akcija u raspodijeljenoj okolini

- **Nedostatci primjene skalarnih oznaka**
  - Ako za događaje *a* i *b* vrijedi da je vremenska oznaka od *a* manja od vremenske oznake od *b*, to ne povlači nužno da je događaj *a* nastupio u vremenu prije događaja *b*
  - $T(a) < T(b)$ *ne povlači* $a \rightarrow b$

Nedostatak primjene skalarnih oznaka



Vektorske oznake vremena

- **Vektorska oznaka opisuje uzročno-posljedične veze između događaja u vremenu**
  - Polje elemenata $V[N]$ koje opisuje broj akcija provedenih na skup ud N računala u raspodijeljenoj okolini
  - Računala razmjenjuju vektorske oznake tijekom razmjene poruka

- **Vektorska oznaka**
  - $V_p[i]$ sadrži broj akcija koje je ostvario proces P_p
  - $V_p[m]$ sadrži broj akcija za koje proces P_p zna da su ostvarene od strane procesa P_m

Vektorske oznake vremena

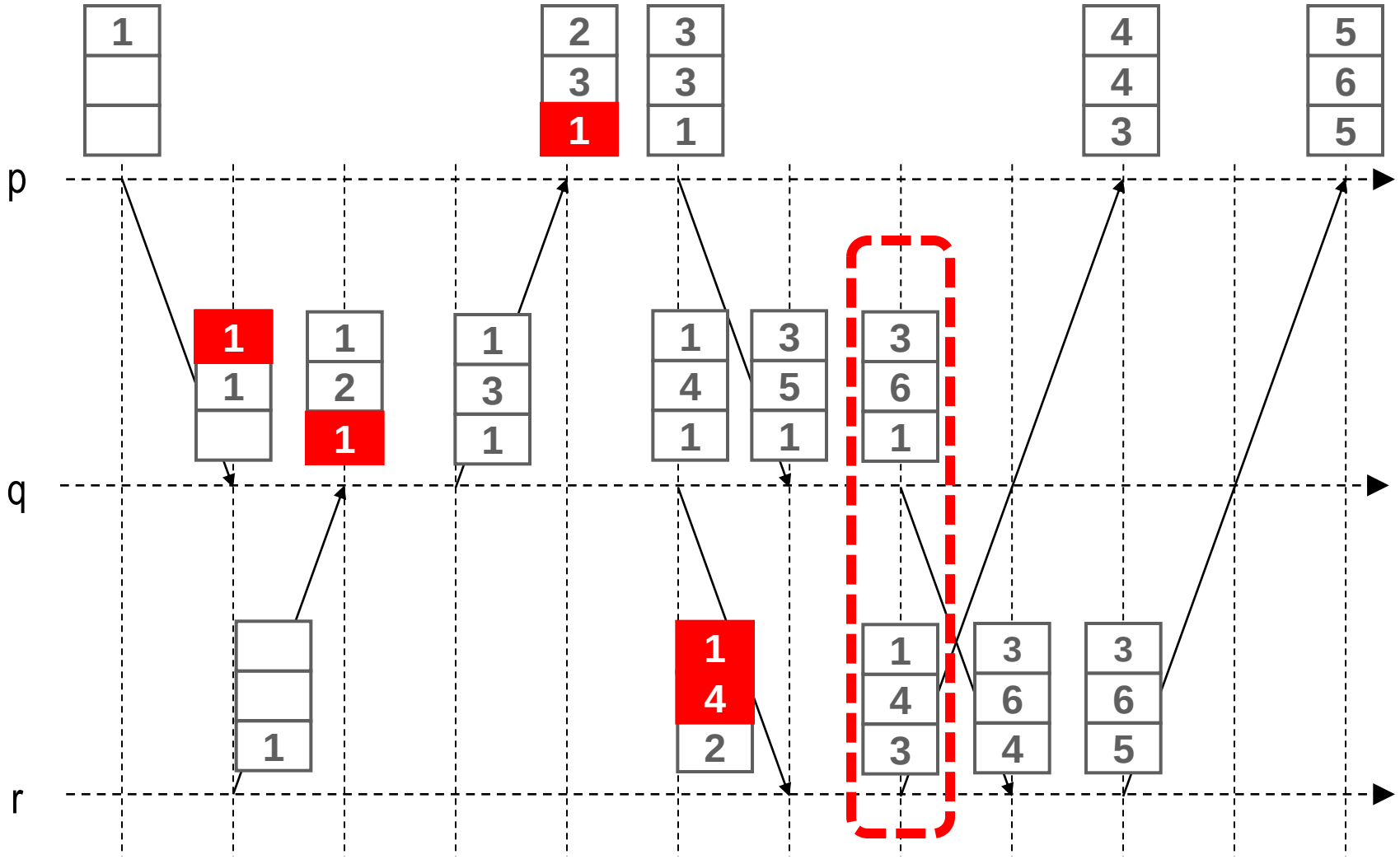
- **Primjena vektorskih oznaka**
  - Ako za događaje *a* i *b* vrijedi $V(a) < V(b)$ tada vrijedi da je događaj *a* nastupio u vremenu prije događaja *b*, $a \rightarrow b$

- **Za vektorske oznake vrijedi $V_p < V_m$ ako:**
  - postoji barem jedan *k* za koji vrijedi $V_p[k] < V_m[k]$,
  - za sve ostale $l \neq k$ vrijedi $V_i[l] \leq V_j[l]$,
  - $p, m, k, l \in [0, N-1]$ i
  - broj procesa u raspodijeljenoj okolini je *N*

Vektorske oznake vremena

- **Koraci algoritma za održavanje vektorskih oznaka:**
  1) Početna vrijednosti svih komponenata je 0
  2) Za svaki interni događaj procesa p uvećaj oznaku $V_p[p]$
  3) Prije slanja poruke na procesu p uvećaj oznaku $V_p[p]$ i poslanoj poruci pridruži izgrađeni vektor V_p
  4) Nakon primitka poruke od procesa p na procesu k uvećaj oznaku $V_k[k]$ dok za sve ostale oznake $j \neq k$ vrijedi: $V_k[j] = V_p[j]$ ako je $V_k[j] < V_p[j]$

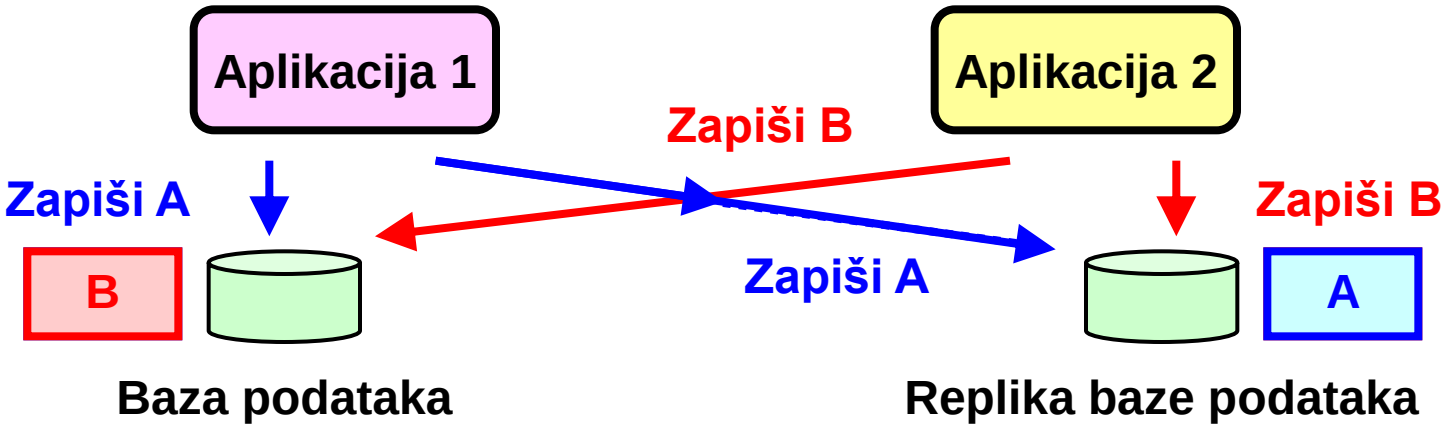
Primjer uporabe vektorskih oznaka



Primjena sata u raspodijeljenoj okolini

- **Uređena razmjena poruka**
  - Primjena skalarnih logičkih oznaka vremena
  - Svi procesi na isti način vide redoslijed događaja

- **Primjena: Održavanje konzistentnosti**
  - Bez vremenskih oznaka nije moguće odrediti pravilni redoslijed akcija u vremenu



Primjena sata u raspodijeljenoj okolini

- **JGroups**
  - Sustav za pouzdanu razmjenu poruka u grupi
  - Ostvaren u programskom jeziku Java
- **Značajke sustava**
  - Pouzdana razmjena poruka
  - Uređenost akcija u vremenu
  - Očuvanje konzistentnosti akcija
- **Dodatne informacije**
  - http://www.jgroups.org

Sadržaj predavanja

- Potreba za sinkronizacijom procesa
- Primjena sata u jednoprocesorskoj okolini
- Primjena sata raspodijeljenoj okolini
- Sinkronizacija tijeka izvođenja procesa
- Međusobno isključivanje procesa

Sinkronizacija tijeka izvođenja procesa

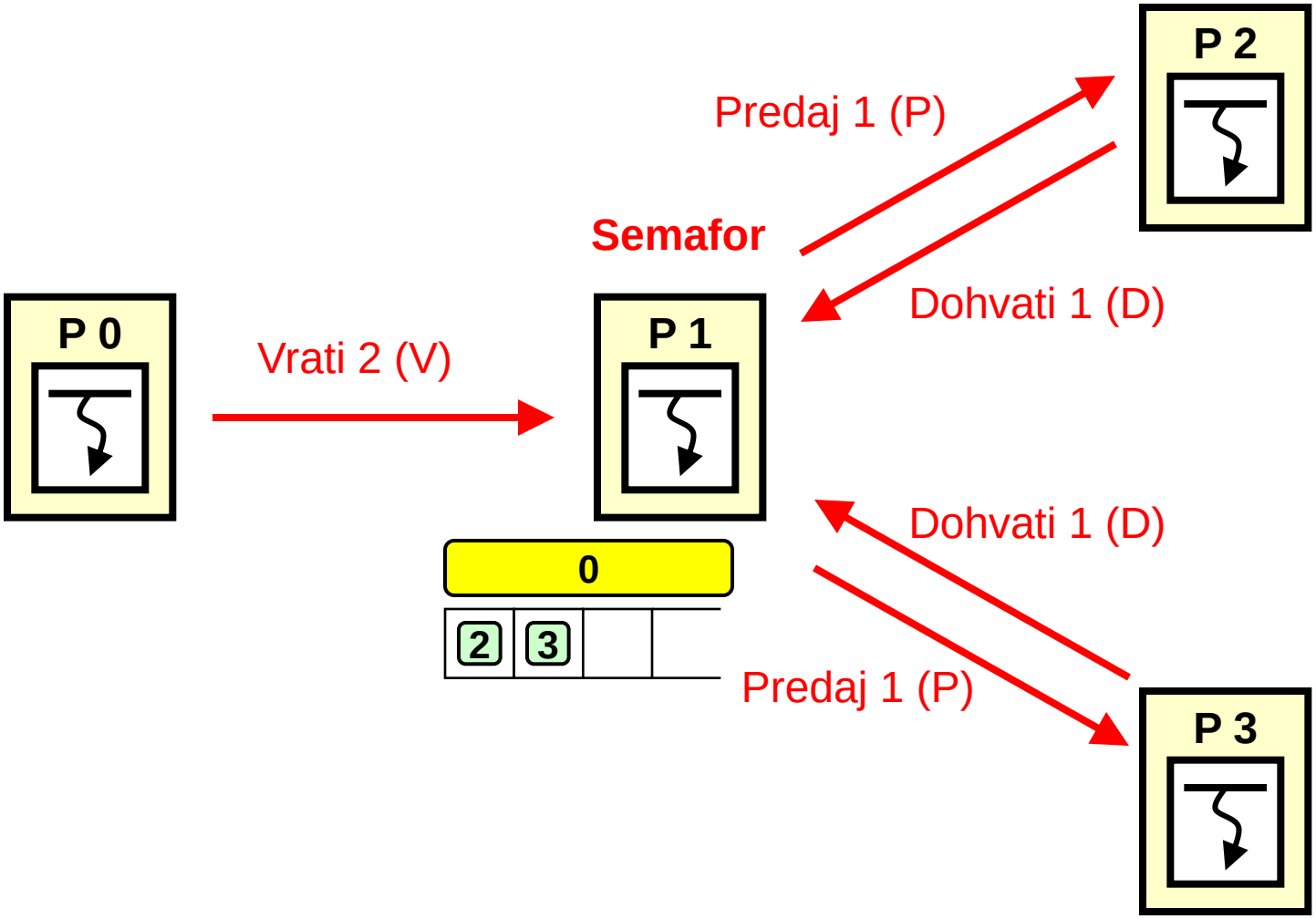
- Primjena semafora u raspodijeljenoj okolini
- Sinkronizacija zasnovana na razmjeni događaja

Primjena semafora u raspodijeljenoj okolini

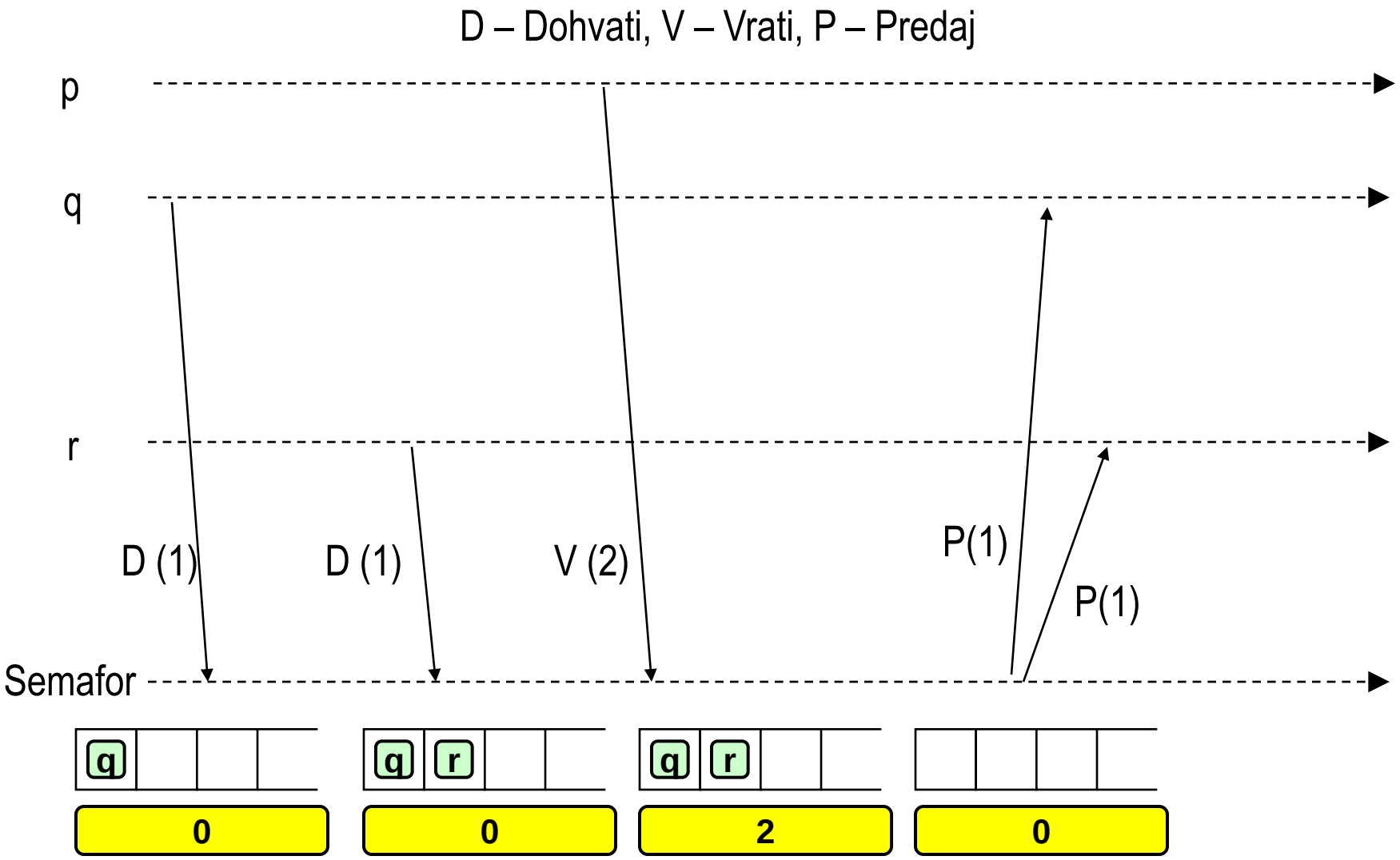
- **Semafor**
  - Proces koji u spremniku čuva n znački
  - Rep čekanja zasnovan na posluživanju zahtjeva prema redoslijedu prispijeća (*FIFO*)

- **Korisnici**
  - Procesi šalju poruke *zahtjev* (*Z*) za dohvat N znački
  - Ako u spremniku postoji traženi broj znački, prosljeđuje se *potvrda* (*P*)
  - Ako u spremniku ne postoji traženi broj znački zahtjev se stavlja u rep čekanja
  - Nakon završetka obrade, procesi vraćaju preuzete značke slanjem poruke *otpusti* (*O*)

Okolina semafora



Primjer sinkronizacije tijeka izvođenja

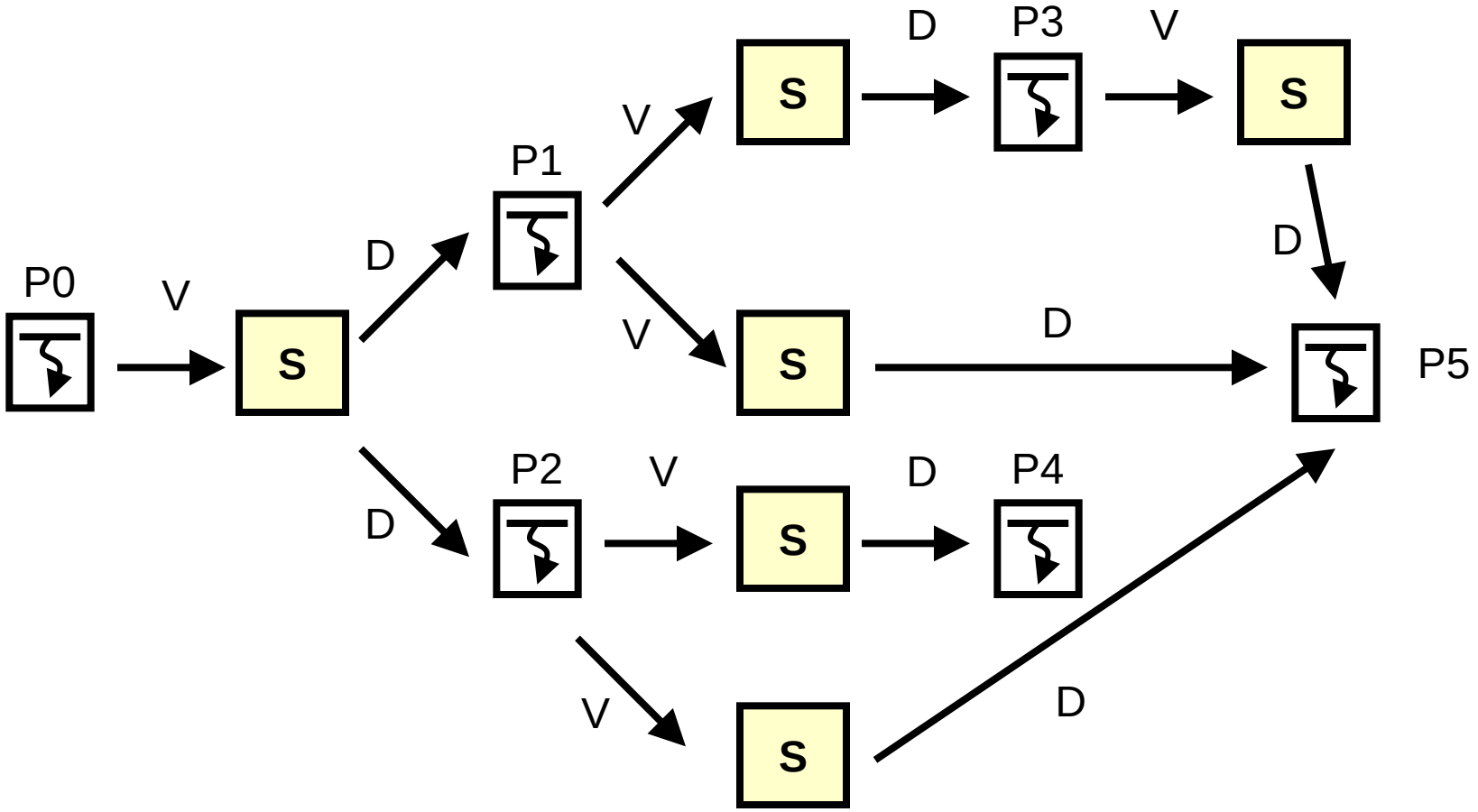


Složeni obrasci sinkronizacije

- **Semafor je osnovni element za ostvarivanje soženih obrazaca sinkronizacije**

- **Graf raspodijeljenog tijeka izvođenja procesa**
  - Grananje tijeka izvođenja
  - Spajanje tijeka izvođenja
  - Ponavljanje tijeka izvođenja

Graf raspodijeljenog tijeka izvođenja

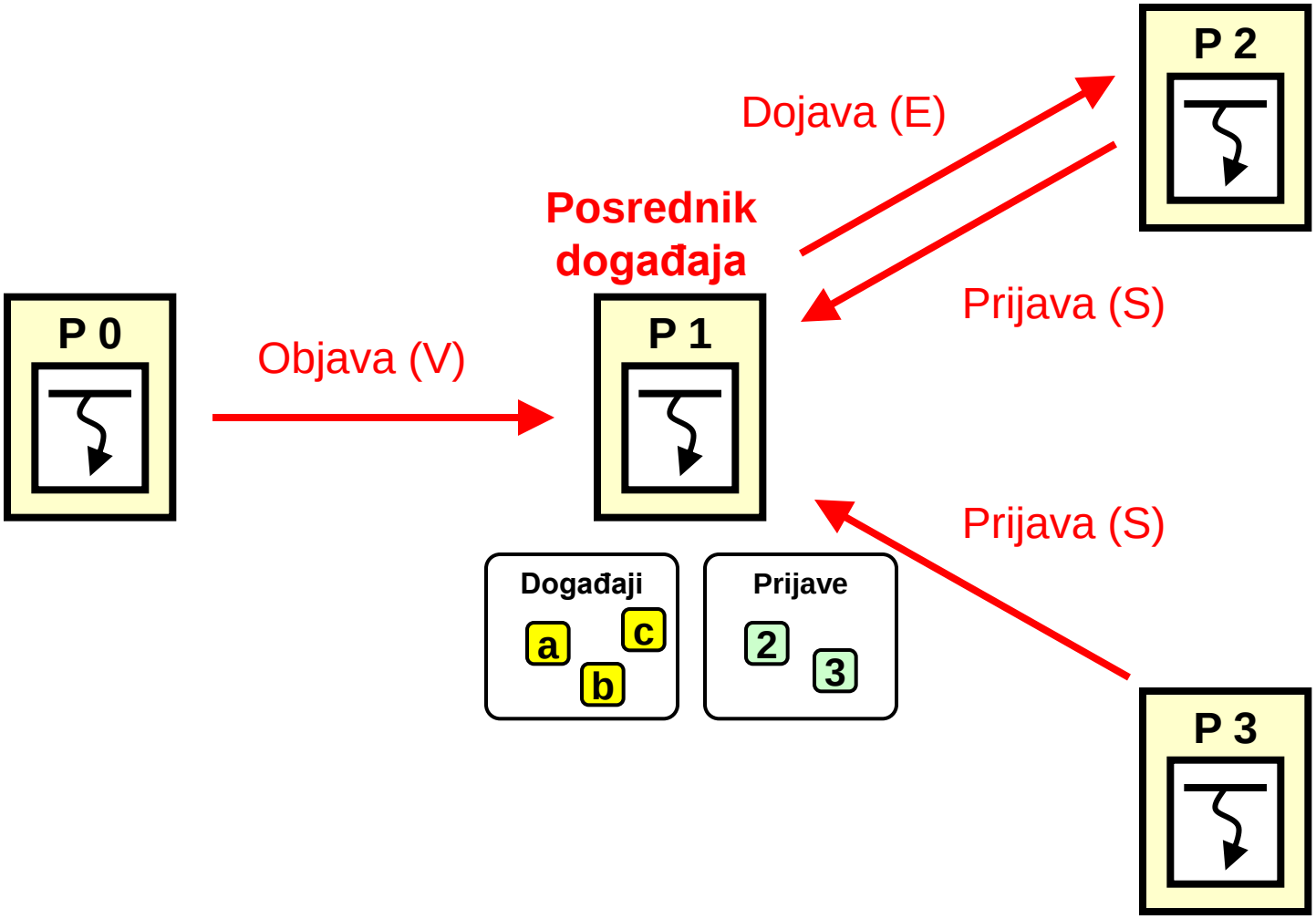


Sinkronizacija razmjenom događaja

- **Posrednik događaja**
  - Sadrži spremnik s objavljenim događajima, spremnik prijava
  - Ostvaruje postupak uparivanja događaja i prijava

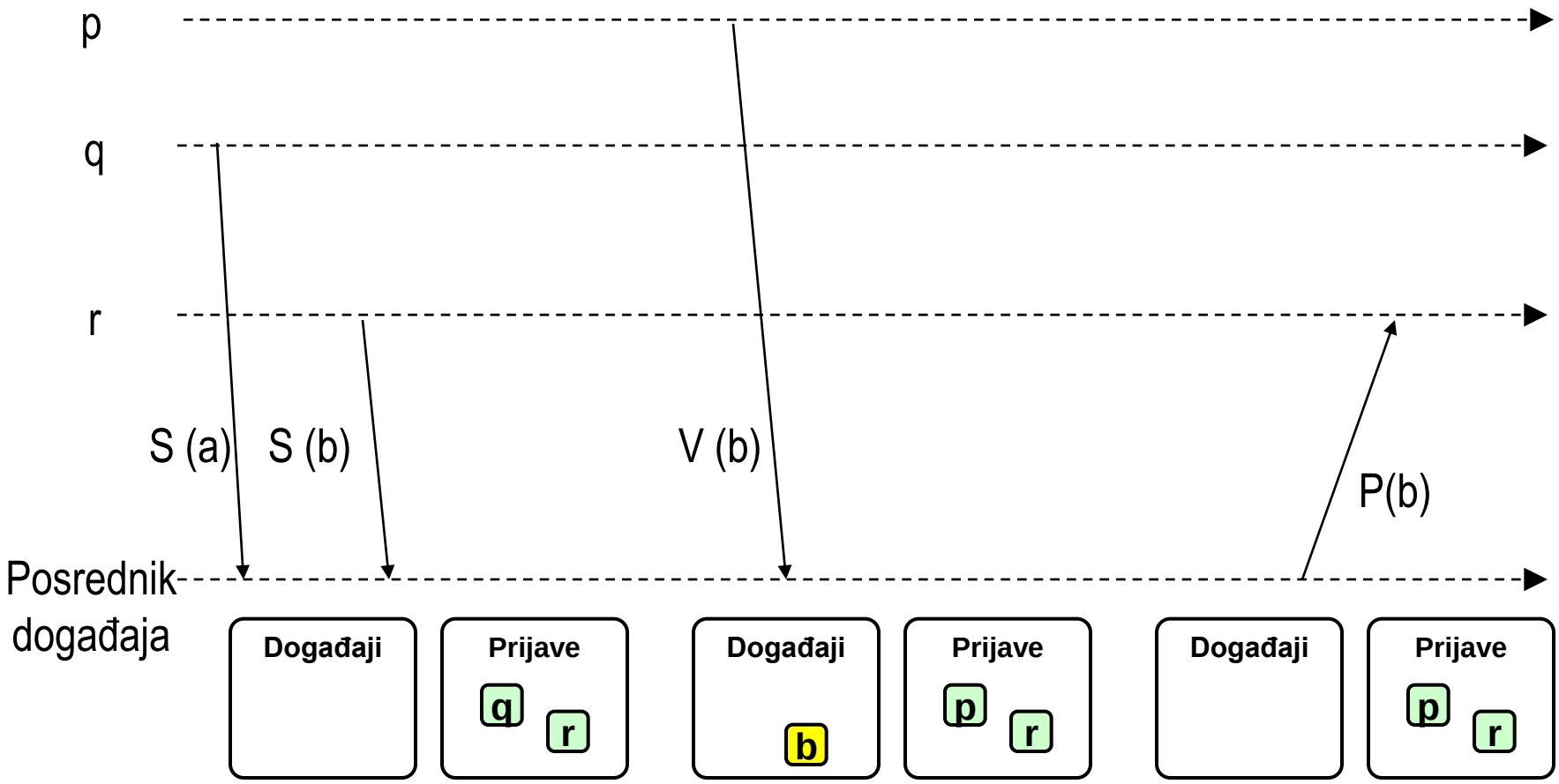
- **Korisnici**
  - Procesi šalju poruke s događajima (*E*)
  - Procesi šalju prijave posredniku (*S*)
  - Ako posrednik ima pretplatu za objavljeni događaj prosljeđuje događaj procesu pretplatniku u poruci dojave (*N*)

Okolina posrednika događaja



Primjer sinkronizacije razmjenom događaja

S (x) – Prijava za događaje tipa x, V(x) – Objava događaja tipa x,
P(x) – Predaj događaj x



Međusobno isključivanje procesa

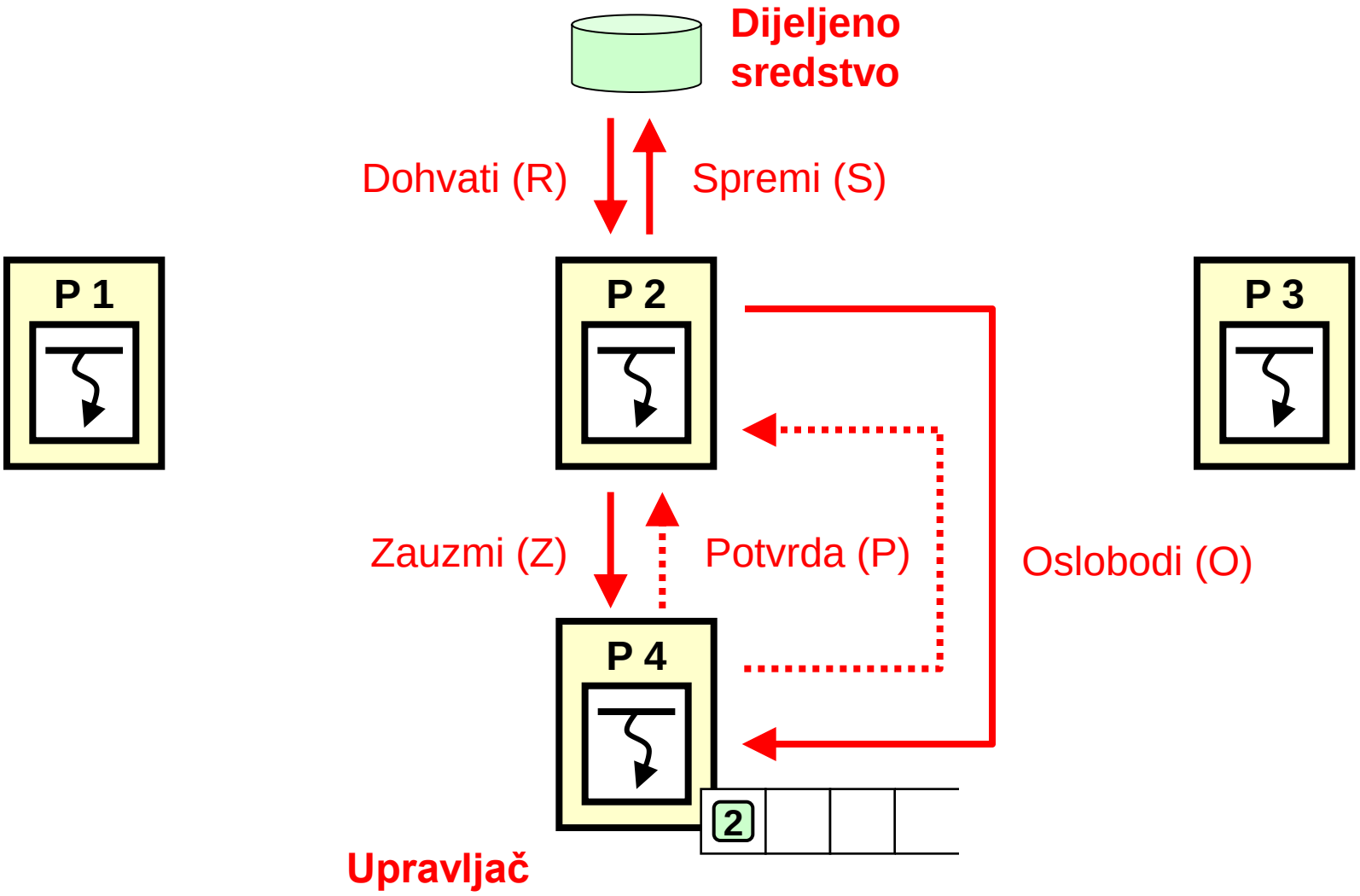
- Središnji upravljač s repom čekanja
- Raspodijeljeno međusobno isključivanje
- Isključivanje zasnovano na primjeni prstena

Međusobno isključivanje procesa

- **Središnji upravljač s repom čekanja**
  - Proces koji čuva stanje repa čekanja
  - Rep čekanja zasnovan na posluživanju zahtjeva prema redoslijedu prispijeća (*FIFO*)

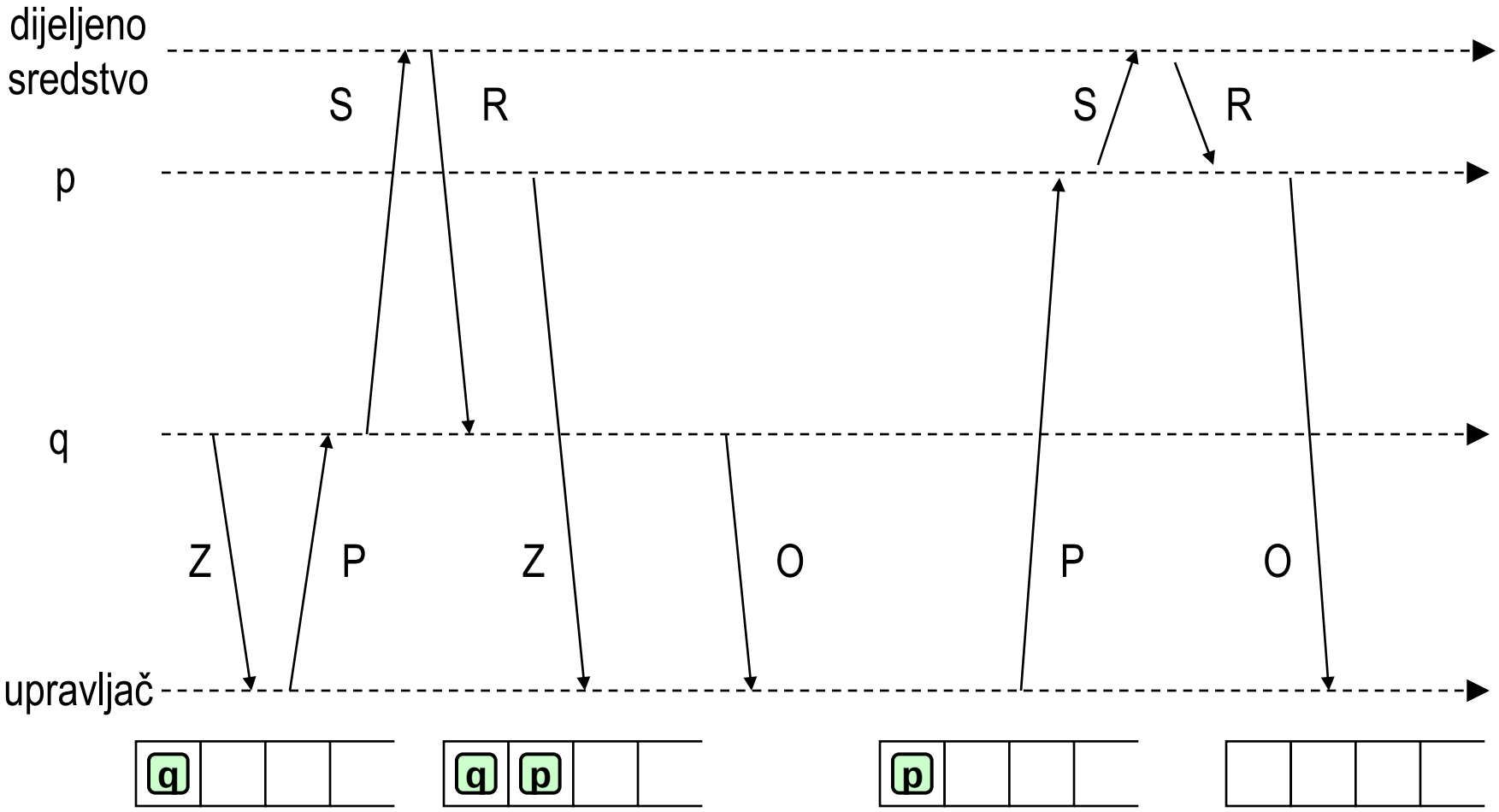
- **Korisnici**
  - Procesi šalju poruke *zahtjev* (*Z*) za pristup sredstvu
  - Procesi ostvaruju pristup sredstvu nakon primitka poruke *potvrda* (*P*)
  - Nakon završetka obrade, procesi otpuštaju zauzeto sredstvo slanjem poruke *oslobodi* (*O*)

Središnji upravljač s repom čekanja



Međusobno isključivanje procesa

R – Dohvati, S – Spremi, Z – Zauzmi, P – Potvrda, O – Oslobodi

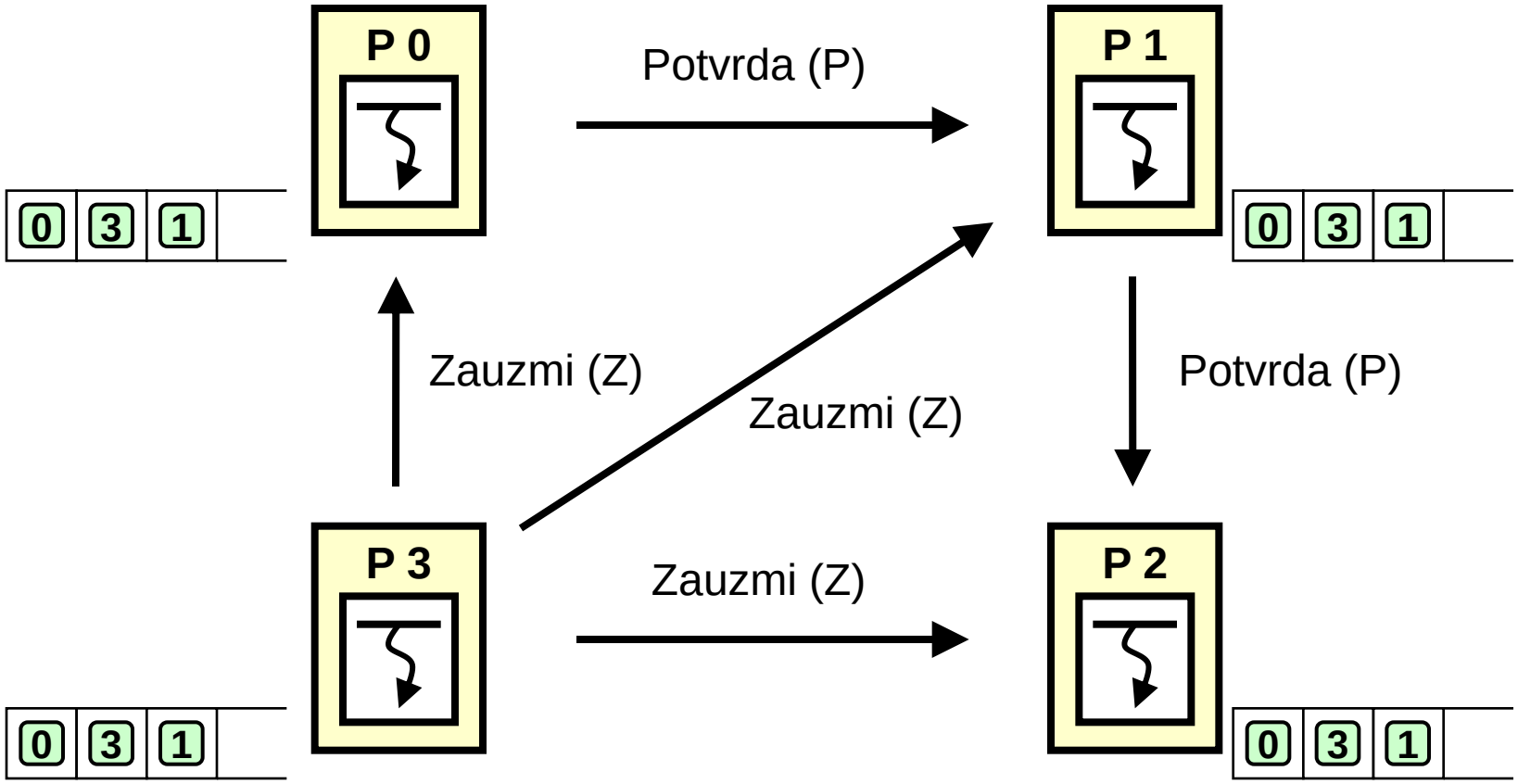


Raspodijeljeno međusobno isključivanje

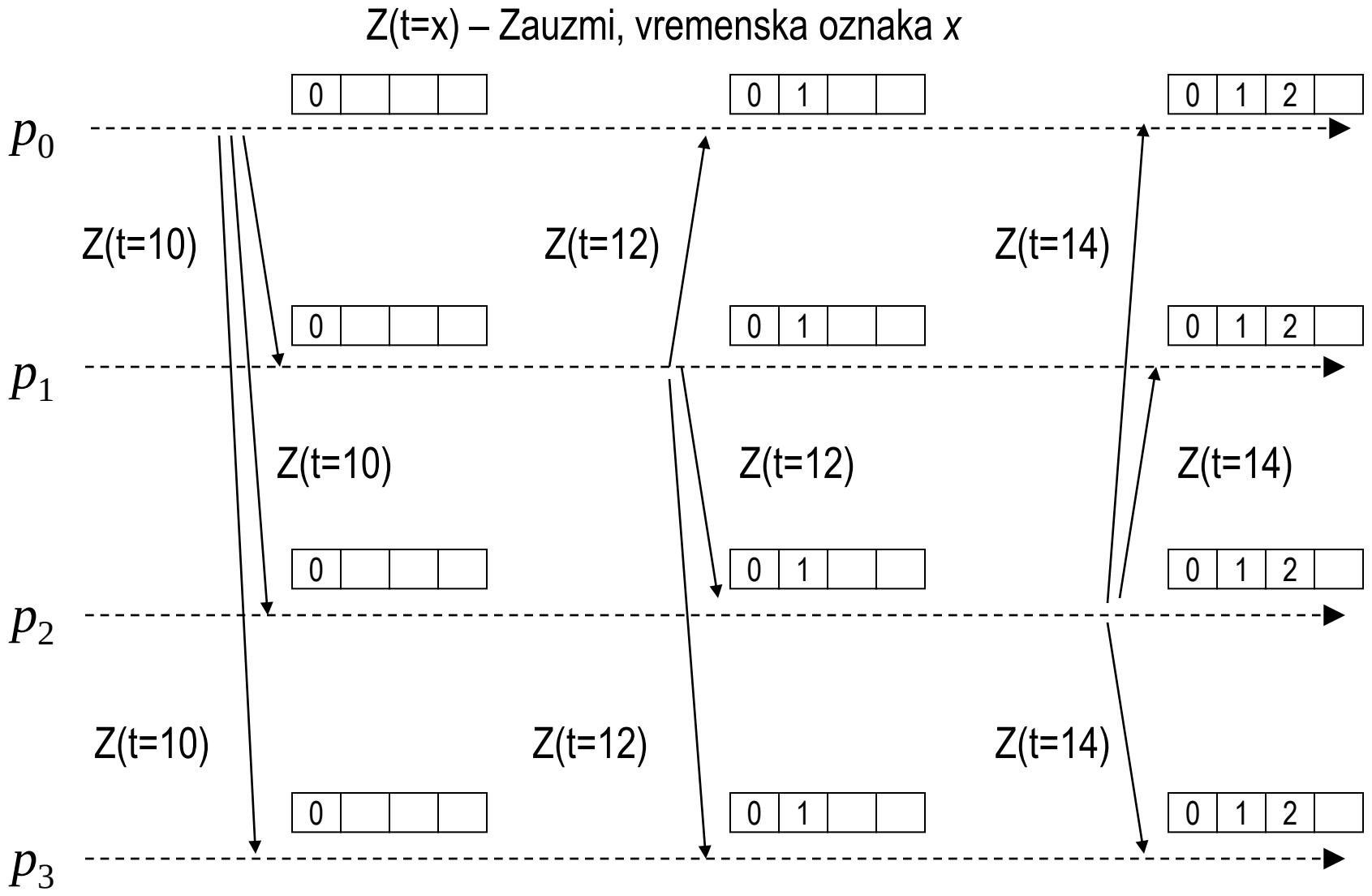
- **Raspodijeljeni rep čekanja**
  - Svaki proces ima lokalni rep čekanja
  - Procesi razmjenjuju informacije potrebne za usklađivanje stanja svih repova čekanja u sustavu

- **Pretpostavke**
  - Svaki proces ima lokalni satni mehanizam koji je usklađen s ostalim procesima
  - Svaki zahtjev za pristup sredstvu uključuje oznaku trenutka u kojem je proces uputilo zahtjev
  - Procesi ostvaruju pristup u skladu s vremenskim oznakama upućivanja zahtjeva

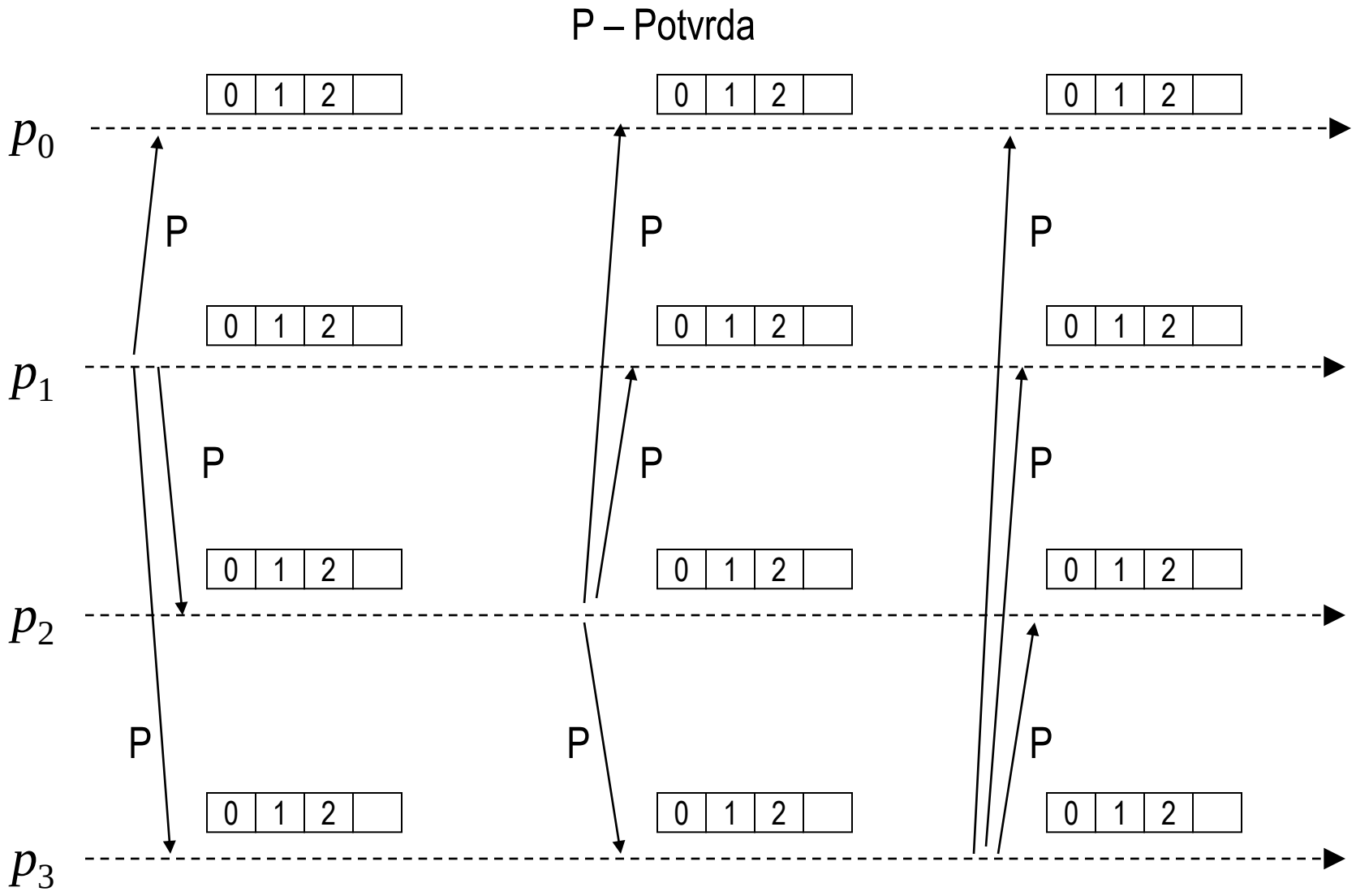
Elementi raspodijeljenog isključivanje



Raspodijeljeno međusobno isključivanje

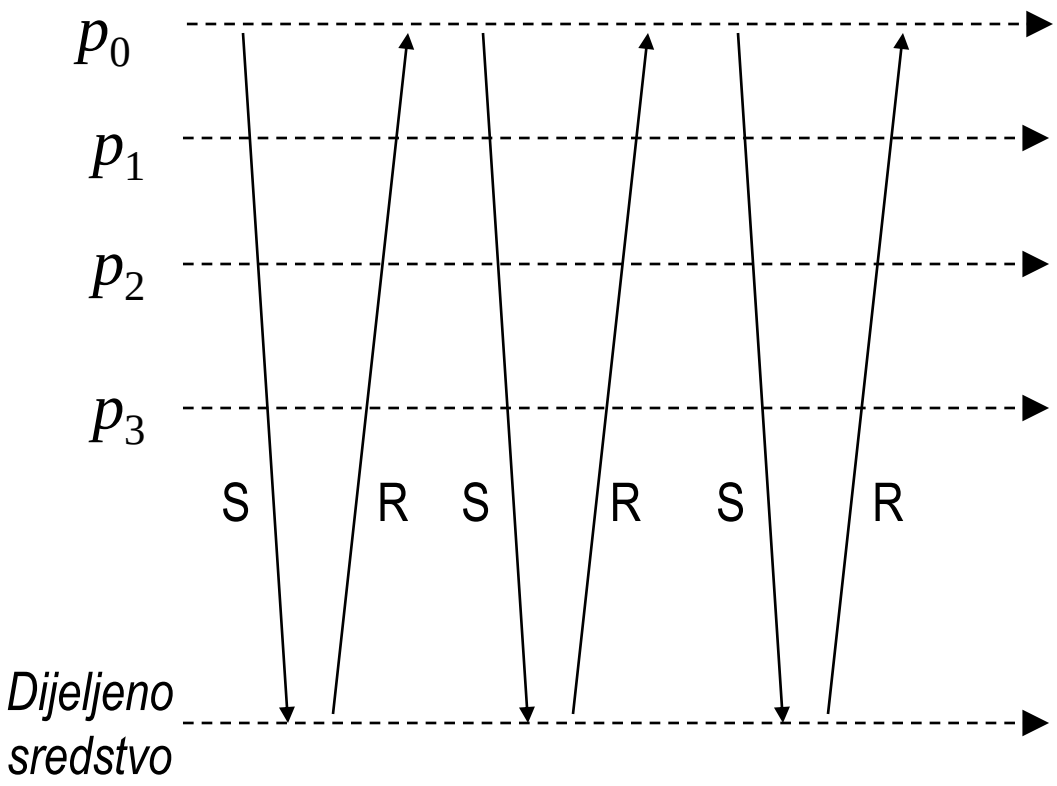


Raspodijeljeno međusobno isključivanje

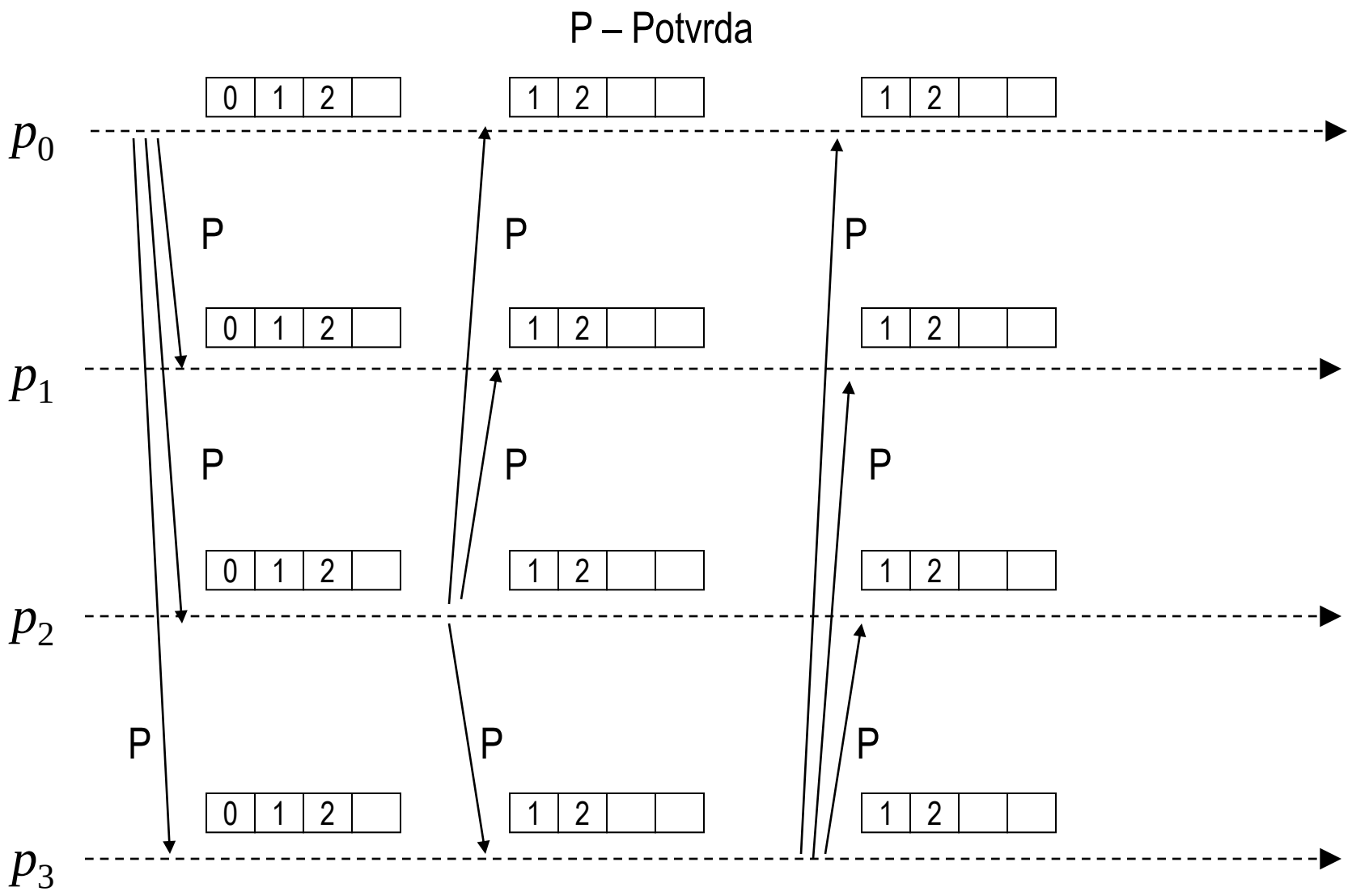


Raspodijeljeno međusobno isključivanje

R – Dohvati, S – Spremi



Raspodijeljeno međusobno isključivanje



Međusobno isključivanje primjenom prstena

- **Struktura prstena procesa**
  - Procesi su povezana u logičku mrežu zasnovanu na prstenu
  - Primjenjuju se identifikatori računala u za formiranje prstena
  - Duž prstena ostvaruje se razmjena jednog tokena
- **Pristup dijeljenom sredstvu**
  - Pristup ima samo proces koji u određenom trenutku ima token
  - Nakon završetka pristupa, proces prosljeđuje token susjednom procesu u prstenu

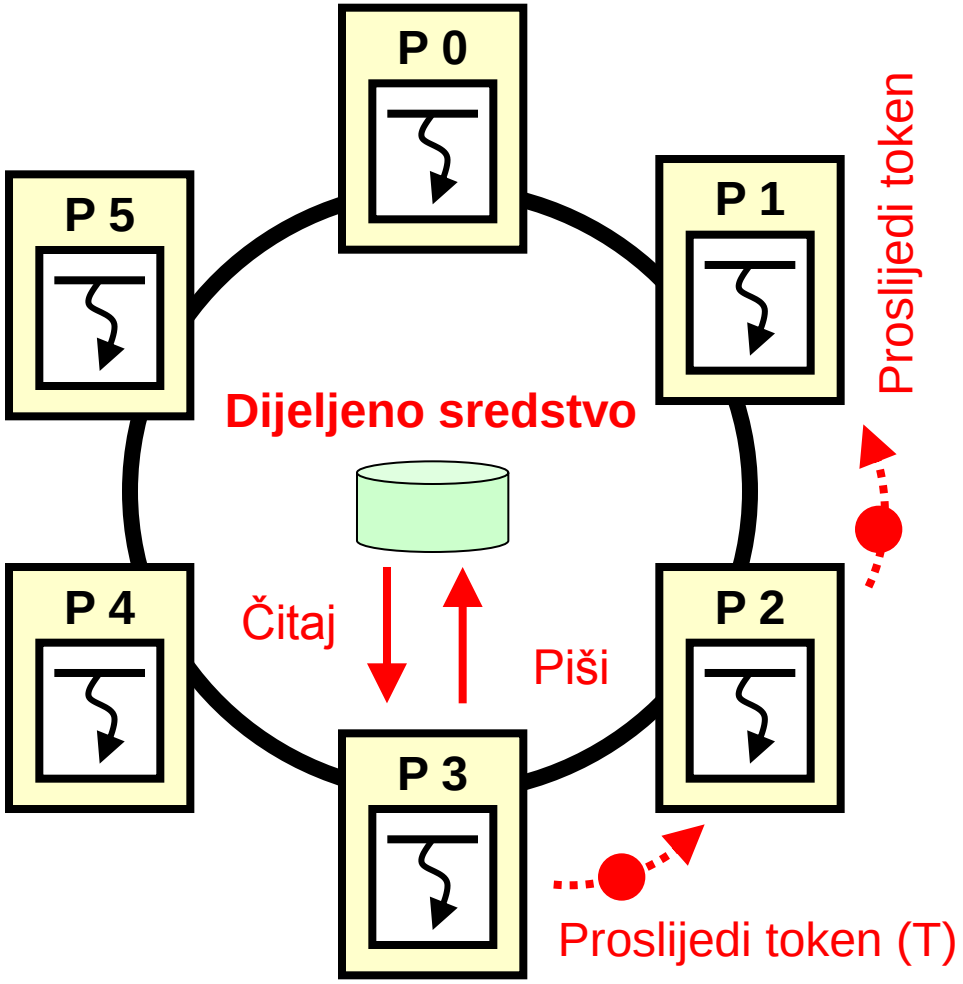
Akcije procesa u prstenu

- **Proces n prima token**
  1) Čitanje podataka iz spremnika
  2) Pisanje podataka u spremnik
  3) Prosljeđivanje tokena procesu (n-1) % N

- **Proces (n-1) % N prima token**
  4) Proces ne zahtjeva pristup spremniku
  5) Prosljeđivanje tokena procesu (n-2) % N

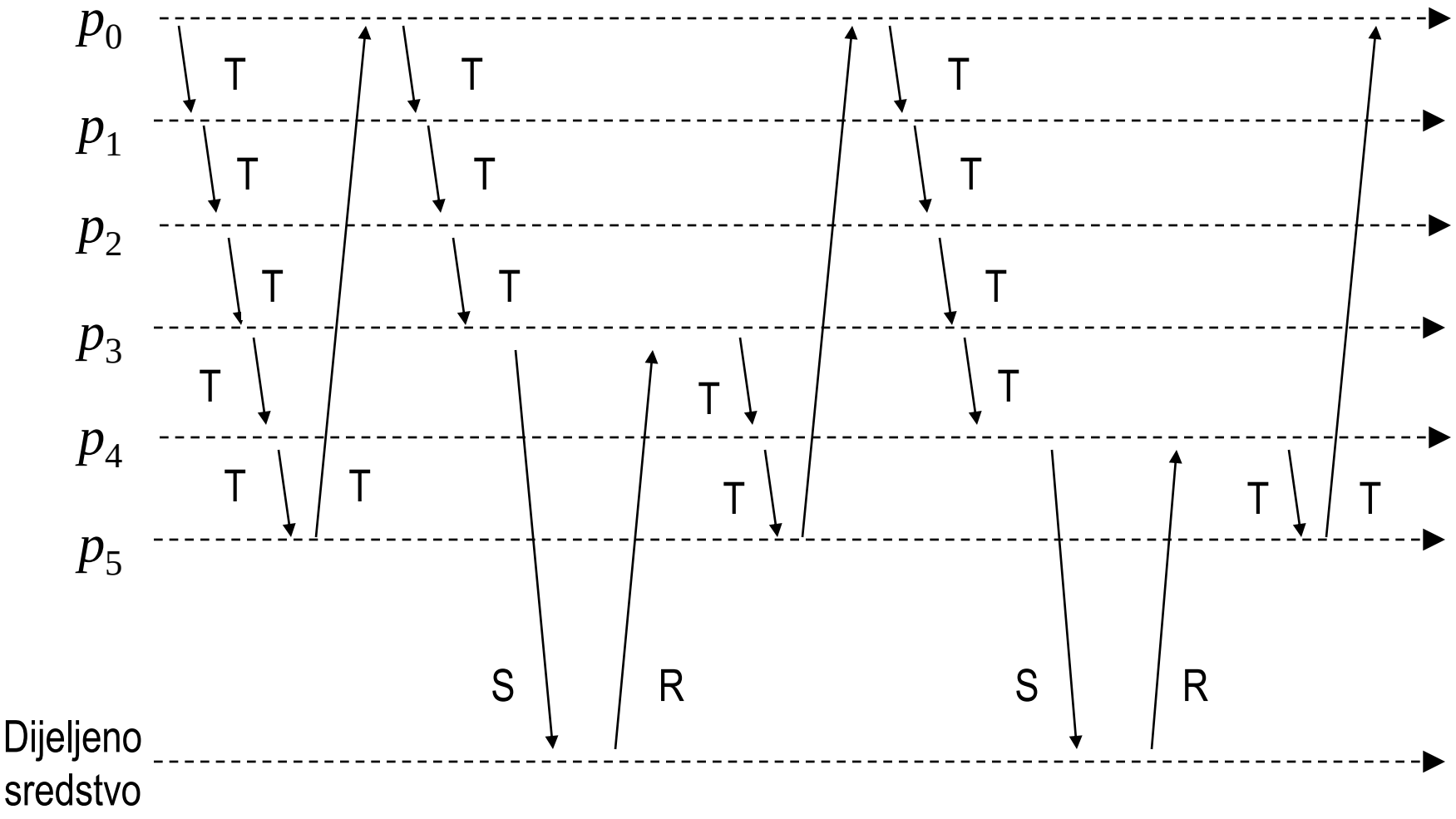
- **Proces (n-2) % N prima token**
  6) …

Međusobno isključivanje primjenom prstena



Raspodijeljeno međusobno isključivanje

T – Prijenos tokena, S – Spremi, R – Dohvati



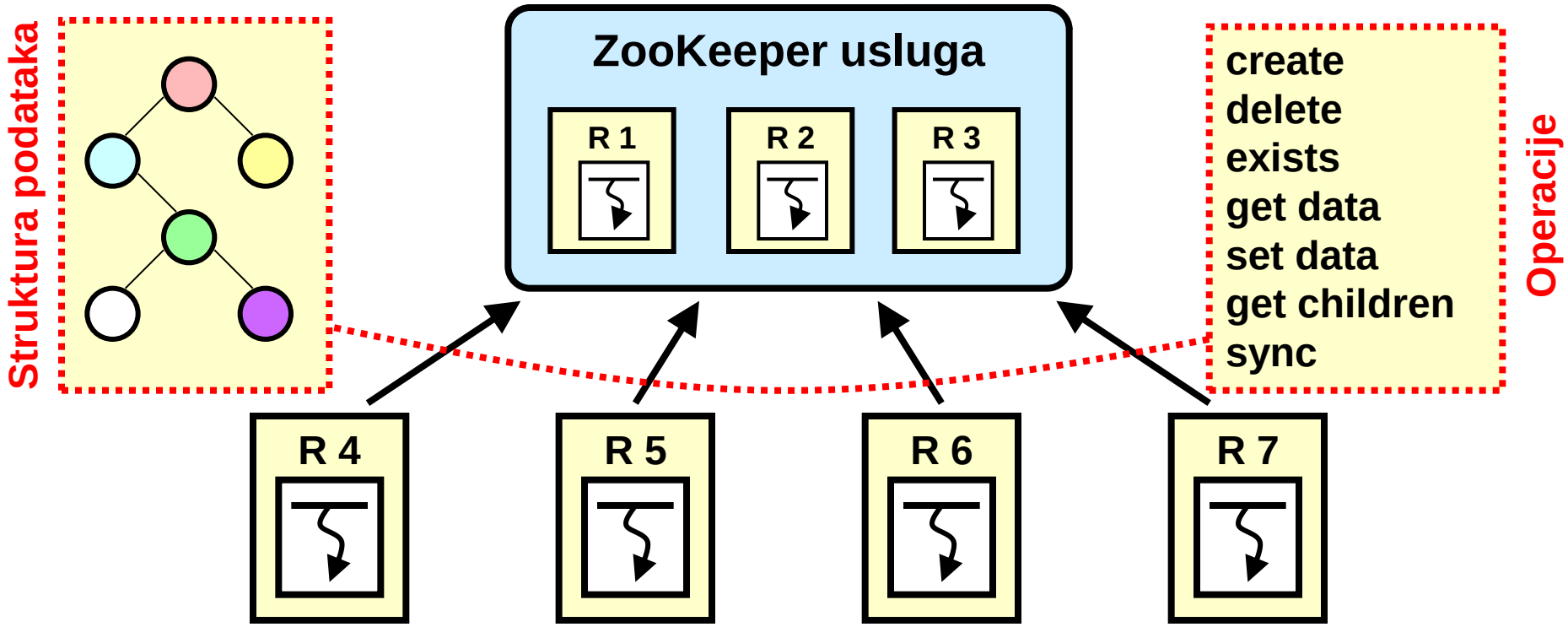
Primjer sustava za sinkronizaciju

- **Usluga ZooKeeper**
  - Usluga za ostvarivanje koordinacije tijeka izvođenja skupa procesa u raspodijeljenoj okolini
  - Ostvarena u jeziku Java
  - http://hadoop.apache.org/zookeeper

- **Okruženje Hadoop Core**
  - Ostvarenje modela za provođenje analize i obrade nad segmentima velikog skupa podataka
  - Ostvareno u jeziku Java
  - http://hadoop.apache.org/core

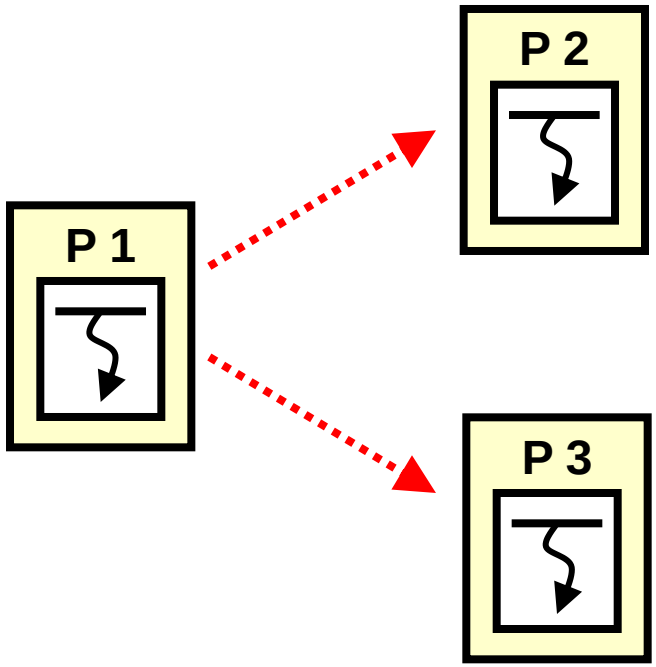
Usluga ZooKeeper

- **Usluga opće namjene za koordiniranje skupa procesa u raspodijeljenoj okolini**
  - Naslovljavanje, sinkronizacija, upravljanje grupama, repovi, donošenje odluka, zaključavanje sredstava



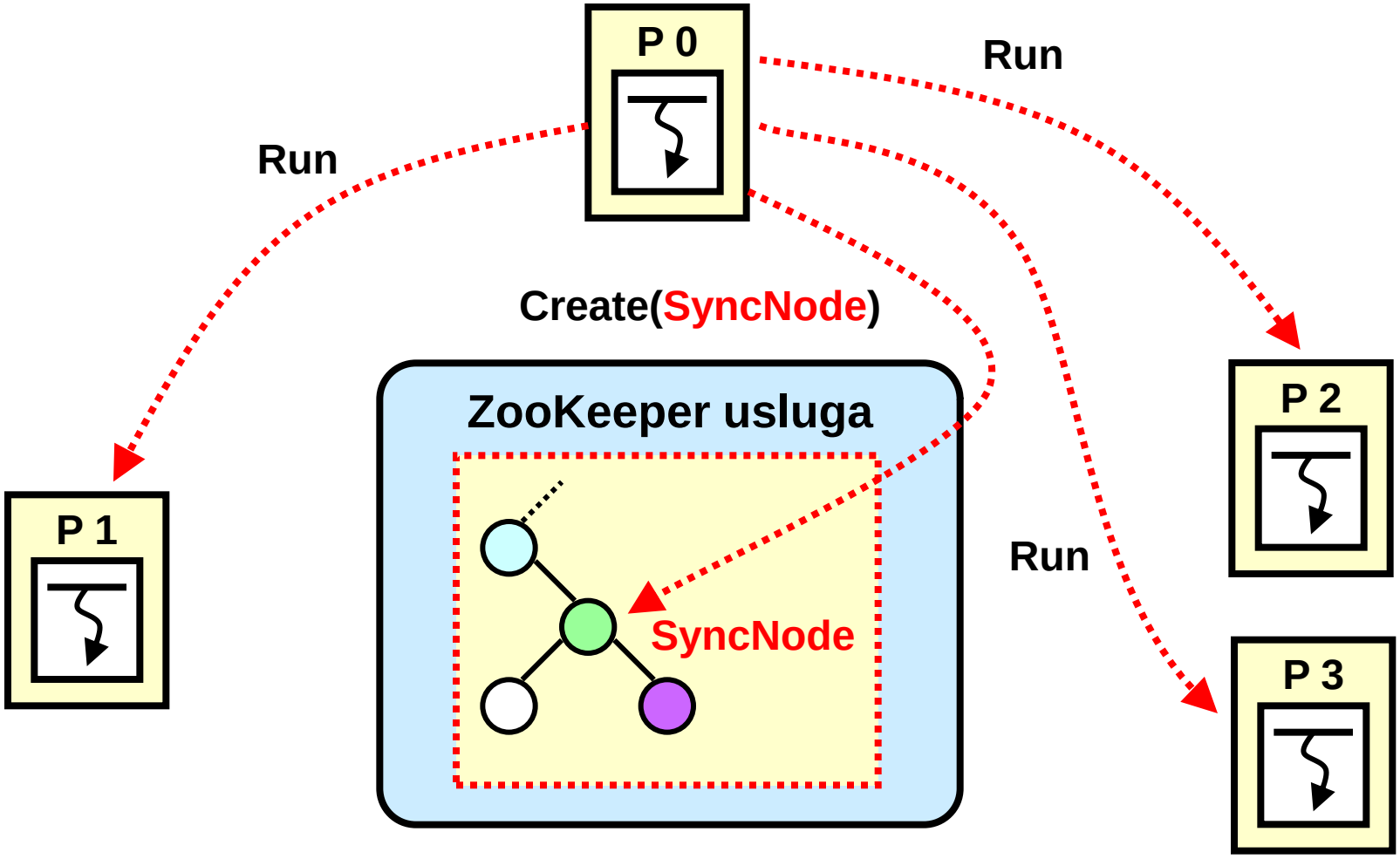
Usluga ZooKeeper

- **Primjer: Sinkronizacija procesa**
  - Procesi P2 i P3 započinju s izvođenjem tek nakon što je proces P1 završio s izvođenjem



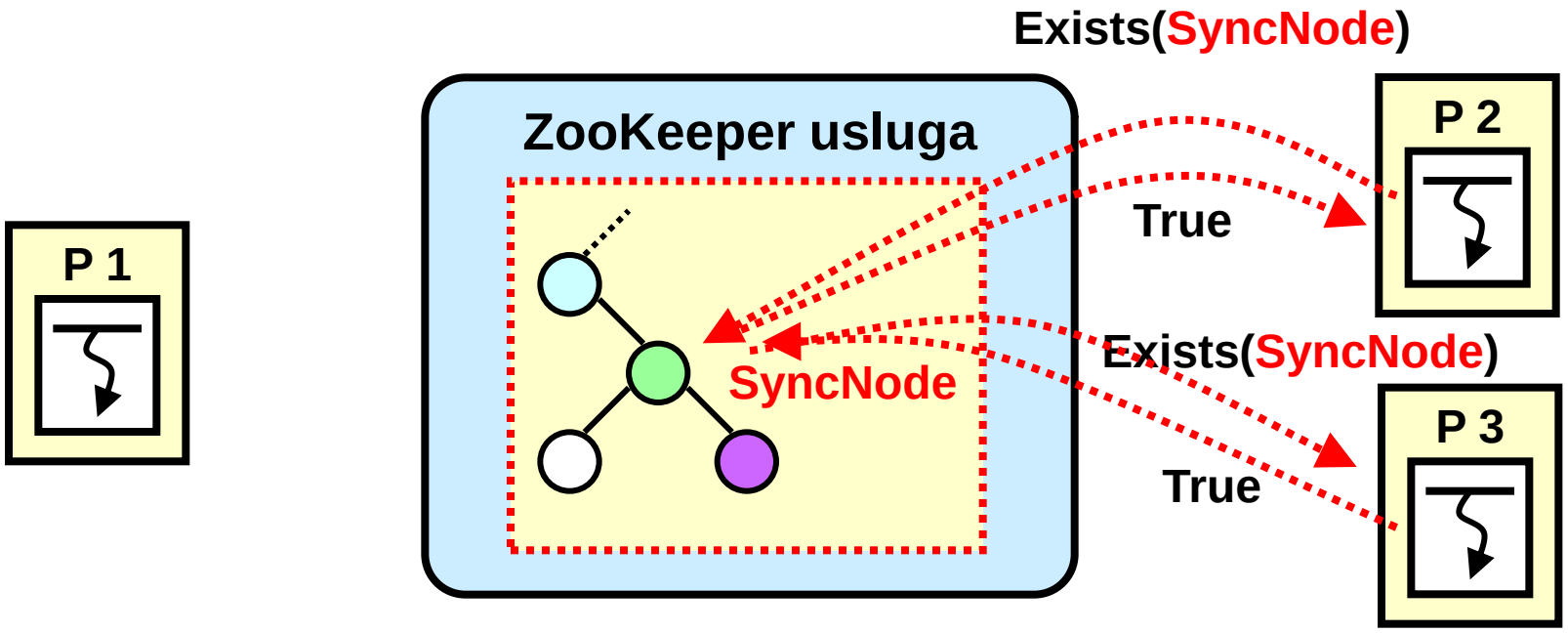
Usluga ZooKeeper

- P0 ostvaruje upravljački proces



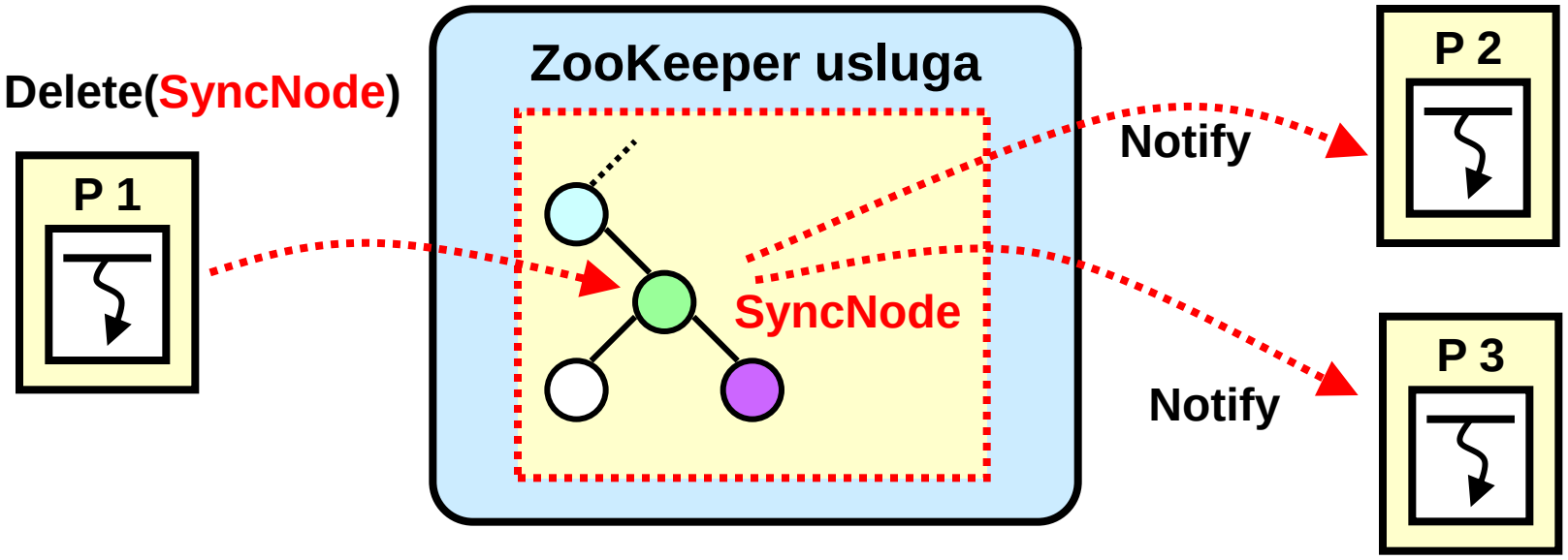
Usluga ZooKeeper

- **Procesi P2 i P3 ispituju postoji li sinkronizacijski čvor**
  - Ako čvor postoji, čekaju na dojavu o brisanju čvora
  - Kada se čvor izbriše, započinju izvođenje



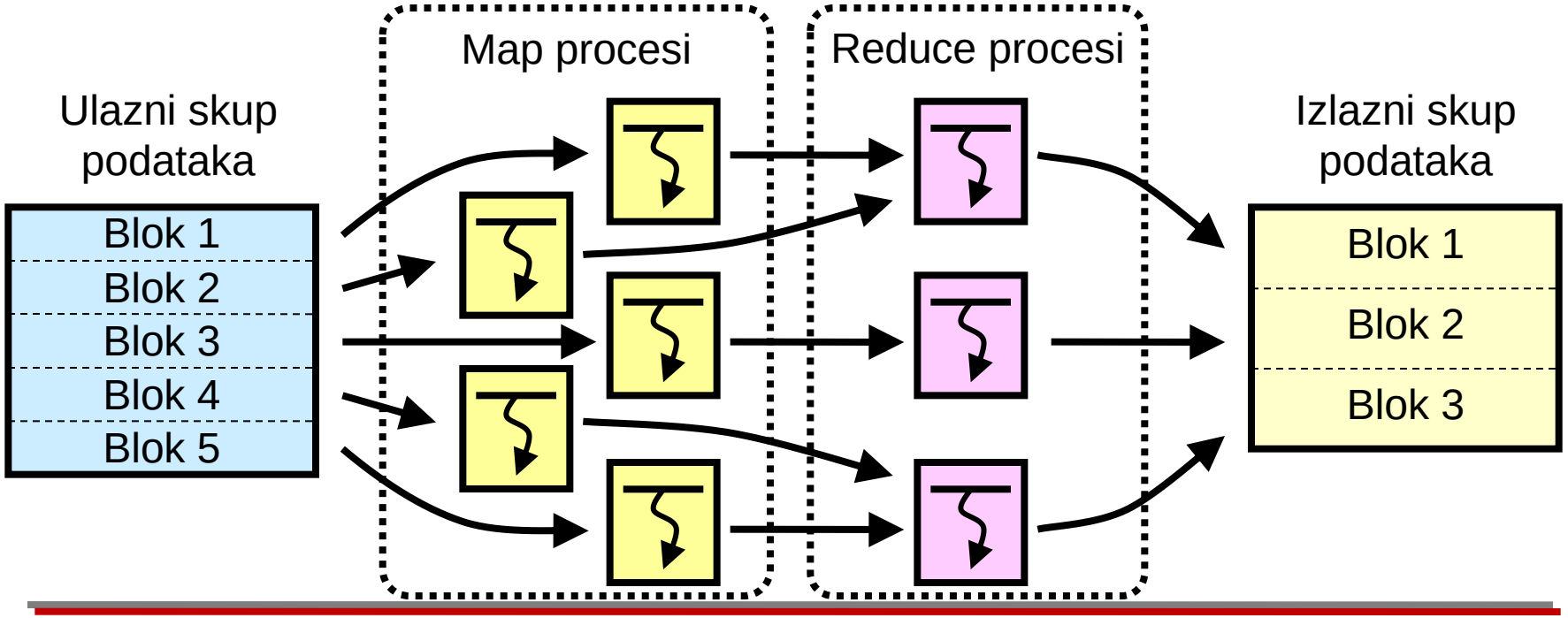
Usluga ZooKeeper

- **Proces P1 završava s izvođenjem**
  - Briše čvor SyncNode
  - Usluga ZooKeeper dojavljuje brisanje čvora
  - Procesi P2 i P3 započinju s izvođenjem



Okruženje Hadoop Core

- **Map-Reduce model analize i obrade podataka**
  - Ulazni skup podataka dijeli se na segmente
  - Segmente istodobno obrađuju nezavisni procesi
  - Rezultati obrade grupiraju se u odredišni skup podataka



Okruženje Hadoop Core

- **Primjer: Prebrojavanje riječi (WordCount)**

```
01: package org.myorg;                                          Zaglavlja
02: import ...

12: public class WordCount {
13:
14:   public static class Map                                   Priprema podataka (Map korak)
          extends MapReduceBase
          implements   Mapper<LongWritable, Text, Text, IntWritable> {

        public void map(LongWritable key, Text value,
                        OutputCollector<Text, IntWritable> output,
                        Reporter reporter) throws IOException { … }
26:   }
```

Okruženje Hadoop Core

- **Primjer: Prebrojavanje riječi (WordCount)**

```
28:  public static class Reduce                    Obrada podataka (Reduce korak)
                 extends MapReduceBase
                 implements Reducer<Text, IntWritable, Text, IntWritable> {

       public void reduce(Text key, Iterator<IntWritable> values,
                          OutputCollector<Text, IntWritable> output,
                          Reporter reporter) throws IOException { … }
37:  }
38:  public static void main(String[] args) throws Exception { … }
```

Okruženje Hadoop Core

- **Primjer: Prebrojavanje riječi (Metoda *map*)**

```
18:   public void map(LongWritable key, Text value,
                      OutputCollector<Text, IntWritable> output,
                      Reporter reporter) throws IOException {
19:     String line = value.toString();
20:     StringTokenizer tokenizer = new StringTokenizer(line);
21:     while (tokenizer.hasMoreTokens()) {
22:       word.set(tokenizer.nextToken());
23:       output.collect(word, one);
24:   }
…
```

Izdvajanje riječi

Okruženje Hadoop Core

- **Primjer: Prebrojavanje riječi (Metoda *reduce*)**

```
29:  public void reduce(Text key, Iterator<IntWritable> values,
                        OutputCollector<Text, IntWritable> output,
                        Reporter reporter) throws IOException {
30:     int sum = 0;
31:     while (values.hasNext()) {
32:        sum += values.next().get();
33:     }
34:     output.collect(key, new IntWritable(sum));
35:  }
…
```

Brojanje riječi

Okruženje Hadoop Core

- **Primjer: Prebrojavanje riječi (Metoda Main)**

```
…
38:  public static void main(String[] args) throws Exception {
39:     JobConf conf = new JobConf(WordCount.class);
40:     conf.setJobName("wordcount");
41:     conf.setOutputKeyClass(Text.class);                    Priprema izlaza
42:     conf.setOutputValueClass(IntWritable.class);
43:     conf.setInputFormat(TextInputFormat.class);
44:     conf.setOutputFormat(TextOutputFormat.class);
45:     FileInputFormat.setInputPaths(conf, new Path(args[0]));
46:     FileOutputFormat.setOutputPath(conf, new Path(args[1]));
47:     conf.setMapperClass(Map.class);                Priprema map i reduce
48:     conf.setReducerClass(Reduce.class);                        koraka
49:     JobClient.runJob(conf);
50:  }
```

Dodatne informacije

- **Knjige**
  - S. Tanenbaum, M. van Steen: **“Distributed Systems: Principles and Paradigms"**, Pretence Hall, 2002. (Poglavlje: *Synchronization*)
  - H. Attya, J. Welch: **“Distributed Computing: Fundamentals, Simulations, and Advanced Topics"**, Wiley, 2004. (Poglavlje: *Causality and Time*)
  - N. A. Lynch: **“Distributed Algorithms"**, Morgan Kaufmann, 1997. (Poglavlje: *Logical Time*)

Dodatne informacije

- **Znanstveni radovi**
  - G. R. Andrews, F. B. Schneider: **“Concepts and Notations for Concurrent Programming"**, ACM Computing Surveys

    http://www.cs.cornell.edu/fbs/publications/LangSurv.pdf

  - G. R. Andrews: **“Paradigms for Process Interaction in Distributed Programs"**, ACM Computing Surveys
    http://web.abo.fi/~kaisa/And91.pdf