Diplomski studij

Informacijska i komunikacijska tehnologija:

Telekomunikacije i informatika

Računarstvo:

Programsko inženjerstvo i informacijski sustavi

Računarska znanost

Ak.g. 2008./2009.

Raspodijeljeni sustavi

8.

Sinkronizacija procesa u vremenu

19.11.2008.

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.

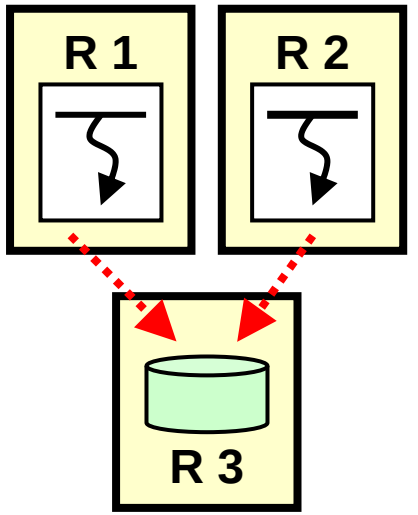
Tekst licencije preuzet je s http://creativecommons.org/.

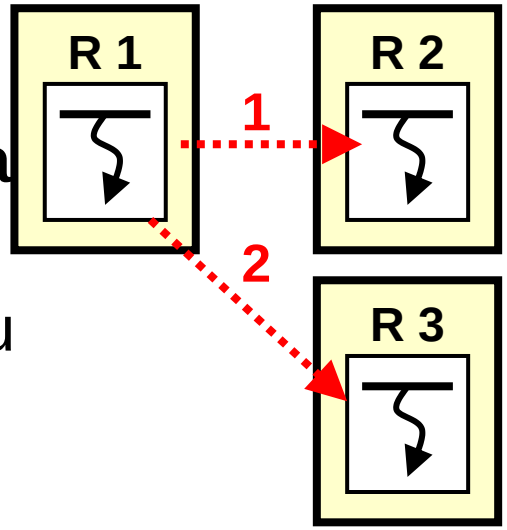
Sadržaj predavanja

- Potreba za sinkronizacijom procesa
- Primjena sata u jednoprocesorskoj okolini
- Primjena sata raspodijeljenoj okolini
- Usklađivanje odluka u raspodijeljenoj okolini
- Sinkronizacija procesa
- Međusobno isključivanje procesa

Potreba za sinkronizacijom procesa

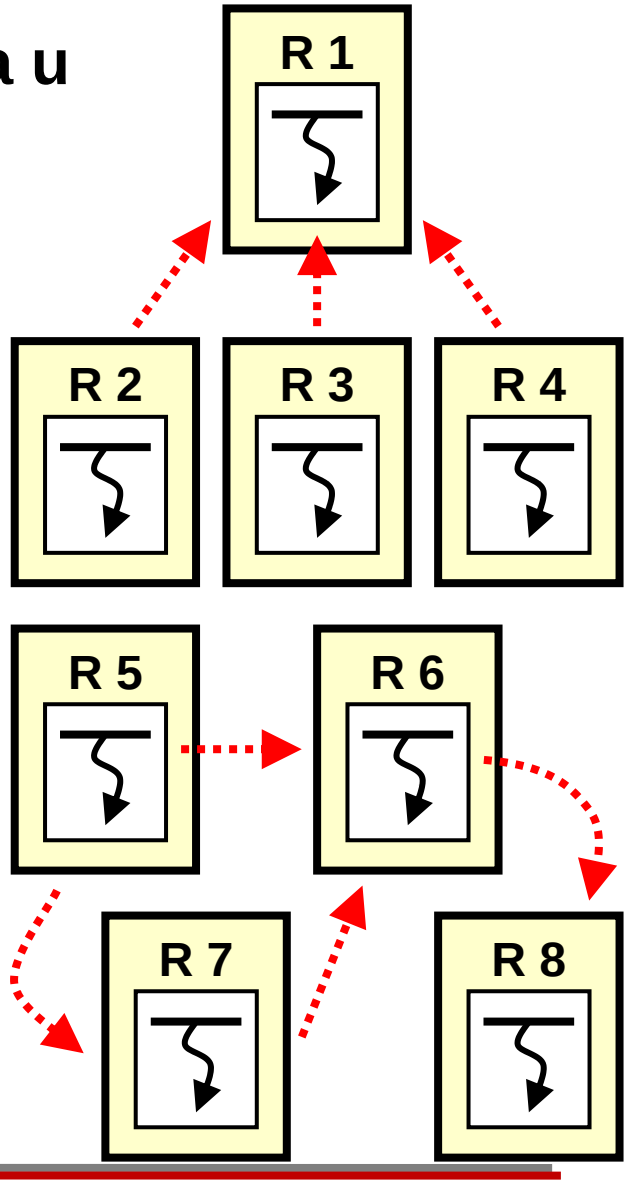
- **Uporaba dijeljenih sredstava u raspodijeljenoj okolini**
  - Procesi istodobno ostvaruju pristup dijeljenim sredstvima
  - Potrebno je ostvariti pristup dijeljenom sredstvu na međusobno isključiv način
  - Dogovor o redoslijedu ostvarivanja pristupa
- **Usuglašavanje vremenskog redoslijeda akcija**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





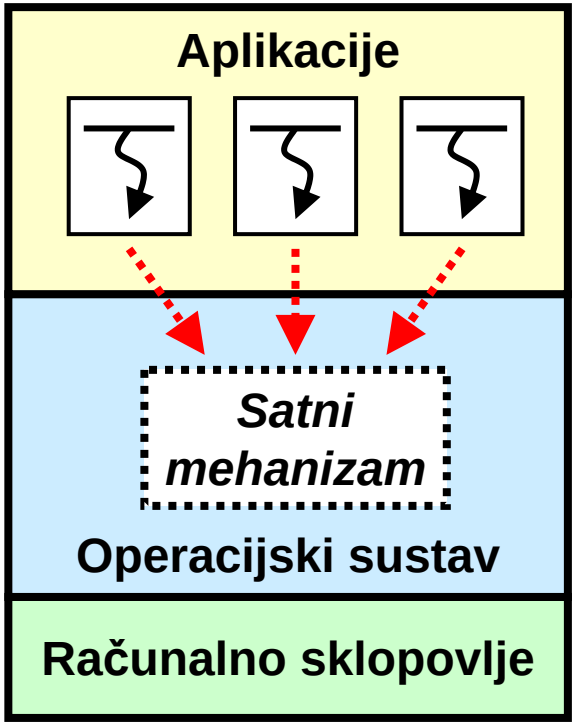
Potreba za sinkronizacijom procesa

- **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



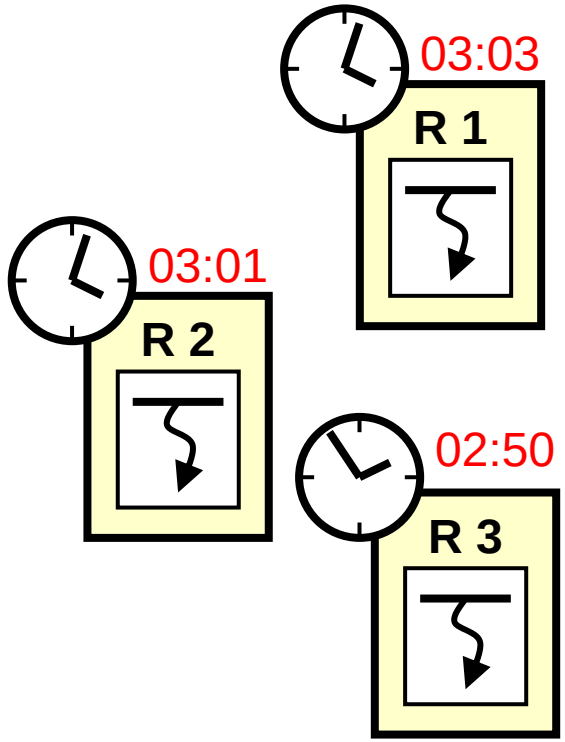
Primjena sata u jednoprocesorskoj okolini

- **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



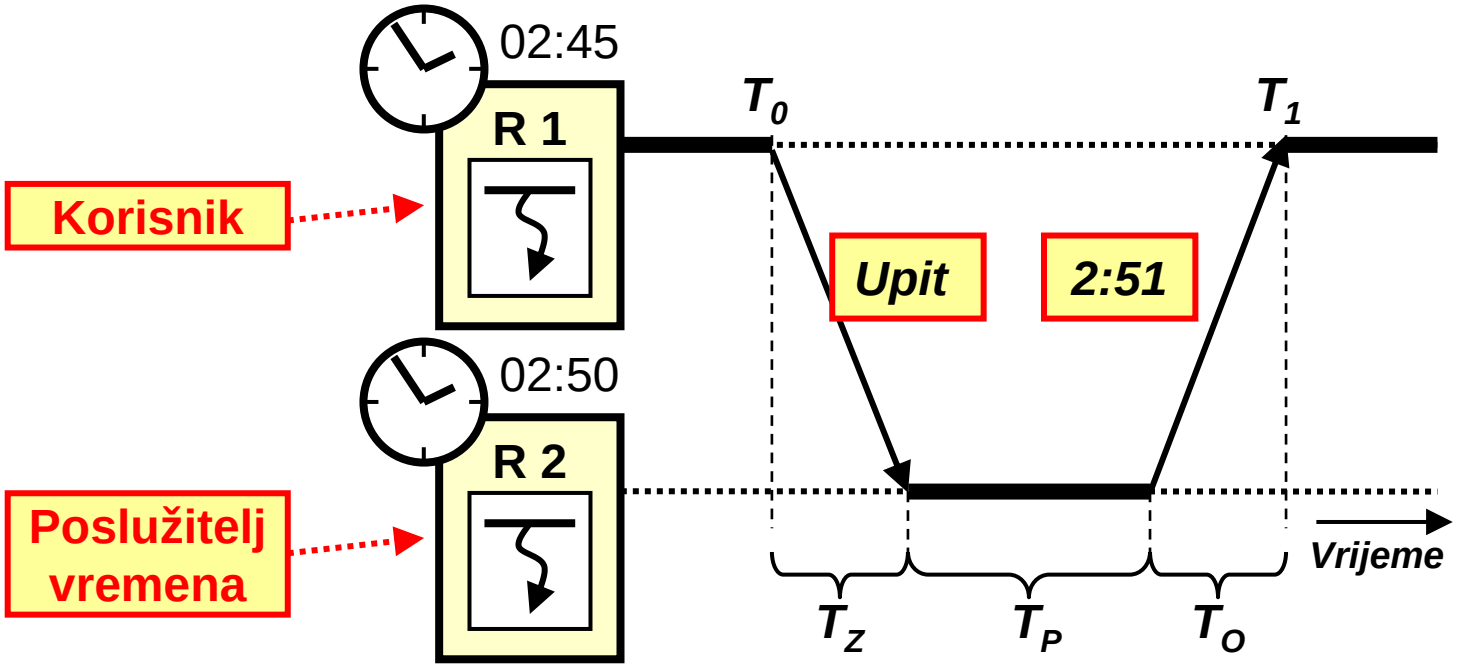
Fizički sat u raspodijeljenoj okolini

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

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

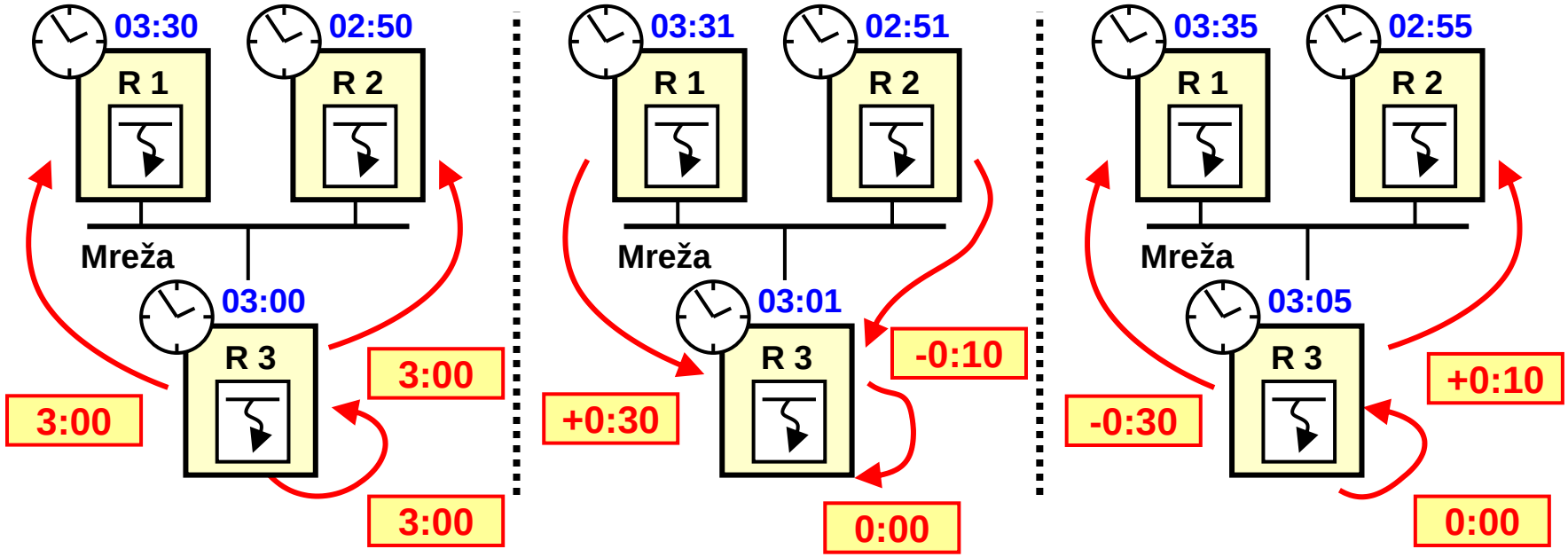
Christian algoritam

- **Primjena poslužitelja vremena**
- **Koraci algoritma**
  1) Korisnik upućuje zahtjev za dohvat vremena
  2) Poslužitelj šalje trenutno vrijeme



Berkeley algoritam

- **Primjena upravitelja vremena**
- **Koraci algoritma**
  1) Upravitelj šalje svoje vrijeme ostalim računalima
  2) Računala šalju razliku vremena
  3) Upravitelj šalje pomak vremena za sva računalo



Logički sat u raspodijeljenoj okolini

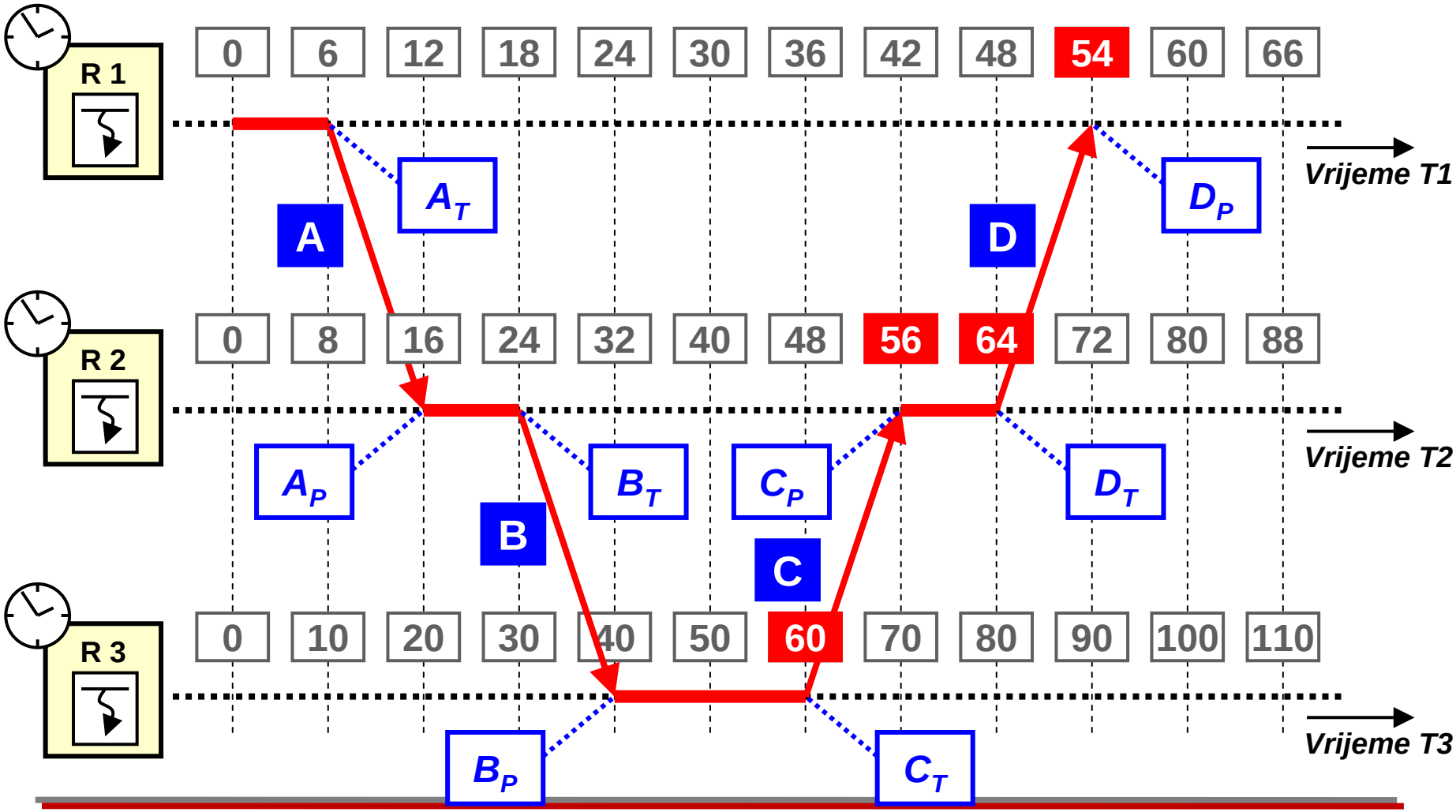
- **Uspostava relacije prethodi**
  - Relacija *prethodi* $\rightarrow$ definira vremensku uređenost akcija u raspodijeljenoj okolini
- **Primjer u jednoprocesorskoj okolini**
  - *a*: primitka poruke na računalu A, *b*: slanja poruke na računalu A
  - akcija *a* nastupila je prije akcije *b*: $a \rightarrow b$
- **Primjer u raspodijeljenoj okolini**
  - *a*: akcija slanja poruke na računalu A prema računalu B, *b*: akcija primitka odaslane poruke na računalu B
  - akcija *a* nastupila je prije akcije *b*: $a \rightarrow b$

Logički sat u raspodijeljenoj okolini

- **Skup fizičkih satnih mehanizama**
  - Satni mehanizmi su potpuno nezavisni
  - Nije moguće uspostaviti relaciju *prethodi* u raspodijeljenoj okolini
- **Usklađivanje globalnog tijeka vremena**
  - Primjena logičkih oznaka vremena
- **Vrste logičkih oznaka**
  - Skalarne oznake vremena
  - Vektorske oznake vremena

Primjena fizičkih satnih mehanizama

$T1(A_T) < T2(A_P)$, $T2(B_T) < T3(B_P)$ **$T3(C_T) > T2(C_P)$** **$T2(D_T) > T1(D_P)$**

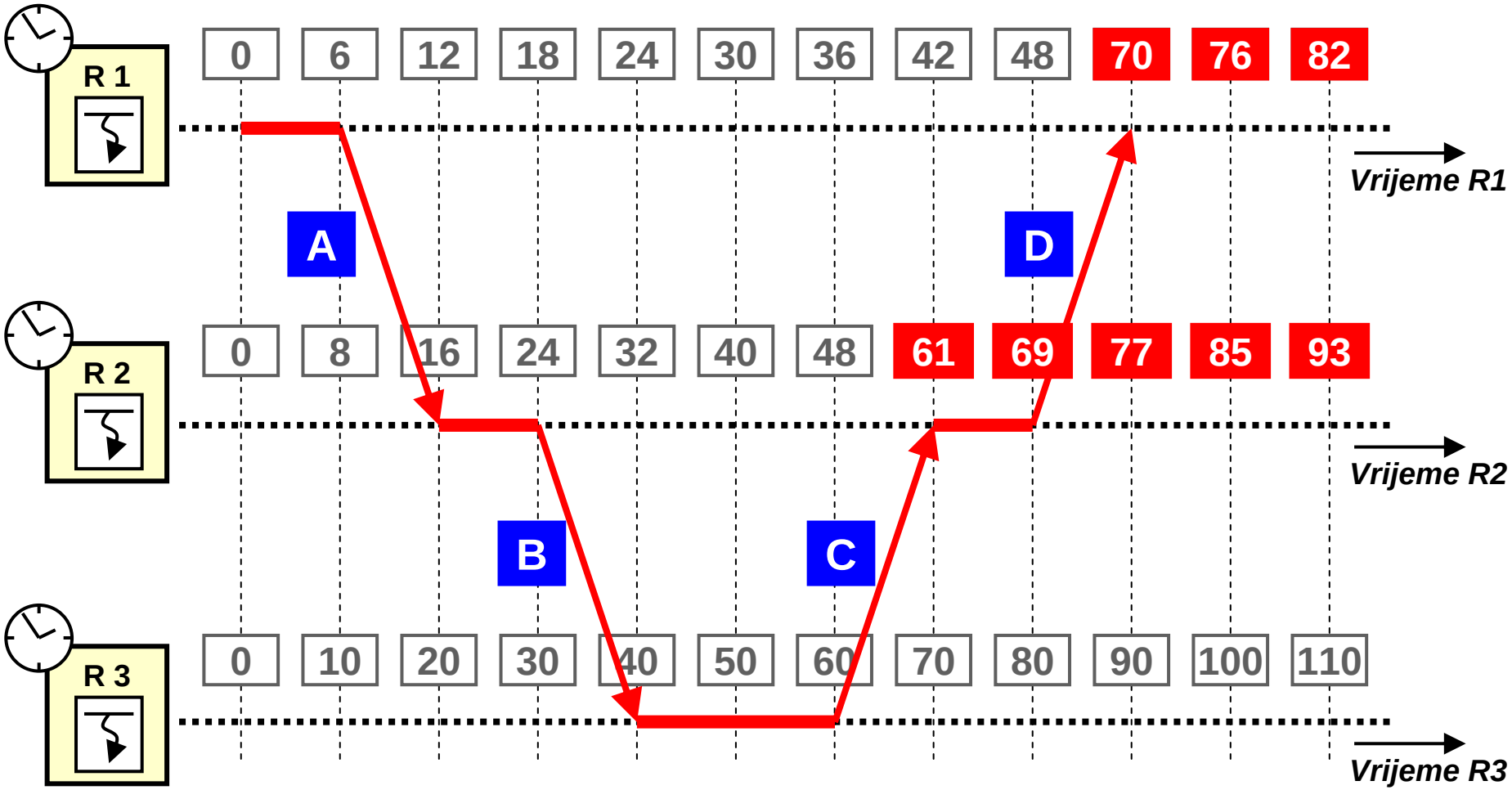


Skalarne oznake vremena

- **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 je akcija *a* ostvarena u vremenu prije akcije *b* tada vrijedi: T(*a*) < T(*b*) i vrijedi relacija $a \rightarrow b$

Skalarne oznake vremena

$T1(A_T) < T2(A_P)$, $T2(B_T) < T3(B_P)$ | $T3(C_T) < T2(C_P)$ | $T2(D_T) < T1(D_P)$

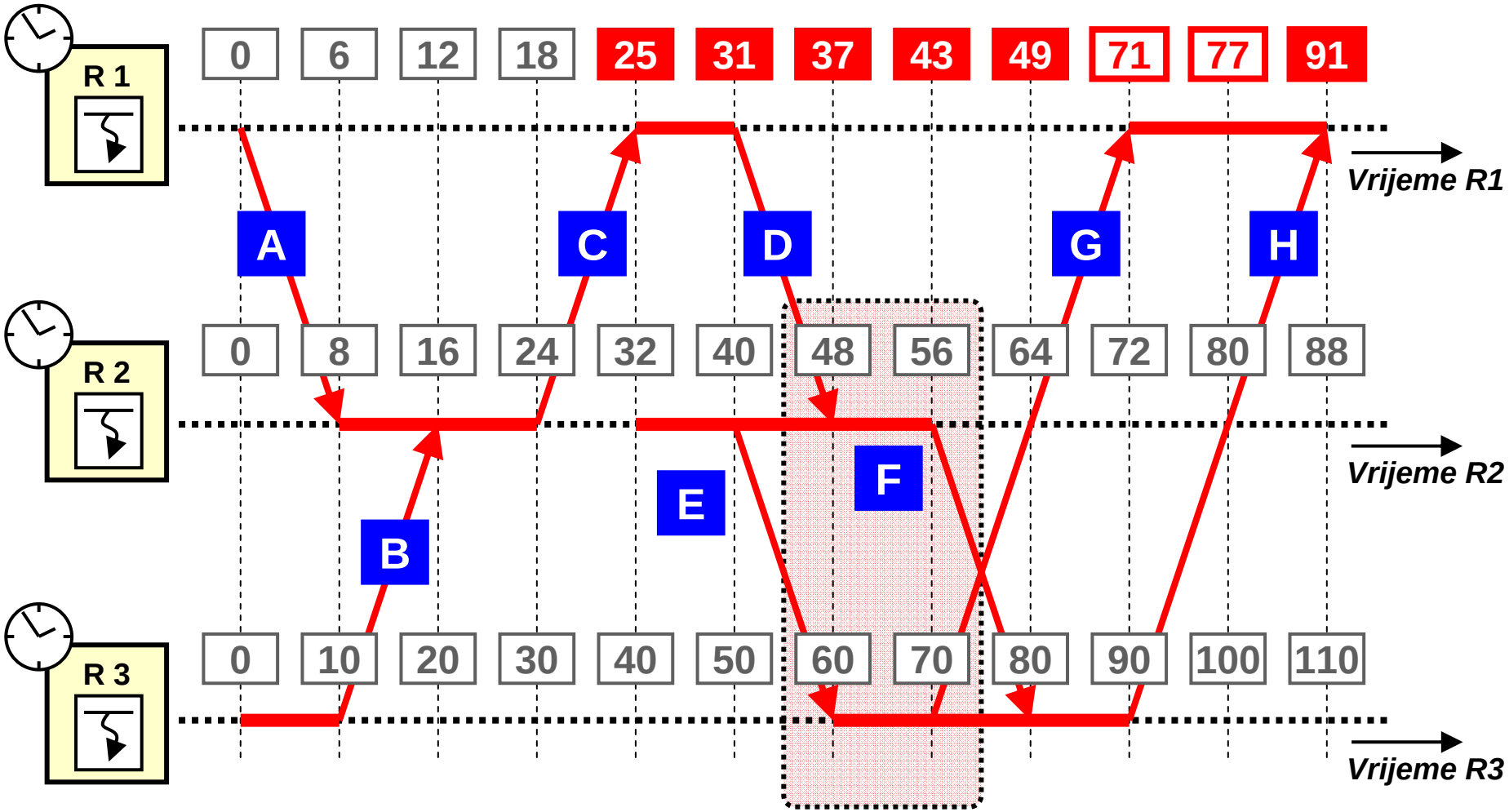


Skalarne oznake vremena

- **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 $V[N]$ koji 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

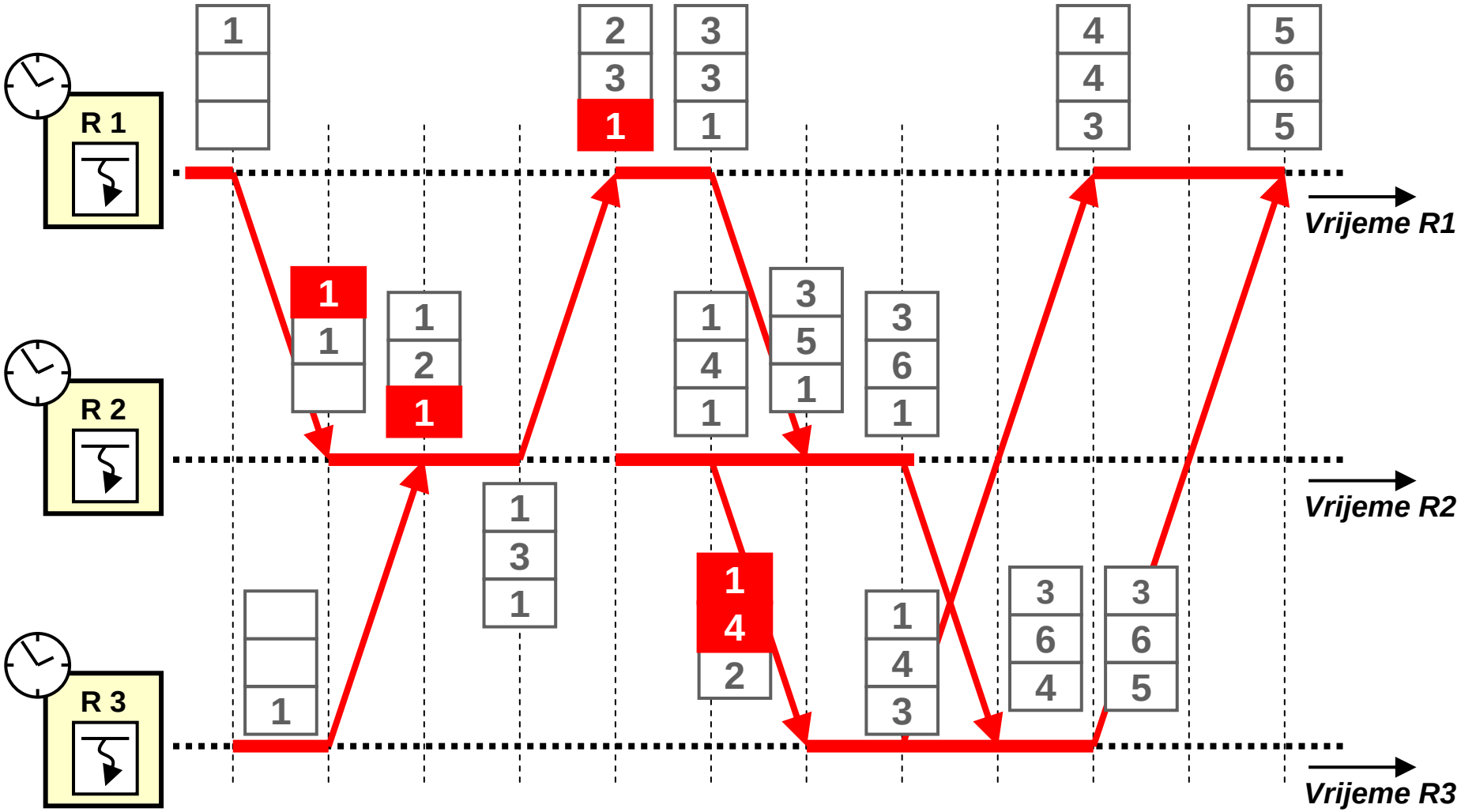
- **Primjena vektorskih oznaka**
  - Ako za događaje *a* i *b* vrijedi $V(a) < V(b)$ tada vrijedi da je događaj *a* nastupio u vremenu prije događaja *b*, $a \rightarrow b$

- **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) Početne 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 $V_k[j] < V_p[j]$

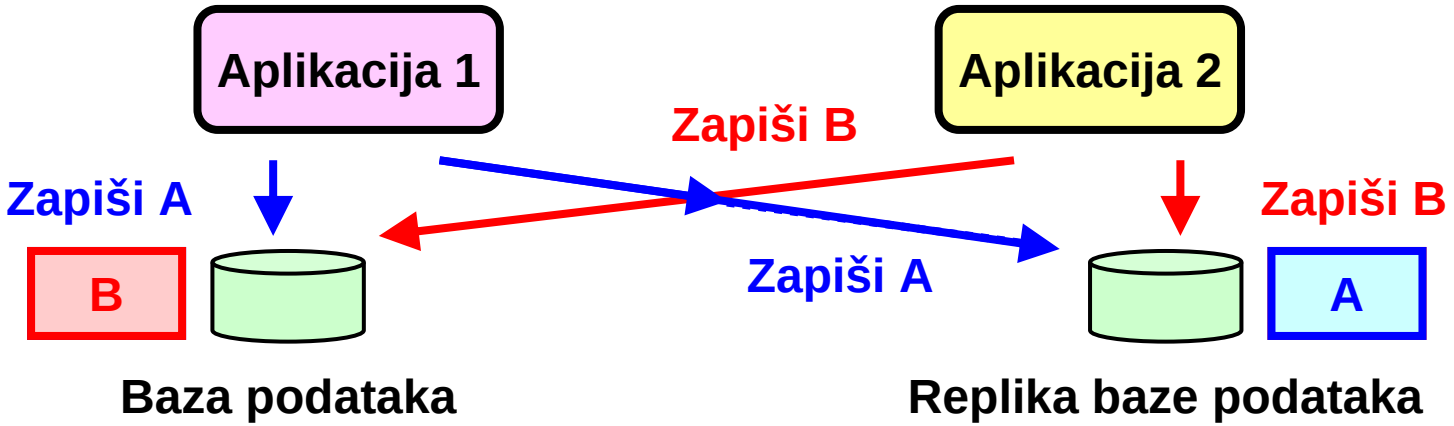
Vektorske oznake vremena



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
- Usklađivanje odluka u raspodijeljenoj okolini
- Sinkronizacija procesa
- Međusobno isključivanje procesa

Usklađivanje odluka u raspodijeljenoj okolini

- **Nametanje odluke**
  - Skup računala međusobno se natječe za preuzimanje uloge upravitelja
- **Pretpostavke**
  - Računala znaju adrese svih ostalih računala
  - Računala ostvaruju izravnu komunikaciju razmjenom poruka
- **Rezultat**
  - Upravitelj postaje računalo s najvećim identifikatorom

Usklađivanje odluka u raspodijeljenoj okolini

- **Ispad postojećeg upravitelja**
  - Upravitelj više ne odgovara na poruke zahtjeva
  - Postupak započinje kada prvo računalo uoči ispad postojećeg računala upravitelja

- **Računalo A koje započinje postupak**
  1) Šalje poruku *izbor* (*I*) svim nadređenim računalima
  2) Ako bilo koje nadređeno računalo odgovori s porukom *potvrda* (*P*), računalo A nije upravitelj
  3) Ako niti jedno nadređeno računalo ne odgovori s porukom *potvrda* (*P*), računalo A postaje novi upravitelj

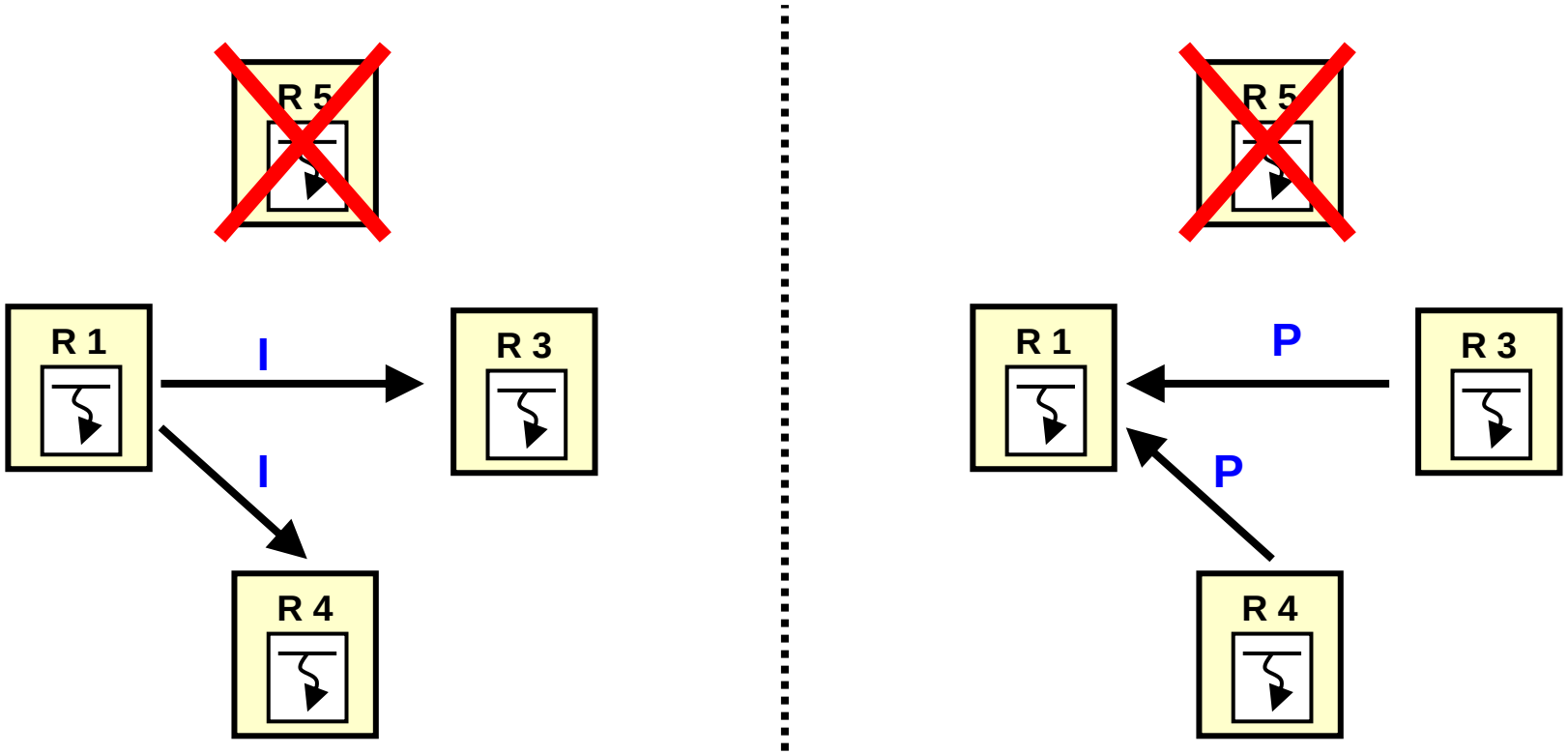
Usklađivanje odluka u raspodijeljenoj okolini

- **Ostala računala**
  1) Prima poruku izbor (*I*) i odgovara s porukom potvrda (*P*)
  2) Ako ne postoji nadređeno računalo, računalo postaje novi upravitelj

- **Računalo koje je postalo upravitelj**
  1) Šalje poruku upravitelj (*U*) svim ostalim računalima u raspodijeljenoj okolini
  2) Zadaje zadatke podređenim računalima

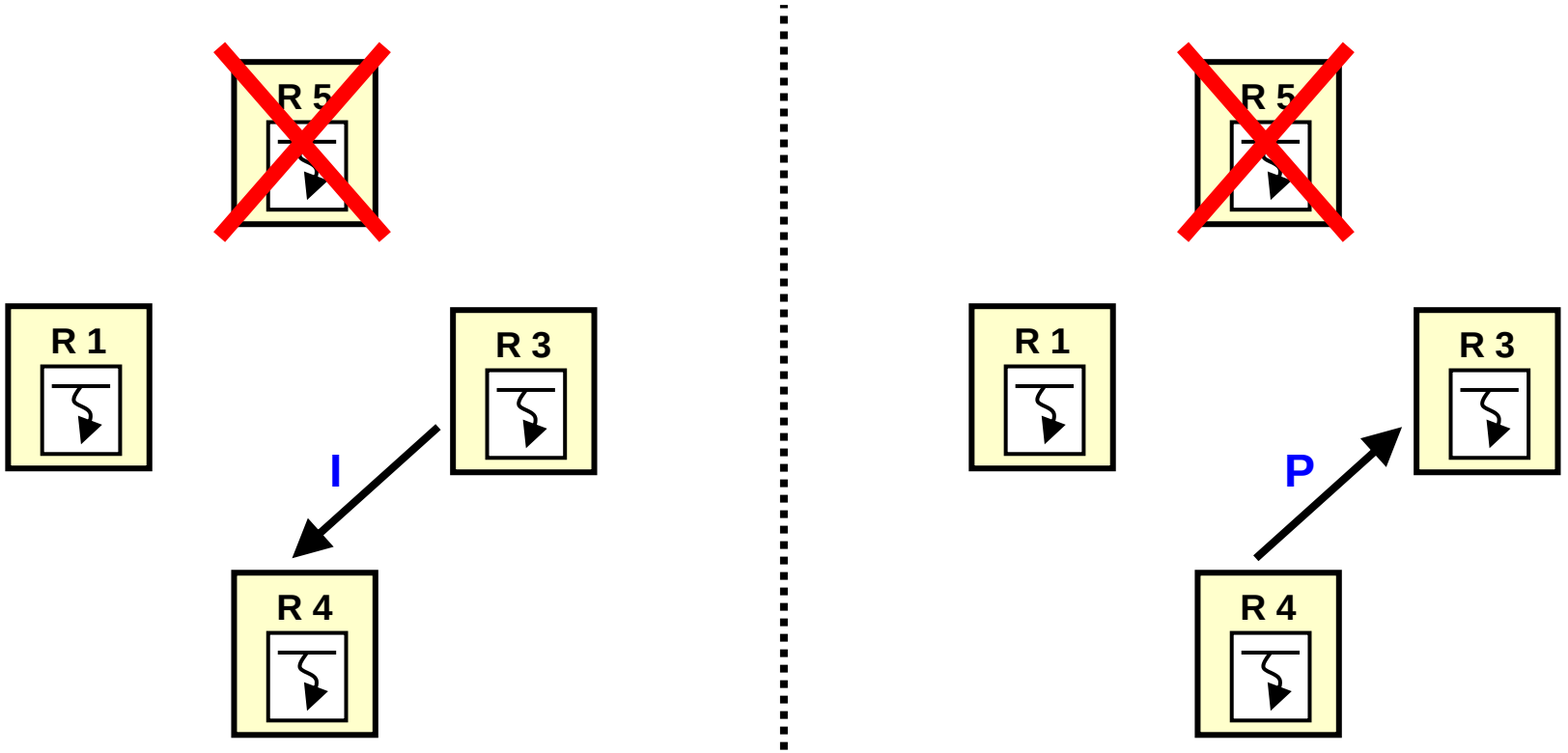
Usklađivanje odluka u raspodijeljenoj okolini

- **Računalo R1 primijetilo je ispad upravljača**
  - Šalje poruku *izbor* (*I*) svim nadređenim računalima
  - Računala odgovaraju s porukom *potvrda* (*P*)



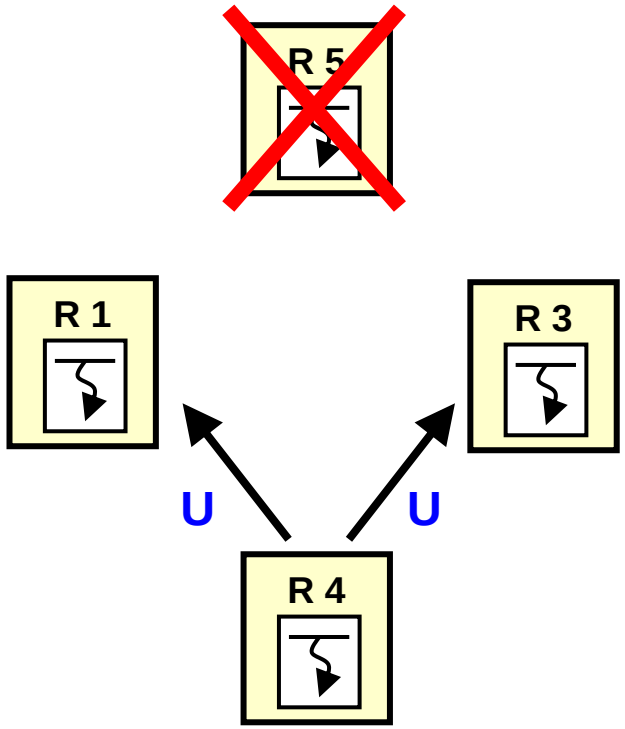
Usklađivanje odluka u raspodijeljenoj okolini

- **Prosljeđivanje poruke izbor**
  - Računalo R3 šalje poruku *izbor* (*I*) računalu R4
  - Računalo R4 odgovara s porukom *potvrda* (*P*)



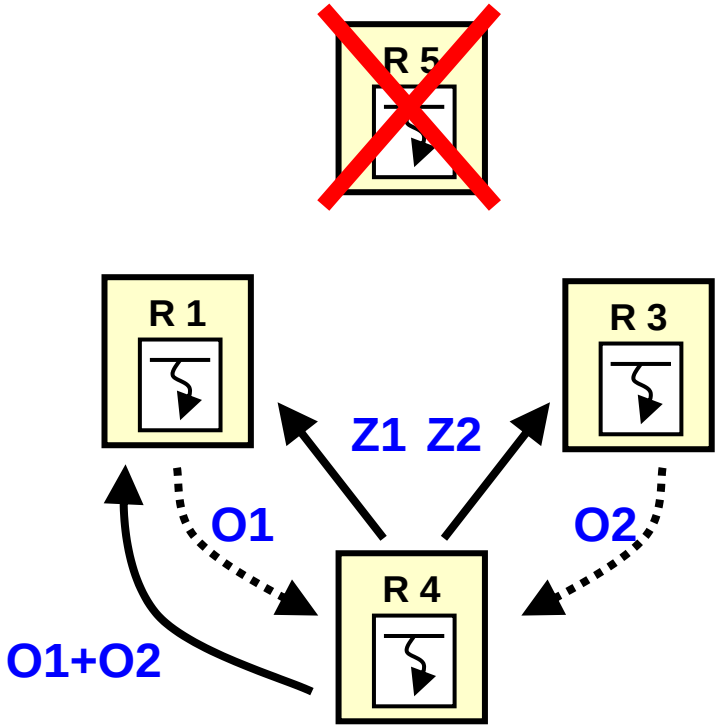
Usklađivanje odluka u raspodijeljenoj okolini

- **Računalo R4 nema nadređenih računala**
  - Postaje novi upravitelj
  - Šalje poruku upravitelj (*U*) svim podređenim računalima



Usklađivanje odluka u raspodijeljenoj okolini

- **Upravljanje tijekom izvođenja zadataka**
  - Prosljeđivanje zadatka Z1 ⇒ R1 i Z2 ⇒ R2
  - Primitak odgovora O1 i O2
  - Slanje odgovora računalu R1 na obradu



Međusobno isključivanje procesa

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

Međusobno isključivanje procesa

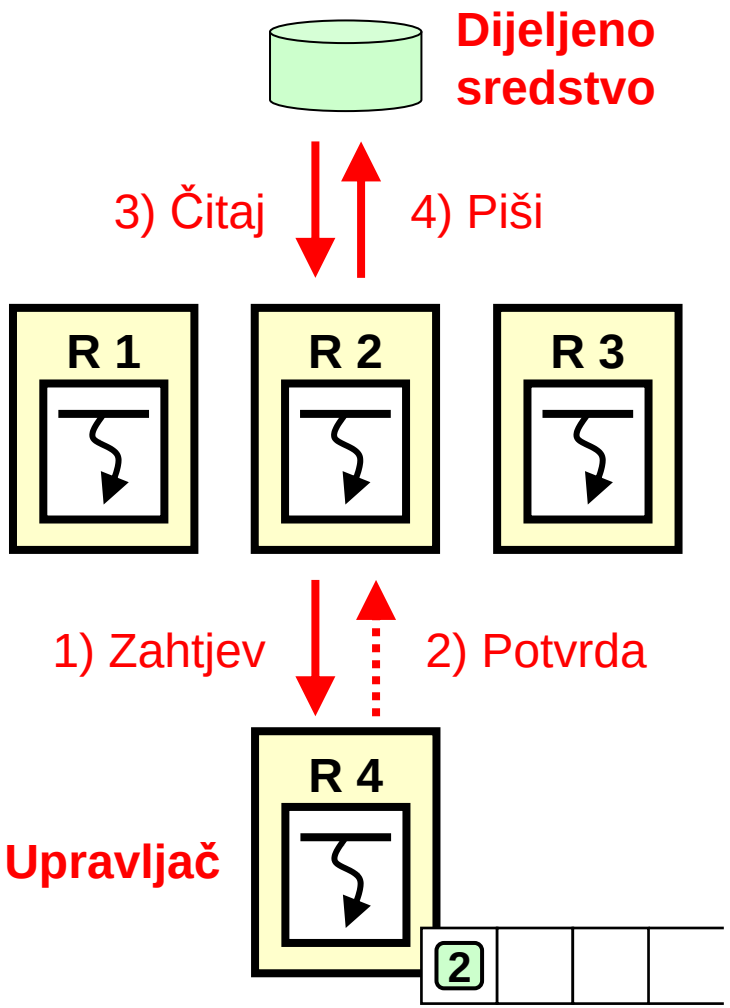
- **Središnji upravljač s repom čekanja**
  - Računalo zaduženo za čuvanje stanja repa čekanja
  - Rep čekanja zasnovan na posluživanju zahtjeva prema redoslijedu prispijeća (*FIFO*)

- **Korisnici**
  - Šalju poruke *zahtjev* (*Z*) za pristup sredstvu
  - Započinju pristup sredstvu nakon primitka poruke *potvrda* (*P*)
  - Ostvaruju poruke *odgovor* (*O*) za oslobađanje pristupa sredstvu

Središnji upravljač s repom čekanja

- **Primjena središnjeg upravljača**

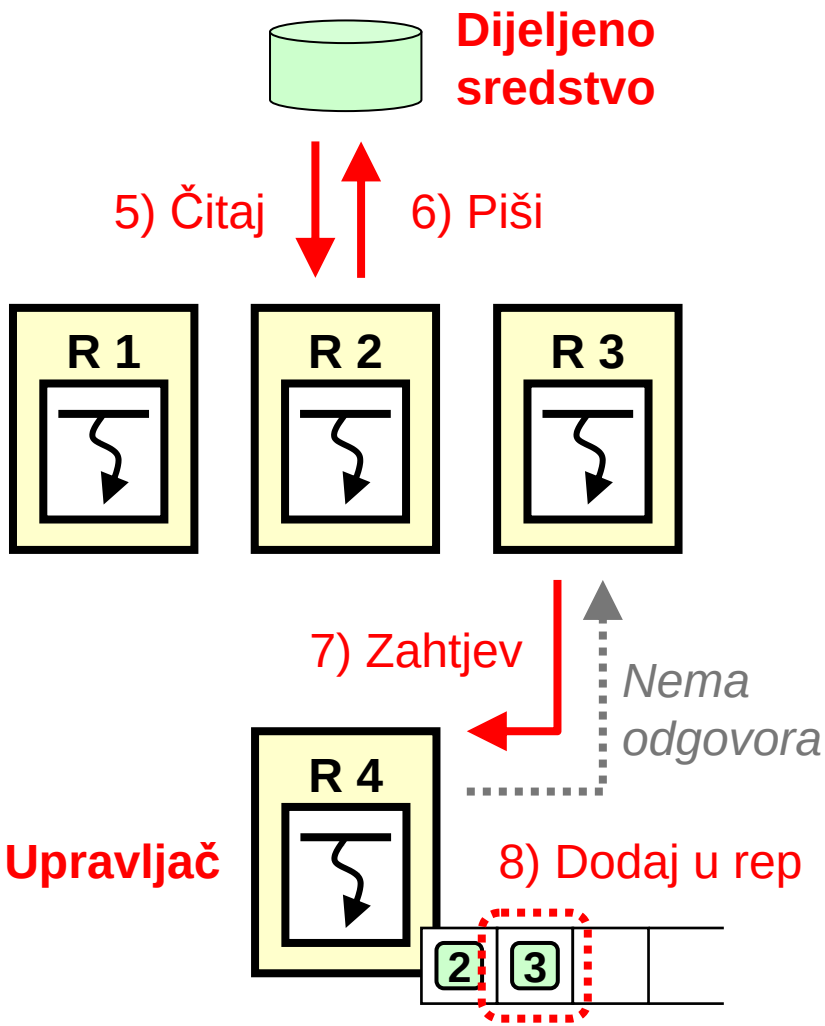
1) Računalo R2 upućuje zahtjev Upravljaču
2) Upravljač stavlja zahtjev u rep čekanja i šalje potvrdu računalu R2
3) Računalo R2 provodi operaciju čitanja
4) Računalo R2 provodi operaciju pisanja



Središnji upravljač s repom čekanja

- **Primjena središnjeg upravljača**

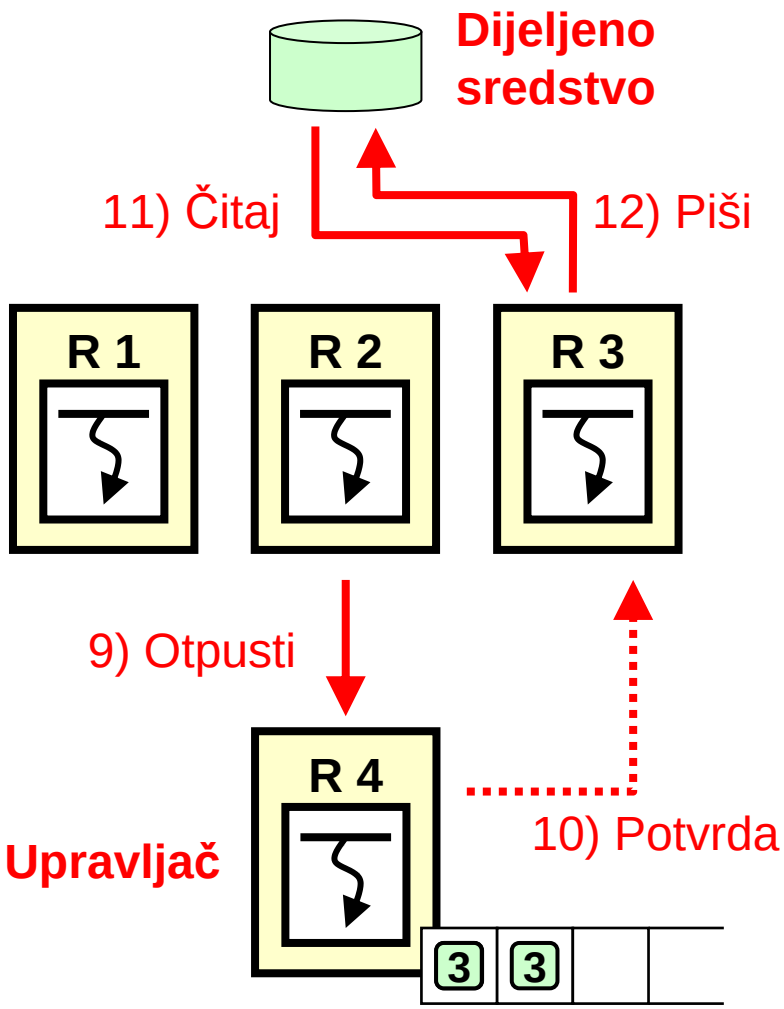
5) Računalo R2 provodi operaciju čitanja
6) Računalo R2 provodi operaciju pisanja
7) Računalo R3 upućuje zahtjev Upravljaču
8) Upravljač stavlja zahtjev u rep čekanja



Središnji upravljač s repom čekanja

- **Primjena središnjeg upravljača**

9) Računalo R2 šalje zahtjev za otpuštanje
10) Upravljač uklanja zahtjev računala R2 iz repa poruka i šalje potvrdu računalu R3
11) Računalo R3 provodi operacije čitanja
12) Računalo R3 provodi operacije čitanja
13) …



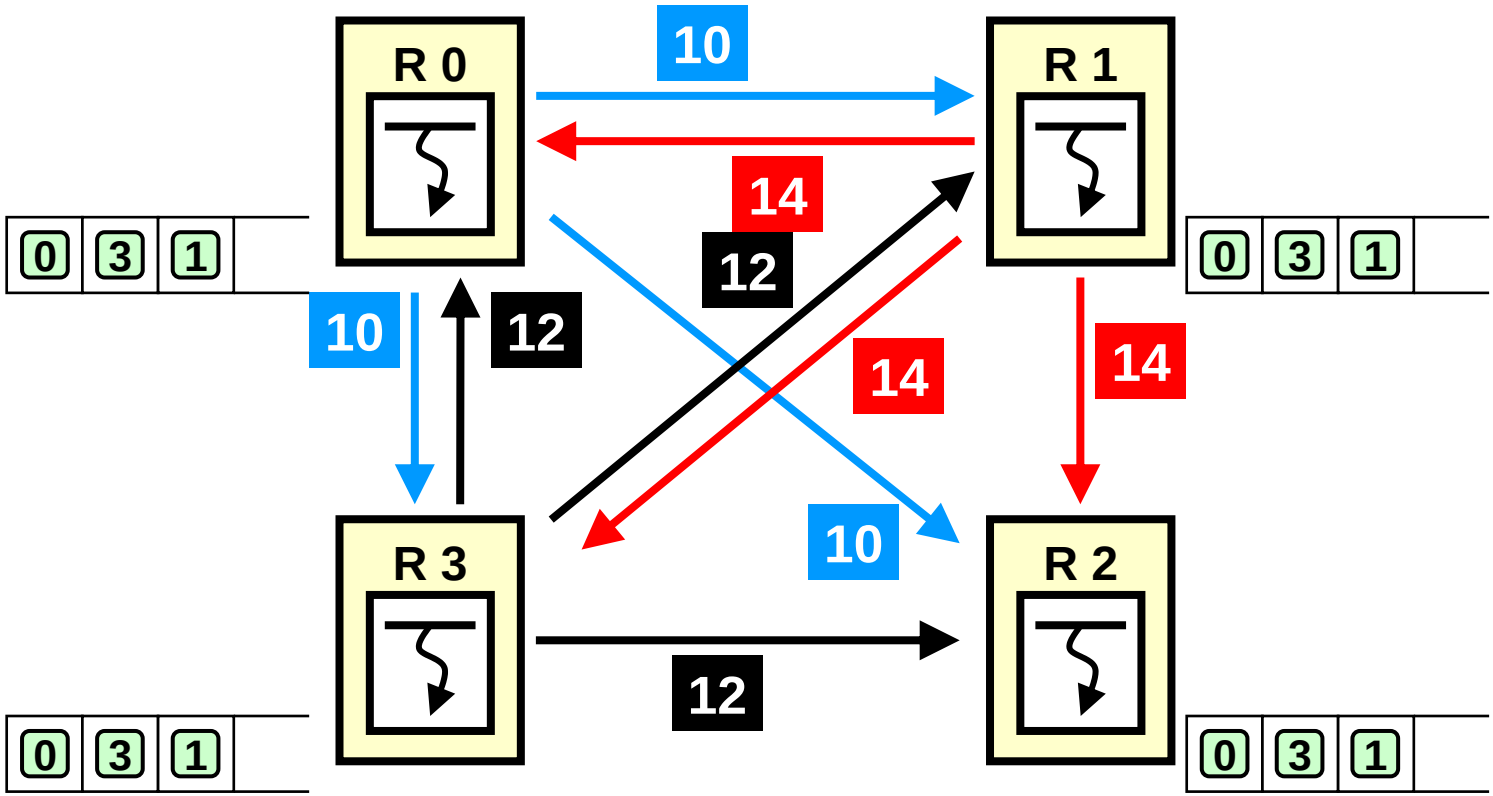
Raspodijeljeno međusobno isključivanje

- **Raspodijeljeni rep čekanja**
  - Svako računalo ima lokalni rep čekanja
  - Računala razmjenjuju informacije potrebne za usklađivanje stanja svih repova čekanja u sustavu

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

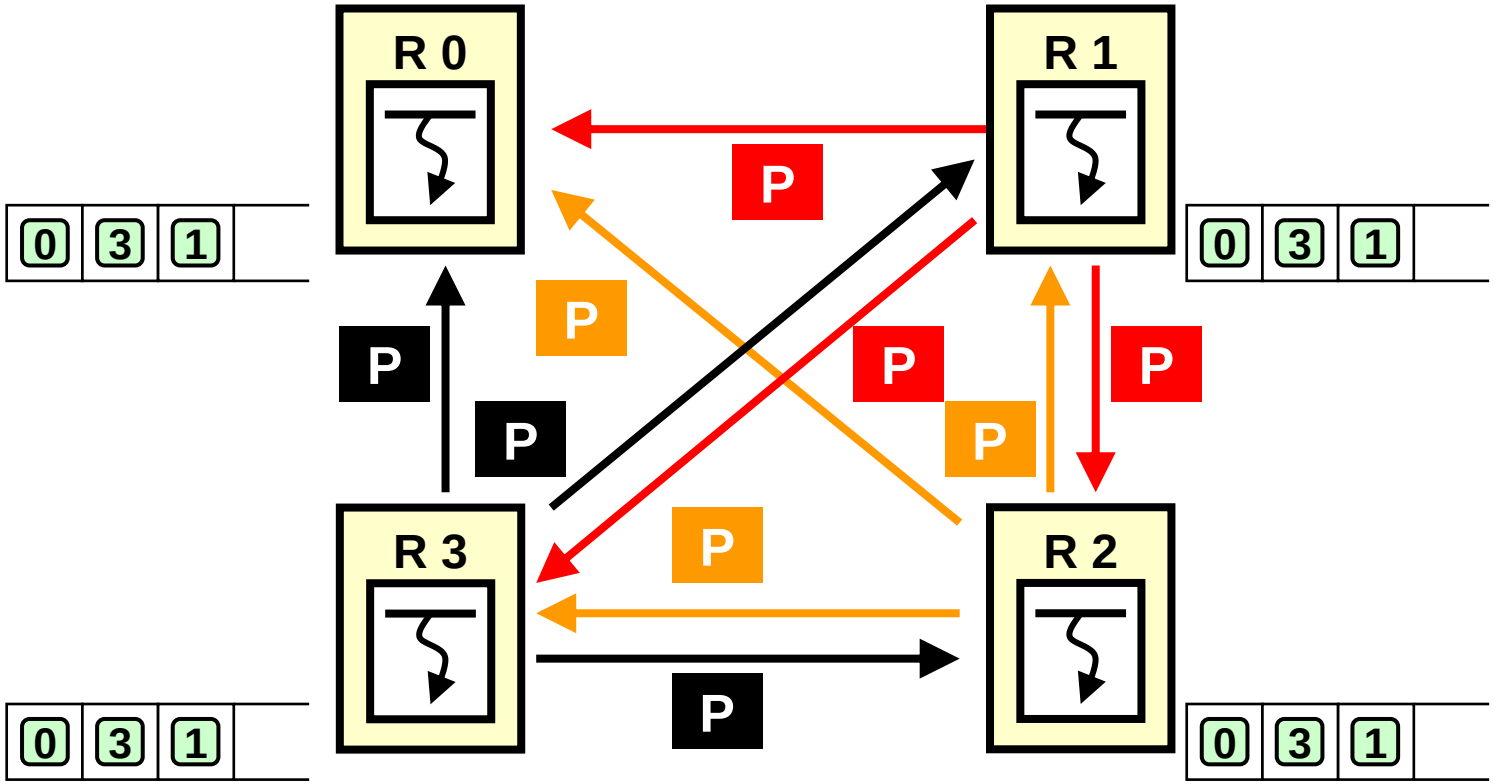
Raspodijeljeno međusobno isključivanje

- Računalo R0, R1 i R2 upućuju zahtjev za pristup dijeljenom sredstvu svim računalima
- Zahtjevi uključuju oznake trenutka upućivanja zahtjeva



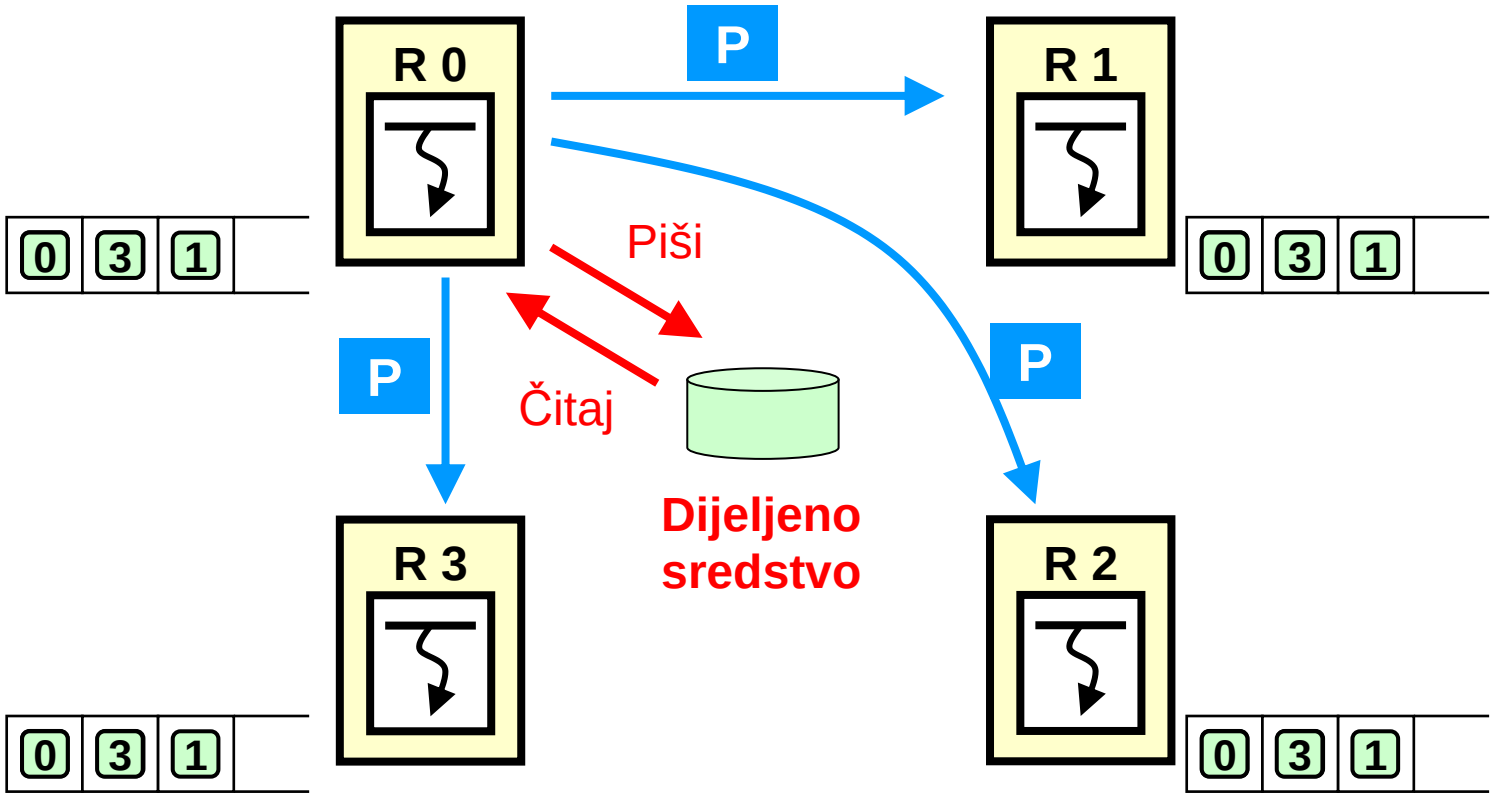
Raspodijeljeno međusobno isključivanje

- Računalo R0 ima zahtjev s najmanjom oznakom vremena, preostali zahtjevi se stavljaju u repove
- Računala R1 i R3 šalju potvrde svim računalima



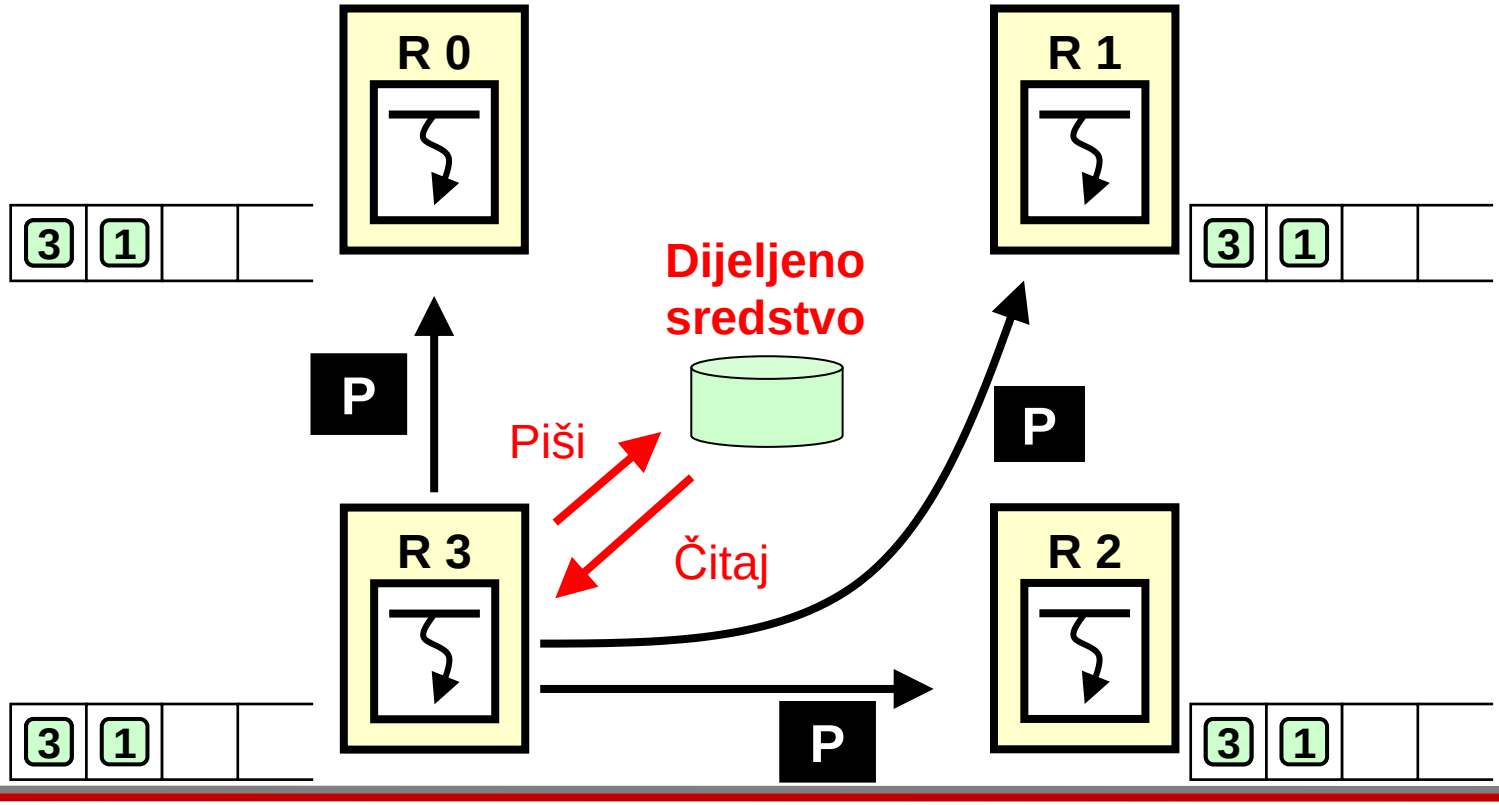
Raspodijeljeno međusobno isključivanje

- Računalo R0 ostvaruje pristup dijeljenom sredstvu
- Računalo R0 šalje potvrdu svim računalima
- Računalo R3 dobiva pravo pristupa



Raspodijeljeno međusobno isključivanje

- Računalo R3 ostvaruje pristup dijeljenom sredstvu
- Računalo R3 šalje potvrdu svim računalima
- Računalo R1 dobiva pravo pristupa

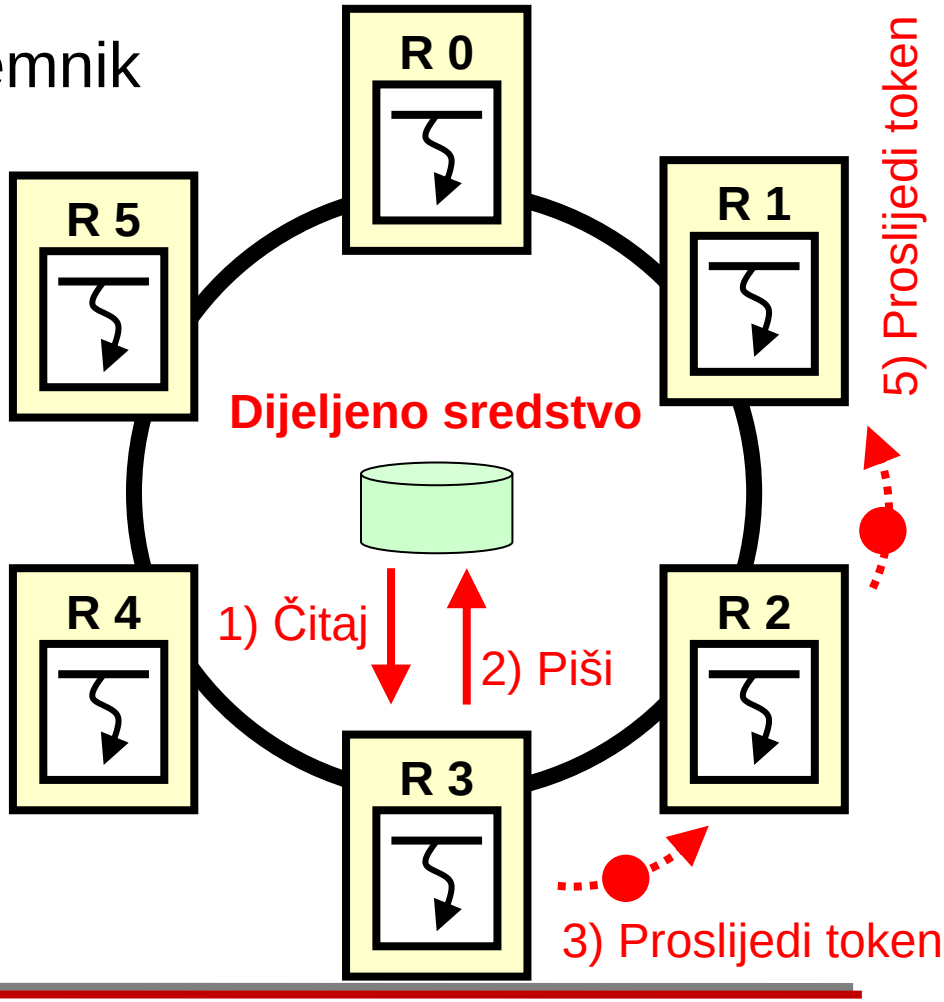


Međusobno isključivanje primjenom prstena

- **Struktura prstena računala**
  - Računala 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 računalo koje u određenom trenutku ima token
  - Nakon završetka pristupa, računalo prosljeđuje token susjednom računalu u prstenu

Međusobno isključivanje primjenom prstena

- **Računalo R3 prima token**
  - **1)** Čitanje podataka iz spremnika
  - **2)** Pisanje podataka u spremnik
  - **3)** Prosljeđivanje tokena računalu R2
- **Računalo R2 prima token**
  - **4)** Računalo ne zahtjeva pristup spremniku
  - **5)** Prosljeđivanje tokena računalu R1



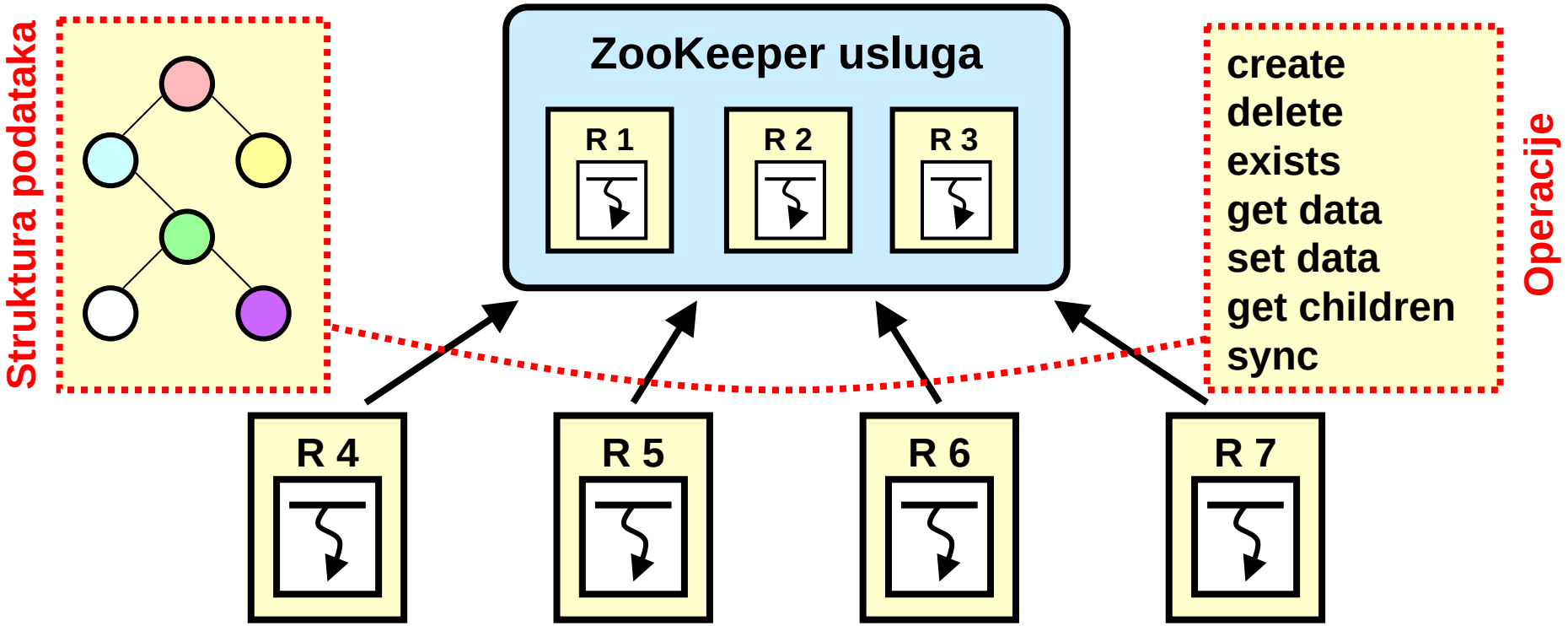
Primjer sustava za sinkronizaciju

- **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

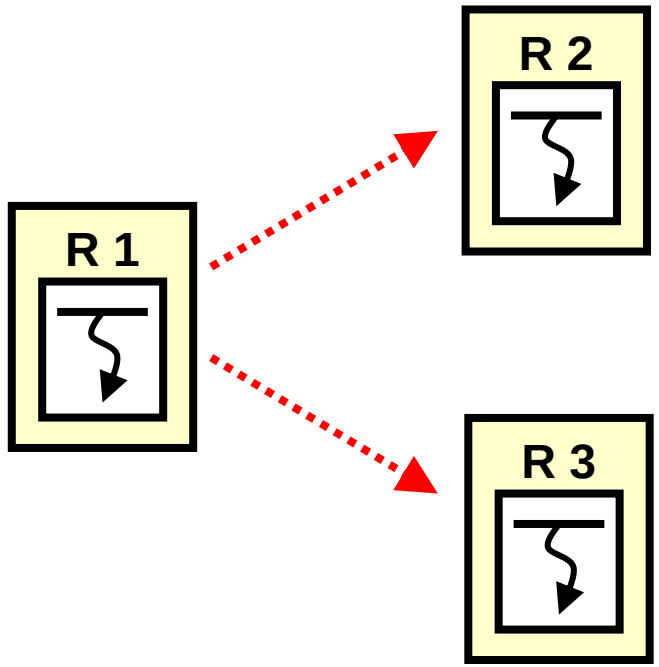
Usluga ZooKeeper

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



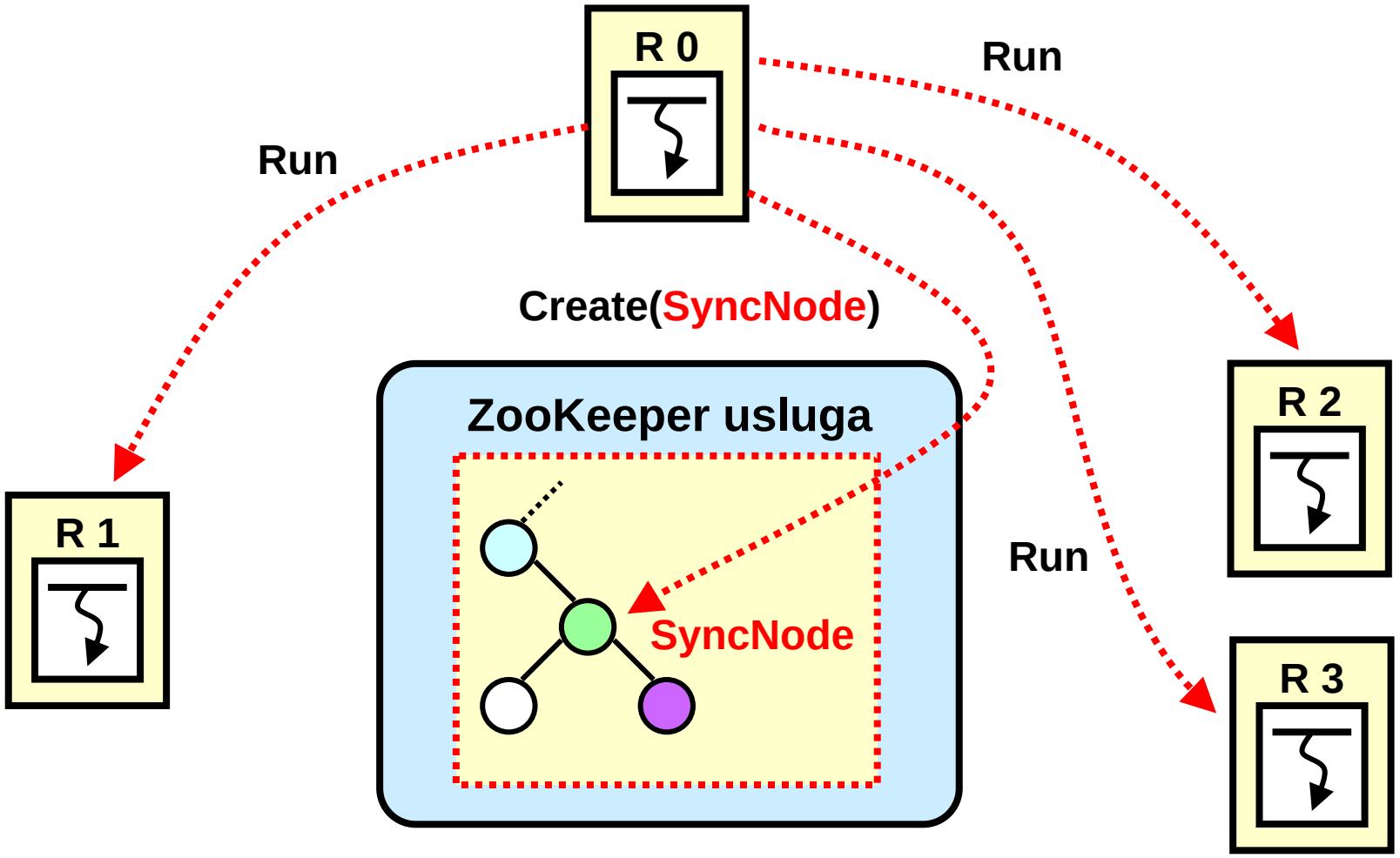
Usluga ZooKeeper

- **Primjer: Sinkronizacija procesa**
  - Procesi na računalima R2 i R3 započinju s izvođenjem tek nakon što je proces R1 završio s izvođenjem



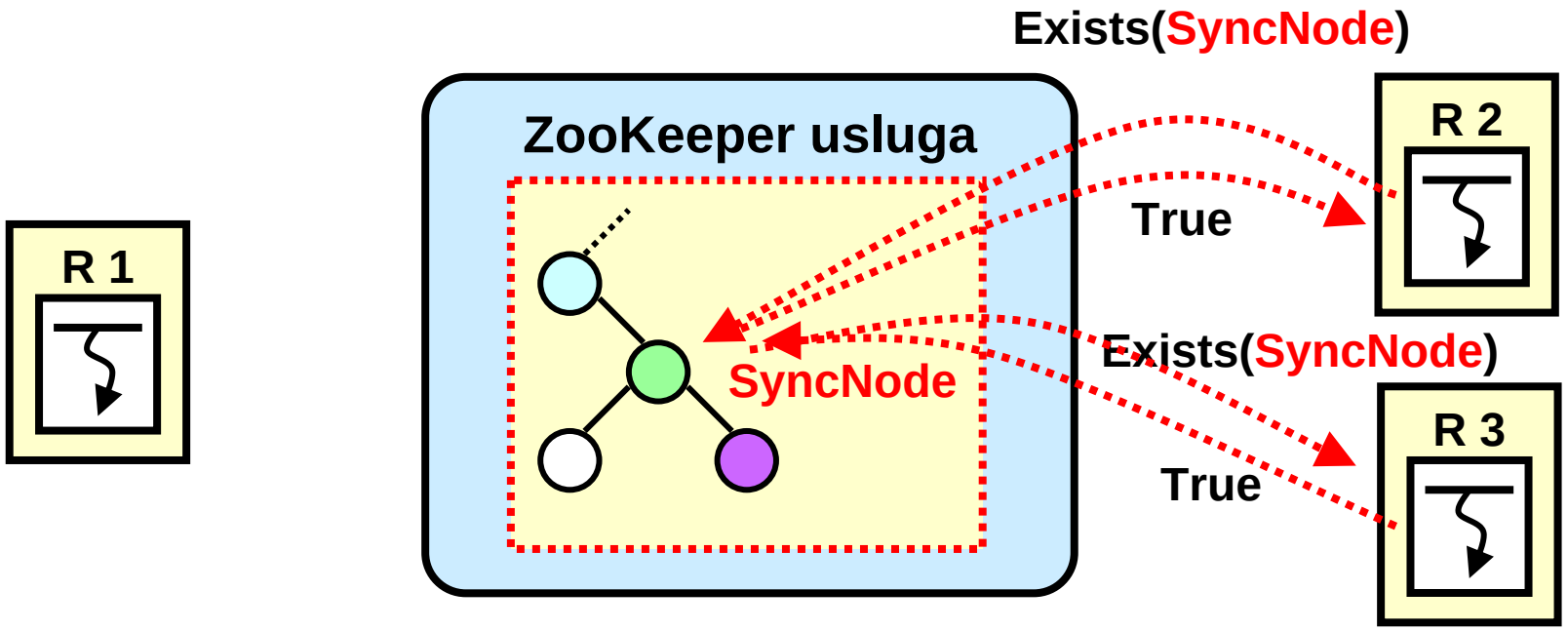
Usluga ZooKeeper

- Računalo R0 izvodi upravljački proces



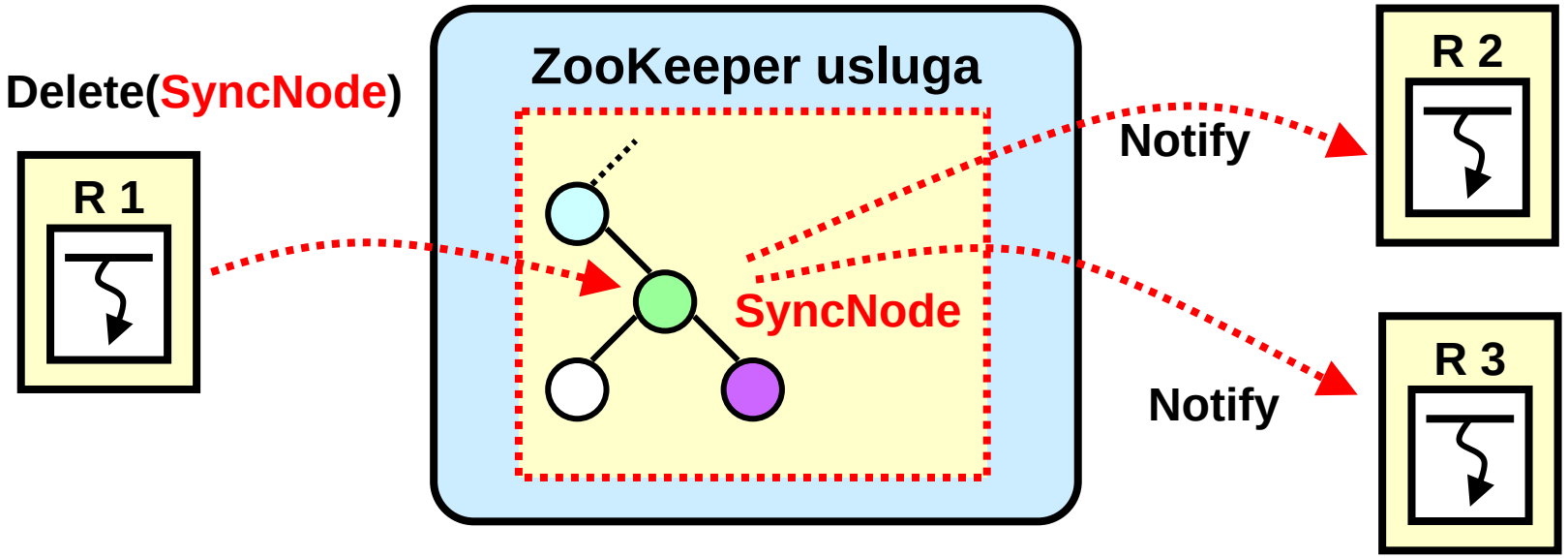
Usluga ZooKeeper

- **Procesi na računalima R2 i R3 ispituju postoji li sinkronizacijski čvor**
  - Ako čvor postoji, čekaju na dojavu o brisanju čvora
  - Kada se čvor izbriše, započinju izvođenje



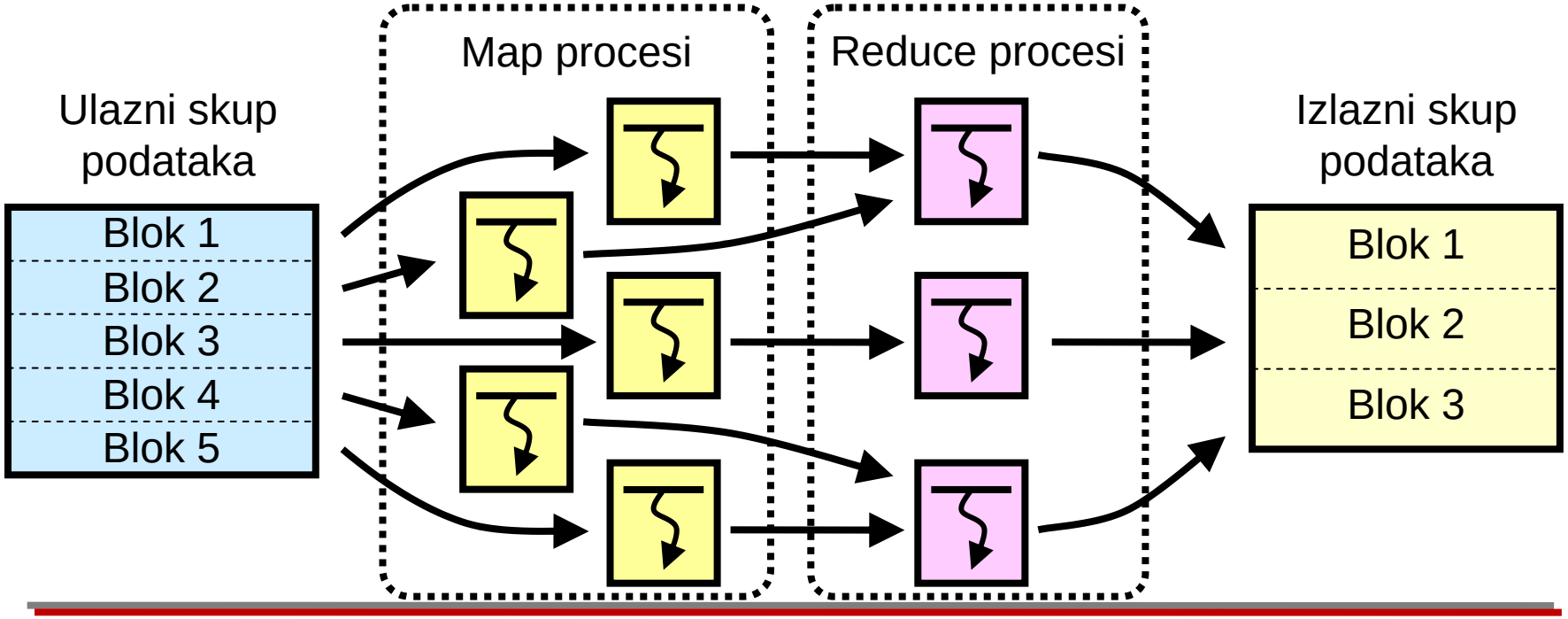
Usluga ZooKeeper

- **Procesi na računalu R1 završava s izvođenjem**
  - Briše čvor SyncNode
  - Usluga ZooKeeper dojavljuje brisanje čvora
  - Procesi na računalima R2 i R3 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()) {                          Brojanje riječi
32:       sum += values.next().get();
33:     }
34:     output.collect(key, new IntWritable(sum));
35:   }
…
```

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