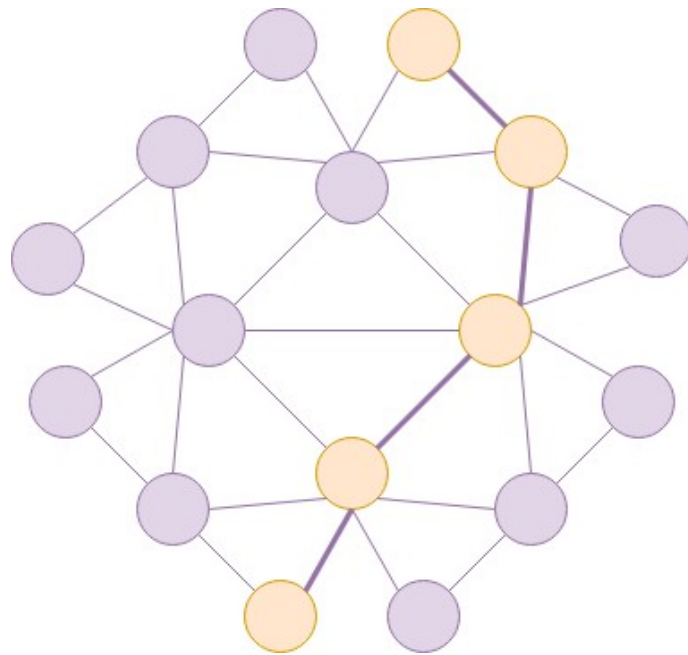


# Distributed Queens

Djordje Velickovic

2019



# Table of Contents

Uvod.....	3
Arhitektura.....	4
Dodavanje cvora.....	4
Kretanje po sistemu.....	5
Izlazak cvora iz sistema.....	6
Posao.....	7
Odredjivanje delova posla.....	7
Komunikacija.....	7
API.....	8
User.....	8
Controls.....	8
POST /control/pause.....	8
POST /control/result.....	8
POST /control/result-simple.....	8
POST /control/start.....	9
GET /control/status.....	9
POST /control/stop.....	9
Node.....	9
Critical section.....	10
POST /critical-section/broadcast.....	10
POST /critical-section/token.....	10
Jobs.....	10
POST /job/queens.....	10
POST /job/queens-pause.....	11
POST /job/queens-result-broadcast.....	11
POST /job/queens-status.....	11
POST /job/queens-status-collector.....	12
POST /job/stealing-collector.....	12
POST /job/stealing-request.....	12
Node endpoints.....	13
POST /node/alter-neighbours.....	13
POST /node/host-ack.....	13
POST /node/host-request.....	14
POST /node/join-broadcast.....	15
POST /node/leave-broadcast.....	15
POST /node/newbie-join.....	15
POST /newbie-accepted.....	16
Bootstrap.....	16
POST /bootstrap/check-in.....	16
POST /bootstrap/check-out.....	17
GET /bootstrap/lock.....	17
GET /bootstrap/unlock.....	17

# Uvod

Projekat je izradjen u Javi u Spring Boot tehnologiji, gde je svaki cvor jedan “mikroservis”.

Cvoru se pri startovanju dodeljuje adresa i port na kojem treba da slusa, kao i adresa i port bootstrap cvora.

Bootstrap cvor je takodje mikroservis, koji dodeljuje redom identifikatore cvorovima (1, 2, 3, ...) ukoliko cvor napusti sistem, bootstrap dekrementira brojac.

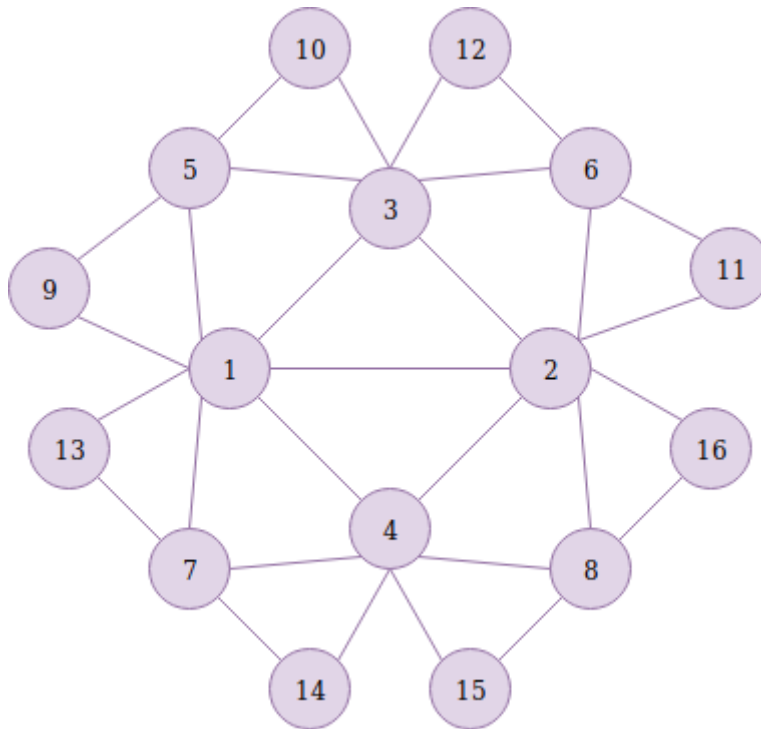
Cvorovi iskljucivo komuniciraju preko HTTP protokola.

Ukoliko su poruke direktne, onda se izvorsavaju po algoritmu koji sledi u narednim poglavljima.

Ukoliko se obavlja broadcast, onda je funkcionise tako sto svaki cvor prosledi poruku svojim susedima.

# Arhitektura

Graf je proizvod mastanja, igre i uocavanja nekih prirodnih pravila.



## Dodavanje cvora

Svako dodavanje cvora prati broadcast, da bi svaki cvor mogao da azurira svog maksimalnog cvora.

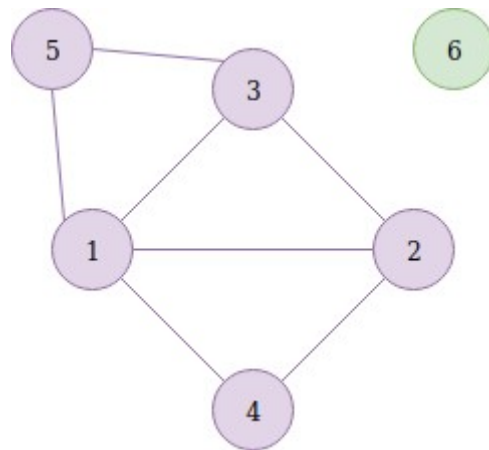
1. Dodavanje prvog cvora je trivijalno.



2. Dodavanje drugog cvora je takodje trivijalno.



n. Dodavanje n-tog cvora se desava na sledeci nacin. Cvor na koga se n povezuje je  $n_p = n / 2 + n \% 2$ .  
2. Ako je cvor neparan onda se on povezuje sa najmanjim susedom cvora  $n_p$ , ukoliko je paran, onda se povezuje sa drugim najmanjim susedom.

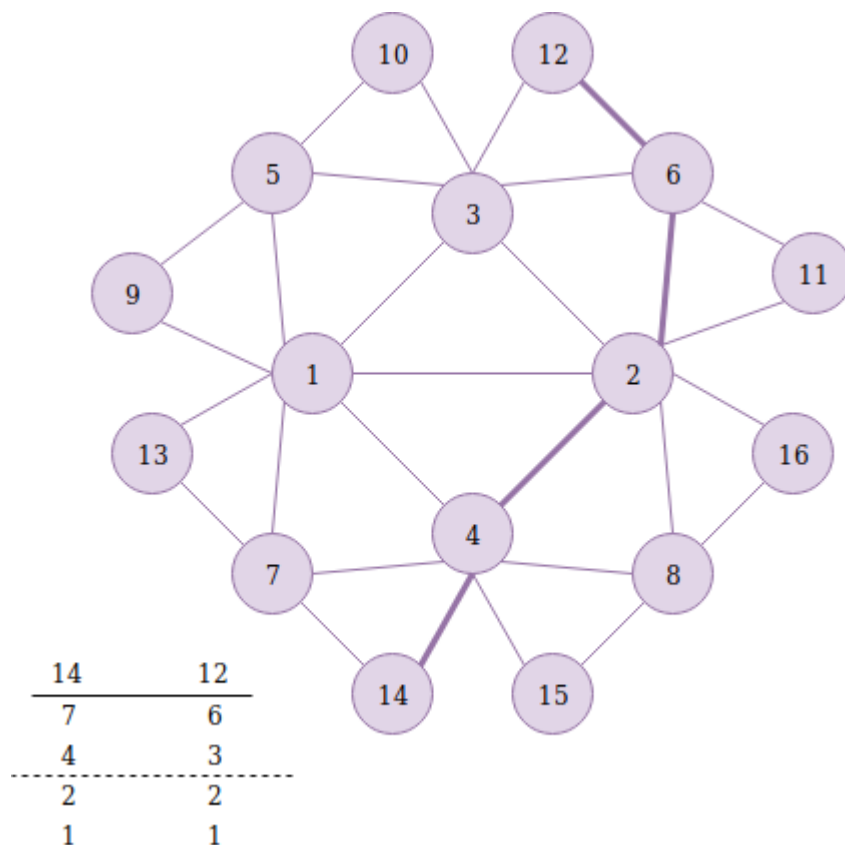


Primer.  $N = 6$ ,  $np = 6 / 2 + 6 \% 2 = 3 + 0 = 3$ , cvor 6 se povezuje sa cvorom 3 i cvorom 2.

## Kretanje po sistemu

Ako cvor **n** zeli da posalje poruku cvoru **m**, sledeci skok u mrezi se odlucuje po sledecem pravilu.

1. Id cvorova se razlozi na delioce brojem 2 zaokuzene na vecu cifru.
2. Pronalazi se njihov najveći zajednicki clan **c**
3. Dok se kretanje odvija po prvoj koloni, sledeci skok se trudi da bude sto blizi **c**
4. Kad se kretanje odvija po drugoj koloni, sledeci skok se trudi da bude sto blizi odredistu, tj cvoru **m**.

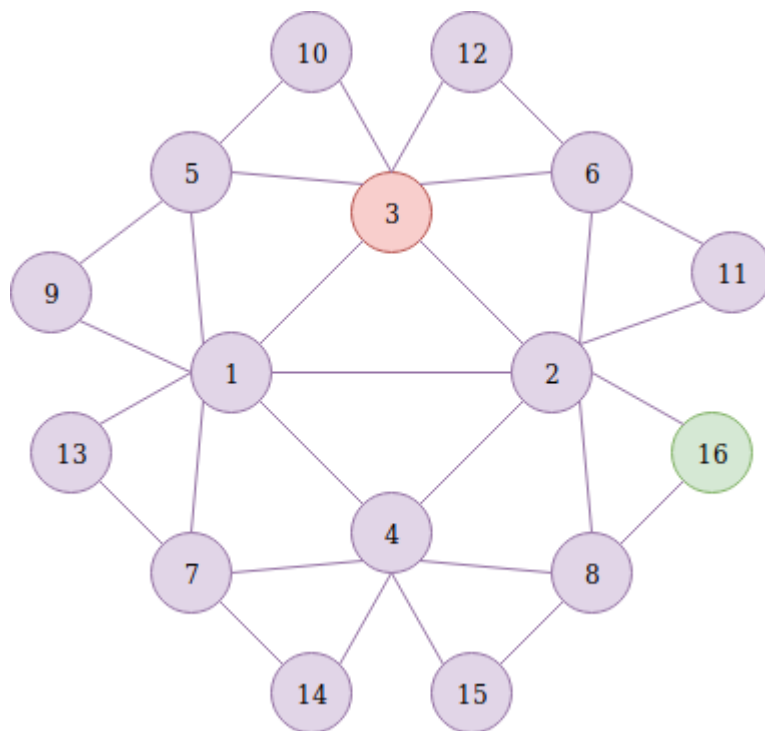


Primer. Ako se kretanje obavlja od cvora 14 do cvora 12, oni se razloze, najveći zajednicki clan je 2. 14 odlucuje da mu je najbolji skok 4, jer je blizi zajednickom clanu (preskace se 7), zatim 4

odlucuje da je najbolji sledeci skok 2, i ovde se završava kretanje po prvoj koloni, kretanje po drugoj koloni zapocinje sa cvorom 2. On sada trazi cvor sto bilizi cvoru 12 u razloženoj koloni. to je 6 (preskace 2) i zatim iz 6 direktno dolazi do 12.

## Izlazak cvora iz sistema

Kada cvor n zeli da napusti sistem, uzima kriticnu sekciju, zaustavlja posao u sistemu, obavestava najveći cvor da napusta sistem i da on treba da zauzme njegovo mesto. Zatim se azuriraju routing tabele, cvorova, obavi se broadcast da je najveći cvor napustio sistem i obavesti se bootstrap cvor.



## Posao

Posao se zapocinje tako sto se uzme kriticna sekcija, posao se podeli na minimalne delove u skladu sa limitom i raspodeli svim cvorovima u sistemu. Kada cvor završi svoj deo posla, uzima nasumicni cvor  $\text{rnd}(1, \text{maxNodeId})$ , a koji nije vec pitan, I pokusava da mu preuzme deo posla. Ukoliko cvor od koga se preuzima nema vise posla, onda se salje prazna lista. Cvor zatim pita sve ostale cvorove u sistemu, I kad završi sa pitanjima, završava posao. Za jedan posao, krade se samo jednom po cvoru. Kad je posao gotov, rezultat se broadcast-uje.

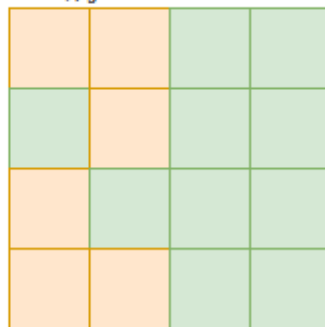
## Odredjivanje delova posla

Posao je ponalazenje svih resenja za matricu dimenzije  $d$ . Posao moze da se deli na celine. U sistemu postoji limit ispod koga ne moze da se deli posao. Ako je limit 1%, onda znaci da moze da postoji maksimalno 100 delova posla. Posto se jedino postavlja pitanje koji opseg tabele dimenzije  $d$  treba da se obidje u jednom poslu, broj delova posla se odredjuje na sledeci nacin

1. Odredi se maksimalni broj poslova
2. Broj delova poslova je inicijalno 1, zatim se broj mnozi sa dimenziom tabele dok god je broj poslova manji od maksimalnog broja poslova.

`limit = 5% -> maxJobs = 20, d = 4, jobNumer = 16`

`jobs=(0...15), job =6 -> matrixAddress = (1,2)`




3. Na osnovi id-ja cvora, odredjuje se pocetna pozicija za kraljice na osnovu koje se dalje brute-force-om obilazi ostatak matrice.

## Komunikacija

Svaka poruka koja se kreće po sistemu ima svog posiljaoca I primaoca ukoliko je direktna, ukoliko je broadcast, onda ima svoj ID.

Ako je poruka direktna, I dodje do cvora koji nije primalac, onda se on trudi sa svojim podacima I ruting tabelom, odredi najbolji sledeci skok za tu poruku.

Ako je poruka broadcast tipa, onda se njen id doda u skup broadcast poruka, I ako je primljena prvi put, onda se prosledi susedima I obradi se, a ukoliko je dosla opet, ignorise se.

Sva komunikacija izmeju cvorova je asinhrona. Cvor primi poruku I odmah posalje response, a zatim drugi thread obavlja posao oko obrade poruke I njenog daljeg eventualnog prosledjivanja.

# API

## User

### Controls

#### ***POST /control/pause***

Uzima kritičnu sekciju i broadcast-a pauzu u celom sistemu.

#### ***POST /control/result***

Ispisuje rezultat racunanja za dimenziju

```
{  
  "dimension": 7  
}
```

Response:

QUEENS: 1

```
   1 2 3 4 5 6 7  
1  _ Q _ _ _ _ _  
2  _ _ _ _ Q _ _  
3  _ _ Q _ _ _ _  
4  Q _ _ _ _ _ _  
5  _ _ _ _ _ Q _  
6  _ _ _ Q _ _ _  
7  _ _ _ _ _ Q _
```

QUEENS: 2

```
   1 2 3 4 5 6 7  
1  _ _ _ _ _ Q _  
2  _ _ _ _ Q _ _  
3  _ _ Q _ _ _ _  
4  Q _ _ _ _ _ _  
5  _ _ _ _ _ Q _  
6  _ _ _ Q _ _ _  
7  _ Q _ _ _ _ _
```

...

#### ***POST /control/result-simple***

Ispisuje rezultat racunanja za dimenziju u formatu liste niza. Zbog uštede memorije i sprečavanja prepunjivanja heap-a za velike rezultate/

```
{  
  "dimension": 7  
}
```



### ***POST /control/start***

Startuje posao za dimenziju

```
{  
  "dimension": 7  
}
```

### ***GET /control/status***

Ispisuje status za sve zapocete poslove

Response:

```
{  
  "7": "done ( 100.0% )",  
  "8": "active ( 83.5323% )",  
  "10": "paused ( 1.32% )"  
}
```

### ***POST /control/stop***

Pokrece proceduru za stopiranje cvora.

## **Node**

Naredne poruke se odvijaju izmedju cvorova. Svaka od request poruka ima sledece attribute

```
{  
  "receiver": 0,  
  "sender": 0,  
  "trace": [  
    {  
      "log": "string",  
      "nodeId": 0  
    }  
  ]  
}
```

oni ce se u svim requestovima dalje podrazumevati.

Svaki response na bilo koji request je sledeceg oblika:

```
{  
  "type": "OK",  
  "redirectNode": null,  
  "comment": null  
}
```

i on se takodje nece specificirati.

## Critical section

### ***POST /critical-section/broadcast***

Broadcast-a zahtev za kriticom sekcijom

```
{
  "id": 0,
  ...
}
```

### ***POST /critical-section/token***

Prosledjivanje tokena

```
{
  "criticalSectionToken": {
    "queue": [
      2
    ],
    "suzukiKasamiNodeMap": {
      "1": 4,
      "2": 2,
      "3": 5
    }
  },
  ...
}
```

## Jobs

### ***POST /job/queens***

Dodeljuje posao cvoru sa kraljicama i zapocinje posao ukoliko nije vec zapocet

```
{
  "dimension": 0,
  "jobs": [
    {
      "dimension": 0,
      "jobId": 0,
      "maxJobs": 0
    }
  ],
}
```

### ***POST /job/queens-pause***

Pauzira sve poslove

```
{
  "id": "string",
  ...
}
```

### ***POST/job/queens-result-broadcast***

broadcast-a rezultate za dimenziju

```
{
  "dimension": 0,
  "id": "string",
  "queensResults": [
    {
      "queensJob": {
        "dimension": 0,
        "jobId": 0,
        "maxJobs": 0
      },
      "results": [
        [
          0
        ]
      ]
    }
  ],
  ...
}
```

### ***POST /job/queens-status***

Zahteva status poslova od cvora

```
{
  "statusRequestId": "string"
  ...
}
```

### ***POST /job/queens-status-collector***

Prima rezultate poslova za status request.

```
{
  "jobStates": [
    {
      "dimension": 0,
      "done": 0,
      "pending": 0,
      "status": "string"
    }
  ],
  "statusRequestId": "string",
}
```

### ***POST /job/stealing-collector***

Prima poslove koje zahtevao za kradju

```
{
  "dimension": 0,
  "stolenJobs": [
    {
      "dimension": 0,
      "jobId": 0,
      "maxJobs": 0
    }
  ],
}
```

### ***POST /job/stealing-request***

Zahtev za kradju poslova za dimenziju.

```
{
  "dimension": 0,
}
```

## Node endpoints

### ***POST /node/alter-neighbours***

Zahtev za azuriranje ruting tabele za sender-a

```
{
  "nodeInfo": {
    "ip": "string",
    "port": 0
  },
  "sender": 0,
  ...
}
```

### ***POST /node/host-ack***

Potvrđuje da preuzima posao cvora koji napusta sistem time sto ga notifikuje na ovaj endpoint.

```
{
  "receiver": 0,
  "sender": 0,
  "trace": [
    {
      "log": "string",
      "nodeId": 0
    }
  ]
}
```

## ***POST /node/host-request***

Request koji se salje najvećem cvoru, zahtev da on preuzme posao

```
{
  "myself": {
    "ip": "string",
    "port": 0
  },
  "routingTable": {
    "1": {
      "ip": "string",
      "port": 0
    },
    "3": {
      "ip": "string",
      "port": 0
    },
    "5": {
      "ip": "string",
      "port": 0
    }
  },
  "sender": 0,
  "stoppedJob": 5,
  "suzukiKasamiCounter": 0,
  "unfinishedJobsForDimension": {

    "6": [
      {
        "dimension": 0,
        "jobId": 0,
        "maxJobs": 0
      }
    ],
    "8": [
      {
        "dimension": 0,
        "jobId": 0,
        "maxJobs": 0
      }
    ],
    "5": [
      {
        "dimension": 0,
        "jobId": 0,
        "maxJobs": 0
      }
    ]
  },
  ...
}
```

### ***POST /node/join-broadcast***

Broadcast poruka kada se cvor pridružuje sistemu

```
{
  "id": "string",
  "sender": 0,
  ...
}
```

### ***POST /node/leave-broadcast***

Broadcast poruka kad cvor napusta sistem

```
{
  "id": "string",
  "sender": 0,
  ...
}
```

### ***POST /node/newbie-join***

Direktna poruka kad se novi cvor pridružuje i njegova adresa da može da bude iskontaktiran

```
{
  "newbie": {
    "ip": "string",
    "port": 0
  },
  ...
}
```

### ***POST /node/max-leave***

Kad graf napusta cvor koji je ujedno i maksimalni, onda on obavesti nasumični cvor na ovom endpoint-u da nastavi posao nakon napustanja

```
{
  "activeJob": 0,
  ...
}
```

## ***POST /newbie-accepted***

Poruka kad se cvor ukljucuje u sistem

```
{
  "collectedResults": {
    "8": {
      "23": [
        [...]
      ],
      "42": [
        [...]
      ],
      "53": [
        [...]
      ]
    },
    "7": {
      "1": [
        [...]
      ],
      "2": [
        [...]
      ],
      "3": [
        [...]
      ]
    }
  },
  "currentlyActiveDim": 0,
  "finishedJobs": [
    7,8,9
  ],
  "firstNeighbour": {
    "id": 0,
    "ip": "string",
    "port": 0
  },
  "receiver": 0,
  "secondNeighbour": {
    "id": 0,
    "ip": "string",
    "port": 0
  },
  ...
}
```

## **Bootstrap**

### ***POST /bootstrap/check-in***

Prijavljivanje cvora

```
{
  "ip": "string",
  "port": 0
}
```



Odgovor:

```
{
  "id": 2,
  "nodeInfo": {
    "ip": "0.0.0.0",
    "port": 12345
  }
}
```

### ***POST /bootstrap/check-out***

Izlazak cvora iz sistema

```
{
  "id": 0,
  "nodeInfo": {
    "ip": "string",
    "port": 0
  }
}
```

### ***GET /bootstrap/lock***

Zaključava bootstrap da daje cvorovima id-jeve

### ***GET /bootstrap/unlock***

Otključava bootstrap da daje cvorovima id-jeve