Messaging, NATS

Servisno orijentisane arhitekture

Univerzitet u Novom Sadu
Fakultet tehničkih nauka

Problem

- Jedan od najvećih izazova prilikom prelaska sa monolitne na mikroservisnu arhitekturu aplikacije jeste izmena u načinu komunikacije između komponenti (lokalni vs udaljeni pozivi procedura)
- Sinhroni pozivi zahtevaju istovremenu dostupnost svih učesnika (problem znatno izraženiji kada imamo ulančane pozive)
- Težimo da komunikaciju između servisa smanjimo na minimum, kao i da ti pozivi, kad god je moguće, ne budu blokirajući (RPC vs messaging)

Tipovi komunikacije

Tipove komunikacije možemo podeliti na dva načina:

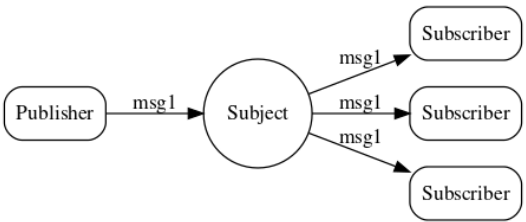
- Prema prirodi komunikacionog protokola:
  - **Sinhroni** - Klijent šalje zahtev i blokiran je sve dok ne dobije odgovor od servera
  - **Asinhroni** - Pošiljalac šalje poruku ne čekajući na odgovor klijenta (ukoliko ga uopšte očekuje)
- Prema broju primalaca poruke/zahteva:
  - **Jedan** - Zahtev dobija i obrađuje tačno jedan primalac
  - **Više** - Klijent šalje poruku preko nekog kanala, iz kog poruke preuzima i obrađuje potencijalno više primalaca

Messaging

- Asinhrona komunikacija, gde servisi razmenjuju poruke upotrebom kanala
- Možemo ostvariti različite stilove komunikacije:
  - **Request/response** - Šaljemo poruku jednom primaocu i čekamo odgovor
  - **Notifications** - Šaljemo poruku jednom primaocu ne očekivajući nikakav odgovor
  - **Request/asynchronous response** - Šaljemo poruku jednom primaocu i očekujemo da ćemo nekada u budućnosti dobiti odgovor
  - **Publish/subscribe** - Šaljemo poruku koju može preuzeti više primalaca
  - **Publish/asynchronous response** - Šaljemo poruku koju može preuzeti više primalaca, koji nam nakon toga mogu vratiti odgovor

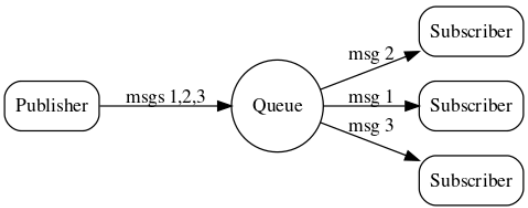
NATS

- NATS je posrednik za razmenu poruka između servisa
- Razlikujemo dva tipa učesnika u komunikaciji
  - **Publisher** - Objavljuje poruku na neku **temu** (eng. subject)
  - **Subscriber** - Prijavljuje se da prima poruke iz određenog subject-a
- Svaka poruka sadrži:
  - subject
  - paylod (niz bajtova)
  - headers
  - reply subject (opciono)



Queue grupa

- Dodatna funkcionalnost koju NATS nudi je **queue**
- Svaki subscriber preuzima poruke iz jednog subject-a
- Subscriber-u se može dodeliti queue kom pripada, a svi subscriber-i koji pripadaju istom queue-u čine queue grupu
- Svaka poruka koja se objavi u subject biće isporučena samo jednom članu queue grupe



Garancije isporuke

NATS nudi dva režima rada, u zavisnosti od toga kakve garancije isporuke poruke žeimo:

- **Core NATS** - Pruža **at most once** garanciju isporuke, ako publisher objavi poruku, a neki subscriber trenutno nije dostupan, poruka mu neće biti dostavljena
- **NATS JetStream** - Pruža **at least once** garanciju isporuke, čak i ako subscriber trenutno nije dostupan, poruka mu može btii isporučena kasnije, zato što će NATS čuvati poruke neki definisani period (u memoriji ili na disku)

Go i NATS

- Klijentska biblioteka za Go:
  `github.com/nats-io/nats.go`
- Konekcija sa NATS serverom:

```
func Conn() *nats.Conn {
        conn, err := nats.Connect("nats://localhost:4222")
        if err != nil {
                log.Fatal(err)
        }
        return conn
}
```

Publish-Subscribe stil komunikacije

- Pošiljalac poruke:

```
err := conn.Publish(subject, []byte("hello world!"))
...
```

- Primaoci poruke:

```
_, err := conn.Subscribe(subject, func(message *nats.Msg) {
      fmt.Printf("RECEIVED MESSAGE: %s\n", string(message.Data))
})
```

Notifications stil komunikacije

- Pošiljalac poruke:

```
err := conn.Publish(subject, []byte("hello world!"))
...
```

- Primaoci poruke:

```
_, err := conn.QueueSubscribe(subject, queue, func(message *nats.Msg) {
      fmt.Printf("RECEIVED MESSAGE: %s\n", string(message.Data))
})
```

Request-Response stil komunikacije

- Pošiljalac poruke:

```
// waiting on the first response
response, err := conn.Request(subject, []byte("Hello world"), 5*time.Second)
...
fmt.Printf("RESPONSE: %s\n", string(response.Data))
```

- Request metoda kreiraće novi naziv subject-a na kom će očekivati odgovor primaoca poruke (svaka NATS poruka sadrži reply subject polje)
- Primaoci poruke:

```
_, err := conn.QueueSubscribe(subject, queue, func(message *nats.Msg) {
      fmt.Printf("RECEIVED MESSAGE: %s\n", string(message.Data))
      reply := []byte(fmt.Sprintf("reply to %s", string(message.Data)))
      err := conn.Publish(message.Reply, reply)
      ...
})
```

Request-Async Response stil komunikacije

- Pošiljalac poruke:

```
_, err := conn.Subscribe(replySubject, func(message *nats.Msg) {
      fmt.Printf("RESPONSE: %s\n", string(message.Data))
})
...
err = conn.PublishRequest(subject, replySubject, []byte("hello world"))
...
fmt.Println("message sent, doing something else in the meantime ...")
```

- Primaoci poruke:

```
_, err := conn.QueueSubscribe(subject, queue, func(message *nats.Msg) {
      fmt.Printf("RECEIVED MESSAGE: %s\n", string(message.Data))
      reply := []byte(fmt.Sprintf("reply to %s", string(message.Data)))
      err := conn.Publish(message.Reply, reply)
      ...
})
```

Publish-Async Response stil komunikacije

- Pošiljalac poruke:

```
_, err := conn.Subscribe(replySubject, func(message *nats.Msg) {
      fmt.Printf("RESPONSE: %s\n", string(message.Data))
})
...
err = conn.PublishRequest(subject, replySubject, []byte("hello world"))
...
fmt.Println("message sent, doing something else in the meantime ...")
```

- Primaoci poruke:

```
_, err := conn.Subscribe(subject, func(message *nats.Msg) {
      fmt.Printf("RECEIVED MESSAGE: %s\n", string(message.Data))
      reply := []byte(fmt.Sprintf("reply to %s", string(message.Data)))
      err := conn.Publish(message.Reply, reply)
      ...
})
```