Fundamentos de PPD

Padrões de comunicação

- Anel -

Fonte: material próprio

Fernando Luís Dotti

PUCRS | ESCOLA POLITÉCNICA

Comunicação em Anel

O padrão em anel também é recorrente em sistemas concorrentes e distribuídos.

Exemplos:

- rede token ring
- protocolo de exclusão mútua
- implementações de consenso
- implementações de ordenação total de mensagens

Comunicação em Anel

Caso: Implementar o funcionamento de uma rede token ring, usando canais.

Token Ring:

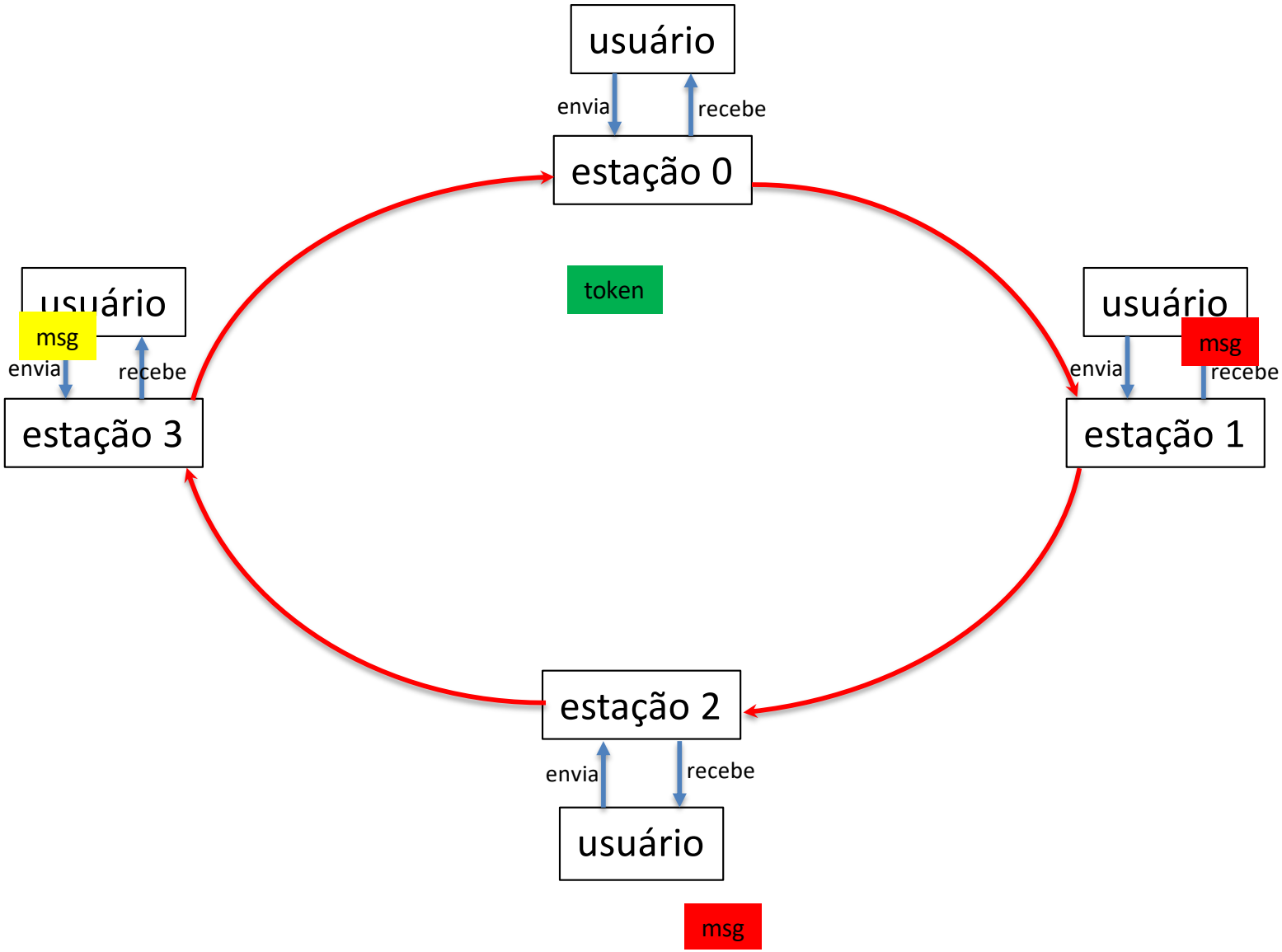
- estações dispostas em anel
- comunicação em um sentido (ex.horário)
- token dá permissão de envio de mensagem

Comunicação em Anel

Token Ring:

- Estação tem um usuário que manda e recebe mensagens
- A qualquer momento, usuário pode postar uma mensagem a enviar
- Um token fica circulando pelo anel. Quando uma estação recebe um token, ela pode passar o token adiante ou enviar uma mensagem criada pelo usuário (se existir).
- Se uma mensagem é enviada ela circula até o originador, que então retira a mensagem do anel e passa o token adiante.

Token Ring



Token Ring

Usuário

- se cria mensagem, coloca em buffer da estação
- recebe mensagem e print na tela

Estação

- se tem token
  - se tem mensagem do usuário: manda
    senão: repassa token
  - se recebe sua mensagem (então deu a volta):
    consome e repassa token
- se não tem token
  - se recebe token: vá para o caso anterior
  - se recebe mensagem
    para outro: repassa no anel
    para si: repassa ao usuário E no anel

Comunicação em Anel

Token Ring - Rede

- crie os canais entre estações,
- crie as estações com seus canais de entrada e saída
- coloque um token na rede

```
package main
const N = 4
type Msg struct {
    sender   int
    receiver int
    message  string
}
type Packet struct {
    token bool // se true é token,
    msg   Msg  // ou tem uma mensagem
}

func user(id int,
        send chan Msg,
        rec chan Msg) {
    // usuario pode mandar e receber
    // concorrentemente
    go func() {
        for i := 0; i <= N; i++ {
            send <- Msg{id, i, "msg"}
        }
    }()
    go func() {
        for {
            m := <-rec
            println("Pacote recebido",
                id, m.sender,
                m.receiver,
                m.message)
        }
    }()
}
```

```
func main() {
    var chanRing [N]chan Packet
    var chanSend [N]chan Msg
    var chanRec [N]chan Msg
    for i := 0; i < N; i++ {
        chanRing[i] = make(chan Packet)
        chanSend[i] = make(chan Msg)
        chanRec[i] = make(chan Msg)
    }
    for i := 0; i < (N - 1); i++ {
        go node(i, false, chanSend[i], chanRec[i],
                          chanRing[i], chanRing[i+1])
        go user(i, chanSend[i], chanRec[i])
    }
    go node(N-1, true, chanSend[N-1], chanRec[N-1],
                       chanRing[N-1], chanRing[0])
    go user(N-1, chanSend[N-1], chanRec[N-1])
    <-make(chan struct{})
}
```

```
func node(id int, hasToken bool,
        send chan Msg, receive chan Msg,
        ringMy chan Packet, ringNext chan Packet) {
    println("node ", id)
    for {

      COMO IMPLEMENTAR ESTE PROCESSO ?

    }
}
```