- Nous considerons un système distribuer local(LAN) ou global(WAN) qui relie avec des canaux bidirectionnel different nœuds(objets physique) comuniquant a l'aide de messages.
- Les canaux sont parfais, pas de données perdu et le temps de transmission est delimité.
- Les messages sont transmit avec deux fonctions :
    - Send(MessageType, Destination [,Parametre})
    - Receive(MessageType [,parametre])
- Dans le cas d'un système distribuer, une base de donné est un exemple de ressource critique.
- Nous allons nous baser sur l'algorithme du boulanger pour trouver une solution a ce probleme, mais dans notre cas, il n'a pas de mémoire partager, il faudra donc transmettre des messages entre les nœuds pour comparer les tickets.

Version simplifier de l'algorithme de Ricart-Agrawala

Ricart-Agrawala algorithm (outline)	
integer myNum ← 0	
set of node IDs deferred ← empty set	
	main
p1:	non-critical section
p2:	myNum ← chooseNumber
p3:	for all *other* nodes N
p4:	send(request, N, myID, myNum)
p5:	await reply's from all *other* nodes
p6:	critical section
p7:	for all nodes N in deferred
p8:	remove N from deferred
p9:	send(reply, N, myID)
	receive
	integer source, reqNum
p10:	receive(request, source, reqNum)
p11:	if reqNum < myNum
p12:	send(reply,source,myID)
p13:	else add source to deferred

- P4 : On envoie a tout les autres nœuds notre ID et notre ticket pour leurs signaler qu'on souhaite entrer en SC.
- P5 : On attend une reponse de tout les autres nœuds pour pouvoir entrer en SC.
- P8 : on vide notre stack des moins prioritaire.
- P9 : On envoie une reponse a tout les nœuds dans la stack, ce qui permet au nœud qui attendait notre reponse d'entrer en SC.
- Receive est le code executer a la reception de message.
- P11 : Si l'emeteur de message est plus prioritaire(num du ticket plus petit) on lui envoi une reponse pour qu'il puisse entrer en SC.
- P13 : Sinon, on l'ajoute dans notre stack des moins prioritaire

- Cette algorithme n'est pas complet car il ne decris pas comment le numero de ticket est choisi, l'algorithme complet est decris dans la section suivante.

Algorithme de Ricart-Agrawala

- Nous allons maintenant voir comment un process choisi sont numero de ticket.
- Chaque nœud doit savoir quel est le plus grand numero de ticket utiliser jusqu'à maintenant, pour ca nous avons la variable « highestNum».

```
                Ricart-Agrawala algorithm
                integer myNum ← 0
set of node IDs deferred ← empty set
integer highestNum ← 0
boolean requestCS ← false
          Main
          loop forever
p1:           non-critical section
p2:           requestCS ← true
p3:           myNum ← highestNum + 1
p4:           for all other nodes N
p5:               send(request, N, myID, myNum)
p6:           await reply's from all other nodes
p7:           critical section
p8:           requestCS ← false
p9:           for all nodes N in deferred
p10:              remove N from deferred
p11:              send(reply, N, myID)
```

```
          Ricart-Agrawala algorithm (continued)

          Receive
          integer source, requestedNum
          loop forever
p1:           receive(request, source, requestedNum)
p2:           highestNum ← max(highestNum, requestedNum)
p3:           if not requestCS or requestedNum ≪ myNum
p4:               send(reply, source, myID)
p5:           else add source to deferred
```

- P2 : le boolean requestCS, initialiser a false, eust mit a vrai pour indiquer qu'on souhaite entrer en SC.
- P3:Notre numero de tiquet est le plus grand numero de tiquet utiliser jusqu'à maintenant +1.
- P8 : Une fois sorti de la SC, on remet le boolean requestCS a false
- Receive P2 : on modifie la valeur de highestNum si on recoit un ticket plus grand.
- P3 : renvoi une reponse que si on a pas besoin d'entrer en SC(requestSC) ou que l'emeteur est plus prioritaire.