RTOS_concurrence_2.3.5

- L'utilisation d'un système distribuer permet d'ameliorer la fiabilité d'un système en dupliquant les calculs effectuer dans different processeurs independant.
- Il y'a deux propriétés a faire respecter :
 - Fail-safety : Un ou plusieurs erreurs ne peuvent causer des dommages au système ou ces utilisateurs
 - o Fail-tolerence: Le système continue a remplir ces obligateurs meme en cas d'erreurs.

Consensus

- Consensus : accord général parmi les membres d'un groupe(noeuds), pouvant permettre de prendre une décision sans vote préalable.
- Chaque nœud du système prend une valeur initial, il faut que tout les nœuds ce mettent d'accord(consensus) sur une de ces valeurs.
- Si aucune erreur ne survient, on peut concevoir une solution simple au probleme :
 - o Chaque nœud envoi ca solution a tous les autres.
 - o Un algorithme commun fait un choix entre toute les solutions, basé sur la majorité.
 - o Comme tout les nœuds utilise le meme algo et les meme donées, le vote est commun.
- Voici deux types d'echecs possible qui peuvent survenir en pratique :
 - o Crash failure : Un nœud arrete d'envoyer des messages
 - Facilement detectable via un time-out
 - o Byzantine failure : Un nœud envoie des messages erronés ou arbitraires.

Probleme des generaux byzantins

- Considerons plusieurs generaux byzentins attaquant la meme ville.
- Les generaux utilisent des messages pour communiquer.
- Ils doivent etre d'accord pour un plan d'action commun :
 - o « A » pour attaquer
 - o « R » pour battre en retraite
- Mais certains generaux peuvent etre des traitres, ceux-ci pourraient essayer de corrompre le resultat du vote des commandants loyaux.
- Il faut donc que les generaux trouvent un algorithme pour garantir que :
 - o Tous les generaux loyaux ce mettent d'accord sur un meme plan.
 - o Un petit nombre de traitres ne peuvent induire les generaux loyaux en erreur.
- Dans le cas d'un système distribuer, les generaux sont des nœuds et les canaux de communication sont leurs messages.

Consensus – Algorithme simple a un round

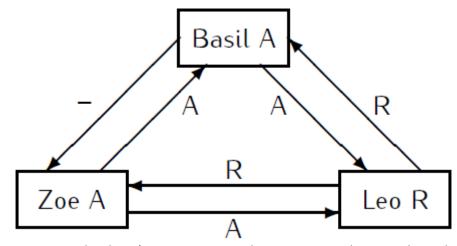
Consensus - one-round algorithm			
planType finalPlan			
planType array[generals] plan			
p1: plan[myID] ← chooseAttackOrRetreat			
p2: for all <i>other</i> generals G			
p3: send(G, myID, plan[myID])			
p4: for all <i>other</i> generals G			
p5: receive(G, plan[G])			
p6: finalPlan ← majority(plan)			

- P6 : En cas d'égalité, la fonction « majority » renvoi « R »
- Considerons que nous avons 3 generaux :

o Deux Ioyaux : Leo et Zoe

o Un traitre : Basil

• Voici ce qui peut arriver si Basil crash aprs avoir envoyer une reponse a Leo



• Ces tables represente les données recuperer par chaque generaux loyaux et leurs decisions :

Leo			
general	plan		
Basil	А		
Leo	R		
Zoe	А		
majority	А		

Zoe			
general	plan		
Basil	_		
Leo	R		
Zoe	А		
majority	R		

• Comme on peut le voir, ils prennent une decisoin different, l'algorithem ne fonctionne donc pas.

Consensus – Algorithme final a deux rounds

- Nous allons maintenant créer une nouvelle solution avec deux rounds.
- Durant le premier tour, chaque general transmet aux autres sont plan.
- Et dans le deuxieme, les generaux transmettent les plans qu'ils ont recu.
- Les generaux loyaux vont transmettre les bonnes informations et si ils sont assez, ils pourront prendre une decision commune.

Consensus – Byzantine Generals algorithm				
planType finalPlan				
planType array[generals] plan, majorityPlan				
planType array[generals, generals] reportedPlan				
p1: $plan[myID] \leftarrow chooseAttackOrRetreat$				
p2: for all <i>other</i> generals G	// First round			
p3: send(G, myID, plan[myID])				
p4: for all <i>other</i> generals G				
p5: receive(G, plan[G])				
p6: for all <i>other</i> generals G	// Second round			
p7: for all <i>other</i> generals G' except G				
p8: send(G', myID, G, plan[G])				
p9: for all <i>other</i> generals G				
p10: for all <i>other</i> generals G' except G				
p11: receive(G, G', reportedPlan[G, G'])				
p12: for all <i>other</i> generals G	// First vote			
p13: majorityPlan[G] \leftarrow majority(plan[G] \cup 1	reportedPlan[*, G])			
p14: majorityPlan[myID] ← plan[myID]	// Second vote			
p15: finalPlan ← majority(majorityPlan)	· 45 43 43 5			

• Voici les tableaux representant les plans recu et la decisions prise par les generaux loyaux.

-	•			•		
Leo						
general	plans	reported by		majority		
		Basil	Zoe			
Basil	А		Α	А		
Leo	R			R		
Zoe	А	А		А		
majority				А		

Zoe						
general	plans	reported by		majority		
		Basil	Leo			
Basil	А		Α	А		
Leo	R	_		R		
Zoe	А			А		
majority				А		

• Les deux generaux prennent la meme decision.