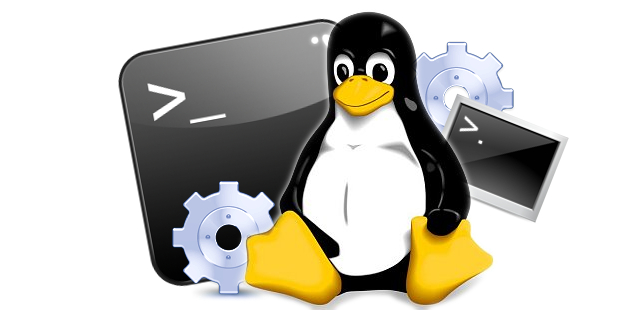
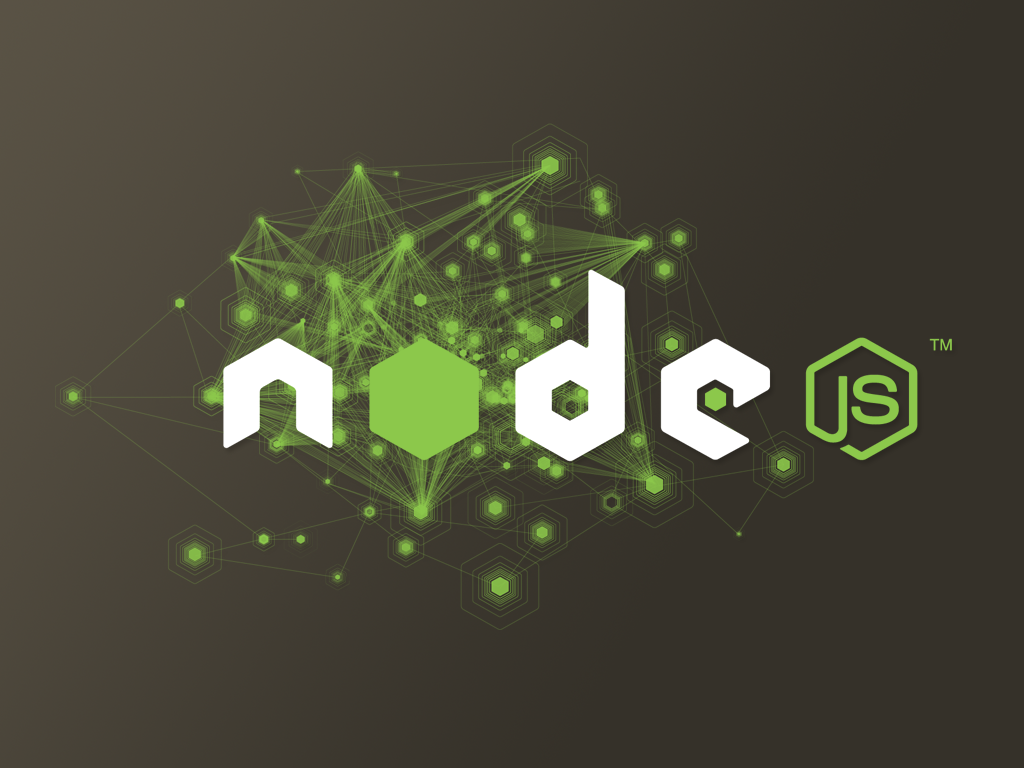
Projet Linux

Node





-Anthony Ait lhadj

-Ilyan Alouani

-Allan Contaret

-Ivan Peric

Sommaire

1. Présentation
2. Choix & Répartition
3. Fonctionnalité
4. Conclusion
5. Présentation

Le but de ce projet est de réaliser une application pour gérer et accéder à des objets sur le réseau Internet. Pour cela une application web doit également être créée pour récupérer les informations de l’objet et elle doit afficher les statistiques.

1. Choix & Répartition

Pour cela nous allons utiliser les outils suivant :

Ce projet sera codé en C++ grâce a l’IDE Qt et pour la base de donnée mongodb. L’interface graphique est faite grâce a Google Charts.

En ce qui concerne la répartition des taches :

- Anthony c’est occupé de l’occupation de la mémoire et de la rédaction de la documentation et de la partie nodejs.

- Ilyan lui c’est chargé des flux entrant et sortant sur les connecteurs virtuels comme par exemple Ethernet, Il c’est aussi occupé de réaliser la IHM.

- Allan a réalisé la partie sur Qt le but était de réaliser une IHM pour permettre d’accéder à l’objet créer par les autres membres du groupe. Cette interface doit permettre une modification des paramètres ainsi que d’obtenir un compte rendu graphique.

Ivan c’est occupé du taux d’occupation du CPU et de définir une méthode et un protocole réseau pour accéder à l’interface. Et également du nodejs.

1. Fonctionnalité

Pour gérer les accès concurrentiels sur un réseau il existe deux méthodes, la première est l’UDP. Pour un serveur créé et utilisant l’UDP, il envoie une requête via la passerelle qui est destiné au client. Lorsque le client veut interroger le serveur, la passerelle ne sait pas où renvoyer cette nouvelle requête qui est destinée au serveur UDP.

Cette méthode n’est donc pas appropriée car la requête ne parvient pas au destinataire.

La méthode que nous utiliserons donc est la deuxième qui est l’utilisation du TCP/IP. Cette deuxième méthode permet entre autre de mettre en relation un client et un serveur qui fait fonctionner des capteurs. Le client peut interroger le serveur les différents capteurs qui sont dans différents processus. Grâce au Multicast qui est utilisable sur le serveur, le client a la possibilité d’interroger les capteurs un à un, ce qui lui donne le droit de lecture.

Trois options s’offrent donc à nous : les sémaphores IPC, les sémaphores POSIX qui donnent un accès multicast, le serveur queue IPC qui n’utilise pas de sémaphore.

Pour accéder à l’interface offerte par système embarqué, nous avons décidé d’utiliser l’exclusion mutuelle en y intégrant des mutex. Nous avons décidé d’utiliser l’exclusion mutuelle avec mutex car une seule personne ne pourra utiliser le processus pour obtenir les informations sur le CPU, la mémoire et le flux TCP et l’exclusion mutuelle permet ça car comme c’est indiqué dans le nom, elle permet d’empêcher des entités qui sont en compétition d’entrer simultanément dans leur section critique.

Qu’est-ce qu’une section critique ?

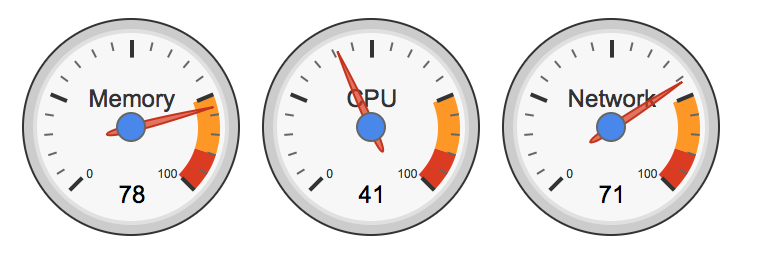
Une section (de code) critique est du code devant pouvoir faire l’hypothèse qu’il utilise la ressource de manière exclusive.

Il faut manipuler les sections critiques avec précaution car si on n’en prend pas, rien n’empêchera plusieurs entités d’utiliser le même processus. On doit donc faire attention principalement à deux points pour que les entités puissent utiliser le processus l’un après l’autre sans encombre :

* Si plusieurs processus attendent pour entrer en SC, et si aucun processus n’est déjà en SC, alors un des processus qui attend doit pouvoir entrer en SC au bout d’un temps fini.

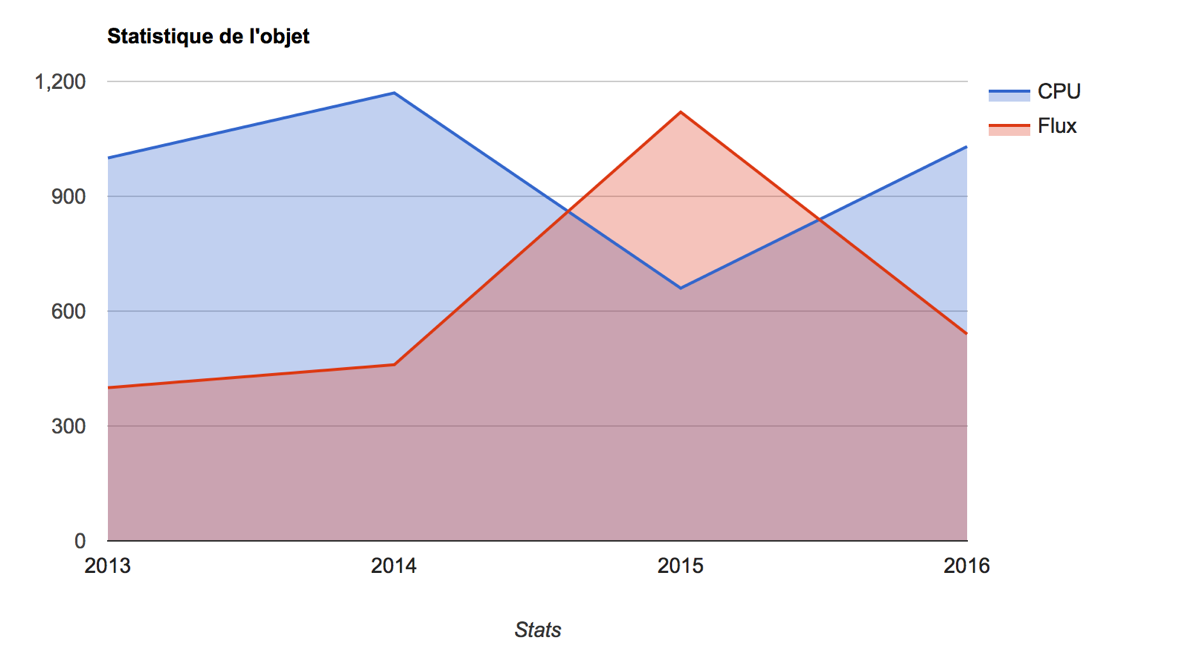
Un processus qui se trouve hors de sa SC et hors du protocole contrôlant l’accès à la SC ne doit pas empêcher un autre processus d’entrer dans sa SC : un processus ne doit pas ralentir un autre.

Pour la partie utilisant QT, nous devions réaliser une IHM qui permettra à un utilisateur d’accéder à l’objet créé précédemment, via le réseau en utilisant le protocole défini un peu plus haut. Pour cela nous devions pouvoir modifier les paramètres et le comportement de l’objet et pouvoir obtenir un compte rendu graphique. Nous avions différents codes : pour avoir le taux d’occupation du CPU de la machine virtuelle, le taux d’occupation de la mémoire en faisant la différence entre la mémoire totale, la mémoire free, les buffers et la mémoire cache. Et pour finir les flux Ethernet entrants et sortants. Nous avons réuni tous ces codes pour les affichés ensembles en mode graphique. Pour cela nous avons utilisés QtCreator.



Pour la partie sur le Node l’affichage graphique est réalisé grâce à Google Charts cela nous permet de visualiser les données.

La dernière étape pour la partie sur node c’est d’utiliser un graphe qui nous permettrait de voir en temps réel les données de la machine virtuelle avec un programme qui le rafraichirait toute les n secondes.



1. Conclusion

Le projet nous a tout d’abord appris a continuer notre travail en équipe, il nous a également permis de nous perfectionner en C++ et en Node. De plus il nous a permis d’apprendre à mieux utiliser Github.