Cahier Des Charges
-
Simulation de réseaux de files d'attente

ADOUANE Cylia
ALI AHMEDI Mycipssa
BONNET Ludivine
DOUBI Dylan
HAMMAD Amir
HELLAL Ouiza
LAMOUR Lauriane
MOLINER Emma

Mercredi 11 Mars 2020

Table des matières

1 Motivations du projet

1.1 Définition du sujet

1.1.1 Concepts

Un réseau de files d'attente est un ensemble de files d'attente inter-connectées. Elles peuvent être vues de manière générale comme un regroupement de clients pouvant être des individus ou des requêtes informatiques, attendant de manière organisée leur tour pour un service fourni par un serveur. Elles résultent d'une demande supérieure à la capacité d'écoulement d'une offre.

Elles ont fait l'objet d'étude d'une branche des probabilités, la théorie des files d'attente. Cette théorie émergea en 1917 suite aux travaux de l'ingénieur danois Agner Krapup Erlang sur la gestion des réseaux téléphoniques de Copenhague et se développa par la suite notamment grâce à David George Kendall à qui l'on doit la notation du même nom permettant la description des files d'attente.

Cette théorie s'applique à de nombreux champs d'applications tels que les télécommunications, les guichets dans les bureaux de poste ou la gestion de stocks de production pour n'en citer que trois.

1.1.2 Objectifs

L'application consiste à l'aide d'un échéancier à effectuer la simulation à évènements discrets d'un réseau de files d'attente existant ou crée par l'utilisateur. Cette simulation découle sur un calcul de performances permettant d'identifier les failles du réseau.

1.1.3 Utilisateurs

L'application est utilisable par toute personne désirant simuler un réseau de files d'attente. Elle est destinée aux professionnels souhaitant améliorer leur réseau et aux particuliers souhaitant en apprendre davantage sur les réseaux de files d'attente et leur fonctionnement.

1.2 Parties prenantes

Les parties prenantes sont des acteurs externes au projet mais elles y contribuent tout aussi bien. Il s'agit principalement du public à qui est destiné le livrable de l'application et sources de financement ayant permis sa réalisation. Dans notre cas, notre partie prenante est Mme Kloul qui est notre commanditaire et nous accompagne tout au long du développement du projet.

2 Contraintes du projet

2.1 Contraintes obligatoires

L'objectif de l'application est de livrer un outil réalisant la simulation d'un réseau de files d'attente. Cet outil devra être portable et universel. Il permettra une analyse de performances dont les résultats seront interprétés par un spécialiste en guise d'améliorer le réseau. Le rendu du projet doit se faire avant le 29 mai, date limite imposée par le commanditaire. Sa réalisation en temps est estimée à 280 h et requiert une équipe de 8 personnes.

2.2 Conventions de noms et de définitions

- **File d'attente :** Ensemble de clients et de serveurs.
- **Serveur :** Chaque file en possèdent un ou plusieurs. Les clients y effectuent leur service.
- **Classe :** Certains clients possèdent une classe. Cela définit leur routage et leur priorités.
- **Routage :** Le routage est la phase où les clients sont redirigés vers les autres files ou vers la sortie.
- **Lois de poisson, uniforme, constante :** Lois de probabilités définissant le taux d'arrivée.
- **Lois exponentielle, constante :** Lois de probabilités pour le temps de service.
- **Ordonnancement :** Permet de placer les clients selon un certain ordre dans la file
- **Échéancier :** Il s'agit d'un calendrier contenant tous les événements à traiter.
- **Temps de service moyen(pour une file) :** Moyenne du temps de service des clients entrés dans le serveur.
- **Temps de réponse moyen :** Somme du temps d'attente moyen et du temps de service moyen.
- **Taux d'utilisation du serveur :** Rapport entre le temps d'utilisation du serveur et la durée totale de la simulation.
- **Temps d'inter-arrivée moyen :** Durée moyenne entre l'arrivée de deux clients dans la file.
- **Temps de séjour moyen :** Moyenne du temps passé par chaque client(sorti du réseau uniquement) dans le réseau.
- **Temps d'attente moyen(pour chaque file) :** Moyenne du temps d'attente de tous les clients passés par la file.

- **Nombre moyen de clients dans le réseau :** Moyenne du nombre de clients présent dans le réseau à chaque instant.
- **Nombre moyen de clients dans une file :** Moyenne du nombre de clients présent dans une file à chaque instant.
- **Taux de perte(pour une file) :** Rapport entre le nombre de clients rejetés par la file et le nombre de clients présentés à la file.
- **Débit d'entrée du réseau :** Moyenne du nombre de clients entrés dans le réseau à chaque instant.
- **Débit de sortie du réseau :** Moyenne du nombre de clients sortis dans le réseau à chaque instant.

3 Exigences fonctionnelles

3.1 Portée du produit

L'application permet de simuler un réseau de files d'attente d'un certain type (ouvert/fermé, mono-classe/multi-classe) dont les files ont une taille finie ou infinie et un nombre de serveurs fini. Le réseau peut être créé par l'utilisateur ou bien choisi parmi ceux proposés et peut être sauvegardé dans un fichier. L'application propose également à l'utilisateur de connaître les performances du réseau simulé et de les sauvegarder dans un fichier. Le choix des lois d'arrivée, de service et d'ordonnancement pour chaque file est laissé libre à l'utilisateur. Les lois d'arrivée proposées sont la loi de Poisson et la loi constante. Les lois de services sont les lois exponentielle, uniforme et constante. On propose aussi les ordonnancements FIFO, LIFO, priorités et aléatoire.

3.2 Exigences fonctionnelles et sur les données

L'organigramme se trouve en annexe.

3.2.1 Module Calculs de Performances

Ce module réalise les calculs des performances suivantes :

- Temps de réponse moyen dans le réseau
- Nombre moyen de clients dans le réseau
- Temps de séjour moyen dans le réseau
- Débit d'entrée dans le réseau
- Débit de sortie du réseau
- Nombre moyen de clients pour chaque file
- Temps d'attente moyen pour chaque file
- Temps de service moyen pour chaque file
- Temps d'inter-arrivée moyen pour chaque file
- Taux de perte pour chaque file
- Taux d'utilisation du serveur pour chaque file

Les résultats permettent à l'utilisateur de détecter les problèmes liés à son réseau et ainsi de les régler afin de l'optimiser.

Selon le type du réseau, toutes les performances ne seront pas calculées. Pour un réseau fermé, le temps de séjour moyen dans le réseau, le débit d'entrée dans le réseau et le débit de sortie du réseau ne seront pas calculés car il n'existe pas d'entrée/sortie dans ce type de réseau.

3.2.2 Module Interface Graphique

L'interface graphique de notre application permet à l'utilisateur de créer son propre réseau et de lui donner les caractéristiques qu'il souhaite. Il en est de même pour les entités qui le composent (classes, serveurs, file d'attentes). L'interface permet principalement de suivre l'évolution du réseau crée et de le modifier à sa guise. L'utilisateur a aussi la possibilité de charger un réseau existant, de sauvegarder son réseau et les résultats de performances obtenus.

3.2.3 Module Gestion du réseau

Il s'agit du chef d'orchestre de l'application. Ce module est chargé de créer le réseau, de le stocker logiquement et de modifier son état en cours de simulation. Cette simulation se déroule grâce à un échéancier. C'est lui qui donne le rythme en traitant les événements suivant un ordre chronologique. Le prochain événement est traité, les événements résultants sont ajoutés à l'échéancier et le processus continue jusqu'à ce que la durée de la simulation soit atteinte. Cette étape est précédée par une phase d'initialisation qui consiste à créer les premiers événements qui engendreront les suivants. Il existe 3 types d'événements :

- Arrivée d'un client dans une file
- Arrivée d'un client dans un serveur
- Sortie d'un client du serveur

Chaque intervalle de temps t, les clients ayant terminé leur service sont routés dans les files suivant la politique d'arrivée des clients choisie par l'utilisateur (loi constante ou de Poisson) puis la simulation se poursuit.

Algorithm 1 Algorithme de routage (pendant la simulation)

Si $A = 0$ (nombre de files d'attente à alimenter) et $Nt > 0$ (nombre de clients à router) **alors**
  Tant que Nt(nombre de clients à router) > 0 **faire**
    Pour chaque file d'attente F **faire**
      $Np++$ (nombre de clients présentés à la file F)
      Si la taille de la file F $<$ à la taille maximale de la file F **alors**
        Ajouter dans E l'évènement "Arrivée d'un client C dans la file F" daté à la date actuelle de l'horloge H et actualiser ses attributs
        $Nf++$ (nombre de clients pour la file d'attente)
        Supprimer ce client des clients à router
        $Ne++$ (nombre de clients entrés dans la file)
        $NET++$ (nombre de clients entrés dans la file à l'instant t)
        $Nt--$ (nombre de clients à router)
      FinSi
    FinPour
  FinTantQue
FinSi
Pour chaque client multi-classe des clients à router **faire**
  Router le client C selon son routage
  Si la file qu'il doit rejoindre n'est pas pleine **alors**
    Ajouter dans E l'évènement "Arrivée du client C dans la file F" daté à la date actuelle de l'horloge H et actualiser ses attributs
    $NET++$ (nombre de clients entrés dans la file F à l'instant t)
  Sinon
    l'ordre de visite est corrompu et le client est perdu, définir sa date de départ à l'horloge et calculer son temps de séjour
  FinSi
FinPour

Explications de l'algorithme de routage

Cet algorithme sert à distribuer les clients ayant terminé leur service.

Pour les files d'attente dont le taux d'arrivée est imposé, on envoie autant de client que ce taux. Sinon, on génère un taux d'arrivée λ en fonction du nombre de clients à router et du nombre de files à alimenter qui servira à calculer le nombre de client à envoyer dans la file. Ce nombre varie d'une file à l'autre. Les clients multi-classe seront envoyés dans la file correspondant à leur routage.

Si une file n'est pas remplie et qu'un client y est envoyé, on ajoute un nouvel événement d'entrée dans une file à la date actuelle dans l'échéancier. On actualise à chaque fois les données de la file et du réseau.

Complexité : O(n), n : nombre de clients a router.

Algorithm 2 Algorithme de routage (pendant la simulation) suite

Pour chaque client mono-classe de l'ensemble des clients à distribuer **faire**
  Tant que $A > 0$ (nombre de files d'attentes à alimenter) et que $Nt > 0$ (nombre de clients à router) **faire**
    Pour chaque file d'attente F dont le taux d'arrivée NC est imposé **faire**
      $tmp = NC$
      $Np++$ (nombre de clients s'étant présentés à la file F)
      Si taille de la file F $<$ taille max de la file F **alors**
        Tant que $tmp > 0$ (nombre de clients entrant de la file actuelle) **faire**
          Ajouter dans échéancier l'évènement " Arrivée d'un client C dans l'ensemble de clients à router " daté à la date actuelle de l'horloge H et actualiser ses attributs
          $Nf++$ (nombre de clients dans la file F)
          Supprimer ce client de L
          $Ne++$ (nombre de clients entrés dans la file F)
          $NET++$ (nombre de clients entré dans la file F à l'instant)
          $tmp--$ (nombre de clients entrant de la file F)
          $Nt--$ (nombre de clients à router)
        FinTantQue
        $A--$ (nombre de files à alimenter)
      FinSi
    FinPour
  FinTantQue
  $tmp = 0$
  Pour toutes les autres files F dont le taux d'arrivée n'est pas imposé **faire**
    Np (nombre de clients présentés à la file F) ++
    Si taille de la file F $<$ taille max de la file F **alors**
      $\lambda = \frac{Nt}{A}$
      $tmp =$ nombre de client à insérer dans la file
      Tant que $tmp > 0$ **faire**
        Ajouter dans E l'évènement "Arrivée d'un client C dans la file F" à la date actuelle de l'horloge H et actualiser ses attributs
        $Nf++$ (nombre de clients dans la file F)
        Supprimer ce client des clients à router
        $Ne++$ (nombre de clients entrés dans la file F)
        $NET++$ (nombre de clients entrés dans la file F à l'instant)
        $tmp--$ (nombre de clients à insérer dans la file F)
        $Nt--$ (nombre de clients à router)
      FinTantQue
    FinSi
    $A--$ (nombre de files à alimenter)
  FinPour
  $tmp = 0$
FinPour
Pour chaque file F **faire**
  Sauvegarder sa valeur NET (nombre de clients entrés à l'instant t) à la date actuelle et créer un nouveau champ pour la prochaine date
FinPour

Algorithm 3 Boucle (en cours de simulation)

Tant que l'horloge $H <$ la durée de simulation D
Chaque intervalle t défini pour le routage :
Pour chaque file F **faire**
    Initialiser NET à 0 (nombre de clients entrés à l'instant actuel)
    Gérer l'entrée et la sortie de nouveaux clients (seulement pour un réseau ouvert)
    Initialiser IN à 0 (nombre de clients à faire entrer dans le système)
    Initialiser OUT à 0 (nombre de clients à faire sortir du système)
FinPour

Pour chaque file d'attente F dont le nombre de clients entrants NC est défini (loi constante) **faire**
    Ajouter NC à IN
FinPour
Ajouter un nombre tiré aléatoirement à IN
$NET = IN$
Sauvegarder le nombre de clients entrés dans le réseau à l'instant actuel avec la valeur NET

Tirer une valeur aléatoire et affecter à OUT
Sauvegarder le nombre de clients sortis du réseau à l'instant actuel avec la valeur OUT

Tant que $IN > 0$ **faire**
    Créer un client C et remplir ses attributs
    Ajouter C aux clients à router
    Ajouter C aux clients entrés dans le réseau
    $IN - -$
FinTantQue

Tant que $OUT > 0$ **faire**
    Prendre un client C et actualiser sa date de départ et son temps de séjour
    Supprimer C des clients à router
    $OUT - -$
FinTantQue
Lancer l'algorithme de routage sur les clients à router
Sinon si l'intervalle t n'est pas atteint, ne rien faire

Procédures

- ***Procédure 1 :*** Arrivée d'un client dans une file
  Date de traitement : Date de traitement à l'horloge actuelle.
  — Retirer le client de la liste des clients à router
  — Insérer le client dans sa file
  — Appliquer l'algorithme d'ordonnancement de la file
  — Mettre les attributs du client à jour (notamment le calcul du temps de service et d'attente) et les attributs de la file à jour
- ***Procédure 2 :*** Entrée d'un client dans un serveur
  Date de traitement : Date de traitement de l'arrivée d'un client dans une file + temps d'attente du client
  — Retirer le client de la file
  — Entrer le client dans le serveur
  — Mettre à jour les attributs du client, les attributs des clients dans la file, les attributs de la file et les attributs du serveur
- ***Procédure 3 :*** Sortie du client d'un serveur
  Date de traitement : Date de traitement de l'entrée du client dans le serveur + temps de service du client
  — Retirer le client du serveur
  — Mettre à jour les attributs du serveur
  — Ajouter le client à la liste des clients à router
  — Mettre à jour les attributs du réseau

Algorithm 4 Boucle (suite)

Tant que l'échéancier E n'est pas vide **faire**
    Retirer le premier événement de E
    H (Horloge) = date de traitement de l'événement retiré

    Si la nature de l'événement = arrivée dans la file : **alors**
        Faire **la procédure 1** (Arrivée d'un client dans une file)
        Si la politique d'ordonnancement de la file est du FIFO, ajouter un événement de nature "Entrée dans le serveur" dans E (échéancier) et affecter sa date de traitement DT à la date $H + TA$ son temps d'attente
    FinSi
    Sinon ajouter un événement de nature "Entrée dans le serveur" dans E (Échéancier) et affecter sa date de traitement DT à la date $H + T\infty$(date très éloignée à actualiser lorsque le client arrive en tête de file)
    Si la nature de l'événement = entrée dans un serveur **alors**
        Faire **la procédure 2** (Entrée d'un client dans un serveur)
        Ajouter un événement de nature "Sortie du serveur" dans E et affecter sa date de traitement DT à la date $H + TS$ (du client)
    FinSi
    Si la nature de l'événement = sortie du serveur **alors**
        Faire **la procédure 3** (Sortie du client d'un serveur)
    FinSi
    Trier l'échéancier E et passer au prochain évènement
FinTantQue

Actualiser le reste des données pour les performances
$NR = 0$ (Nombre de clients présents dans le réseau)
$NF = 0$ (Nombre de clients présents dans la file)
Pour chaque file d'attente F **faire**
    Affecter à N, le nombre de clients présents dans F
    Sauvegarder NF à la date actuelle de l'horloge
    $NR+ = NF$
FinPour
Pour chaque serveur S **faire**
    Si S est occupé **alors**
        $NR++$
    FinSi
FinPour

Sauvegarder le nombre de clients actuellement présents dans le réseau avec la valeur NR
Trier E par date de traitement croissante

Explications de la boucle de simulation

Pendant la simulation, à chaque intervalle de temps t (prédéfini), on route les clients sortant des serveurs vers les différentes files ou vers la sortie pour un réseau ouvert. Dans ce type de réseau, l'on doit d'abord insérer de nouveaux clients en fonction du taux d'arrivée des files.

Ensuite on traite les différents événements de l'échéancier en appliquant la procédure correspondante et enfin on actualise les données du réseau. On réitère l'algorithme jusqu'à ce que la durée de simulation soit atteinte.

3.2.4 Module Gestion de fichiers

Ce module permet la sauvegarde d'un réseau ou des mesures de performances dans des formats spécifiques de fichiers. Il permet également de charger depuis l'application un réseau à partir d'un fichier existant.

3.2.5 Liste des informations circulant entre les modules

1. Demande de données du réseau
2. - Demande du nom du fichier pour la sauvegarde du réseau
   - Demande du nom du fichier à charger
   - Demande du nom du fichier pour la sauvegarde des performances
   - Données sur le réseau chargé
     - *Caractéristiques du réseau :* type mono-classe/multi-classe, nombre de files et nombre de clients.
     - *Caractéristiques des files :* distribution d'arrivée, taille, loi d'ordonnancement, loi de service, nombre de serveurs par file et service proposée.
     - *Caractéristiques des classes :* routage et priorité des clients de la classe en fonction des clients des autres classes.
3. - Signal de lancement de la simulation
   - Signal de mise en pause de la simulation
   - Signal d'arrêt de la simulation
   - Signal de reprise de la simulation
   - Durée de la simulation
   - Données sur le réseau (voir (2))
4. Demande de calculs de performances
5. - Nom du fichier de sauvegarde pour le réseau
   - Nom du fichier de sauvegarde pour les performances
   - Nom du fichier à charger
   - Signal de sauvegarde du réseau
   - Signal de sauvegarde des performances
   - Signal de chargement
6. - Demande des données pour les calculs
   - Données pour la mise à jour des données des files : temps de service moyen pour chaque file et temps d'attente moyen pour chaque file
7. Résultats de performances
   - Temps de réponse moyen dans le réseau
   - Nombre moyen de clients dans le réseau
   - Nombre moyen de clients dans la file
   - Temps de séjour moyen dans le réseau
   - Temps d'attente moyen par file
7. - Temps de service moyen par file
   - Temps d'inter-arrivée moyen par file
   - Taux de perte par file
   - Débit d'entrée/sortie du réseau
   - Taux d'utilisation par serveur
8. Résultats de performances (voir (7))
9. Données sur le réseau à sauvegarder (voir (2))
10. Données pour les calculs de performances
    - Nombre de clients dans le réseau à chaque instant
    - Nombre de clients entrés dans le réseau à chaque instant
    - Nombre de clients sortis du réseau à chaque instant
    - Temps de séjour des clients sortis du réseau
    - Durée de la simulation
    - *Par file :* nombre de clients à chaque instant, nombre de clients servis, temps d'attente moyen actuel, temps d'attente du client quittant la file, temps de service moyen actuel, temps de service du client entrant dans le serveur, nombre de clients entrés à chaque instant, nombre de clients présentés, nombre de clients entrés
    - Temps d'utilisation par serveur
11. Changements dans le réseau
    - *Pour la file :* nombre de clients, nombre de serveurs, nombre de clients perdus, nombre de clients présentés, nombre de clients ayant effectué leur service, ordonnancement, loi de service
    - *Pour le serveur :* occupation, client l'occupant, file dans lequel il se trouve
    - *Pour le réseau :* nombre de clients actuel dans le réseau, nombre de clients passés dans le réseau, nombre de classes et nombre de clients par classe, type du réseau
    - *Pour le client :* temps de service, temps d'attente, classe, file dans laquelle il se trouve
    - *Pour chaque classe :* le routage, la priorité.
    - *Pour l'échéancier :* événements se trouvant dans l'échéancier (la durée, le nom de l'événement), l'état de l'horloge.

4 Exigences non fonctionnelles

4.1 Exigences d'apparence

Nous avons décidé de représenter l'interface graphique par des widgets. Chacune de ces fonctionnalités envoie un signal à la gestion du réseau qui l'interprète. L'interface graphique est organisée de façon à séparer les widgets utilisables durant la simulation (en haut) et les widgets non utilisables durant la simulation (à gauche).

La maquette se trouve en annexe.

1. **File :** Ce widget affiche une liste d'option telle que la sauvegarde "Save" ,qui permet de sauvegarder le réseau courant dans un fichier et "Open" permettant d'ouvrir un fichier existant. L'utilisateur à la possibilité d'ouvrir un fichier contenant un réseau déjà existant, comme celui de Jackson. L'accès à ce fichier n'est pas possible, il ne peut qu'être ouvert grâce à l'application.
2. **Start :** On demande à l'utilisateur la durée de la simulation. Ainsi la simulation débute.
3. **Pause :** Ce widget permet de mettre de pause la simulation et laisse la possibilité à l'utilisateur de modifier le réseau.
4. **Resume :** Reprendre la simulation.
5. **Stop :** Arrêter la simulation : Afficher les performances calculées et la simulation est finie.
6. **Reset :** L'utilisateur peut supprimer le réseau et tous les éléments qui le composent.
7. **Results :** Affiche sur une nouvelle fenêtre graphique les résultats de calculs de performances.
8. **History :** Affiche un historique des performances
9. **Zoom in :** Agrandir les objets.
10. **Zoom out :** Rétrécir les objets.
11. **Add widget :** Ajout d'un objet.
12. **Remove widget :** Suppression d'un objet.
13. **Move widget :** Cliquer sur un objet puis sur un emplacement le déplacera à cet endroit
14. **Configure widget :** Cliquer sur un objet, une nouvelle fenêtre graphique s'affiche avec ses caractéristiques et il est possible de les modifier.
15. **Add class :** Permet d'ajouter une classe au réseau.
16. **Remove class :** Supprimer une classe dans le cas d'un réseau multi-classe.

4.2 Exigences d'utilisabilité

L'application doit être facilement utilisable par des personnes ne maîtrisant pas les fondements des réseaux de files d'attente. Elle fournira une aide sous forme de définitions des différents concepts techniques afin que l'utilisateur puisse en apprendre davantage sur les réseaux de files d'attente. Elle fournira également un manuel d'utilisation de l'application décrivant les différentes étapes à suivre pour créer son réseau et expliquant les différents boutons se trouvant sur l'interface. L'interface doit être claire et intuitive pour assurer le confort de l'utilisateur.

5 Questions sur le projet

5.1 Exigences mises en attente

Nous avons décidé de pouvoir représenter des réseaux de files d'attente ouverts et fermés. Nous avons fait le choix de de ne pas représenter les réseaux mixtes pour se consacrer à l'essentiel. D'autres ordonnancements (processor sharing, ajout de la préemption...) auraient aussi pu être inclus, ainsi que de nouvelles lois d'arrivées et de service. Nous pourrions envisager d'ajouter ces changements afin de proposer une application encore plus complète.

5.2 Risques

En gestion de projet, on peut rencontrer un certain nombre de risques :

- **Risques concernant le respect du planning :** le délai peut être un peu trop court.
- **Risques humains :** mauvaise communication et difficultés de répartition des tâches de chacun.
- **Risques techniques :** manque d'expérience pour certaines fonctionnalités que nous voulions coder.

5.3 Manuel utilisateur et formations

Un manuel d'utilisation est mis à la disposition de l'utilisateur afin de l'aider à utiliser toutes les fonctionnalités de l'application. Chaque terme complexe sera accompagné d'une définition permettant à l'utilisateur de mieux comprendre ce qu'il manipule.

5.4 Tâches à livrer pour le projet

- **11 Mars :** Remise du Cahier des Charges
- **17 Mars :** Soutenance du Cahier des Charges
- **21 Avril :** Remise du Cahier des Spécifications
- **25 Mai :** Remise de l'application et du compte-rendu
- **29 Mai :** Présentation Finale

5.5 Estimation des coûts

- **Tableau d'estimation des coûts**

Module	Estimation en ligne de code	Estimation en volume horaire	Nom et prénom
Interface Graphique	1500	72h	ADOUANE Cylia BONNET Ludivine HAMMAD Amir HELLAL Ouiza
Gestion de fichiers	400	24h	HELLAL Ouiza
Gestion du réseau	2000	72h	ALI AHMEDI Mycipssa DOUBI Dylan LAMOUR Lauriane MOLINER Emma
Calculs de performances	400	12h	ADOUANE Cylia

- **Module Interface**

Fonctionnalités - Module Interface	Personne chargée de coder la fonctionnalité
Ajouter une station	BONNET Ludivine
Retirer une station	BONNET Ludivine
Déplacer une station	BONNET Ludivine
Configurer une station	BONNET Ludivine
Ajouter une classe pour un réseau multiclasse	HAMMAD Amir
Retirer une classe du réseau multiclasse	HAMMAD Amir
Visualiser les mesures de performances	ADOUANE Cylia
Lancer la simulation	HAMMAD Amir
Mettre en pause la simulation	HAMMAD Amir
Reprendre la simulation	HAMMAD Amir
Arrêter la simulation	HAMMAD Amir
Sauvegarder le réseau	HELLAL Ouiza
Ouvrir un fichier	HELLAL Ouiza
Proposer un réseau déjà existant	HELLAL Ouiza
Zoomer	ADOUANE Cylia
Dézoomer	ADOUANE Cylia
Supprimer le réseau	BONNET Ludivine

- **Module Gestion du réseau**

Fonctionnalités - Module Gestion du réseau	Personne en charge de coder cette fonctionnalité
Distribuer le client dans les files au début de la simulation	ALI AHMEDI Mycipssa
Gérer l'échéancier	LAMOUR Lauriane
Distribuer les clients dans les files/sortie pendant la simulation	DOUBI Dylan
Ordonnancer les clients dans les files	MOLINER Emma
Calculer le temps d'attente de chaque client	MOLINER Emma
Calculer le temps de service de chaque client	ALI AHMEDI Mycipssa
Calculer le taux d'arrivée pour chaque file	DOUBI Dylan

- **Module Gestion de fichiers / Module Calculs de performances**
  En ce qui concerne le module Gestion de fichiers, toutes les fonctionnalités seront codés par HELLAL Ouiza et pour le module Calculs de performances, ADOUANE Cylia.

6 Choix du langage

L'application consiste à simuler un réseau de files d'attentes. Cette simulation faite grâce à un échéancier vise à calculer les performances du réseau créé.

Un réseau de file d'attente est principalement composé de trois entités : les files d'attente, les serveurs et les clients. Chacune de ses entités possède des caractéristiques qui lui sont propres et ne devant pas être modifiées par des facteurs extérieurs car la fiabilité des données est nécessaire à l'exactitude des calculs de performances et au bon déroulement de la simulation. Ceci nécessite alors la notion d'encapsulation qui permet de protéger l'information contenue dans les objets et de les rendre manipulables uniquement par ceux qui en possèdent les droits.

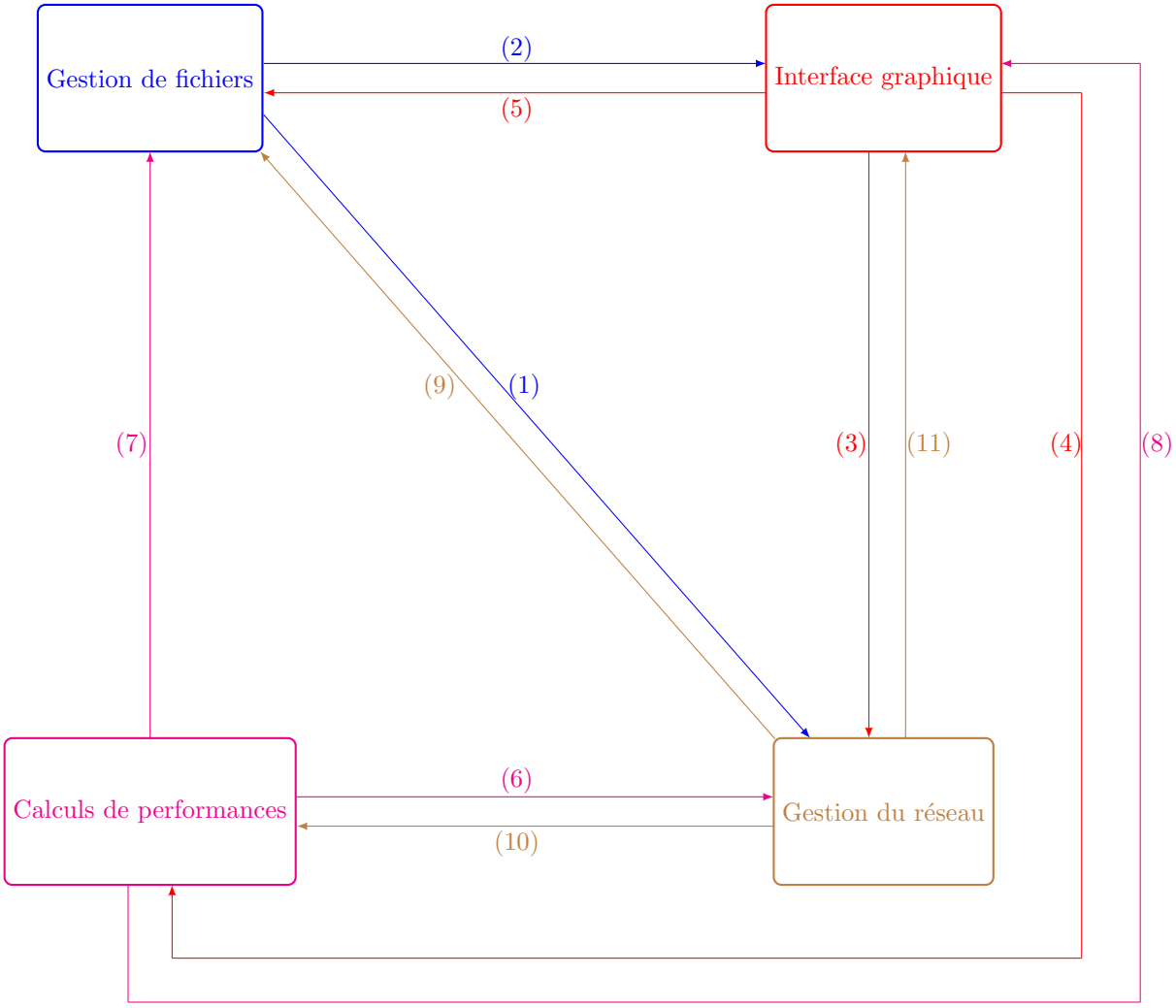
De plus, il faudra traiter deux types de clients, les clients d'un réseau mono-classe et les clients d'un réseau multi-classe, qui auront tous deux des caractéristiques en commun comme la sauvegarde de leur temps d'attente ou de leur temps de service. Néanmoins il faudra les différencier sur certains points comme le fait qu'un client dans un réseau multi-classe appartienne à une classe. L'héritage sera utile pour ce faire.

Le réseau sera affiché sur l'interface en fonction de ses caractéristiques, un réseau ouvert comportera une entrée et une sortie à la différence d'un réseau fermé qui n'en a pas. La méthode chargée d'afficher le réseau sera donc différente en fonction du type de réseau. Cet aspect requiert la notion polymorphisme.

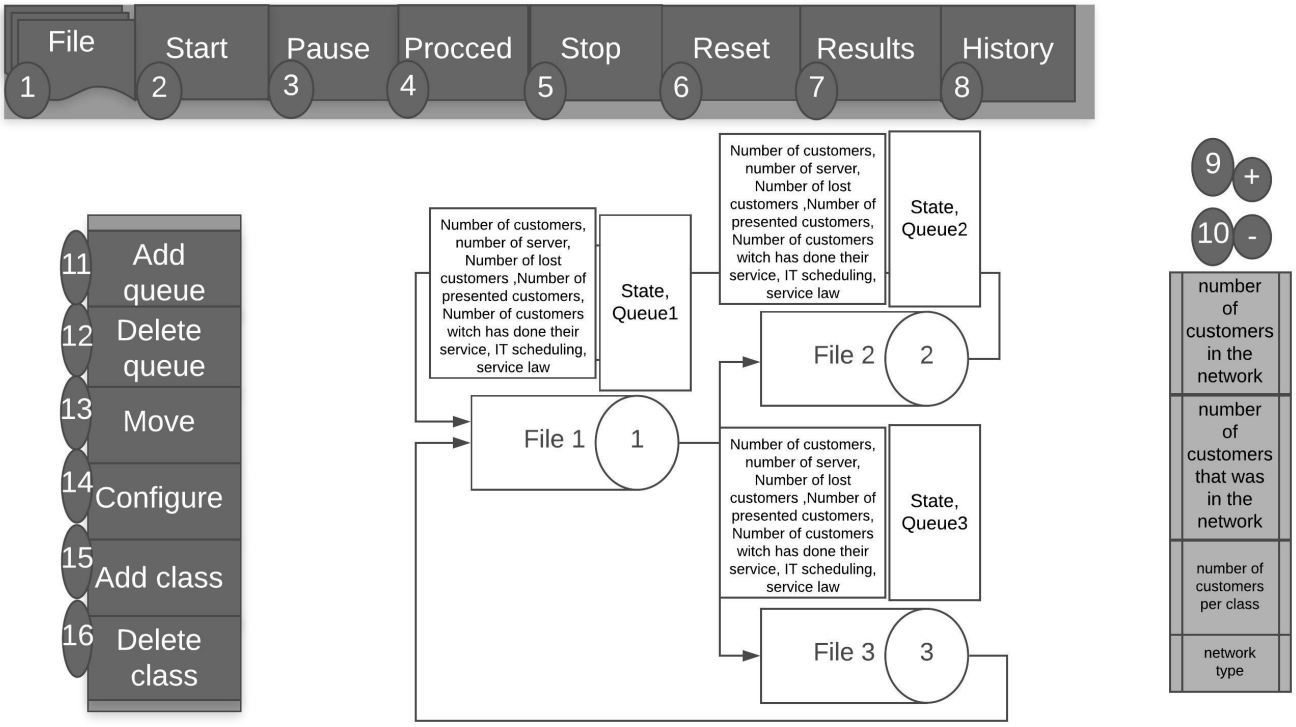
Par ailleurs, un langage alliant performance et vitesse d'exécution est indispensable au bon fonctionnement de notre application. Des calculs assez lourds seront effectués. Le réseau stocke énormément d'informations en mémoire auxquelles il faudra rapidement accéder lors des changements d'états. Il faudra tenir compte lors de la suppression d'un élément de toutes les autres modifications qu'elle implique. Gérer manuellement la mémoire sera utile.

En conclusion, nous avons besoin que le langage choisi gère l'encapsulation pour éviter des modifications inattendues faussant les calculs de performances. Il doit également pouvoir proposer l'héritage, nécessaire notamment pour représenter les différents types de clients possibles. Il doit permettre une vitesse d'exécution élevée pour les calculs. Enfin, il doit laisser la possibilité de gérer la mémoire. Ainsi le langage C++ est le plus adapté. Il est portable, stable, opensource, extensible, compatible avec de nombreuses bibliothèques dédiées aux interfaces graphiques et très connu des développeurs pour une éventuelle maintenance. On le choisit donc pour la réaliser.

7 Annexes



Organigramme



Maquette de l'interface de l'application

8 Bibliographie

Articles sur internet :

1. Auteur : Claude HASSENFORDER - Processus stochastique : Modelisation", Université de Toulouse Le Mirail
2. Auteurs : Wissem SANA, Mounira GUEJDALI, Université A/MIRA de Béjaia(Algérie) - Mémoire de fin de cycle en vue de l'obtention du Diplôme de Master Thème :Les réseaux de files d'attente à Forme Produit
3. Auteur : Céline DECOSSE, Centre de recherche public Henri TUDOR 2006 - Plan de cahier des charges et spécification des exigences non fonctionnelles avec Volere
4. Auteurs : Fabienne PERRONNIN, Polytech Grenoble - Queuing Networks

Livres :

1. "Simulation à événements discrets"
   Auteurs : Gérard FLEURY, Philippe LACOMME, Alain TANGUY
   Éditeur : Eyrolles 2006
2. "Expression des besoins pour SI : guide d'élaboration du cahier
   Auteur : Yves CONSTANTINIDIS 2015
   Éditeur : Eyrolles 2ème édition (2015)
3. "Introduction aux processus stochastiques et à la simulation "
   Auteur : Gérard-Michel COCHARD
   Éditeur : ISTE 2019
4. "Simulation par événements discrets"
   Auteurs : Erard, Pierre-Jean ; Déguénon, Pontien
   Éditeur : Lausanne Paris : Presses polytechniques et universitaires romandes (1996)
5. Processus aléatoires à temps discrets
   Auteurs : Franchi, Jacques
   Editeur : Ellipses 2013
6. "Probabilités, Processus stochastiques et applications"
   Auteurs : Valérie, Girardin, Nikolaos, Limnios
   Editeur : Vuibert 2014
7. " Processus de Markov et applications"
   Auteurs : Étienne, Pardoux
   Editeur : Dunod 2007
8. "Probability and computing"
   Auteurs : Mitzenmacher, Michael
   Éditeur : Cambridge New York N.Y. : Cambridge University Press 2017