Réf: 98/ I/ I Soutenu à la session de Juillet 1998

Université de TUNIS II
ECOLE NATIONALE DES SCIENCES DE L'INFORMATIQUE

RAPPORT
de Projet de Fin d'Études

Présenté en vue de l'obtention du titre
D'INGÉNIEUR EN INFORMATIQUE

par:
Abdelaziz HOUAIDI
Fethi FILALI

Sujet:

Mise en d'un atelier de conception et d'optimisation de réseaux

Encadrés par:

Mr. Farouk KAMOUN
Professeur, ENSI

Mr. Mohamed Kamel BEN RHOUMA
Maître de conférences, CCK

Mme. Sonia ZGHAL
Ingénieur, CCK

Organismes:

Centre de Calcul El Khawarizmi (CCK)
Ecole Nationale des Sciences de l'Informatique (ENSI)

Année universitaire 1997/1998

DÉDICACES

À ma douce et adorable mère,
À mon père qui a tout sacrifié pour que j'arrive là où je suis,
À mes frères et squi n'hésitent pas à m'adresser toute sorte d'aide,
À tous ceux que j'aime et qui supportent mon manque quasi permanent
de disponibilité à leur égard.
FETHI

À mes très chers parents
pour leur sacrifices et leur patience,
À mes frères et s pour leur soutien et leur encouragement,
À toute ma famille qui n'a jamais manqué de m'apporter toute sorte d'aide,
Je dédie ce mémoire de fin d'études.
ABDELAZIZ

REMERCIEMENTS

Au terme de ce travail de fin d'études, réalisé au **C**entre de **C**alcul de **K**hawarizmi (CCK) en collaboration avec l'**É**cole **N**ationale des **S**ciences de l'**I**nformatique (ENSI), nous tenons à remercier vivement, le professeur **Farouk KAMOUN**, Directeur de l'ENSI, pour nous avoir proposé ce sujet, pour son assistance lors de la réalisation de ce projet, pour sa disponibilité qui nous a permis de mener à terme ce travail. Son encadrement nous a été très bénéfique.

Nous voudrions, également, exprimer toute notre gratitude à Monsieur **Mohamed Kamel Ben RHOUMA**, Directeur du Centre de Calcul Khawarizmi, pour nous avoir offert l'opportunité de ce stage, pour son intervention comme conseiller, pour ses remarques précieuses qui nous ont permis de mener efficacement ce travail.

Nous exprimons notre reconnaissance à Monsieur **Yassine JAMOUSSI**, pour son suivi de près de l'avancement de ce projet, pour les conseils judicieux qu'il n'a cessé de nous prodiguer, ainsi que pour la précieuse aide qui a permis d'améliorer la qualité de ce rapport.

Un grand merci à Monsieur **Rafik BRAHAM**, pour la patience dont il a fait preuve à notre égard, pour ses suggestions, son aide et ses encouragements.

Nous sommes très reconnaissants à Madame **Sonia KALLEL** née **ZGHAL**, d'avoir eu la gentillesse de nous fournir la documentation nécessaire. Ses conseils techniques nous ont été très précieux.

Nous remercions Monsieur **Larbi BATTI**, ingénieur en TUNIPAC, d'avoir voulu nous fournir des explications sur les tarifs télécoms.

Enfin, nous voulons remercier tous nos **enseignants**. Sans le savoir faire qu'ils n'ont cessé de nous communiquer, ce stage n'aurait pu être mené à bien.

Abstract

The Wide Area Network design considering performance criteria (Response time, reliability ...) on the one hand, and the telecom offer (communication media, ...) on the other hand, is the principal problem encountred by network designers looking for an optimal network layout.

The project aim is to develop a software engineering environnement capable to solve the general problem of optimal WAN design and so, all its sub problems. To do this, an intelligent computing core has been implemented. It allows the extraction of tariffs data from a tariff data base, carries out all optimization calculation to display results for designers on a convivial interface.

The project application has been done on the National Academic Network, designed to ineterconnect all tunisian universities in order to benefit from Internet services.

***Keywords**: Topology, WAN, Design, Optimization, Tariffs, Capacity, Flow, Reliability, Connectivity.*

Résumé

La conception des réseaux étendus moyennant des critères de performances (temps de réponse, fiabilité ...) d'une part, et l'offre télécom (technologies de communication,...) d'autre part, est le problème majeur rencontré par les concepteurs de réseaux cherchant à établir un maillage optimal entre les divers sites.

Ce projet vise à offrir un atelier logiciel capable de résoudre le problème général de dimensionnement de réseaux de transit ainsi que les sous problèmes qui en découlent. Pour ce faire, un noyau intelligent de calcul a été implémenté. Il permet d'extraire des données tarifaires à partir d'une base de données des tarifs, d'effectuer les calculs d'optimisation nécessaires et d'afficher les résultats au concepteur via une interface conviviale.

L'application de ce travail, a été faite sur le projet Réseau National Universitaire (RNU), qui vise à interconnecter l'ensemble des institutions universitaires tunisiennes, les équipes et laboratoires de recherche et de mettre à leur disposition tous les services Internet.

***Mots clés**: Topologie, WAN, Conception, Optimisation, Tarifs, Capacité, Flot, Fiabilité, Connectivité.*

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

1.1 Présentation du sujet

Un réseau informatique est un ensemble de ressources de communication, d'ordinateurs et de clients cherchant à exploiter ces ressources. Pour simplifier, un réseau est un ensemble d'équipements partageables et géographiquement distribués. Dès que leur nombre dépasse quelques stations, il n'est plus possible de les interconnecter par une simple ligne multipoint et il devient nécessaire d'organiser les échanges sur un réseau qui met en jeu un certain nombre de nœuds interconnectés par des liaisons (cf. figure 1.1).

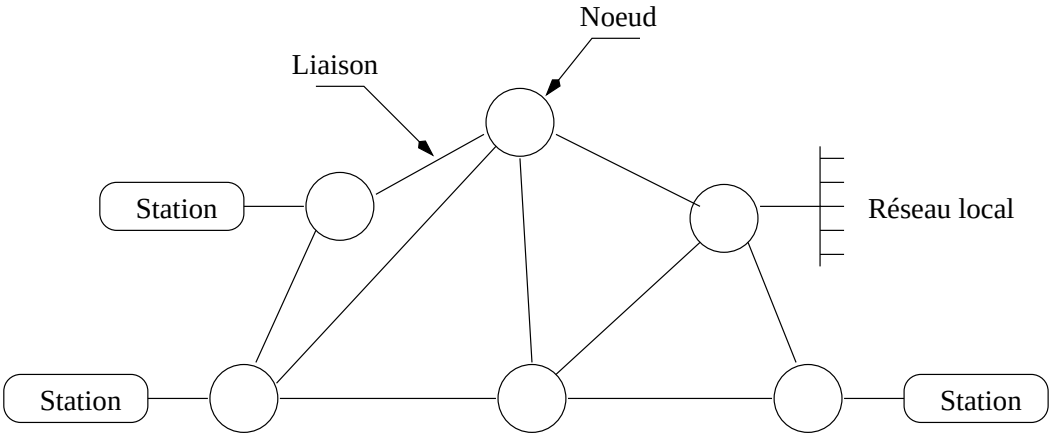


FIG. 1.1 – Un réseau informatique

Une telle structure pose un grand nombre de problèmes nouveaux par rapport à une simple liaison. En effet, au niveau de la conception, il convient de déterminer la topologie du réseau en fonction des contraintes et des critères imposés au concepteur. Au niveau de l'exploitation, le réseau doit être capable d'assurer le routage des messages et de garantir le fonctionnement correct malgré les surcharges et les défaillances des équipements.

En partant d'une spécification du système réel décrivant les données de trafic et les contraintes dont il faut tenir compte, le concepteur cherche à trouver un système idéal optimisant la fonction objective. Ce processus se compose en général de quatre étapes essentielles: la modélisation, l'analyse, la conception et l'évaluation. La figure 1.2 décrit l'interaction existante entre ces étapes. Ainsi, suite à une modélisation du système par un réseau de

files d'attente, il convient de l'analyser afin de déterminer les expressions mathématiques des différentes variables qui seront prises en compte pendant la phase de la conception. Durant cette phase et selon la nature du problème, des méthodes analytiques ou heuristiques sont appliquées pour donner une nouvelle configuration du système. En confrontant les résultats d'une simulation ou d'une résolution mathématique et les objectifs à atteindre, une évaluation du système est menée pour décider de sa validité.

Du fait du coût élevé des communications, la topologie des réseaux informatique doit être étudiée parfaitement. En effet, si cette étude ne permet de réduire les coûts que de quelques pourcent, les sommes ainsi économisées peuvent être considérables.

Le problème d'optimisation d'un réseau informatique peut être formulé de la façon suivante: étant donné des équipements informatiques à interconnecter, leur localisation, le trafic entre stations, ainsi que le coût des liaisons et les différents équipements du réseau, le concepteur, durant le processus de conception, cherche à minimiser le coût du réseau. Cette optimisation doit être effectuée en tenant compte d'un certain nombre de contraintes dont les principales sont généralement d'assurer une fiabilité minimale au réseau et de garantir que le temps de traversée du réseau par les paquets ne dépasse pas une limite donnée.

Le problème, ainsi posé, ne peut pas être résolu par une étude exhaustive, car le nombre de combinaisons possibles devient gigantesque dès que le réseau comporte plus de quelques nœuds [NUS 87]. De plus, la complexité du problème est telle qu'il n'existe aucune méthode d'analyse permettant d'arriver à coup sûr à un réseau optimum. C'est pourquoi, les outils logiciels de planification et d'aide à la décision relatifs au dimensionnement prennent une nouvelle importance.

Le but du présent projet de fin d'études est justement d'étudier les problèmes d'allocation de ressources et d'implémenter un outil appelé *"WanDesigner"* permettant la conception et l'optimisation des réseaux et ce, compte tenu des besoins en communication d'une part, et des supports et technologies d'interconnexion d'autre part.

En général, une simplification du problème de conception du réseau consiste à diviser ce dernier en deux parties: le réseau de transit et le réseau local, et à traiter séparément ces deux parties. Le réseau de transit est le cœur du réseau informatique, et il est constitué par des lignes à grand débit reliant des nœuds comportant chacun un commutateur de paquets. Le réseau local relie les stations des abonnés aux nœuds du réseau de transit, soit directement, soit par l'intermédiaire de concentrateurs ou de lignes multipoint.

1.2 Plan du rapport

La solution qui est généralement adoptée et à laquelle nous avons fait recours, consiste à diviser le problème en une série de sous-problèmes qui sont traités séparément, soit par des méthodes analytiques exactes, soit par des heuristiques. L'organisation du rapport est bâtie, en conséquence, sur cette même décomposition.

Comme tout problème d'optimisation faisant appel aux théories de files d'attente et de graphes, un modèle de files d'attente doit être défini.

Pour la mise en contexte, nous commencerons par la description du modèle adopté. Nous essayerons aussi, à partir d'une étude complète du problème d'allocation des ressources de transmission de données, de dégager les variables pertinentes pouvant influer sur les performances et les coûts du réseau à concevoir et les contraintes dont il faut tenir compte. Ces variables nous permettront de décrire les sous-problèmes à traiter.

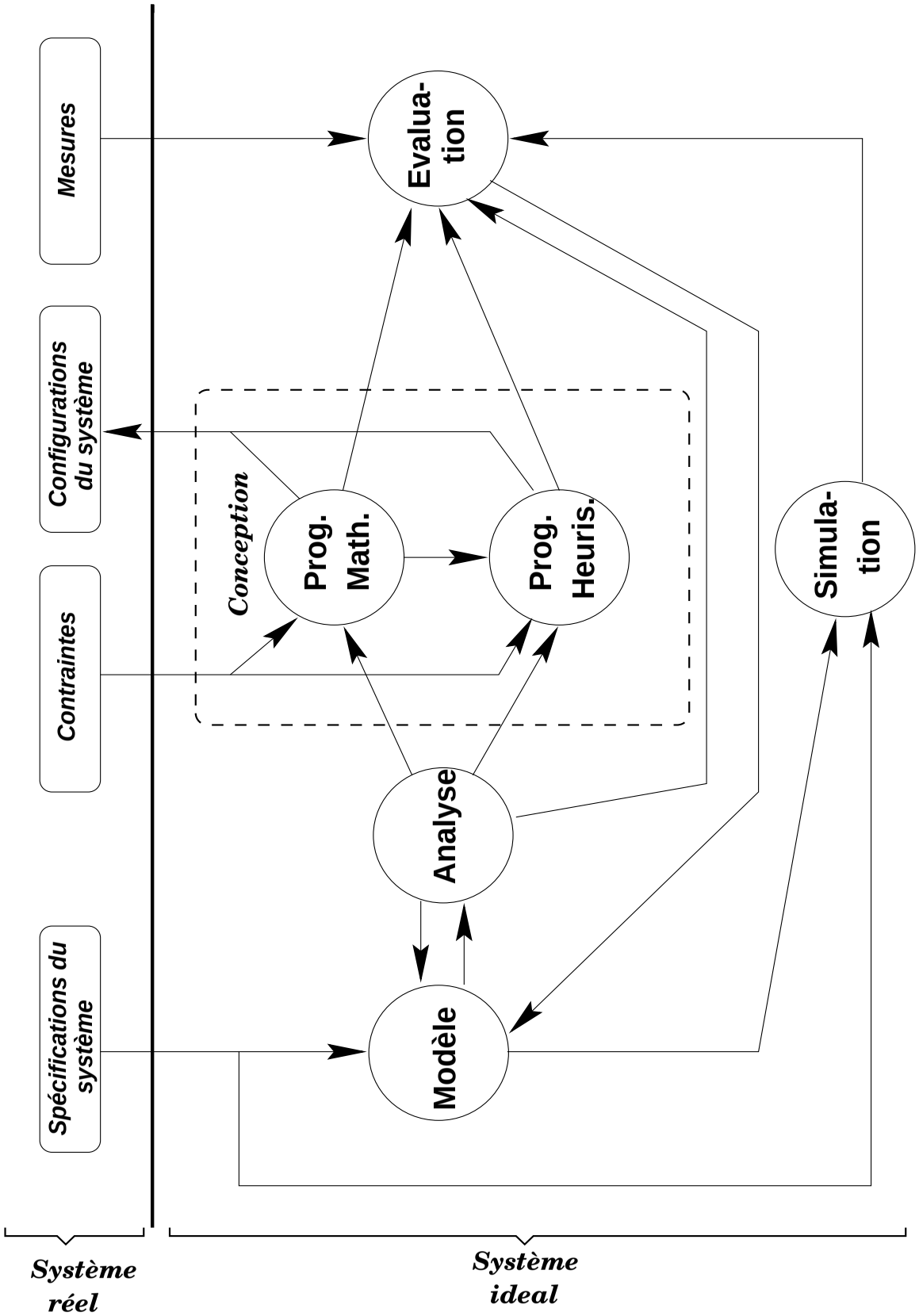


FIG. 1.2 – Interaction entre modélisation, analyse, conception et évaluation [GER 73]

Dans une première étape, nous étudierons les différentes technologies existantes (RTC, X25, LS, RNIS) utilisées pour l'interconnexion inter-sites. Nous menons également, dans ce chapitre, une étude comparative des tarifications dans l'espoir de cerner le problème de caractérisation des coûts.

Dans une seconde étape, nous étudierons le problème d'allocation de capacité (CA: Capacity Assignment). Ce chapitre propose des éléments de solutions sous forme d'algorithmes analytiques pour le cas des capacités continues et des heuristiques pour le cas des capacités discrètes.

Dans une troisième étape, nous traiterons le problème d'allocation de flot (FA: Flow Assignment) qui est intimement dépendant du choix de la politique de routage. A ce propos, nous présenterons la démarche suivie pour la résolution du problème de routage fixe.

Dans une quatrième étape, nous chercherons des solutions pour le problème d'allocation de capacité et de flot (CFA: Capacity and Flow Assignment) à partir des éléments ayant été fournis dans les deux chapitres précédents.

Dans une dernière étape, nous présenterons le problème général de conception de topologie (GAP: Generalized Assignment Problem). Dans ce contexte, trois types de topologies: maillée, en arbre et en étoile feront l'objet de ce chapitre.

Nous finirons cette partie par la présentation de quelques résultats expérimentaux issus de la conception et l'optimisation du Réseau National Universitaire (RNU).

Pour évaluer et classer les divers algorithmes disponibles pour la résolution des sous-problèmes de conception, nous étudions dans chaque chapitre traitant un problème d'optimisation, la complexité des algorithmes utilisés.

Pour mieux comprendre le fonctionnement des algorithmes, nous illustrons chaque chapitre par une application faite sur le RNU.

La deuxième partie du rapport décrira le processus de mise en œuvre de l'atelier moyennant une méthode de conception orientée objet. Après avoir compris et modélisé les besoins et les fonctionnalités, une architecture de l'atelier est déterminée durant la phase de conception. Enfin, une conception concrète des algorithmes développés dans la première partie, des tests de validation seront détaillés.

En dernier lieu, nous achèverons par une conclusion et les perspectives de ce projet.

1.3 Présentation de l'organisme: CCK

Le Centre de Calcul EL-KHAWARIZMI est un établissement dépendant de l'Université des Sciences et Techniques et de Médecine de Tunis de Ministère de l'Enseignement Supérieur. Il a été créé en Octobre 1976 suite à l'introduction de l'informatique en tant que discipline autonome dans les milieux universitaires.

Ce centre offre les services suivants:

- mettre à la disposition des chercheurs, des doctorants et des enseignants les moyens et les services nécessaires pour favoriser le développement de la recherche scientifique;
- répondre aux demandes des services de l'administration de la Faculté des Sciences de Tunis;
- fournir l'accès Internet pour toutes les institutions d'enseignement supérieur: enseignants, doctorants, étudiants et les centres de recherche.

Première partie

Etude théorique et résultats expérimentaux

Chapitre 1

Modèle, variables et contraintes de conception

Ce chapitre sera consacré en premier lieu à la présentation du modèle à utiliser durant les étapes de conception d'un réseau informatique étendu. Moyennant ce modèle, nous déterminons l'expression mathématique du temps de réponse moyen. Enfin, nous passons à l'énumération et à la définition des différentes variables et contraintes de conception de réseaux.

1.1 Le modèle

1.1.1 Présentation du modèle

Pour des raisons de routage, les messages arrivant à un nœud sont distingués selon leur destination (cf. chapitre 3). Les messages ayant la même destination appartiendront à la même classe de clients. De cette façon, on aura tant de classes de clients que de nœuds dans le réseau. Selon la politique de routage utilisée, un message de la classe i arrivant au nœud k est délivré vers une file d'attente particulière ou il est rejeté [1]

Le modèle ainsi proposé est un réseau de files d'attente. La figure 1.1 montre que le nœud k route le message ayant pour destination i vers la ligne (k, l) avec la probabilité $P_{k,l}^{(i)}$ qui dépend de la politique de routage choisie. En effet, il sera constant pour un routage déterministe et variable (dépend de l'état global du système) pour un routage adaptatif.

En se basant sur ce modèle, nous déterminons dans la section suivante l'expression de temps de réponse moyen du réseau informatique.

1.1.2 Détermination du temps de réponse moyen

Le modèle présenté ci-dessus, est un peu général, il ne spécifie pas les lois d'arrivées de messages, les types des files d'attente... Nous essayons de le simplifier en se basant sur des hypothèses liées entre autres à la probabilité d'erreurs, au temps de traitement des messages...

1. si le message présente des erreurs ou si le buffer est plein.

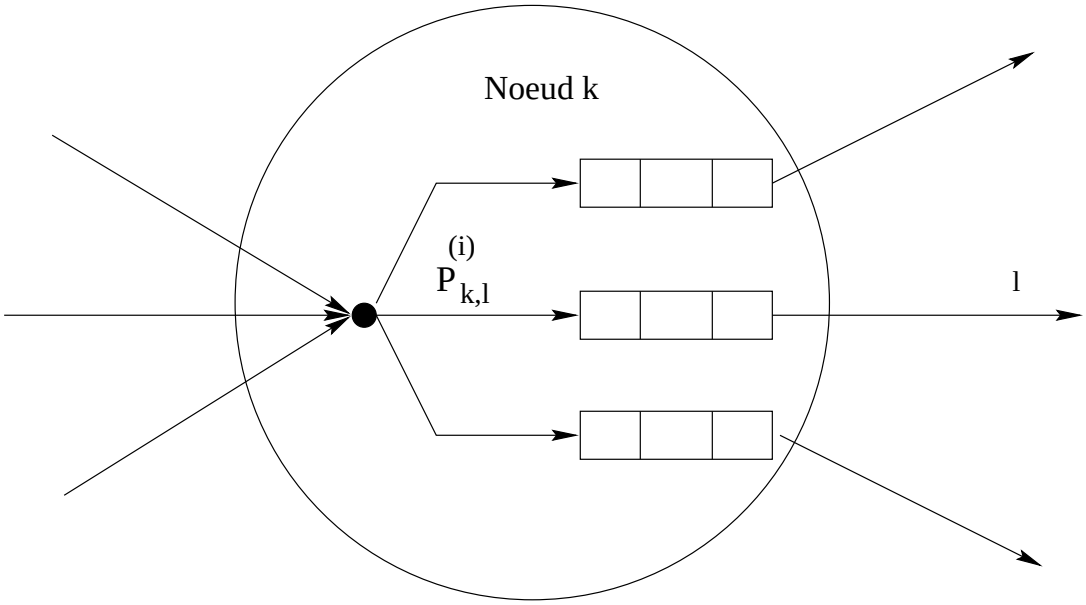


FIG. 1.1 – Modèle d'un nd'un réseau de type Store/Forward

Dans tout ce qui suit, les notations suivantes seront utilisées:

N: nombre de nœuds,
M: nombre de lignes,
N_i: nœud i,
M_i: ligne i;
$(N_i\ ,\ N_j)$: la ligne entre N_i et N_j.

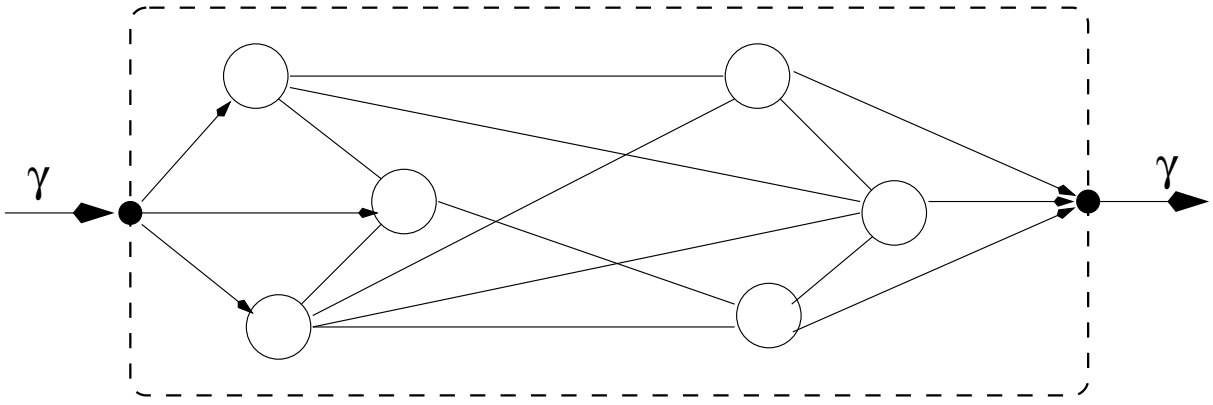


FIG. 1.2 – Résultat de Little

Nous commençons tout d'abord par exprimer le temps de transmission moyen T. Le théorème de Little (cf. section A.2) appliqué au système de la figure (1.2) à l'équilibre, sous l'hypothèse d'absence de pertes, nous permet d'écrire:

$$\gamma T = \overline{n} = \sum_{k=1}^{M} \overline{n_k} \tag{1.1}$$

Avec $\overline{n}$: le nombre moyen de messages dans le réseau;
$\overline{n_k}$: le nombre moyen de messages dans la file d'attente de la ligne k.
Pour la ligne k le théorème de Little donne:

$$\overline{n_k} = \lambda_k T_k \tag{1.2}$$

Où λ_k : le taux moyen d'arrivée à la ligne k;
T_k: le temps de réponse moyen de la ligne k.
D'après (1.1) et (1.2) on a:

$$T = \frac{\sum_{k=1}^{M} \lambda_k T_k}{\gamma} \tag{1.3}$$

C'est une expression simple et générale, elle présente une certaine difficulté dans la détermination de T_k et λ_k. En effet, ceci nécessite la connaissance du routage, des lois d'arrivées, des lois de services... Pour ce faire, on suppose que [KLE 64]:

1. les arrivés externes sont poissonniennes,
2. la longueur de messages suit une distribution exponentielle,
3. les buffers ont une taille infinie,
4. le routage est déterministe,
5. la probabilité d'erreur est nulle (lignes parfaitement fiables),
6. le temps de traitement est nul,
7. l'hypothèse d'indépendance [2] de **Kleinrock**.

Avec ces hypothèses, le système peut être considéré comme un réseau de files d'attente de type Jackson [JAC 57] avec plusieurs classes de clients. Dans ce cas, chaque file d'attente est considérée de type M/M/1 (cf. section A.2) et le temps de réponse moyen de la ligne k est donné par:

$$T_k = \frac{1}{\mu C_k - \lambda_k} \tag{1.4}$$

Où $1/\mu$: la longueur moyenne de messages (bits/sec),
C_k: la capacité de la ligne k (bits/sec),
λ_k: le taux moyen d'arrivée de messages à la ligne k (messages/sec).
Comme le routage est déterministe, les λ_k seront facilement déduits de la matrice de trafic externe et du routage.

Le temps de réponse moyen T est donné par (en remplaçant T_k dans (1.3) par son expression de (1.4)

$$T = \frac{1}{\gamma} \sum \frac{\lambda_k/\mu}{C_k - \lambda_k/\mu} \tag{1.5}$$

ou encore

$$T = \frac{1}{\gamma} \sum \frac{f_k}{C_k - f_k} \tag{1.6}$$

Où $f_k = \lambda_k/\mu$ est le taux d'arrivée en bits par seconde.

2. Les paquets perdent leurs identités à leur arrivée dans un nœud et ce dernier reconstruit des paquets avec une longueur distribuée exponentiellement.

1.2 Les variables et les contraintes

1.2.1 Les variables de conception

Nous présentons ici une liste non exhaustive des variables de conception qui seront utiles à la description des problèmes d'allocation de ressources.

La topologie

Pour un ensemble de N nœuds, il y aura $\frac{N(N-1)}{2}$ lignes duplex possibles et par la suite $2^{\frac{N(N-1)}{2}}$ topologies qui s'obtiennent en variant l'ensemble de lignes. Par exemple, pour $N = 10$ on aura $4 * 10^{13}$ représentations qui rendent impossible d'explorer toutes les solutions à la recherche de la topologie optimale [TAN 81]. En effet, si le temps de traitement d'une topologie est de $1ms$, il nous faut une durée de l'ordre de 10 siècles pour le cas de $N = 10$.

Les lignes de communication

Pour chaque arc M_i est associé une capacité C_i et un coût D_i. L'expression du coût dépend fortement de la technologie de communication utilisée. En effet, il peut être fonction du volume de données transférées (X25), de capacités (LS), de la durée de connexion (RTC, RNIS). Le chapitre suivant, traitera en détail le problème de caractérisation des coûts.

La matrice de trafic

La matrice de trafic (notée Γ) est une donnée fondamentale de tout problème de conception de réseau. L'élément γ_{ij} de la matrice donne le nombre de messages par seconde émis par le nœud N_i vers le nœud N_j. Cette matrice est déterminée par des outils d'analyse du trafic [ROM 98] dans le cas où le réseau est déjà existant et elle est estimée dans le cas de conception d'un nouveau réseau. Une approximation de γ_{ij} dans ce dernier cas est donnée par [KLE 76]:

$$\frac{nb\,utilisateurs(i) * nb\,utilisateurs(j)}{distance(i,j)^{\alpha}} \tag{1.7}$$

Où $\alpha = 1,\ 2,\ ...$

La politique de routage

Il existe plusieurs politiques de routage possibles: fixe, adaptatif, hiérarchique, ... Dans notre cas, nous considérons la politique de routage déterministe pour deux raisons essentielles. La première, que nous avons déjà indiquée, découle de la facilité de la détermination des flots de différents canaux de communications à partir des matrices de routage et de trafic. La deuxième, est liée à l'hypothèse d'indépendance de **Kleinrock** qui peut être vérifiée, par des simulations, pour le cas du routage déterministe mais elle ne le serait jamais pour le cas d'une politique de routage adaptative.

Les flots sur les canaux de communication

Si tous les paquets circulant dans le réseau ont la même longueur moyenne, le nombre moyen λ_i de paquets qui arrivent par seconde sur chacune des lignes M_i, avec $i = 1, ..., M$, est déterminé en prenant la somme de tous les trafics γ_{ij} qui passent par cette ligne.

$$\lambda_i = \sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_{kl}^{i} \gamma_{kl} \tag{1.8}$$

tel que, le trafic $\alpha_{kl}^{i} \gamma_{kl}$ suit la route passant par la ligne *i.*

1.2.2 Les contraintes

Pendant la résolution des problèmes d'allocations de ressources, certaines contraintes doivent être respectées. Nous citons, dans cette section, une liste de ces contraintes.

Contraintes de flux

– Condition de conservation de flot pour chaque paire (i, j) de nœuds

$$\sum_{j=1}^{N} f_{ji}^{(k,l)} - \sum_{j=1}^{N} f_{ij}^{(k,l)} = \begin{cases} \frac{-\gamma_{kl}}{\mu} \; si \; i = k \\ +\frac{-\gamma_{kl}}{\mu} \; si \; i = l \\ 0 \; sinon \end{cases} \tag{1.9}$$

où $f_{ij}^{(k,l)}$ est le flux produit par le nœud *i* vers le nœud *j* passant par le lien *(k,l).*

– Condition de positivité

$$f_{i}^{(k,l)} \geq 0 \tag{1.10}$$

Contrainte de capacité

$$\lambda_i \leq \mu C_i \quad ou \quad f_i \leq C_i \tag{1.11}$$

Contrainte de coût ou objectif à optimiser

$$D \leq D_{max}$$

où D_{max} est le budget maximal qui est une donnée pour certains problèmes.

Contrainte de délai de transit ou objectif à optimiser

$$T \leq T_{max}$$

où T_{max} est le temps de réponse à ne pas dépasser.

Contraintes de topologie

- Connectivité: pour des raisons de fiabilité, il est impératif, dans certains cas, qu'il existe au moins k (degré de connectivité) chemins différents (n'ayant en commun que les nœuds source et destination) entre chaque paire de nœuds.
- Degré maximal de connexion: pour des raisons techniques, chaque nœud doit être lié à un nombre des nœuds voisins inférieur à un nombre maximal. À titre d'exemple, ce nombre correspond au nombre d'entrée maximal dans un routeur.

1.3 Conclusion

Après avoir esquissé le processus général de conception décrit par la figure 1.2 du premier chapitre, le deuxième chapitre a présenté le modèle de **Kleinrock** adopté par notre démarche, puis envisagé les variables et contraintes de conception qui sont, par leurs combinaisons, à la base de la distinction entre différents sous-problèmes d'allocation de ressources à savoir: allocation de capacité, allocation de flot, allocation de capacité et de flot et problème général de conception. L'étude détaillée de ces sous-problèmes ainsi que les algorithmes analytiques et heuristiques associés, feront l'objet des quatre chapitres qui suivent.

Chapitre 2

Caractérisation des coûts

À l'heure où le marché international des services télécom connaît une concurrence entre les opérateurs, les outils logiciels de planification et d'aide à la décision relatifs au dimensionnement des réseaux longues distances de voix et de données prennent une nouvelle importance. Ces logiciels d'optimisation des réseaux dépendent, en grande partie, des services et tarifs offerts par les opérateurs. En particulier, l'outil à réaliser ne peut, en aucun cas, éviter cette dépendance. En effet, le concepteur de réseau voulant implémenter un réseau privé se trouve confronté à une multitude de possibilités rendant la décision pas évidente à prendre dès le premier coup. Une comparaison des tarifs des lignes télécom s'avère nécessaire à ce propos.

Dans l'espoir de cerner le problème de caractérisation des tarifs, nous menons dans ce chapitre une étude détaillée du marché, suivie d'une analyse des données tarifaires existantes pour finir par une conclusion sur l'opération de calcul des coûts et des services télécom et la décision à prendre au profit du choix arrêté.

2.1 Etude de l'offre du marché

Si nous examinons les différentes techniques de communication offertes par le service télécom, nous constatons qu'elles sont trop variées. En plus des techniques traditionnelles (point à point, maillée, multipoint, ...), s'ajoutent des techniques nouvelles (commutation de paquets, satellite, radio, ...) offrant ainsi une large gamme d'alternatives pour les concepteurs de réseaux. Nous constatons également que la procédure de facturation de ces services varie d'un média à un autre à cause de la pluralité des paramètres régissant ces tarifs. Parmi ces paramètres, nous trouvons ceux qui sont liés au trafic, distance, temps et autres ...

À cause de la diversité de ces paramètres, les méthodes d'allocation de capacité ne permettent, en réalité, que d'obtenir une première solution approchée des débits à prévoir sur les lignes. Ceci est du, en fait à la fonction coût-capacité. Elle n'est pas, en général, une fonction linéaire, mais plutôt une fonction en marche d'escalier ou affine par intervalle. De plus, lorsque les lignes sont empruntées à des réseaux télécom différents, le coût d'une liaison ne dépend plus seulement de son débit et de sa portée mais également du réseau particulier par lequel passe cette liaison. Tous ces facteurs rendent la tâche de calcul des coûts, la tâche la plus cruciale à aborder.

Nous limitons notre étude aux quatre techniques les plus utilisées et qui sont fournies par TUNISIE TELECOM (le fournisseur des services Télécom en Tunisie) à savoir LS (ligne spé-

cialisée), X25, RTC (Réseau Téléphonique Commuté) et RNIS (Réseau Numérique à Intégration de Services). La structuration des tarifs des lignes dépend essentiellement du type de service et de la qualité de service qu'elle peut offrir.

2.1.1 Structuration des tarifs des Lignes Spécialisées (LS)

Les grandes sociétés et les organismes importants qui veulent un accès Internet, s'orientent vers une liaison spécialisée. Cette approche permet l'accès à tous les niveaux de services. Le fournisseur de services mettra en place une ligne spécialisée louée chez un opérateur de télécommunications. Le débit pourra être adapté à l'utilisation souhaitée (plus le débit est important, plus le coût augmente).

L'accès dédié offre une possibilité de configuration flexible et assure une connexion permanente à l'Internet depuis quelques kbits/s jusqu'à plusieurs mégabits de débits. En effet, chaque ordinateur connecté au réseau interne peut accéder à tous les services disponibles. Cependant, il est plus adapté à une large structure ayant des besoins et des moyens importants qu'un utilisateur individuel.

Nous distinguons trois classes de lignes "LS": LS à 2 fils, LS télégraphique à 2 fils et LS à 4 fils. La tarification de ces lignes est fixée par le fournisseur de services télécom et dépend de la classe même et de la distance à vol d'oiseau. Le choix d'une liaison spécialisée dépend du niveau de qualité cherché, du débit offert et également de l'évolution permise. S'il n'y a plus de facturation à la durée pour les liaisons spécialisées, le Fournisseur de Service Internet (FSI) qui assure la connexion impose, généralement, un débit mensuel maximal au delà duquel le débit supplémentaire est facturé. Nous récapitulons dans le tableau 2.1.1 les redevances annuelles des liaisons spécialisées en DT [1] :

Avec:

TR: Taxe de raccordement,

NJ: Liaison n'empruntant pas une jonction,

J: Liaison empruntant une jonction.

La procédure de facturation est donnée par la formule 2.1:

$$D = TaxeFixe + TaxeVariable * Distance \tag{2.1}$$

À partir de la figure 2.1, nous constatons que les courbes représentant la variation de coût en fonction de la distance pour trois types de LS, sont affines par intervalle. Au delà de 200 km, le coût devient indépendant de la distance.

0 8 1 0 0 0 0Classe	TR	Taxe fixe en DT	Taxe variable / km	LS 2 F	160	0 - 10 km	10 - 50 km	50 - 200 km	≥200 k
50 - 200 km	≥200			NJ	J	NJ	J	4853	1288
36.400	0			280	1036	1152	2412		
		LS 4 F	320	504	1260	1469.300	2729.300	6304	20220.2
65.800	0	LS 2 FT	160	264.600	457.800	2358.300	4458.300	68.600	49.70

TAB. 2.1 – Tarification des lignes spécialisées

1. Extrait de l'article 37 du JORT du 13/05/94.

2.1.2 Structuration des tarifs pour les lignes louées X25

Le raccordement à Internet par X25 permet d'utiliser le réseau public TUNIPAC pour se relier à un fournisseur. Différentes solutions sont disponibles pour ce type de connexion: équipement dédié ou carte ajoutée à un des équipements déjà connectés au réseau local. On utilise le plus souvent le protocole d'encapsulation IP sur X25. Quant au protocole PPP, il est plus souvent utilisé pour la connexion des particuliers par TUNIPAC, par l'intermédiaire d'un PAD (*Packet Assembler Disassembler*).

Les capacités disponibles sont des valeurs discrètes et varient de 1200 à 64000 bit/seconde. Nous récapitulons dans le tableau 2.2 les redevances mensuelles de la tarification X25 en accès direct. La représentation graphique de la fonction Coût-Capacité est donnée par la figure 2.2, elle présente une allure plus ou moins concave.

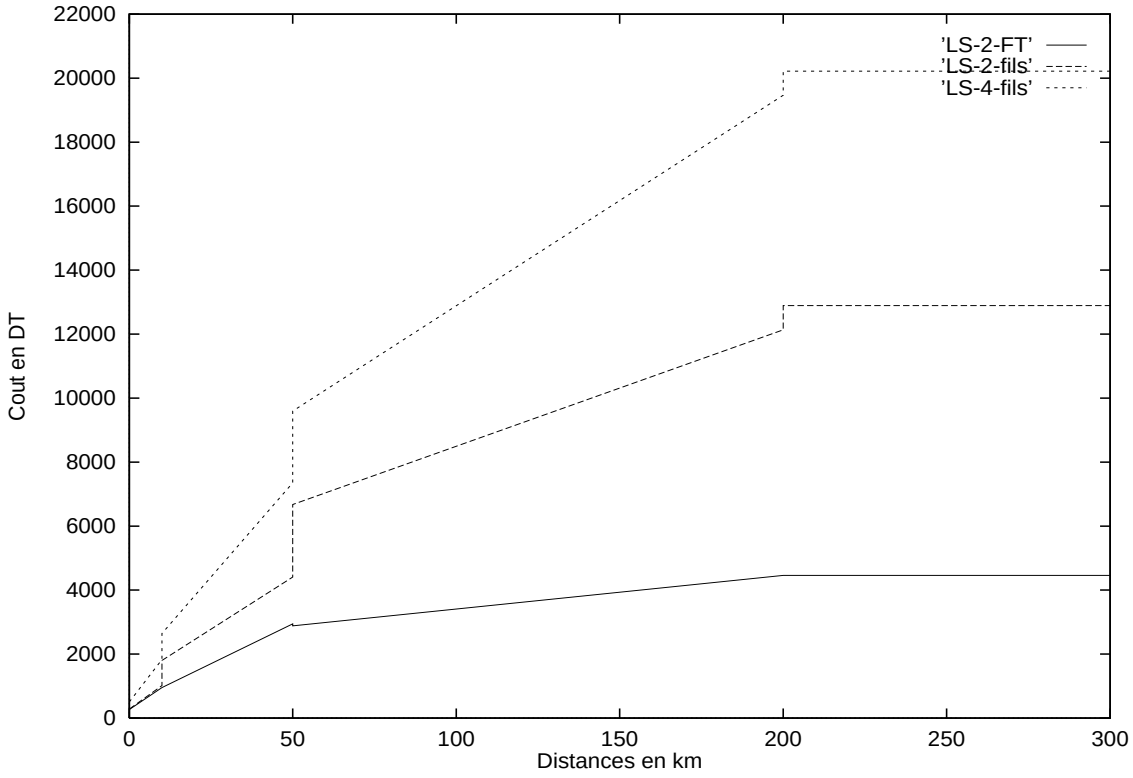


FIG. 2.1 – Tarification des lignes spécialisées en fonction de la distance

La méthode la plus simple pour tenir compte du coût réel des lignes consiste à évaluer l'approximation linéaire la plus proche de la courbe de la figure 2.2 et à appliquer les méthodes d'allocation de capacité linéaire pour calculer les débits à donner aux lignes. Ces débits sont ensuite arrondis aux valeurs discrètes normalisées (2400, 4800, 9600, ...) en vérifiant que ces dernières permettent encore d'écouler convenablement le trafic lorsqu'elles sont ajustées à une valeur inférieure à la valeur calculée.

En régime national (communications internes), la taxation de X25 se fait au volume de données transférées (par MO).

La formule 2.2 donne la procédure de calcul du coût des lignes louées X25.

0 8 1 1 0 0 0vitesse (bit /seconde)	Redevances mensuelles d'abonnement(DT)
1200	20
2400	20
4800	50
9600	90
14400	100
28800	120
64000	200

TAB. 2.2 – Tarification des lignes louées X25

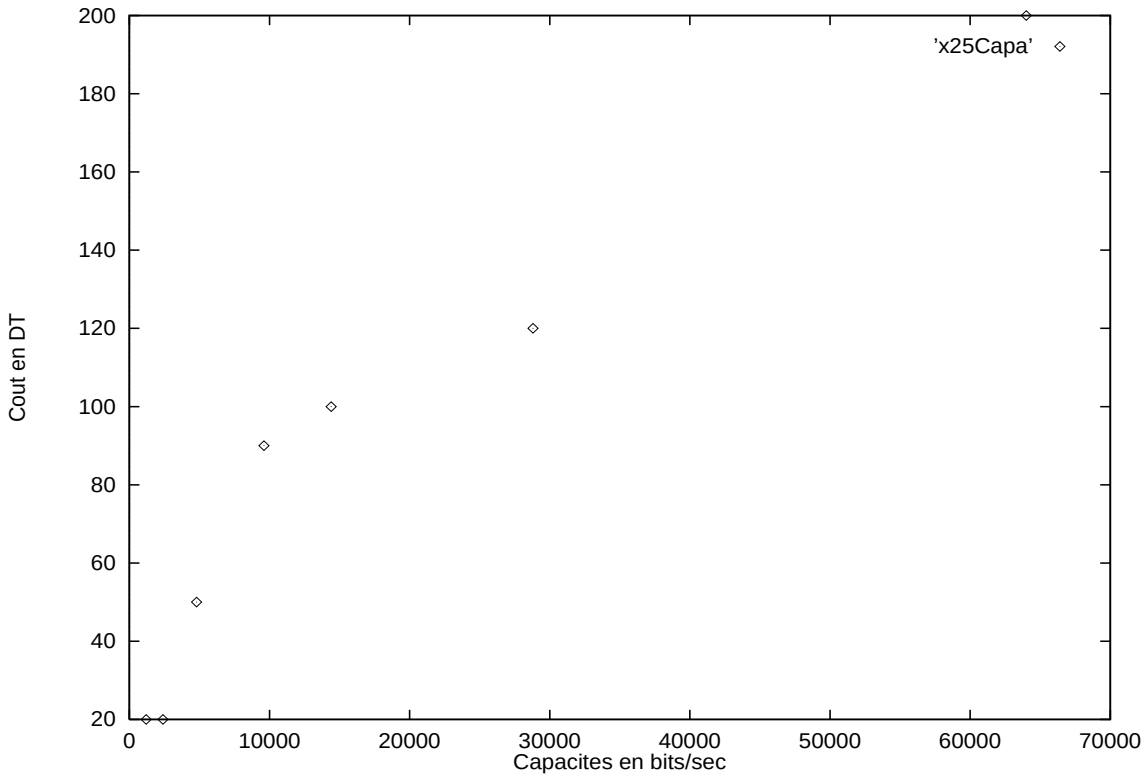


FIG. 2.2 – Tarification des lignes louées X25 en fonction des capacités

$$D = Redevance\,d'abonnement(DT) + Charge\,de\,base(DT) + Tarif/MO * Volume \quad (2.2)$$

La figure 2.3 illustre, pour 3 capacités particulières (1200 bits/s, 9600 bits/s et 64000 bits/s), la variation du coût total en fonction du volume de données. Ces courbes sont parallèles et affines par intervalles.

0 8 1 0 0 0 0Palier	Charge de base	Tarif par MO
entre 0 et 5 MO	0 DT	16 DT
entre 5 et 15 MO	20 DT	12 DT
entre 15 et 25 MO	30 DT	10 DT
au delà de 25 MO	0 DT	16 DT

TAB. 2.3 – Tarification X25 en fonction du volume de donnée

L'optimisation peut être conduite d'une manière plus fine en définissant une fonction continue dont l'allure générale est très proche de la fonction affine par intervalle. Dans ce cas, la méthode d'optimisation reste très proche de celle qui est utilisée avec une fonction de coût linéaire.

2.1.3 Accès RNIS

RNIS est l'acronyme du Réseau Numérique d'Intégration de Services. Pour ce type de raccordement, il faut acquérir une carte RNIS ou un adaptateur de terminal RNIS qu'il faudra relier à un équipement déjà connecté au réseau local ou un routeur dédié. Dans la majorité des cas, le fournisseur propose d'utiliser le protocole PPP pour transporter les paquets IP entre le réseau local du client et le sien.

L'accès RNIS offre un débit assez important pour un coût relativement faible. Une ligne RNIS inclut 2 canaux B à 64 Kbits/s et un canal D à 16 Kbits/s pour la signalisation. En Tunisie, le réseau numérique d'intégration de services TUNIRIS est en cours d'expansion et représente une évolution du réseau téléphonique actuel. Il propose la continuité numérique de bout en bout:

- en multipliant les débits de transmission actuels,
- en intégrant la voix, le texte, les donnés et l'image,
- en ouvrant la voie à l'intégration du système de communication (téléphonique, informatique et bureautique,...).

Ce type de réseau a plusieurs avantages. En effet, TUNIRIS améliore la communication interne et externe sans pour autant en bouleverser les structures. Il intègre les applications qui nécessitaient auparavant des réseaux distincts:

- on peut téléphoner et transférer des fichiers en même temps,
- on peut accéder à distance aux systèmes d'archivage électronique.

Les services TUNIRIS sont facturés mensuellement selon la formule 2.3:

$$D = RedevAbonn + NbAcc(64Kbits/s) * TaxeTeleph * 1.30 \quad (2.3)$$

La redevance d'abonnement actuelle est de *200* DT par mois.

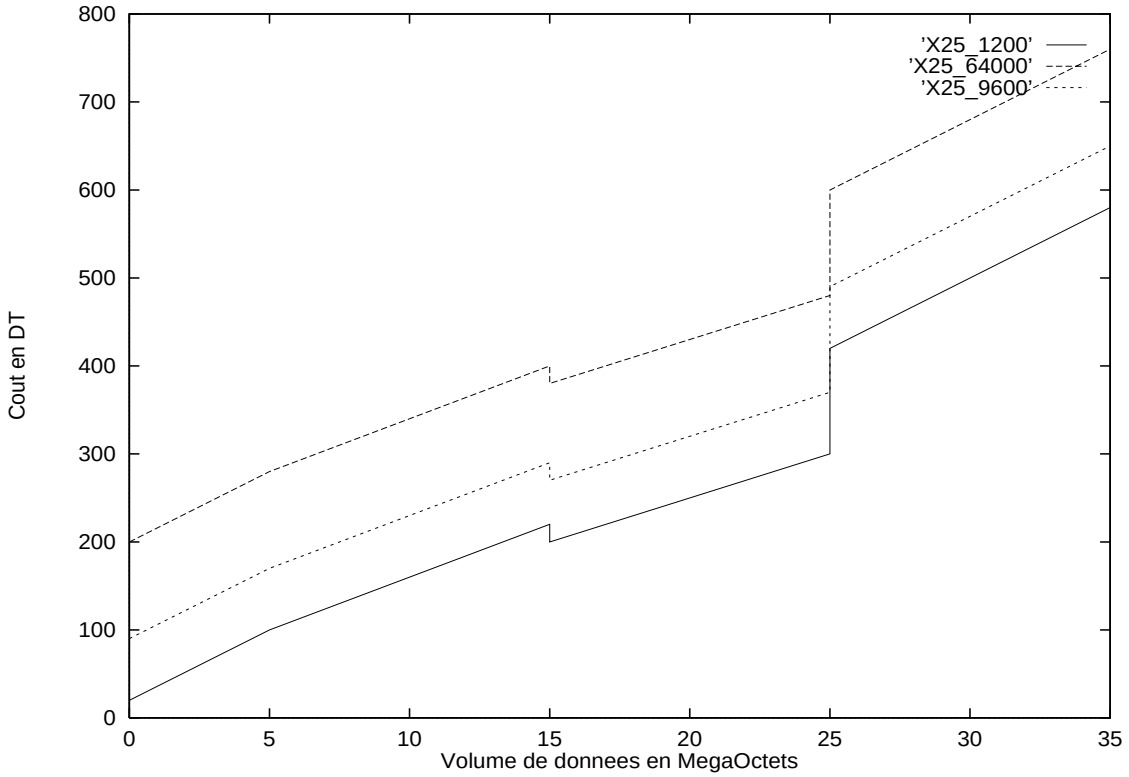


FIG. 2.3 – Tarification des lignes louées X25 en fonction du volume de données

2.1.4 Réseau Téléphonique Commuté (RTC)

Que faire si nous ne pouvons envisager un accès dédié, ni utiliser SLIP ou PPP? Dans ce cas, il nous faut obtenir un compte sur un ordinateur ayant accès à Internet et nous connecter à cet ordinateur pour effectuer nos commandes à distance en temps partagé. Ce type d'accès n'est pas aussi satisfaisant mais il est plus facile à mettre en œuvre et moins coûteux pour un utilisateur individuel. Notre ordinateur n'est pas vu de l'Internet, il accède juste à un ordinateur connecté de manière permanente au réseau. Dans la mesure où nous partageons la connexion avec d'autres, le coût du service est plus faible.

Cette solution élémentaire permettant l'accès au "réseau des réseaux" est la solution la plus répandue pour l'accès ponctuel ou individuel. Une petite société mettra facilement en œuvre cette solution qui se trouve être la moins onéreuse. Cette connexion permet de disposer d'une boite aux lettres, d'échanger des courriers électroniques, de consulter le web, même, selon le fournisseur d'accès (ISP: Internet Service Provider), de disposer de quelques pages HTML pour présenter ses activités...

2.2 Choix du réseau télécom

Le moyen d'accéder au "réseau des réseaux" différera selon que l'on veut doter les abonnés d'un e-mail, permettre la recherche ponctuelle d'informations sur le web ou télécharger des logiciels... Il sera également différent selon qu'il s'agit de mettre en œuvre un système de téléphonie, d'assurer une veille marketing, d'interconnecter un réseau local Intranet ou

non, d'ouvrir un serveur web institutionnel ou un site marchand important. En fonction de tout cela, on choisira entre: RTC, RNIS, une liaison spécialisée, X25 (cf. tableau **??**).

Outre ces critères de choix, il existe des qualités de service à ne pas négliger. Des qualités de services meilleures à coût élevé et vice versa. Dans l'absolu, il n'existe pas de réseau idéal, le choix du réseau télécom dépend des besoins de l'entreprise.

2.3 Conclusion

Les études faites dans le but de caractériser les coûts ne sont, en fait, qu'une étape préliminaire nécessaire pour les outils de conception et d'optimisation des réseaux. Nous pouvons dire que la caractérisation de la fonction coût-capacité reste difficile à réaliser. En effet, un bon modèle de coût doit être en mesure de suivre les évolutions continues des tarifs dont les seuls responsables sont les opérateurs télécom. Ces derniers doivent fournir des bases de données tarifaires et leurs évolutions. Cette condition contraignante rend la tâche de maîtrise des coûts, en temps réel, difficile à atteindre. Dans ce chapitre, nous avons étudié les technologies X25, LS, RTC, RNIS. D'autres moyens type ATM, Frame Relay, FDDI n'étaient pas considérées car ils ne sont pas offerts par Tunisie Télécom.

Chapitre 3

Problème d'allocation de capacité

Nous abordons dans ce chapitre le problème d'allocation de capacité, un parmi les problèmes les plus délicats en matière de conception de réseaux. Il consiste à sélectionner les capacités optimales parmi un ensemble discret. Plusieurs approches heuristiques peuvent être envisagées pour résoudre ce problème, cependant les résultats analytiques exactes sont en général rares. Après une formulation générale du problème, nous discutons trois types de solutions répondant successivement aux trois cas de figures de la nature de la fonction coût capacités : linéaire, concave et discrète.

3.1 Formulation du problème

Comme tout problème d'optimisation combinatoire, nous commençons par poser le problème d'allocation de capacité sous forme de: données, variables, objectif et contraintes:

Données:

- Topologie;
- Matrice de trafic externe;
- Politique de routage.

Variables:

- Capacités.

Objectif:

- minimiser le temps de réponse T (ou le coût D).

Contraintes:

- $T \leq T_{max}$ (ou $D \leq D_{max}$);
- $C_i \geq f_i$.

La première étape de cette optimisation consiste à déterminer les taux moyens des arrivées λ_i. Pour ce faire, il est nécessaire de connaître la politique de routage. À ce stade, nous supposons que le routage des messages est fixe et connu. Dans ces conditions, les données concernant le trafic à écouler par le réseau peuvent être résumées par la matrice de trafic Γ (cf. section 1.2.1).

Comme nous l'avons indiqué au deuxième chapitre, nous supposons que les arrivées des paquets forment un processus de Poisson et que les longueurs des paquets sont distribuées exponentiellement. Ces hypothèses sont assez bien vérifiées par l'expérience et sont nécessaires pour trouver une solution simple au problème.

Connaissant la matrice de trafic Γ et la politique de routage, il est trivial de déterminer le flux λ_i traversant la ligne M_i en appliquant la formule 1.8.

Après ces différentes opérations, nous connaissons pour chacune des M lignes du réseau le trafic moyen λ_i, ainsi que la longueur moyenne $\frac{1}{\mu}$ des paquets et **il faut maintenant déterminer le débit C_i qu'il faut prévoir sur chaque ligne pour minimiser le temps de réponse moyen $\overline{T}$ (ou bien le coût du réseau D) tout en garantissant un coût D inférieur à une limite donnée D_{max} (un temps de réponse moyen T inférieur à une limite donnée T_{max}).**

3.2 Cas où la fonction Coût-Capacité est linéaire

On commence par le cas le plus simple, à savoir lorsque la fonction $D_i(C_i)$ [1], variation du coût du canal en fonction de la capacité, est linéaire. C'est le cas par exemple des opérateurs télécom qui loueraient des lignes sur la base d'une taxe mensuelle fixe de raccordement d_{i0}, plus une taxe mensuelle d_i proportionnelle au débit. Avec cette hypothèse, le coût total D_i des lignes du réseau est donné par :

$$D_i(C_i) = d_i C_i + d_{i0} \tag{3.1}$$

Les deux problèmes cités ci dessus sont en réalité deux problèmes duaux. Nous choisissons à les traiter séparément :

3.2.1 Optimisation du temps de réponse

Rappelons que nous sommes toujours sous les mêmes hypothèses de **Kleinrock**. De ce faire, et d'après ce qui a été démontré dans la section 1.1.2, l'expression du temps de réponse moyen $\overline{T}$ est donnée par la formule 1.5.

L'optimisation de ce problème convexe [2], moyennant la méthode de Lagrange (cf. section A.3), donne les expressions optimales des C_i et $\overline{T}$ suivantes:

$$Ci = \frac{\lambda i}{\mu} + \frac{D_{max} - \sum_{i=1}^{M}(\lambda_i d_i/\mu + d_{i0})}{\sum_{i=1}^{M}\sqrt{di\lambda_i/\mu}}\sqrt{\frac{\lambda_i}{\mu d_i}} \tag{3.2}$$

$$\overline{T} = \frac{1}{\gamma}\sum_{i=1}^{M}\frac{\left(\sum_{i=1}^{M}\sqrt{\lambda_i d_i/\mu}^2\right)}{D_{max} - \sum_{i=1}^{M}(\lambda_i d_i/\mu + d_{i0})} \tag{3.3}$$

3.2.2 Optimisation du coût du réseau

La résolution de ce problème moyennant toujours la méthode de Lagrange (cf. section A.4), donne les expressions optimales des C_i et D suivantes:

1. On notera la fonction coût-capacité par $D_i(C_i)$.
2. T est la somme de fonctions convexes et l'espace des solutions réalisables est aussi convexe.

$$C_i = \frac{\lambda_i}{\mu} + \frac{\sum_{i=1}^{M} \sqrt{d_i \frac{\lambda_i}{\mu}}}{\gamma T_{max}} \sqrt{\frac{\lambda_i}{\mu d_i}} \tag{3.4}$$

$$D = \sum_{i=1}^{M} (\frac{\lambda_i d_i}{\mu} + d_{i0}) + \frac{\left(\sum_{i=1}^{M} \sqrt{di \frac{\lambda i}{\mu}}\right)^2}{\gamma T_{max}} \tag{3.5}$$

3.3 Cas où la fonction Coût-Capacité est concave

Supposons, dans ce cas ci, que la fonction coût-capacité est concave et croissante. L'espace

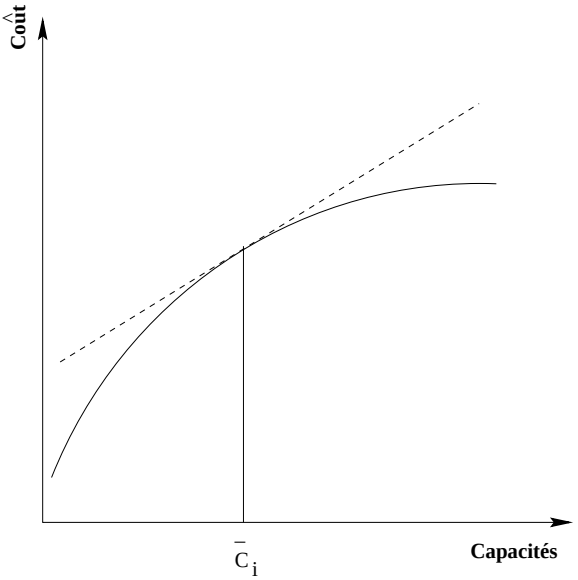


FIG. 3.1 – Fonction coût-capacité concave

des solutions réalisables dans ce cas est non convexe (cf. figure 3.1). Donc, il existe en général au moins un minimum local. Si l'on suppose, en plus que les fonctions $d_i(C_i)$ sont continues et différentiables pour $C_i \geq f_i$, les minima locaux sont caractérisés dans ce cas par les deux propriétés suivantes [GER 73]:

1. Si $\overline{C}$ est un minimum local pour le problème concave, alors il est aussi un minimum global pour le problème linéaire au voisinage de $\overline{C}$[3] (cf. figure 3.1).
2. Si C^n est une allocation faisable et C^{n+1} est une solution du problème linéaire autour de C^n, alors:

$$\overline{T}(C^{n+1}) \leq \overline{T}(C^n) \tag{3.6}$$

$$D_{con}(C^{n+1}) \leq D_{lin}(C^{n+1}) \leq D_{max} \tag{3.7}$$

Où $D_{con}(C^{n+1})$ est le coût calculé pour le problème concave et $D_{lin}(C^{n+1})$ est le coût correspondant au problème linéaire (cf. figure 3.2).

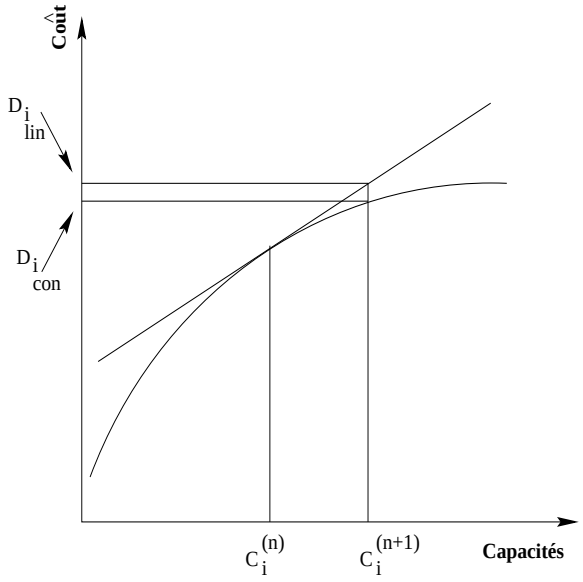


FIG. 3.2 – Linéarisation de la fonction concave

La relation (3.6) est immédiate du fait que C^{n+1} minimise le problème linéaire. La relation (3.7) résulte de la concavité de la fonction Coût-Capacité. Donc, étant donné C^n, la propriété 1 permet d'obtenir une allocation adéquate des capacités. Ces propriétés permettent d'énoncer l'algorithme de détermination des capacités optimales suivant:

Algorithme de CA pour fonction coût-capacité concave

1. Soit C^0 une allocation faisable (généralement prendre $C^0 = f$),
   $n \leftarrow 0$,
   $T_0 \leftarrow \infty$ ($D_0 \leftarrow \infty$),
2. Calculer C^{n+1} solution du problème linéaire au voisinage de C^n et calculer le temps $T_{n+1} \equiv T(C^{n+1})$ (le coût $D_{n+1} \equiv D(C^{n+1})$)
3. Si $\mid (T_{n+1} - T_n)/T_n \mid \leq \varepsilon$ ($\mid (Dn+1 - Dn)/Dn \mid \leq \varepsilon$), où ε est une tolérance choisie,
   alors sortir, et C_{n+1} est la solution minimale à ε près.
   Sinon $n \leftarrow n+1$ et aller à 2

De la seconde propriété, on déduit que la suite T_n est strictement décroissante et converge vers $\overline{T}$. L'algorithme précédent reste aussi valable pour le cas des fonctions coût-capacité non différentiables, continues, concaves et croissantes (cf. figure 3.3) [GER 73].

3.4 Cas où les capacités sont discrètes

On suppose que pour chaque ligne i, il existe K_i valeurs de capacités faisables, C_{i1}, C_{i2}, ...C_{iK_i} tels que $C_{i1} \leq C_{i2} \leq ... \leq C_{iK_i}$. On suppose également que pour chaque valeur C_{ik} est as-

3. Cette propriété est immédiate du fait qu'un déplacement élémentaire δC au voisinage de $\overline{C}$ est le même pour le problème linéaire que concave.

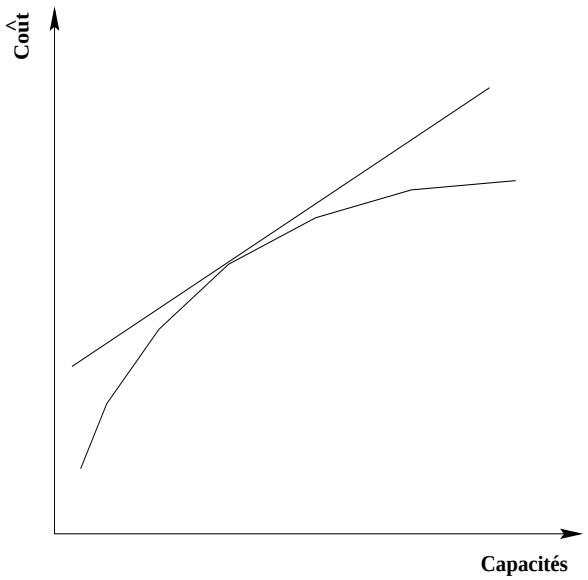


FIG. 3.3 – Cas où la fonction Coût-Capacité est non différentiable

socié un coût d_{ik} tels que $d_{i1} \leq d_{i2} \leq ... \leq d_{iK_i}$ (en général, les coûts augmentent avec les capacités).

Le problème d'allocation des capacités discrètes est un problème de théorie des nombres et ne peut être résolu que par les techniques de la programmation. Deux techniques sont connues à ce propos: la Programmation Dynamique et la Décomposition de Lagrange. Nous décrivons par la suite plutôt la deuxième, celle, utilisée pour l'optimisation non linéaire, sujet à des contraintes non linéaires.

3.4.1 Méthode de décomposition de Lagrange (LD)

La méthode LD est très efficace pour les problèmes séparables où l'optimisation d'un problème de dimension n peut être décomposée en n problèmes de dimension 1. Elle est souvent utilisée pour sa simplicité. Cependant, elle n'est pas valable, surtout si les problèmes sont non convexes. Le principe de cette méthode est le suivant [GER 73]:

$$\begin{array}{c} \min\ f(x) \\ sous\ contraintes\ g(x) \leq g_0 \end{array} \tag{3.8}$$

Où f et g sont deux fonctions arbitraires définies sur R^n. Le Lagrangien $L(x, \lambda)$ pour ce problème s'exprime

$$\begin{array}{c} L(x, \lambda) = f(x) + \lambda\,(g(x) - g_0) \\ avec\,\lambda\ \geq 0 \end{array} \tag{3.9}$$

Soit $\lambda = \lambda^*$, où λ^* est un multiplicateur positif et soit x^* le minimum de $L(x, \lambda^*)$, alors x^* est la solution du problème (3.8) vérifiant $g(x^*) = g_0$. Le problème (3.8) peut être, donc, résolu en minimisant $L(x, \lambda)$ pour différentes valeurs de λ jusqu'à avoir $g(x^*) = g_0$.

Ces résultats restent aussi applicables pour le cas des problèmes discrets et plus particulièrement, le cas d'allocation des capacités discrètes. Il existe, pour ce faire, un algorithme

très efficace, pour la détermination des solutions lagrangiennes, se basant sur la propriété suivante:

Supposons qu'à l'itération n, la solution non dominée [4] trouvée est:

$$C^{(n)} = C^{(n)}_{1,k_1},\ C^{(n)}_{2,k_2},\ ....,\ C^{(n)}_{M,k_M}$$

avec un coût D_n et un temps de réponse T_n. Pour passer à la prochaine solution C^{n+1} sur la courbe (T, D) de la figure 3.4, on considère pour chaque arc i le rapport défini par:

$$\rho_i^n = \frac{-(t^{(n)}_{i,k_i+1} - t^{(n)}_{i,k_i})}{(d^{(n)}_{i,k_i+1} - d^{(n)}_{i,k_i})} \tag{3.10}$$

et soit :

$$\rho_s = \max_i\ \rho_i \tag{3.11}$$

La solution C^{n+1} est déduite de C^n en faisant coulisser l'arc s, c'est à dire: $C^{(n+1)}_{s,K_s} \leftarrow C^{(n)}_{s,K_s+1}$et en laissant les autres arcs intacts.

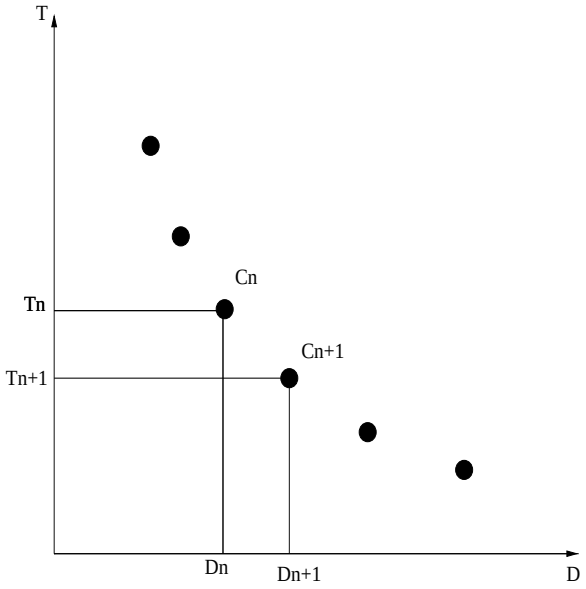


FIG. 3.4 – T en fonction de D

Cette propriété est valable uniquement pour les fonctions décroissantes, convexes $\forall i = 1, ..., M$ (rappelons que les coûts augmentent avec les capacités ce qui implique que les T_i sont décroissantes avec d_i). L'algorithme de la Décomposotion de Lagrange est le suivant:

Algorithme de la Décomposition de Lagrange

---

1. `Soient`

   - `n=0,`

4. (T_1, D_1)est dominé par (T_2, D_2) si $D_1 \geq D_2$ et $T_1 \geq T_2$.

– C^0 le minimum des capacités (c'est à dire $k_i = 1, \forall i = 1, ..., M$)

2. Soit s l'indice de l'arc de plus grand rapport $\rho_s^{(n)} = max\, \rho_i^{(n)}$ où :

$$\rho_i^{(n)} = \begin{cases} Comme\ celle\ qui\ est\ definie\ dans\ 4.11 & si\ k_i^{(n)} < K_i \\ 0 & si\ k_i^{(n)} = K_i \end{cases}$$

3. si $\rho_s^{(n)} = 0\ ou\ D \geq D_{max}$ alors sortir, la solution est trouvée.
Sinon, $k_i^{(n+1)} = \begin{cases} k_i^{(n)} & pour\ i \neq s \\ k_i^{(n)} + 1, & pour\ i = s \end{cases}$,
n ← n+1 et aller à 2.

Cet algorithme génère toutes les solutions de Lagrange. Si nous voulons résoudre un problème donné d'allocation de capacités discrètes sous contrainte $D \leq D_{max}$, nous effectuons la recherche d'un multiplicateur λ ($\lambda = \rho_s^{(n)}$) satisfaisant la contrainte de coût ($D = D_{max}$).

3.5 Conclusion

La méthode d'allocation de capacités optimales est l'un des problèmes les plus délicats rencontrés par les concepteurs de réseaux. La résolution de ce problème est intimement liée à la nature de la fonction Coût-Capacité $D_i(C_i)$. Quant aux coûts linéaires, les solutions exactes sont données par les équations A.11 et A.17 pour le cas d'optimisation du temps de réponse et par les équations A.20 et A.21 pour le cas d'optimisation du coût du réseau. Quant aux coûts logarithmiques (problème concave), une solution approchée est décrite par l'algorithme 4.4. Quant aux capacités discrètes, une méthode heuristique est donnée dans la section 3.4.

Chapitre 4

Problème d'allocation de flot

Comme nous avons mentionné dans le deuxième chapitre, nous ne traitons que le cas de la politique de routage déterministe non adaptative. Le but est donc de trouver un ensemble optimal de chemins qui minimise une fonction objective définie telle que le temps de transit moyen, le coût total...

Dans ce chapitre, nous discutons ce problème. Nous commençons par une formulation du problème: données, variables, objectif et contraintes. Nous décrivons, ensuite, la méthode de déviation de flux (*Flow Deviation*) qui est une méthode analytique donnant, en général, des solutions optimales dans le cas d'une politique de routage alternée, et sous-optimale dans celui d'une politique de routage fixe. Nous finissons ce chapitre par présenter des résultats expérimentaux issus de l'application de cette méthode sur le réseau national universitaire.

4.1 Le problème de routage

4.1.1 Formulation du problème

Etant donnée un réseau informatique représenté par N_i nœuds $(i = 1...N)$ et A_j arcs $(j = 1...M)$ (cf. figure 4.1). Dans ce réseau, il est demandé de router une quantité γ_{ij} de messages par seconde de N_i (nœud source) à N_j (nœud destination) pour toute paire de nœuds.

Le problème de routage consiste à obtenir l'ensemble optimal de chemins qui minimise la fonction objective en respectant certaines contraintes. Ce problème se formalise de la manière suivante:

Données:

- Matrice de trafic externe;
- Emplacements géographiques de nœuds;
- Les capacités de lignes.

Variables:

- flux $\{f\}$.

Objectif :

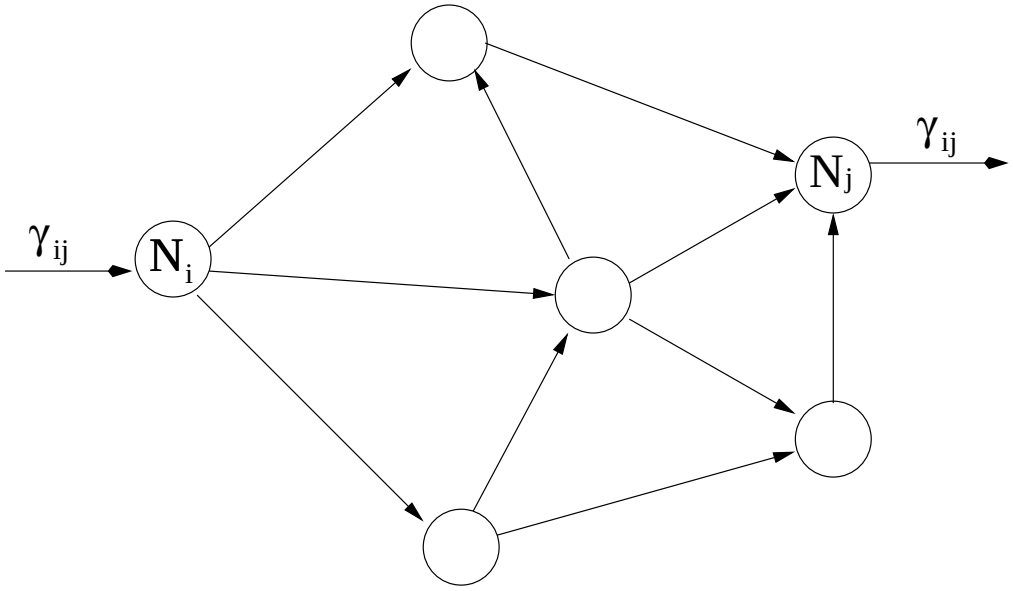


FIG. 4.1 – Flots dans un réseau informatique

– minimiser $T = \frac{1}{\gamma} \sum_{i=1}^{M} \frac{f_i}{C_i - f_i}$ avec $f = (f_1, f_2, ..., f_M)$ où $f_i = \lambda_i / \mu$.

Contraintes

1. f satisfait la matrice de trafic Γ
2. $f_i \leq C_i$

Le problème revient à déterminer l'ensemble F de solutions faisables. Cet ensemble peut être vu comme l'intersection de deux ensembles F_1et F_2 où

$$F_1 = \{f / f \text{ satisfait la contrainte } 1\}$$
$$F_2 = \{f / f \text{ satisfait la contrainte } 2\}$$

Nous pouvons démontrer que l'ensemble F des solutions réalisables est un ensemble convexe et qu'on peut se débarrasser de la deuxième contrainte (cf. section A.5) qui est introduite comme fonction de pénalité. Dans ce qui suit, nous développons la méthode de déviation de flux qui est une méthode efficace pour la résolution du problème non linéaire sans contraintes.

4.2 La méthode de déviation de flux

4.2.1 Description de la méthode

Considérons le problème d'allocation de flot cité ci-dessus, le temps de réponse moyen $T(f)$[1] donné par (1.6) est une fonction strictement *convexe* (somme de fonctions séparables convexes) et l'ensemble de solutions $F = F1 \cap F_2$ est un *polygone convexe*. D'où, si le problème est faisable, il existe un point stationnaire unique qui est le minimum global. Les contraintes

1. Comme les capacités sont données, la fonction coût est non significative et l'objectif sera de minimiser le temps de réponse moyen.

supplémentaires (contraintes de capacité et de flots) sont introduites dans $T(f)$ comme des fonctions de pénalité [GER 73]. Avec l'obtention d'une solution initiale faisable f_0, le problème peut être considéré comme un problème sans contraintes et par la suite elle se résout par la méthode de déviation de flux (*flow deviation*).

Cherchons maintenant si $T(f)$ satisfait les conditions de convergence [2] . Les dérivées première et seconde de *T* sont:

$$\frac{\delta T}{\delta f_i} = \frac{1}{\gamma} \frac{C_i}{(C_i - f_i)^2} \tag{4.1}$$

et

$$\frac{\delta^2 T}{\delta f_i \delta f_j} = \begin{cases} \frac{1}{\gamma} \frac{2C_i}{(C_i - f_i)^3} & si\ i = j \\ 0 & sinon \end{cases} \tag{4.2}$$

La solution optimale f^*, si elle existe (si le problème est faisable), satisfait les contraintes de capacités et par la suite,

$\forall f \in F_1\ on\ a$:

$$\frac{\delta T}{\delta f_i} > 0 \quad et \quad \frac{\delta T}{\delta f_i \delta f_j} < \infty \tag{4.3}$$

L'algorithme de déviation de flux appliqué au problème d'allocation de flot converge donc vers un minimum global. Cet algorithme, appliqué au problème de routage, se déroule en deux phases: il s'agit dans la première phase, de déterminer une solution initiale faisable (si elle existe sinon le problème est déclaré non faisable), dans la seconde phase le routage optimal est obtenu en partant de la solution initiale trouvée dans la première phase.

À la recherche d'une solution initiale faisable $f_0 \in F$, plusieurs méthodes sont applicables. La méthode que nous présentons a été décrite dans [KLE 76]. Elle consiste à choisir une solution $f \in F$, puis on fait varier les flots de tous les arcs par un facteur de décalage ρ jusqu'à l'obtention d'une solution faisable $f_0 = \rho * f \in F$ (f_0 satisfait la matrice de trafic $\Gamma_0 = \rho\Gamma$). La méthode de déviation de flux est par la suite, appliquée avec le flot f_0 comme flot initial et Γ_0 comme matrice du trafic. Après chaque itération, la valeur de ρ est augmentée pour être proche du point de saturation.

Le processus se répète jusqu'à ce que $\rho \geq 1$ et un flot initial faisable est obtenu ou que le réseau est saturé, $T(f)$ est minimum et $\rho < 1$. Dans ce dernier cas, le problème est non faisable avec les précisions utilisées.

Complexité de Flow Déviation:

La phase la plus gourmande en nombre d'opérations est celle de détermination de chemins les plus courts. En effet, la complexité de l'algorithme de **Dijkstra** [PRI 96] s'évalue en $O(N2)$ où N est le nombre de nœuds. La complexité globale de la méthode de déviation de flux est intimement liée à la précision souhaitée de temps de réponse et aux données du problème.

L'algorithme général est le suivant:

2.

1. $T(f) \geq 0 \quad \forall f \in F$
2. $T(f) - T(f') = 0 \Longrightarrow f\ est\ stationnaire$ où f' est le flux issu de la déviation du flux *f*.

Algorithme de déviation de flux

Première phase: détermination d'une solution initiale

1. Soient $\rho_0 = 1$, f^0 le flot obtenu avec les chemins les plus courts et où les distances sont $l_k = [\delta T/\delta f_k]_{f_k=0} = 1/\gamma C_k$ et $n = 0$
2. Soit $\sigma_n = \max_k(\frac{f_k^n}{C_k})$.
   Si $\sigma_n/\rho_n < 1$ alors $f_0 \longleftarrow f^n/\rho_n$ et aller à la phase 2.
   Si non $\rho_{n+1} = \rho_n(1 - \varepsilon(1 - \sigma_n))/\sigma_n$ où $0 < \varepsilon < 1$.
   Soit $g^{n+1} = f^n(\rho_{n+1}/\rho_n)$. Aller à 3
3. Soit $f^{n+1} = (1 - \lambda^n)g^{n+1} + \lambda^n V^n$.
   Où:
   - V^n est le vecteur flux obtenu par les plus courts chemins,
   - λ^n est la valeur qui minimise le temps de réponse (peût être déterminée par la méthode de Fibonacci [MIN 83]).
4. Si $n = 0$, aller à 6
5. si $|\sum_{k=1}^{M} l_k(v_k - g_k^{n+1})| < \theta \, et \, |\rho_{n+1} - \rho_n| < \delta$, où θ et δ sont des précisions bien choisies, et v et le flot calculé en utilisant les chemins les plus courts de g^{n+1}, alors arrêter: le problème est non faisable avec les tolérances θ et δ. Sinon aller à 6.
6. $n \longleftarrow n + 1$ et aller à 2.

Deuxième phase: détermination du routage optimal

1. Soit $n = 0$;
2. $f^{n+1} = (1 - \lambda^n)f^n + \lambda^n V^n$
3. si $\frac{T(f^n) - T(f^{n+1})}{T(f^n)} < \varepsilon$ alors arrêter sinon $n \leftarrow n + 1$, aller à 2

4.2.2 Représentation et calcul des tables de routage

Une politique de routage est représentée par un ensemble de tables de routage. La table au nœud i est une matrice $N * A_i \; de \, P^{(i)}(.,.)$ où N est le nombre total de nœuds et A_i est le nombre de nœuds adjacents au nœud i (cf. figure 4.2).

$P^{(i)}(k, j)$ est la fraction de trafic arrivée au nœud *i* et ayant pour destination le nœud *k* qui va être routée vers la ligne *j* (cf. figure 4.3).

Par définition on a :

$$\sum_{j=1}^{A_i} P^{(i)}(k, j) = 1 \; \forall i = 1...N \tag{4.4}$$

Pour calculer les éléments de la matrice de routage, il est nécessaire de laisser des traces sur les flots trouvés lors de l'exécution de l'algorithme de déviation de flux pour chaque destination. Notons par $f_m^{(k)}$ le flot moyen sur la ligne m dû aux messages destinés au nœud k. À partir de ces flots, les éléments des tables de routage sont calculés par la formule 4.5:

	1	2		Ai
k	$P^{(i)}_{(k,1)}$	$P^{(i)}_{(k,2)}$	.....	$P^{(i)}_{(k,Ai)}$

FIG. 4.2 – Table de routage du ni

$$P^{(i)}_{(k,j)} = \frac{f^{(k)}_{i,j}}{\sum_{l=1}^{Ai} f^{(k)}_{i,l}} \tag{4.5}$$

Où (i, j) indique le lien direct entre les nœuds i et j et A_i est le nombre de nœuds adjacents au nœud i.

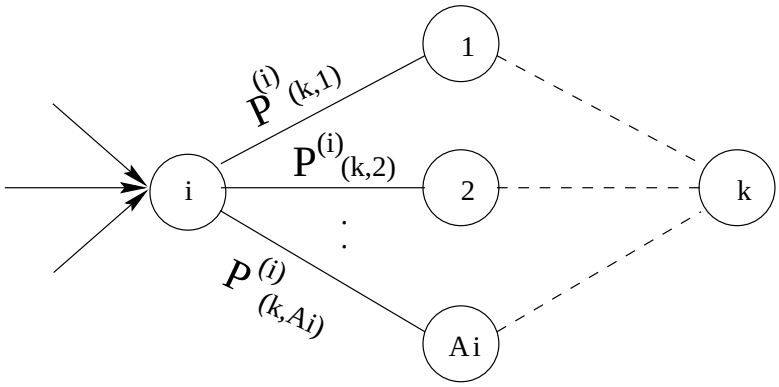


FIG. 4.3 – Le routage alterné de paquets

4.2.3 Résultats expérimentaux

Dans cette section, nous donnons un exemple d'application de la méthode de **déviation de flux** dans le cas de résolution du problème de routage. Le réseau choisi est le relais de nord du Réseau National Universitaire. Les données du problème sont:

- Nombre de nœuds:12;
- Topologie de connectivité 3 (cf. figure 4.4);
- Nombre de liens: 18;
- Capacités initiales (cf. tableau 4.2);

- Valeurs de ε pour la deuxième phase de FD: 10^{-4};
- Nombre d'itérations pour la recherche d'une solution initiale: 3;

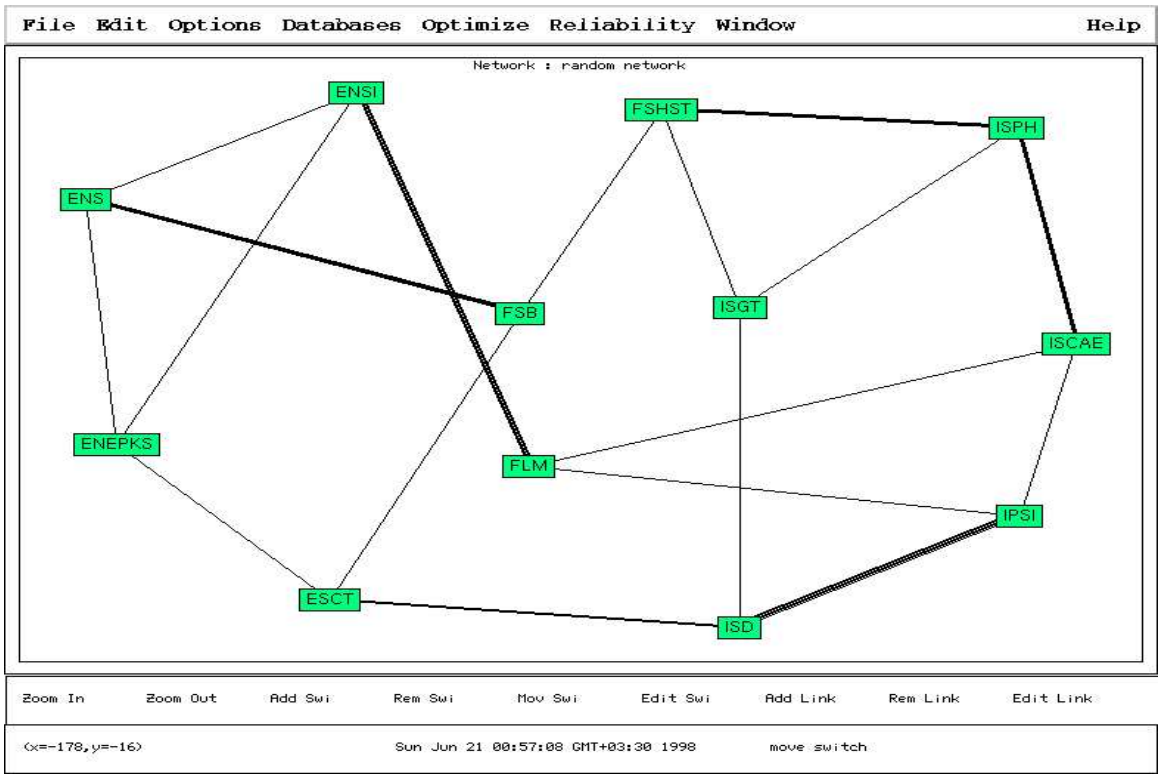


FIG. 4.4 – Topologie maillée du relais du nord du RNU

0 8 1 0 0 0 00.0	0.03	0.03	0.36	0.02	0.30	0.35	0.24	0.16	0.21	0.07	0.17
	0.0	0.36	0.25	0.22	0.02	0.20	0.19	0.15	0.09	0.22	0.13
		0.0	0.21	0.30	0.06	0.17	0.27	0.36	0.01	0.05	0.29
			0.0	0.01	0.21	0.29	0.35	0.12	0.04	0.18	0.32
				0.0	0.18	0.32	0.15	0.14	0.33	0.31	0.00
					0.0	0.13	0.22	0.07	0.10	0.35	0.01
						0.0	0.24	0.35	0.12	0.17	0.29
							0.0	0.20	0.01	0.14	0.12
								0.0	0.14	0.28	0.17
									0.0	0.22	0.13
										0.0	0.33
											0.0

TAB. 4.1 – Matrice de trafic symétrique: générée aléatoirement (en paquets par seconde)

0 8 1 0 0 0 0N° Lien	Lien	Capacité	Flux Optimal
0	ENSI-FLM	14400	5459.95
1	ISPH-ISCAE	4800	0.0
2	FSHT-ISPH	64000	7647.42
3	ENS-FSB	2400	1072.82
4	FSB-ESCT	2400	1707.03
5	ISD-ISGT	1200	768.04
6	FLM-ISCAE	9600	2904.36
7	FLM-IPSI	28800	2399.13
8	ISCAE-IPSI	14400	5985.44
9	ESCT-ISD	14400	5409.28
10	ENEPKS-ESCT	28800	11108.96
11	ISD-IPSI	64000	5194.52
12	ENSI-ENEPKS	9600	8154.06
13	ENS-ENEPKS	28800	4614.05
14	ENSI-ENS	2400	1762.31
15	ENS-FSB	14400	3267.99
16	FSHT-ISGT	14400	10616.73
17	FSB-ISGT	14400	2180.64

TAB. 4.2 – Les vecteurs de capacités et de flux

- Nombre d'itérations exécutées par l'algorithme FD: 10;
- Les algorithmes sont implémentés en Java et ils sont exécutés sur un PC 120 MkZ.

Nous remarquons qu'au bout de 10 itérations il y a stabilisation du temps de réponse. Ceci explique la rapidité de convergence de la méthode de déviation de flux.

Etant donné que nous appliquons une politique de routage déterministe alternée, la table de routage du nœud FSB (cf. tableau 4.3) est une matrice montrant les probabilités de router les messages entrant vers chaque ligne de sortie. Notons aussi que la somme de probabilités de chaque ligne du tableau est égale à 1.

Le tableau 4.2 montre les vecteurs de capacités et de flux qui ont donné le temps de réponse optimal. Nous remarquons que la contrainte de capacité est bien vérifiée par tous les liens du réseau ($\forall\, la\ ligne\ i\ f_i < C_i$).

4.3 Conclusion

Ce chapitre a traité le problème d'allocation de flot. Nous avons présenté principalement la méthode de **déviation de flux** appliquée à une politique de routage fixe. Nous signalons que la rapidité de convergence de cet algorithme est fortement liée aux données initiales, autrement dit, à la solution initiale faisable trouvée.

A ce stade, nous avons évoqué séparément deux sous-problèmes: allocation de capacité et allocation de flot. Nous avons détaillé des méthodes efficaces pour la résolution de chaque problème. En considérant les deux problèmes conjointement, un problème d'allocation de capacité et de flot se pose. Ce problème ainsi que des éléments de solutions y sont associés feront l'objet du chapitre suivant.

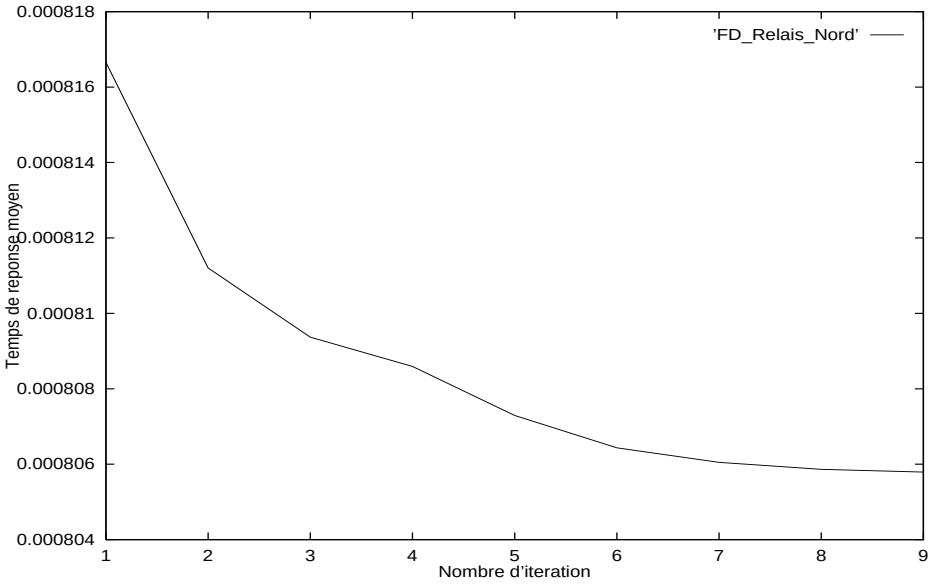


FIG. 4.5 – Convergence du temps de réponse pour le relais **Nord** du RNU

0 8 1 0 0 0 0Nœud Dest\Lien de sortie	0	1	2
0	0.086	0.513	0.399
1	0.020	0.440	0.539
2	0.022	0.449	0.527
3	0.028	0.424	0.547
4	0.038	0.561	0.399
5	0.023	0.450	0.526
6	0.0	0.0	0.0
7	0.097	0.485	0.417
8	0.099	0.506	0.394
9	0.016	0.453	0.529
10	0.020	0.449	0.529
11	0.072	0.500	0.427

TAB. 4.3 – Table de routage optimale probabiliste au nFSB

Chapitre 5

Problème d'allocation de capacité et de flot

Dans le quatrième et cinquième chapitre, nous avons présenté des méthodes pour la résolution exacte du problème d'allocation de capacité et le problème d'allocation de flot séparément. Ce chapitre est consacré au problème d'allocation de capacité et de flot simultanément (CFA). Les solutions développées sont, en fait, sous optimales, car il n'existe pas des méthodes efficaces donnant des solutions exactes et ceci est du en fait aux difficultés cruciales présentées par les fonctions objectives non convexes (plusieurs minima locaux), les capacités discrètes,...

Après une formulation du problème, nous discutons quelques propriétés des minima locaux puis nous introduisons un algorithme général pour la détermination de ces optima. Ensuite, nous passons, en revue, le cas de fonction coût-capacité linéaire pour montrer la pluralité des minima locaux. Enfin, nous donnons quelques résultats comme application de cet algorithme au RNU.

5.1 Formulation du problème

Données:

- Topologie;
- Matrice de trafics Γ;
- Fonction Coût-Capacité.

Variables:

- Capacités$\{C_i\}$
- flots $\{f_i\}$

Objectif:

- Minimiser le coût $D = \sum_{i=1}^{M} d_i(C_i)$

Contraintes:

- Contraintes de flux:($f \geq 0$ et la conservation du flux à chaque nœud);

– contrainte de faisabilité: ($\widetilde{f} \leq \widetilde{C}$)
– $T(\widetilde{f}, \widetilde{C}) = \frac{1}{\gamma} \sum_{i=1}^{M} f_i(\frac{1}{C_i - f_i}) \leq T_{max}$

Dans ce chapitre, nous choisissons d'optimiser le coût du réseau D car le temps de réponse est un critère de performance trop sensible [GER 73].

Ce problème est trop complexe comparé au problème de routage, car l'objectif à minimiser est généralement non convexe. De même, pour la contrainte de temps qui est aussi non convexe. D'où la multiplicité des minima locaux. Nous décrivons dans la section qui suit, quelques caractéristiques de ces minima locaux.

5.2 Propriétés des minima locaux: Algorithme général pour leur détermination

Nous supposons que les fonctions Coût-Capacité sont continues et strictement croissantes.

Théorème[GER 73]:

(f^*, C^*) est un minimum local si et seulement si:

1. f^*et C^* sont faisables (f^* respecte les contraintes de conservation de flux et $f^* \leq C^*$);
2. $T(f^*, C^*) = T_{max}$;
3. f^* minimise $T(f, C^*)$;
4. C^* minimise $D(C)$ sous contrainte $T(f^*, C) \leq T_{max}$.

L'approche utilisée pour trouver ces minima locaux est de commencer par un flux faisable f_0, calculer l'allocation des capacités optimales, exécuter l'algorithme de déviation de flux pour trouver le flot optimal, réexécuter de nouveau le problème d'allocation de capacité (CA) pour ces nouveaux flux et on itère entre CA et FA jusqu'à atteinte d'une solution sous-optimale.

D'où, l'algorithme suivant pour la détermination des minima locaux:

Algorithme d'allocation de capacité et de flot

Partons de (f^0, C^0) solution faisable initiale.
Soient $D_0 = Coût(C^0)$ le coût initial et $n \leftarrow 0$ le nombre d'itérations.

1. Soit $T(f^{n+1}, C^n) = \min_f T(f, C^n)$ (Application de l'algorithme FD).
2. Soit $D_{n+1} = D(C^{n+1}) = min\, D(C)$ sous contrainte $T(f^{n+1}, C) \leq T_{max}$.
3. Si $|\frac{D_n - D_{n+1}}{D_n}| \leq \varepsilon$, où ε est une précision à choisir, alors arrêter et (f^{n+1}, C^{n+1}) est un minimum local à ε près. Sinon $n \leftarrow n+1$ et aller à 1.

Remarquons que l'étape 1, fait appel au problème d'allocation de flot traité dans le quatrième chapitre et l'étape 2 fait appel au problème d'allocation de capacité traité dans le troisième chapitre. Remarquons également, que la suite (f^n, C^n) est convergente vers la limite (f^*, C^*).

Pour obtenir une solution initiale faisable *(f°,C°)*, nous commençons par déterminer le vecteur flot f° en appliquant le routage par les plus courts chemins (les longueurs des lignes

du réseau sont affectées aléatoirement ce qui garant la variation du flux initial d'une itération à une autre). Quant au vecteur capacité C° est généré aléatoirement sous la contrainte de capacité. Ainsi, C° varie d'une itération à une autre.

Cet algorithme peut être appliqué à n'importe quel problème à fonctions coût-capacité continues et strictement croissantes.

5.3 Fonction Coût-Capacité linéaire

Il a été démontré dans la section 3.2.2 que les capacités optimales étaient fonction de flux ($\frac{\lambda}{\mu}$). Plus précisément les équations:

$$C_i = \frac{\lambda_i}{\mu} + \frac{\sum_{i=1}^{M} \sqrt{d_i \frac{\lambda_i}{\mu}}}{\gamma T_{max}} \sqrt{\frac{\lambda_i}{\mu d_i}} \tag{5.1}$$

et

$$D = \sum_{i=1}^{M} (\frac{\lambda_i d_i}{\mu} + d_{i0}) + \frac{\left(\sum_{i=1}^{M} \sqrt{di \frac{\lambda i}{\mu}}\right)^2}{\gamma T_{max}} \tag{5.2}$$

où C_i et D sont respectivement la capacité optimale accordée à la ligne i et le coût optimal du réseau. Si nous introduisons ces expressions dans le problème 5.1, nous obtenons une nouvelle formulation du problème:

Données:

- Topologie;
- Matrice de trafics Γ;
- Fonction Coût-Capacité.

Variables:

- Capacités $\{C_i\}$
- flots $\left\{f_i = \frac{\lambda_i}{\mu}\right\}$

Objectif:

- Minimiser le coût:

$$D(\frac{\lambda_i}{\mu}) = \sum_{i=1}^{M} (\frac{\lambda_i d_i}{\mu} + d_{i0}) + \frac{\left(\sum_{i=1}^{M} \sqrt{di \frac{\lambda i}{\mu}}\right)^2}{\gamma T_{max}} \tag{5.3}$$

Contraintes:

- Contraintes de flux:($f \geq 0$ et la conservation du flux à chaque nœud);
- contrainte de faisabilité: ($\widetilde{f} \leq \widetilde{C}$)
- $T(\widetilde{f}, \widetilde{C}) = \frac{1}{\gamma} \sum_{i=1}^{M} f_i(\frac{1}{C_i - f_i}) \leq T_{max}$

Ce nouveau problème est non linéaire.

5.4 Résultats expérimentaux

Nous supposons que les coûts varient linéairement en fonction des capacités. Nous appliquons l'algorithme d'allocation de capacité et de flot pour le cas des fonctions coût-capacité linéaires.

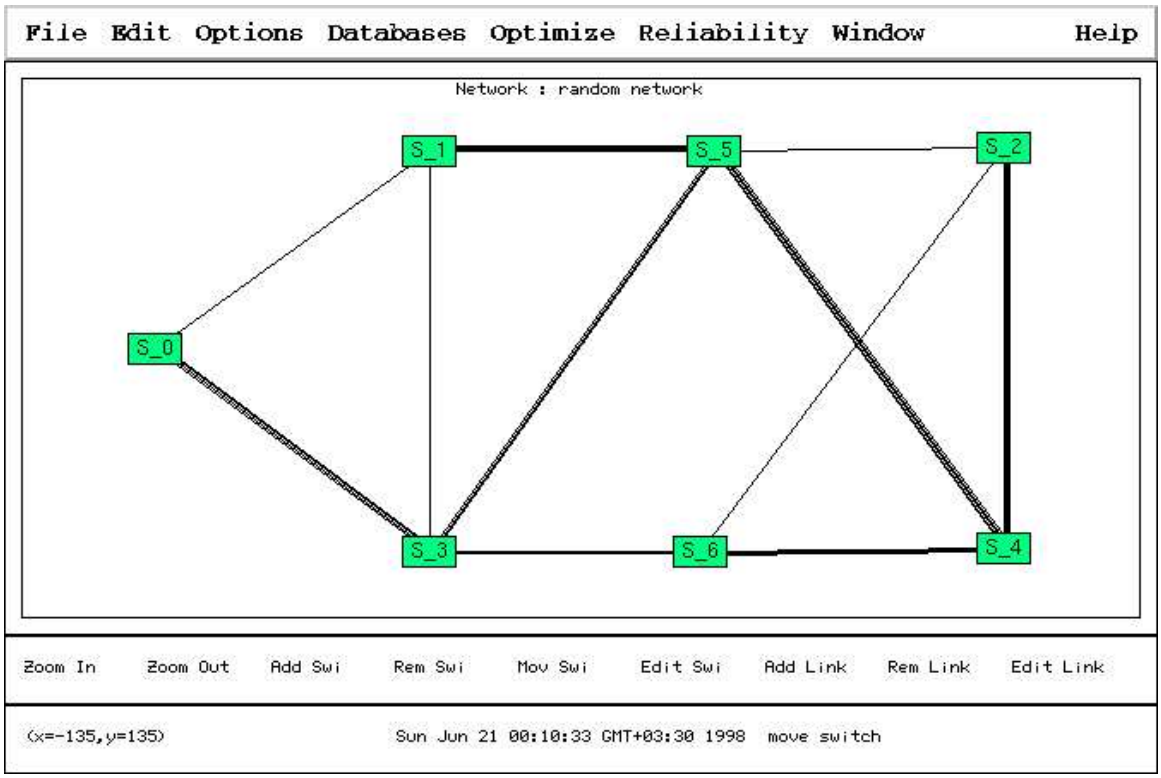


FIG. 5.1 – Topologie d'un réseau maillé

La figure 5.2 illustre la variation du coût du réseau de la figure 5.1 (un réseau maillé de connectivité 3 et ayant 7 nœuds) en fonction du nombre d'itérations exécutées. En fait, à chaque itération nous obtenons un minimum local qui dépend de la solution initiale choisie.

5.5 Conclusion

Malgré l'inexistence de méthodes de résolution exactes donnant des solutions optimales, l'heuristique de recherche itérative présentée dans ce chapitre est performante en temps d'exécution et en nombre d'itérations exécutées à la recherche d'une solution sous-optimale. En effet, pour le cas de la topolgie de la figure 5.1, le temps d'exécution sur un PC pentium 120 MkZ est de l'ordre de **2** minutes pour 100 itérations. La non convexité de la fonction objective et la discrétisation des niveaux de capacité rendent le problème plus délicat à résoudre.

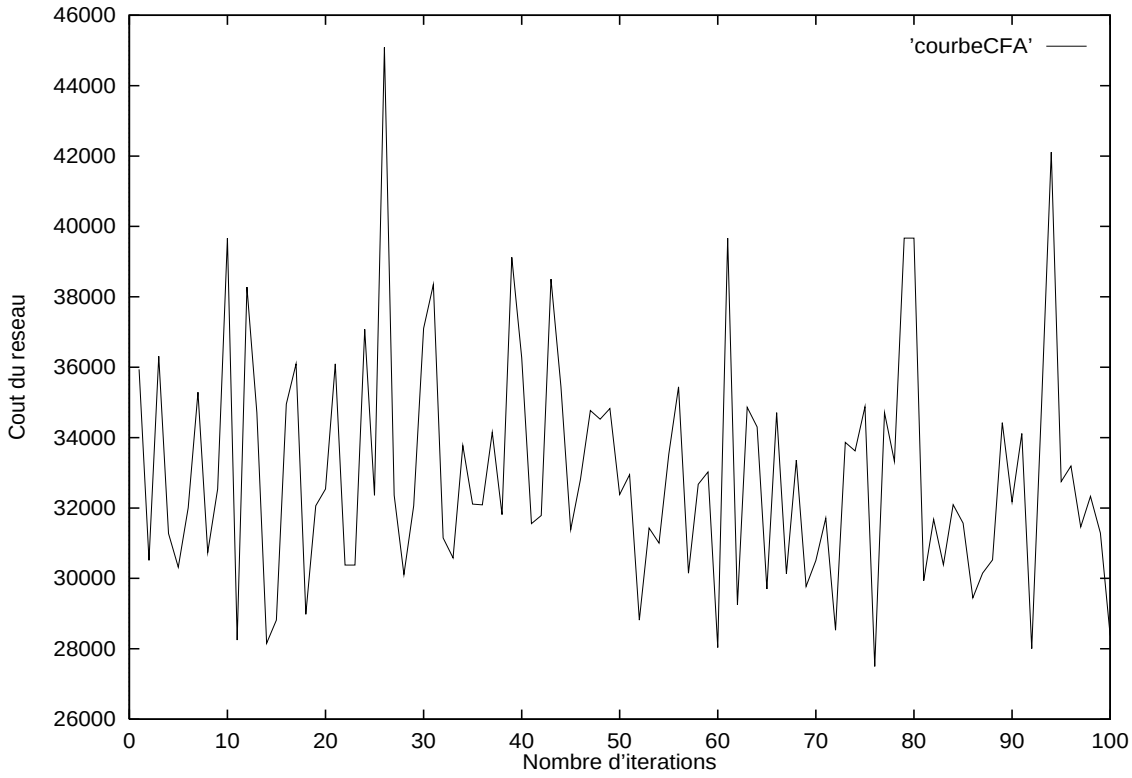


FIG. 5.2 – Variation du coût en fonction du nombre d'itérations

Chapitre 6

Problème général d'allocation

Pour un réseau de transit ayant une topologie **fixée**, les techniques de routage permettent de déterminer les chemins suivis par les différents paquets, et les méthodes d'allocation de capacité permettent de choisir les débits des liaisons en fonction de critères de coûts et de délais. Dans la plupart des cas, la topologie n'est pas imposée à priori et le problème à résoudre consiste à déterminer la meilleure topologie possible pour écouler le trafic du réseau.

Dans ce chapitre, nous décrivons les algorithmes **Recuit Simulé**, **Chandy Russel** et **Cheapest Star** utilisés respectivement pour la résolution des problèmes de topologie: maillée, multipoint et en étoile. Nous donnons pour chaque algorithme quelques résultats expérimentaux. À la fin, nous donnons l'algorithme de **Kleitman** utilisé pour l'étude de la fiabilité d'un réseau informatique.

6.1 Formulation du problème

Le problème d'allocation de la topologie peut être formulé de la manière suivante:

Données:

- Matrice de trafic;
- Localisation des nœuds.

Objectif:

- Minimiser le coût D.

Variables:

- La topologie;
- Les capacités C_i;
- Matrice(s) de routage .

Contraintes:

- Les contraintes de topologie;
- Les contraintes de capacités;
- Les contraintes de flux.

6.2 Conception de la topologie

6.2.1 Topologie maillée

Véritable problème combinatoire, le problème général de conception est le plus difficile à résoudre. En effet, il n'existe pas de méthodes analytiques exactes qui donnent des solutions optimales. Cependant, il existe des méthodes heuristiques de résolution décrites dans la littérature des réseaux dont la plus fréquemment utilisée est celle du Recuit Simulé [PRI 96]. C'est une méthode de recherche globale basée sur la comparaison des minima locaux. Nous donnons dans le paragraphe suivant le squelette de la méthode du Recuit Simulé.

La méthode de Recuit Simulé

Algorithme du recuit simulé

On tire au sort une transformation (c'est à dire une solution s' de $V(s)$) au lieu de chercher la meilleure ou la première solution améliorante;

1. On construit la solution résultante s' et sa variation de coût $\delta_f = f(s') - f(s)$;
2. Si $\delta_f \leq 0$, le coût diminue et on effectue la transformation améliorante comme dans la recherche locale ($s \leftarrow s'$) ;
3. Si $\delta_f > 0$, le coût remonte, c'est un rebond qu'on va pénaliser d'autant plus que la température est basse et que δ_f est grand, une fonction exponentielle a les propriétés désirées. On calcule les probabilités d'acceptation $a = \exp(-\delta_f/\theta)$, puis on tire au sort p dans $[0, 1]$, si $p \leq a$, la transformation est déclarée acceptée bien qu'elle dégrade le coût, et on fait $s \leftarrow s'$, sinon, la transformation est rejetée et on conserve s pour l'itération suivante ;
4. Pour assurer la convergence, θ est diminué lentement à chaque itération, par exemple $\theta = \alpha.\theta$, $\alpha < 1$ mais proche de 1;
5. On s'arrête quand θ atteint un seuil ε fixé, proche de 0.

Le recuit simulé (Simulated annealing) a été inventé par les physiciens Kirpatrick, Gelatt et Vectri en 1983 [SIA 96]. Le recuit simulé s'inspire des méthodes de simulation de Métropolis en mécanique statistique. Le recuit simulé en optimisation combinatoire n'a plus qu'un lointain rapport avec la thermodynamique. L'énergie du système est représenté par un réel arbitraire θ, la température.

À partir d'une recherche locale quelconque pour un problème, on obtient une méthode de recuit comme suit (on part toujours d'une solution réalisable initiale S).

L'algorithme du recuit simulé diffère d'une application à une autre par l'étape de définition de voisinage (étape 1), ainsi que par l'étape de construction de la solution S (étape 2). Ces deux étapes se présentent ainsi pour le cas du problème de topologie maillée:

Définition du voisinage (V(s))

On appelle *voisinage* d'une solution *S*, noté *V(S)*, l'ensemble des solutions que l'on peut obtenir en appliquant une transformation élémentaire à *S*.

Il existe, en fait, 3 types de structures de voisinage possibles pour le problème de conception de la topologie maillée optimale: voisinage Shift, voisinage 2-opt et voisinage 3-opt (cf. [OUR 98]). Le choix du voisinage optimal nécessite des comparaisons poussées des résultats issus de l'exécution de l'heuristique sur diverses données du problème.

Description de l'algorithme générant la solution résultante s'

L'algorithme consiste à générer une topologie initiale, vérifiant les contraintes de connectivité et du temps de réponse, à partir du voisinage de la solution précédente. On y applique l'algorithme CFA pour minimiser le coût ou le délais. Ensuite, une séquence d'actions: perturbation, allocation de capacité, allocation de flot sont appliquées à ce réseau en vue d'améliorer ses performances. On itère jusqu'à atteinte du budget maximum. La solution finale, ainsi obtenue, est sous optimale (cf. figure 6.1).

Description de la méthode de perturbation: échange de lignes (Branch X-Change) [NUS 87]

Sous sa forme la plus simple, cette méthode de perturbation procède par échange de lignes (*Branch X-Change*), et consiste à sélectionner deux liaisons voisins S_1S_2 et S_3S_4 qui mettent en jeu quatre nœuds distincts qui sont ici S_1, S_2, S_3, S_4, dans le cas de la figure 6.2. Les deux liaisons S_1S_2 et S_3S_4 sont ensuite supprimées et remplacées par deux autres liaisons, ici S_1S_3 et S_2S_4, qui mettent en jeu les quatre même nœuds. Le coût du réseau ainsi modifié est évalué en appliquant les méthodes d'allocation de flot et de capacité, et une nouvelle topologie remplace la précédente si son coût est moindre. L'algorithme d'échange s'applique aux liaisons de coût élevé et à faible débit, ainsi que dans le cas où l'introduction d'une liaison directe entre deux nœuds est justifiée par l'importance du trafic direct entre ces deux nœuds.

Résultats expérimentaux

- Nombre de nœuds: 12;
- Budget maximum mis en disposition: $10^6 DT$;
- Topologie optimale (cf. figure 6.3).

6.2.2 Topologie multipoint

Pour des réseaux où la fiabilité n'est pas un besoin contraignant, une topologie multipoint "spanning tree" peut être mieux appropriée. En utilisant cette topologie, il est évident d'allouer des capacités relativement grandes uniquement sur quelques liaisons, dans l'espoir de réduire le temps de réponse moyen des paquets.

Pour la conception de telle topologie, on associe à chaque liaison $(i\,,\,j)$, un poids $p(i\,,\,j)$ qui est le coût de cette liaison et on utilise l'algorithme de l'arbre de recouvrement minimal de Kruskal (ARM) (cf. section A.6). L'inconvénient majeur de cet algorithme est la contrainte de flux qui doit être véhiculé via une liaison et qui ne devra pas excéder la capacité permise. A cause de cette contrainte, le problème multipoint a un aspect combinatoire. C'est pourquoi nous avions recours à une méthode de type Branch and Bound (Chandy Russel).

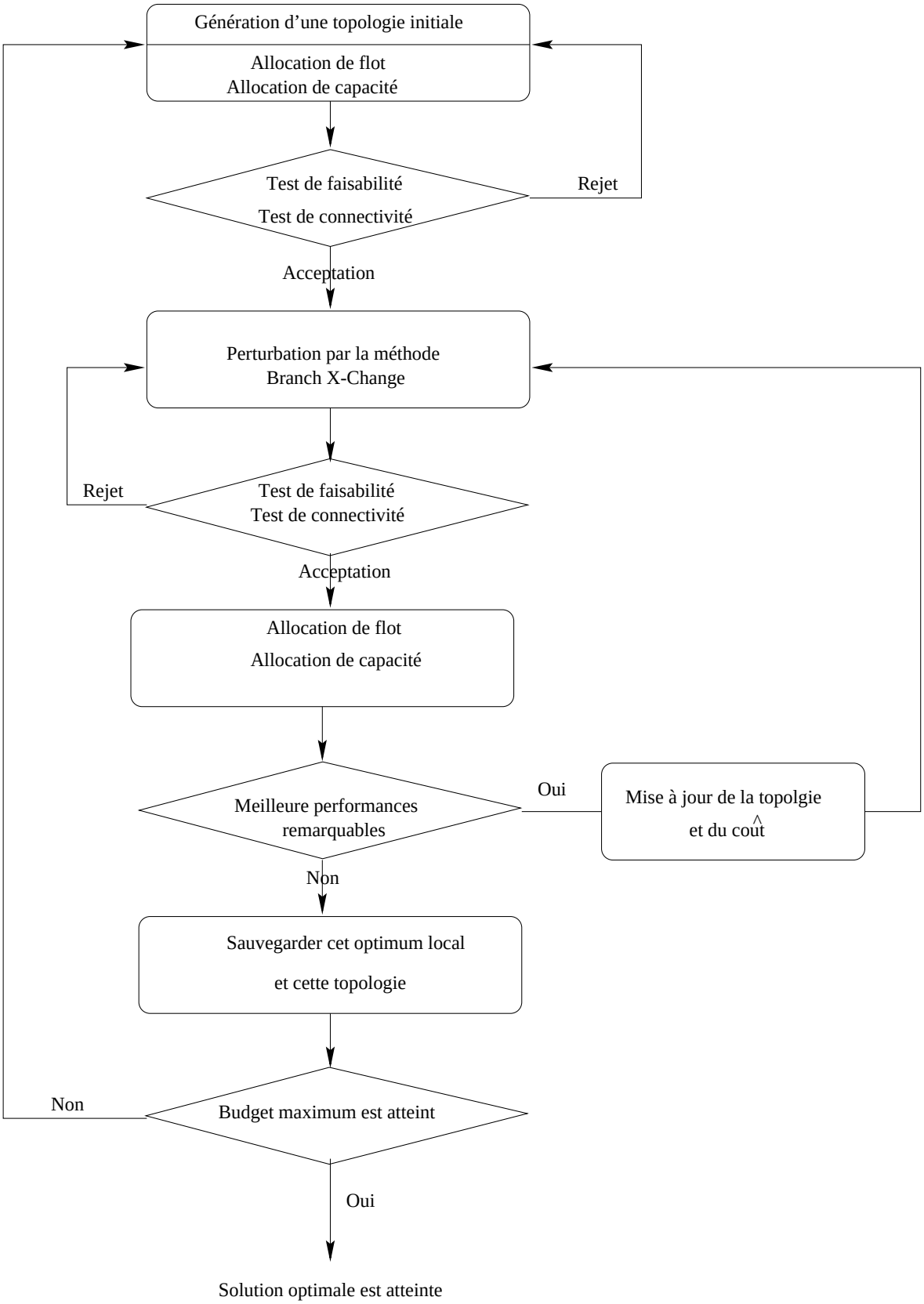


FIG. 6.1 – Etape de la détermination d'une solution résultante faisable (2^{eme} étape de l'algorithme du recuit simulé)

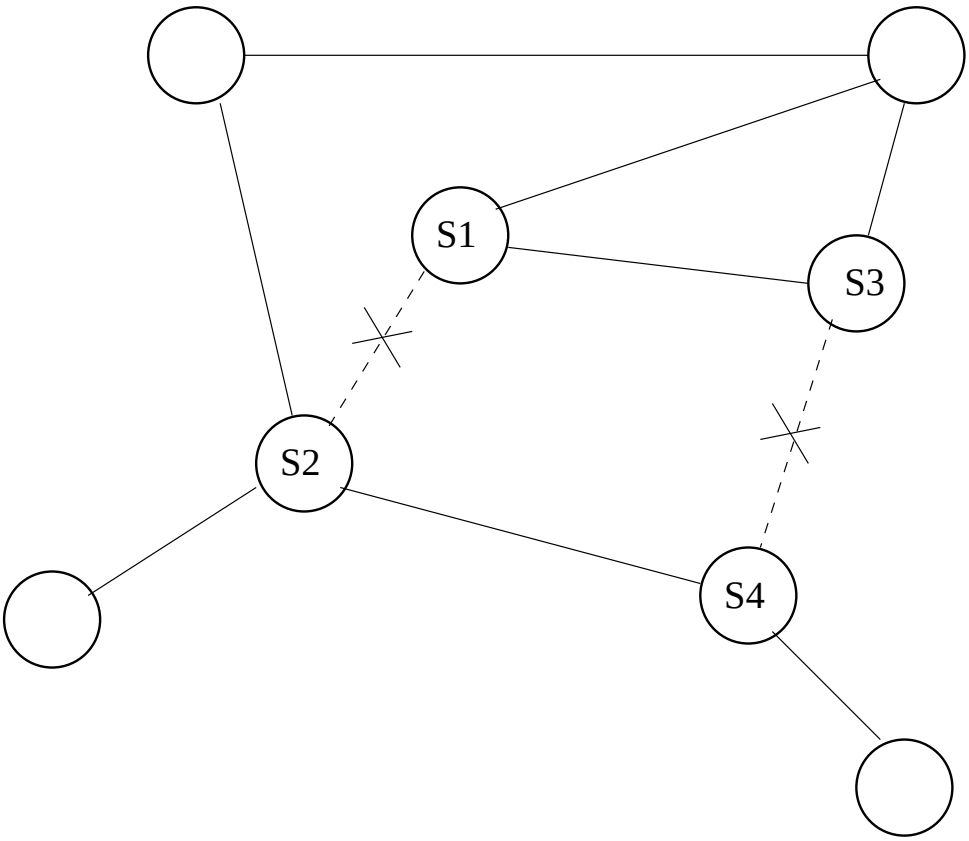


FIG. 6.2 – Perturbation de la topologie

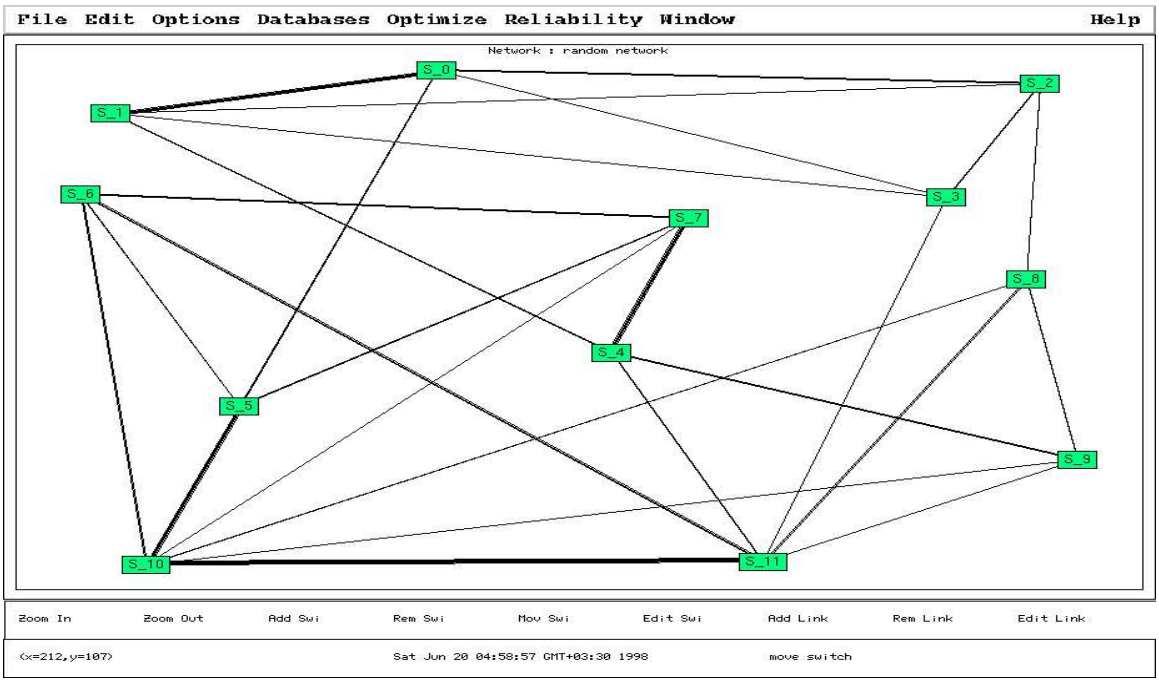


FIG. 6.3 – La topologie optimale trouvée

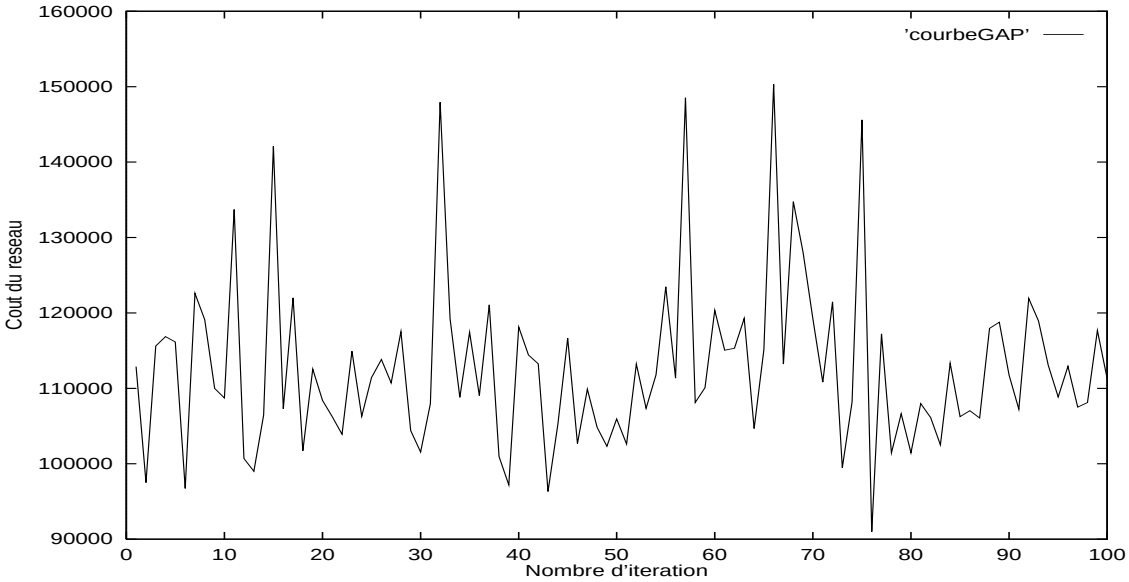


FIG. 6.4 – Variation du Coût en fonction du nombre d'itérations

6.2.3 Topologie en étoile

La topologie en étoile est en fait un cas particulier de la topologie multipoint. Ce type de topologie est souvent utilisé pour interconnecter des agences d'une entreprise à un siège central. Pour étudier ce type de topologie, nous commençons par introduire cette définition:

Chaque nœud u du réseau est défini par 2 paramètres:

- $\alpha(u)$: flux maximum sortant du nœud u;
- $\beta(u)$: flux maximum entrant au nœud u;
- $\gamma(A, B)$: coût de la liaison (A, B).

Nous décrivons dans ce paragraphe les principes de l'algorithme **Cheapest Star** [AND 94]:

Algorithme Cheapest Star

- Pour tout n$u \in N$ faire
  - Calculer $Out(u) \leftarrow$Capacité du lien (u, C), pour tout étoile de centre $C \neq u$
  - Calculer $In(u) \leftarrow$ Capacité du lien (C, u) pour tout étoile
- Fin Pour
- Pour tout n$C \in N$ faire (Calculer le coût de la topologie en étoile de centre C)
  - Coût(C)$\leftarrow \sum_{u \in N-\{C\}} (Out(u)\gamma(u, C) + in(u)\gamma(C, u))$
- Fin Pour
- Trouver le Coût minimal

Complexité de Cheapest Star

Chaque itération coûte $O(N)$. D'où la complexité de cet algorithme est de $O(N2)$.

Résultats expérimentaux

- Nombre de nœuds: 21.
- Topologie en étoile trouvée est donnée par la figure 6.5.

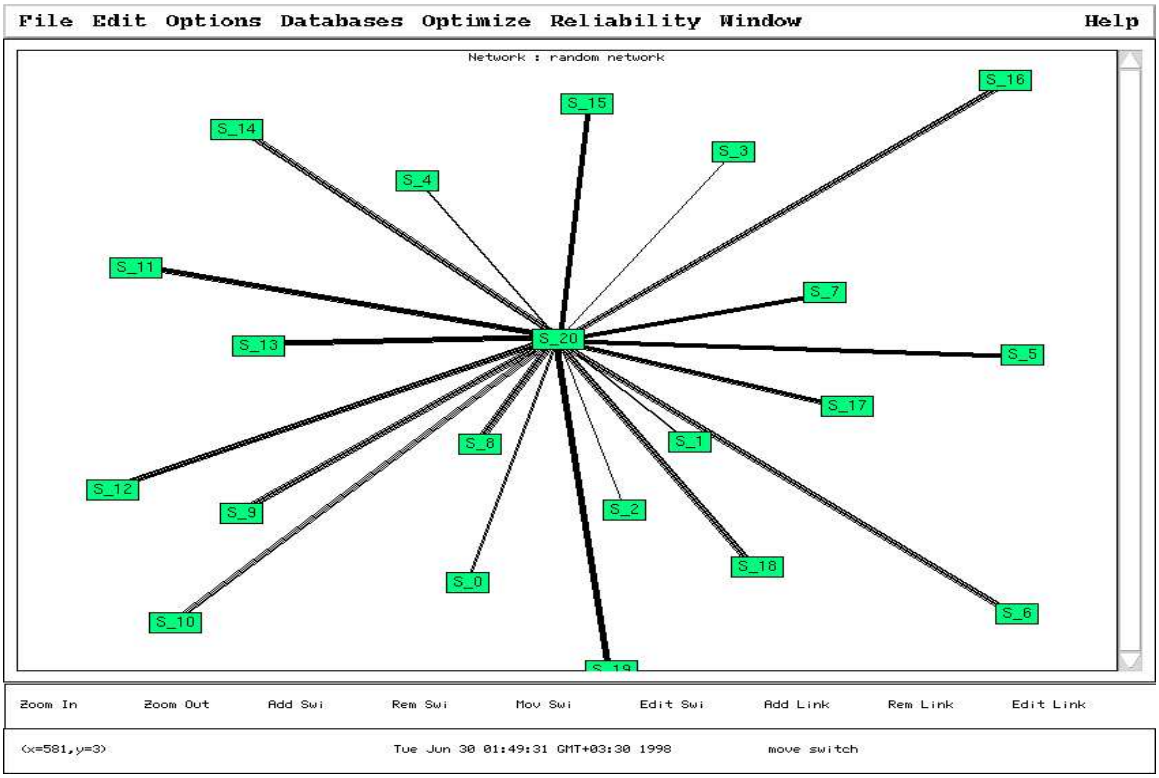


FIG. 6.5 – Topologie en étoile optimale

6.3 Etude de la fiabilité d'un réseau

6.3.1 Détermination de la connectivité d'un réseau

Notre objectif dans ce paragraphe est d'étudier la fiabilité des réseaux: c'est à dire des réseaux pouvant perdre un ou plusieurs nœuds et ne pouvant pas perdre leur connectivité. Pour cela, on énonce le résultat suivant: énoncé par **Ford Fulkerson** en 1962 [TAN 81]:

"Le flot maximum entre deux points quelconques d'un réseau ne peut pas dépasser la capacité de la coupe minimale séparant ces deux nœuds "

Ce résultat est devenu par la suite un théorème et Frank et Fish 1971 ont montré que :

" Flot maximum est égale à la taille de la coupe minimale "

La connectivité au sens arc entre deux nœuds x et y est le nombre minimum d'arcs à supprimer pour déconnecter les deux nœuds, c'est aussi la taille de la coupe minimale entre x et y.

Pour calculer la connectivité au sens arc, il suffit de calculer le nombre de chemins disjoints au sens arcs (chemins n'ayant aucun arcs en commun). Et pour calculer le nombre de chemins disjoints on procède comme suit :

1. Valuer le graphe par des 1;
2. Calculer le flot maximum entre ces deux nœuds par l'algorithme de Malhorta 1978.

L'algorithme de **Malhorta [TAN 81]**, qui détermine le flot maximum, prend en entrée un graphe oriente, une source, une destination et fournit comme résultat le flot maximal faisable.

Algorithme de Malhorta

– Tant que la mise en couche est possible Faire

1. Construire un réseau multi-couches à partir du réseau initial; mettre la source dans la couche 0, tous les arcs connectés à la source dans la couche 1, ainsi de suite,...(la procédure de mise en couche s'arrête dès qu'on atteint la destination), étiqueter les npar distance par rapport à la source;
2. Trouver les arcs faisables (dont le flux effectif peut être augmenté);
3. Trouver les chemins bloquants (ne peuvent pas conduire à la destination);
4. Parcourir tous les ndu réseau en couche pour le flux potentiel pour être utilisé pour améliorer le flux;
5. Trouver le Nréférence (le nayant la moindre différence entre le flux entrant et le flux sortant);
6. Commencer à partir du NRéférence pour améliorer le flux ;
7. Aller à 1.

– Fin Tant que

Complexité de l'algorithme de Malhorta:

La construction du réseau multi-couches coûte $O(N)$, la recherche du Flux Potentiel a la complexité $O(M)$, la recherche de Nœud Référence coûte O(N), d'où la complexité de l'algorithme est $O(M * N^2)$.

Pour montrer que la connectivité d'un réseau est k, nous avons fait recours à l'algorithme de **Kleitman** [BER 92] qui est basé sur l'algorithme de Malhorta et dont le principe est le suivant:

Algorithme de Kleitman

– Tant que la connectivité courante k est différente de 0 Faire

1. Construire, à partir d'un graphe non orienté G(N,M), un graphe G' (2N, 2N+M) orienté ;
2. Attribuer aux liaisons des évaluations de 1 ;
3. Tirer au sort un n, calculer le flux maximum entre ce net tous les autres n, ça doit être supérieur ou égal à k (dont le nombre de chemins disjoints au sens lien est aussi supérieur ou égal à k et donc la connectivité est supérieure ou égal à k).
4. Supprimer ce net tous les liens y sont associés,
5. k := k-1 ;
6. aller à 1

– Fin tant que

La complexité de cet algorithme est de l'ordre de $O(N^2)$ à cause de l'imbrication de deux boucles itératives.

6.3.2 Simulation de la fiabilité d'un réseau

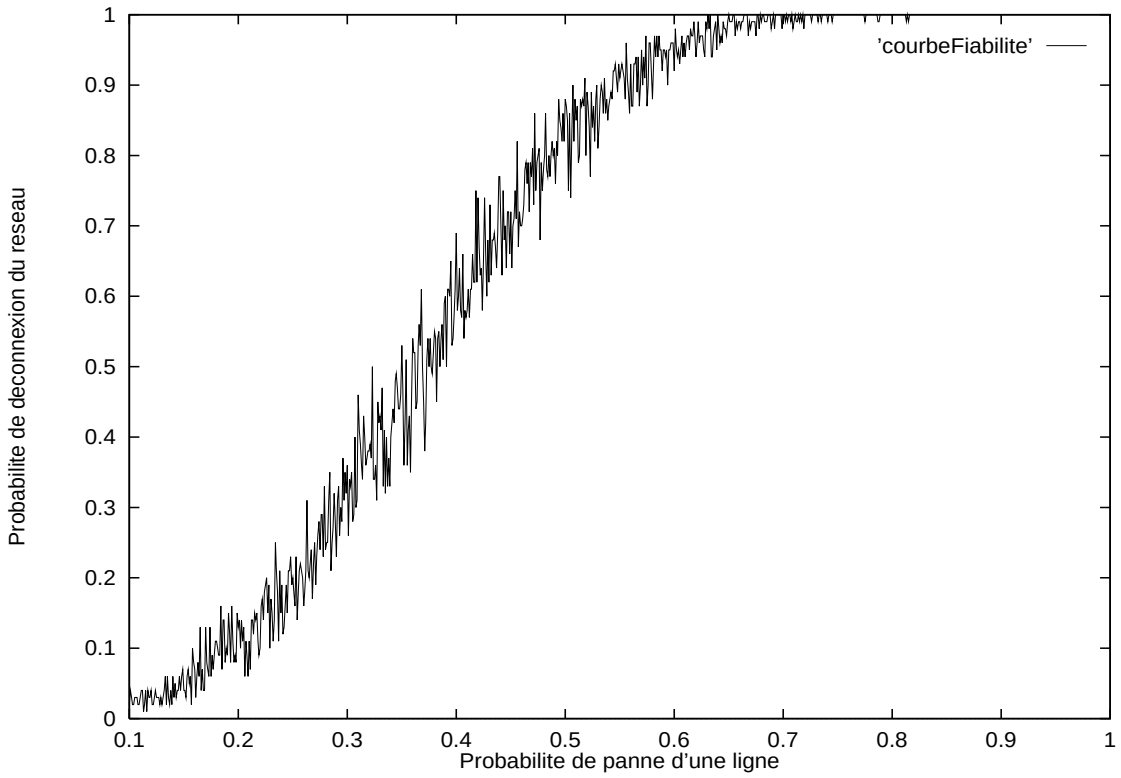


FIG. 6.6 – Probabilité de déconnexion du réseau en fonction de la probabilité de panne d'une ligne

Avec les réseaux larges et non réguliers, notre unique recours, pour étudier la fiabilité de ces réseaux est la simulation. Un modèle, qui n'est pas l'unique, est de diviser le temps en intervalles de temps réguliers. Durant chaque intervalle, un lien ou un switch est positionné ou bien à *ON* ou à *OFF*, la probabilité de panne d'une liaison pendant un intervalle est p. La sortie de cette métrique est la probabilité de déconnexion du réseau en fonction de p. Quant à la connectivité, elle s'avère intéressante quand nous voulons imposer des contraintes de degré de connectivité minimale pour éviter les ruptures inattendues.

La figure 6.6 montre que la probabilité de déconnexion du réseau est croissante avec la probabilité de panne d'une ligne. Cette variation est intimement liée à la connectivité du réseau choisi. La topologie que nous avons choisie dans cet exemple est celle de la figure 6.3.

6.4 Conclusion

En conclusion, il importe de ne pas perdre de vue que le problème de conception de topologie est le plus compliqué à résoudre parmi les problèmes étudiés jusqu'à présent. Il fait partie de la classe des problèmes NP-Complets.

Ce chapitre a été consacré à la description de la démarche suivie par **WanDesigner** pour la résolution du problème général de conception de topologie. L'algorithme **Recuit Simulé** a été présenté, dans la première partie de ce chapitre. Ainsi, la résolution du problème compliqué de conception de topologie maillée est faite grâce à cette heuristique. Par ailleurs, deux autres algorithmes sont proposés: **Chandy Russel** et **Cheapest Star** pour résoudre respectivement le problème de topologie en arbre et en étoile. Enfin, la dernière partie de ce chapitre, présente au concepteur quelques résultats expérimentaux déduits de l'application de WanDesigner à la conception du RNU.

Après avoir achevé la première partie de ce rapport, nous donnons dans le tableau 6.1 un récapitulatif des algorithmes et des heuristiques étudiés jusqu'à présent, leur utilisation et leur complexité.

0 8 1 0 0 0 0Utilisation	Algorithme	Complexité
Allocation de capacité linéaire	Méthode de Lagrange	$O(1)$
Allocation de capacité concave	Méthode heuristique	
Allocation de capacité discrètes	Heuristique de Décomposition de Lagrange	
Allocation de flot	Méthode de déviation de flux	
Recherche du coefficient de déviation	Suite de Fibonacci	$O(N)$
Recherhe des chemins les plus courts	Algorithme de Dijkstra	$O(N^2)$
Allocation de capacité et de flot	Méthode heuristique	
Topologie maillée	Méthode de Recuit Simulé	
Topologie multipoint	Heuristique de Kruskal	$O(Mlog(N))$
Topologie en étoile	Algorithme Cheapest Star	$O(N^2)$
Détermination du flot maximum	Algorithme de Malhorta	$O(MN^2)$
Détermination de la connectivité	Algorithme de Kleitman	$O(N^2)$
Etude de la fiabilité du réseau	Méthode par simulation de panne	

TAB. 6.1 – Récapitulatif des algorithmes étudiés et implémentés

Chapitre 7

Etude de cas: Conception du Réseau National Universitaire

Après avoir esquissé les différents problèmes d'optimisation des réseaux, la dernière étape de cette partie, consiste à examiner les résultats de l'application de **WanDesigner** sur la conception du Réseau National Universitaire tunisien (RNU). Le RNU est un projet qui vise à mailler les institutions universitaires et de mettre à la disposition des laboratoires de recherche, des outils de travail utilisant des nouvelles technologies informatiques et de télécommunication.

Pour des raisons géographiques nous nous limitons dans le cadre de ce projet à une topologie maillée pour la conception du backbone du RNU et des topologies multipoint aux seins des groupements d'institutions. L'interconnexion de toutes les institutions universitaires est établie, tout simplement, en faisant relier chaque groupement au backbone par un nœud d'entrée.

7.1 Architecture globale du RNU

Le Réseau National Universitaire (RNU) sera structuré sous forme de "réseau de réseaux". Par exemple, une institution universitaire ayant plusieurs départements et laboratoires pourra installer ses propres réseaux locaux à raison d'un réseau par département ou par laboratoire (en fonction de sa taille et des moyens disponibles). Ces réseaux locaux seront connectés entre eux pour constituer le réseau de l'institution qui sera à son tour connecté au réseau RNU soit directement, soit à travers un réseau fédérateur reliant un ensemble d'institutions géographiquement proches (réseau de campus).

De même, les différentes directions de l'administration du Ministère, les rectorats et les directions et les offices des œuvres universitaires (Nord, Centre et Sud) auront aussi leurs propres réseaux locaux et seront connectés aussi au réseau RNU.

Par ailleurs, pour les accès aux réseaux internationaux, le centre CCK sera relié directement à l'Agence Tunisienne d'Internet (ATI) par des lignes spécialisées et assurera ainsi l'interface d'accès à Internet.

Dans ce cadre, le centre CCK jouera le rôle de Fournisseur de service Internet (FSI) pour les institutions à l'échelle nationale et internationale.

7.2 Conception du backbone du RNU

Le réseau backbone a pour rôle d'interconnecter les différents relais du RNU au Fournisseur de Service Internet le CCK d'une part et de relier ce dernier au fournisseur national d'accès Internet l'ATI. Ce réseau est constitué de cinq nœuds: l'ATI, le CCK, le relais de nord, le relais de sud, le relais de centre. Pour des raisons de fiabilité, la topologie proposée de ce backbone est une topologie maillée.

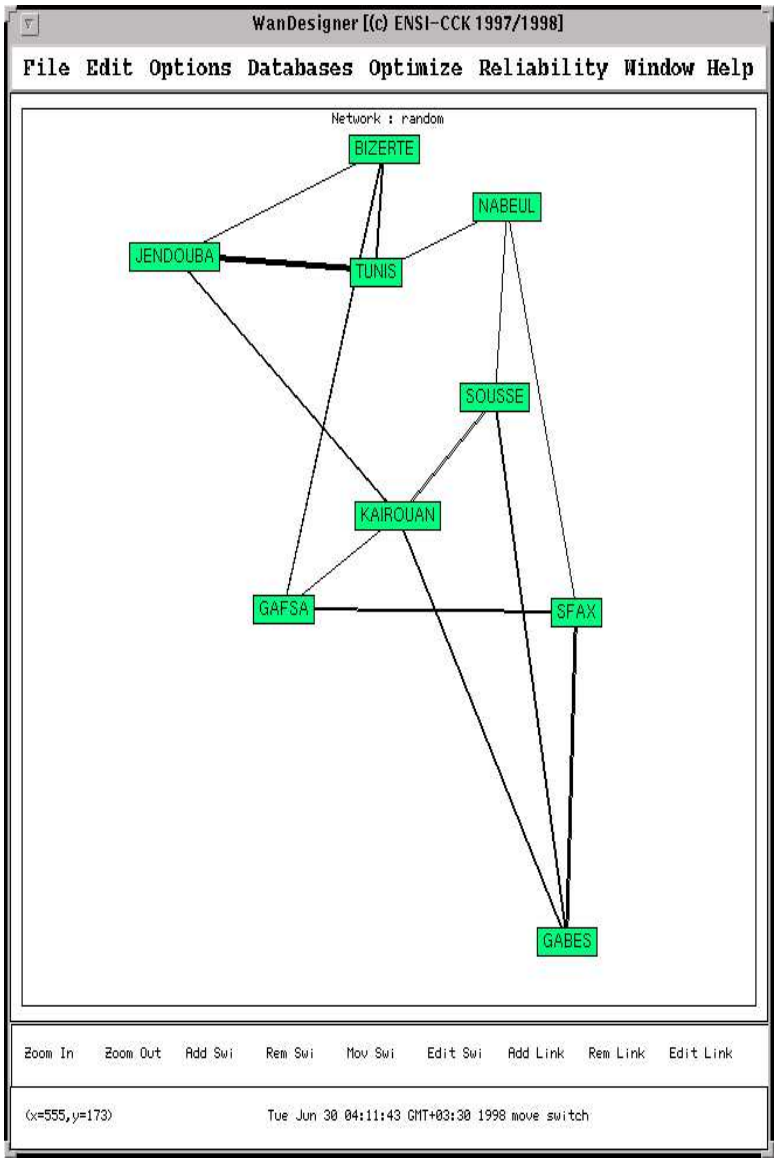


FIG. 7.1 – Topologie maillée du backbone

Pour la conception de ce réseau, certaines données sont nécessaires. Ces données sont relatives essentiellement à:

- la disposition géographique des nœuds constituant le réseau,
- la matrice de trafic entre chaque paire de nœuds du backbone.

Nous considérons à titre indicatif 9 zones éloignées (concentration d'institutions), pouvant être considérées comme les nœuds formant le backbone. Le trafic véhiculé dans ce backbone est prévu élevé, car il interconnecte toutes les institutions du RNU.

La figure 7.1 montre la topologie maillée optimale de ce backbone telle qu'elle est fournie par **WanDesigner**. Cette topologie est obtenue moyennant les distances géographiques estimées et les trafics générés aléatoirement.

7.3 Conception des différents relais

Un relais est constitué d'un ensemble d'institutions universitaires et de centres de recherche géographiquement proches. Le RNU renferme trois principaux relais: relais du nord, relais du sud et relais du centre. Chaque relais correspondra à un réseau multipoint.

Il est à signaler, que les lignes reliant les institutions d'un même relais seront des lignes spécialisées de 64 kbits/s. Le coût de chaque ligne est proportionnel à la distance géographique entre les deux institutions (cf. section 2.1.1).

Nous nous limitons à présenter la démarche de la conception du relais du nord. Celle ci restera valable pour la conception des réseaux des autres relais.

Afin de concevoir le réseau de chaque relais, il est nécessaire de déterminer les paramètres de conception qui sont liées à:

- la matrice de distances géographiques,
- le trafic généré par chaque institution vers le backbone du RNU.

7.3.1 Matrice de distances géographiques

Les distances sont déterminées à vol d'oiseau. Elles sont nécessaires pour décider quel nœud doit être relié au réseau du relais au cours de la conception de la topologie multipoint. Les distances entre les institutions du relais de nord sont générées aléatoirement car nous ne disposons pas des données réelles.

7.3.2 Détermination de la matrice de trafic

Etant donné que les institutions ne sont pas toutes, actuellement, liées au RNU. Il nous a fallu de préciser une procédure d'estimation de trafic généré par chaque institution vers le backbone. Pour ce faire, nous proposons deux techniques d'estimation:

La première, et que nous avons déjà évoquée dans la section 1.2.1, consiste à appliquer une formule mathématique mettant en jeu le nombre d'utilisateurs de chaque institution et la distance entre les deux institutions:

$$\frac{nb\,utilisateurs(i) * nb\,utilisateurs(j)}{distance(i,j)^{\alpha}} \tag{7.1}$$

La deuxième, consiste à déduire les différents trafics de chaque institution à partir du trafic généré par une institution déjà connecté au réseau. Ce trafic est déterminé par des outils de mesure sophistiqués qui ont été déjà installés par nos collègues effectuant des mesures afin de caractériser le trafic Internet du RNU [ROM 98] (principalement l'outil *NeTraMet*). Pour que le trafic estimé soit le plus proche possible de la réalité, nous devons distinguer les institutions par discipline. En effet, il sera plus significatif si nous étudions disjointement les

disciplines littéraire et scientifique et considérons pour chaque ensemble d'institutions un échantillon représentatif.

À titre d'exemple, nous proposons l'ENIT (Ecole Nationale des Ingénieurs de Tunis) comme échantillon représentatif de la discipline scientifique. En effet, les chercheurs de cette école ont souvent accès à Internet pour la recherche de l'information.

Il nous reste à préciser, de manière exacte, la méthode de calcul de la matrice de trafic. Le trafic global généré par une institution *i* est déterminé en multipliant le nombre d'utilisateurs par le trafic généré par un utilisateur de l'institution représentative.

Le nombre d'utilisateurs estimé à avoir accès aux services Internet pour chaque institution est donné dans le tableau 7.1. Le trafic généré par chaque utilisateur est déduit directement des mesures de trafic effectuées sur l'institution représentative. Nous admettons que le nombre d'utilisateurs participant à ce trafic est égal au nombre des comptes mail ouverts pour cette institution.

À titre d'exemple, nous considérons le cas d'institutions scientifiques dont l'ENIT est l'institution représentative. Le nombre de comptes mail réservés pour l'ENIT est 6. Le trafic global généré par cette institution est *4650.973 bits/sec*. D'où le trafic par utilisateur est:

$$4650/6 = 775 \quad bits/seconde$$

Nous appliquons cette démarche pour le reste des groupements et nous calculons le trafic des institutions de chaque groupement.

Nous donnons dans le tableau 7.1 la matrice de trafic du relais du nord du RNU:

0 8 1 0 0 0 0Institution	Nombre d'utilisateurs	Trafic total vers le backbone du RNU (en bits/s)
ENSI	92	71314
ENIT	477	369752
EPT	23	17828
FST	865	670515
FSEGT	451	349598
IHEC	151	117049
INSAT	51	39533
IPEIT	32	24805
IPEST	39	30231
ISCAE	104	80616
ISD	28	21704
ISET	255	197666

TAB. 7.1 – Trafic par institution de relais du Nord du RNU

7.4 Topologie optimale du relais du nord

Cette topologie est issue de l'application de l'heuristique de **Kruskal** donné en annexe (cf. section A.6). La figure 7.2 donne la topologie optimale trouvée par **WanDesigner** compte tenu des matrices de tarfic et de distances déterminées précédemment

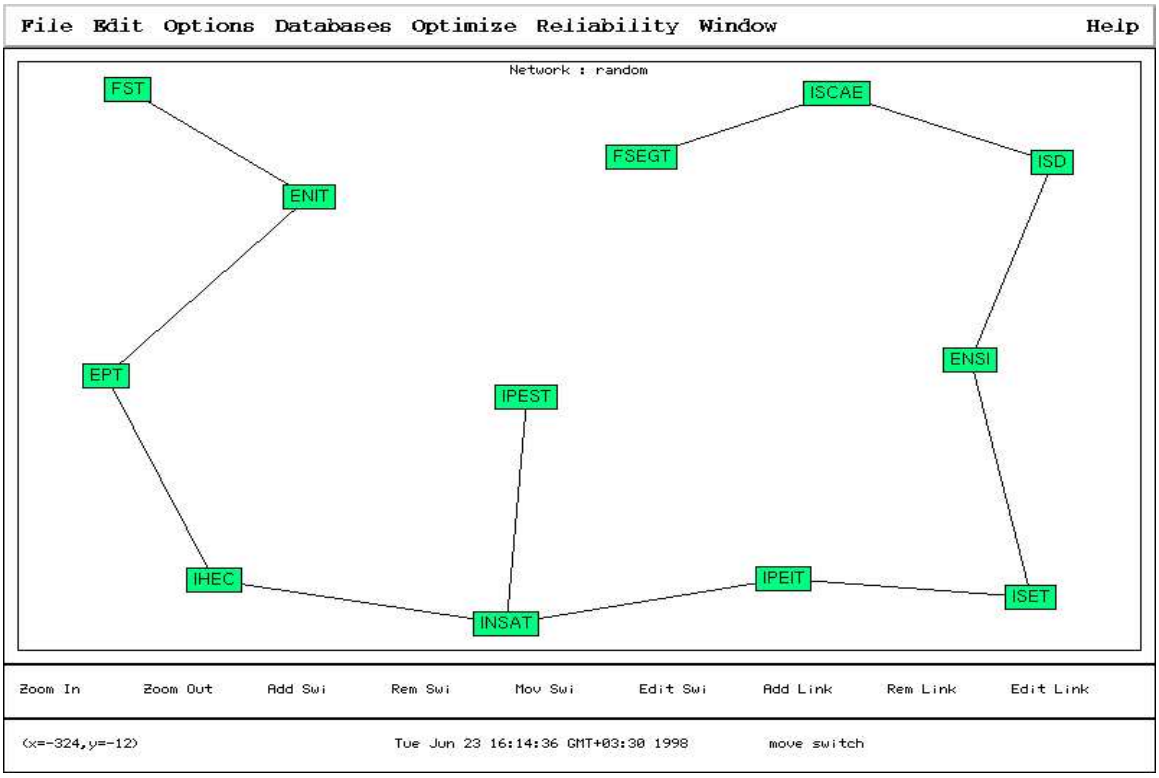


FIG. 7.2 – Topologie optimale du relais du nord du RNU

7.5 Conclusion

Quelques résultats données dans ce chapitre ne tiennent pas compte, en fait, des données réelles du RNU. Cependant, si nous voulons nous rapprocher de la réalité, il suffit de déterminer d'une manière exacte et réelle la matrice de trafic, les dispositions géographiques, le nombre d'utilisateurs de chaque institution.... Peu importe les résultats, l'essentiel c'est qu'il y a optimisation du réseau en fonction des données en entrée.

Deuxième partie

Processus de mise en œuvre de WanDesigner

Chapitre 1

Spécifications de WanDesigner

Avant d'entamer le processus réel de mise en œuvre de notre atelier **WanDesigner**, il est important de spécifier les fonctionnalités attendues indépendamment de la méthode de conception à adopter. Notre souci est de dégager, à partir de la formulation du problème, les besoins fonctionnels ainsi qu'architecturaux de l'outil à concevoir.

1.1 Formulation du problème

Comme tout problème de développement, la première étape consiste à poser le problème. **Le travail demandé consiste à développer un atelier (environnement logiciel) permettant la conception et l'optimisation de réseaux et ce, compte tenu des besoins en communication d'une part, et des supports et technologies d'interconnexion d'autre part. L'application se fera sur la conception du réseau national universitaire.**

Pour extraire les besoins, nous avons utilisé plusieurs sources d'informations. Parmi lesquels des études de l'offre des technologies de communication existantes et leurs coûts. Ces études prévoyaient que l'atelier englobe une base de données tarifaire et interagit éventuellement avec des outils d'analyse de trafic pour l'extraction des matrices de trafic; comme le montre la figure 1.1. Les outils d'analyse de trafic fournissent la matrice de trafic au moteur d'optimisation. Ce dernier effectue les calculs d'optimisation en se basant sur des bases de données tarifaires.

1.2 Besoins fonctionnels

Les besoins fonctionnels de **WanDesigner** peuvent être résumés dans les points suivants:

- Par définition, **WanDesigner** doit regrouper des "moteurs intelligents" permettant la conception et l'optimisation d'un réseau WAN en se basant sur une étude comparative des technologies d'interconnexion existantes et une étude de la fiabilité.
- **WanDesigner** doit être capable de proposer une topologie d'un réseau informatique de coût minimal assurant certaines performances en ce qui concerne le temps de réponse, la disponibilité et une fiabilité acceptable.
- **WanDesigner** doit stocker les données dans une base de données gérée par un système de gestion de base données (SGBD).

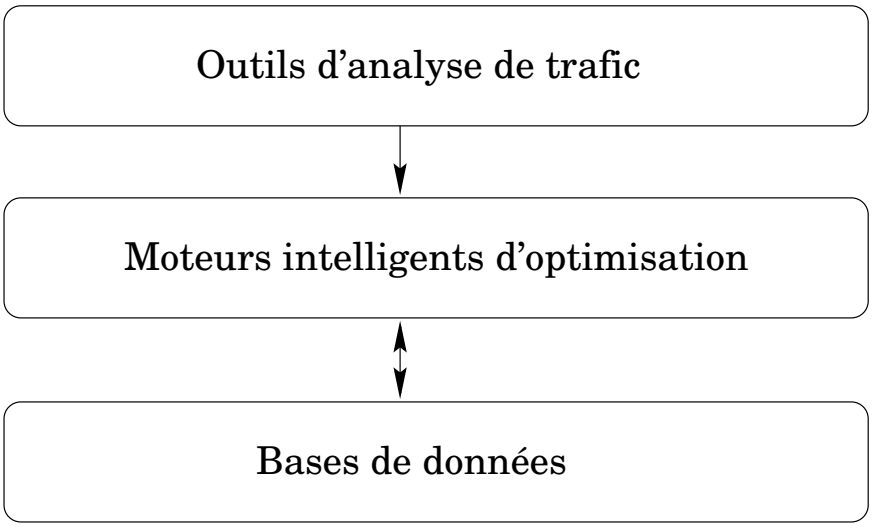


FIG. 1.1 – Vue globale du système

- L'atelier **WanDesigner** doit communiquer avec une base de données tarifaire (locale ou distante). Cette base de données est particulièrement adoptée à l'offre du marché télécom tunisien.
- Les données géographiques doivent être stockées dans une base de données géographique.
- Les réseaux doivent être stockés dans une base de données particulière. Chaque réseau est caractérisé par un ensemble de nœuds, un ensemble de liens, un ensemble des contraintes et des paramètres de performance.
- L'utilisateur doit pouvoir accéder aux bases de données pour les alimenter, les consulter, et déclencher les différents algorithmes d'optimisation. Ceci doit se faire via une interface graphique souple et ergonomique.
- Le logiciel doit être portable sur différentes machines, systèmes d'exploitation, systèmes de gestions de bases de données.
- Le logiciel doit être extensible afin d'inclure des nouvelles fonctions et des nouveaux programmes d'application.
- En sortie, le logiciel doit permettre la visualisation de la configuration du réseau, des courbes graphiques donnant la variation des paramètres de performances en fonction des variables de conception. Ces courbes servent comme support d'aide à la décision pour l'utilisateur.
- Le logiciel doit permettre l'édition des rapports décrivant les paramètres de performance, les matrices de routage, les coûts des différents composants...

1.3 Besoins architecturaux

WanDesigner doit être bâti sur plusieurs composants logiciels à savoir:

- un système graphique pour créer, sélectionner et éditer des composants graphiques du réseau courant,

- une interface vers les bases de données pour insérer, détruire, mettre à jour les données tarifaires, les réseaux et les données géographiques.
- une interface utilisateur pour afficher des résultats d'optimisation et accepter les entrées utilisateur, et enfin,
- des noyaux intelligents d'optimisation.

1.4 Conclusion

Après avoir défini les besoins d'une façon claire et concise, il nous a fallu chercher une démarche de prise de décisions dans le but de proposer des éléments de solution au problème qui vient d'être posé. Partant de cette spécification, nous présentons dans les chapitres suivants la démarche, proposée par OMT, à suivre pour la mise en œuvre de **WanDesigner.**

Chapitre 2

Analyse de WanDesigner

Après avoir achevé la phase de spécification du projet, nous entamons la phase de conception. Nous nous sommes convenus d'utiliser la méthode OMT pour cette phase qui est une méthode de conception **orientée objet** couvrant toutes les étapes du cycle de vie d'un logiciel : elle peut être employée pour analyser les spécifications du problème, concevoir une solution et enfin implémenter cette solution. Dans ce chapitre, sera développée la modélisation du système WanDesigner. Cette modélisation consiste, en fait, en 3 axes différents: axe statique, axe dynamique et axe fonctionnel.

2.1 Modélisation statique

Un examen approfondi de la spécification permet de dégager un ensemble d'objets et des classes qui, ensemble permettent de décrire la structure statique des données du système. Pour cette fin, nous essayons dans cette section d'identifier les classes d'objets, préparer le dictionnaire de données, identifier les associations et regrouper les classes en modules.

2.1.1 Identification des classes d'objets

Commençons par énumérer les classes d'objets candidates. L'examen de la description du problème permet de dégager l'ensemble des classes de la figure 2.1.

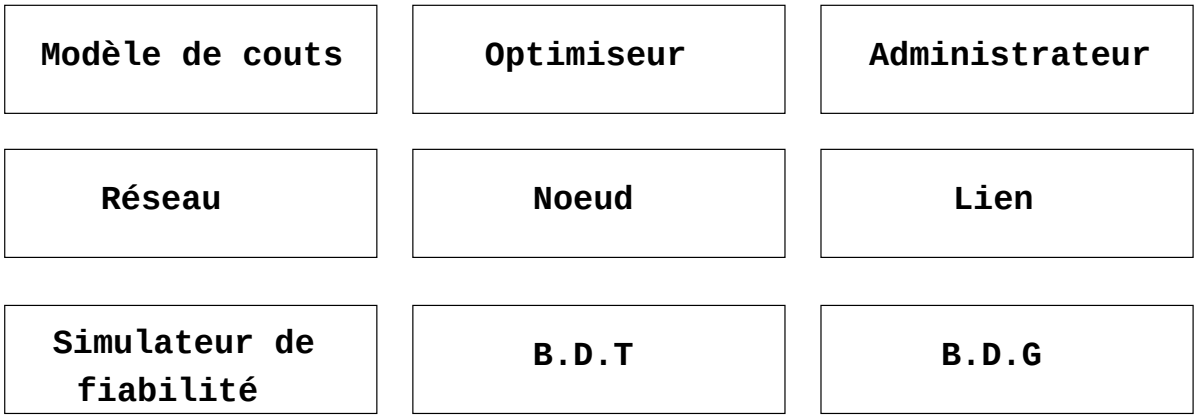


FIG. 2.1 – Classes d'objets

Cet ensemble de classes d'objets n'est pas exhaustif, mais peut être enrichi par des classes qui n'apparaissent pas au début.

2.1.2 Préparation du dictionnaire des données

- **Modèle coût**: Modèle primordial pour caractériser la fonction Coût-Capacité. Ce modèle se base sur les données stockées dans une base de données tarifaire pour estimer la fonction du coût des supports de communication.
- **Lien:** Jonction entre deux nœuds. Cette jonction est caractérisée par son nom, sa longueur, sa capacité, le trafic qu'elle peut écouler...
- **Noeud:** C'est un routeur ou un commutateur caractérisé par son nom, son emplacement,...
- **Réseau:** Un ensemble de nœuds, de liens et de contraintes. Un réseau est caractérisé par le nombre de nœuds, le nombre de liens, la topologie, le coefficient de fiabilité, la charge extérieure qu'il peut écouler,...
- **Optimiseur:** Algorithmes, heuristiques ou analytiques, d'optimisation. Ces algorithmes, moyennant certaines données, minimisent une fonction objective sous certaines contraintes.
- **Simulateur de fiabilité:** Module qui prend en charge le problème de la connectivité du réseau. Ainsi, tout réseau construit doit être tolérant aux pannes pouvant parvenir (préserver la connectivité du réseau malgré la défaillance des lignes ou des nœuds).
- **Base de données de réseaux:** Base de données stockant des réseaux qui ont été déjà étudiés pour une réutilisation future.
- **Base de données géographique:** Base de données stockant des données géographiques des sites (localisation dans une carte...), des liens (distances...).
- **Base de données tarifaire:** Base de données incluant des informations sur l'offre télécom (services, prix, ...). Cette base alimente le modèle coût pour caractériser la fonction de coût.
- **Administrateur:** Concepteur, manipulateur du logiciel et gérant des bases de données. Il peut en conséquence, consulter, modifier, insérer des données.

En examinant l'ensemble des objets dégagés ainsi que la spécification du modèle, on peut extraire une architecture en objets reliés par des relations ou associations et définis par des attributs. Pour notre application, le diagramme d'objets retenu est donné par la figure 2.2 et la liste des attributs des objets est donnée par le tableau 2.1.

2.1.3 Organisation et simplification des classes d'objets en utilisant l'héritage

Le diagramme issu de l'application du concept de l'héritage est donné dans la figure 2.3.

La classe base de données a été ajoutée afin de généraliser les attributs et les opérations qui paraissent communs entre les différentes bases de données. Ainsi, l'administrateur gérant la base de données indiquera les paramètres de localisation de la base. Ces paramètres sont entre autres: le nom de la base, la localisation, le numéro du port d'écoute, le nom de l'utilisateur, son mot de passe... permettant l'authentification de l'utilisateur.

Les moteurs intelligents d'optimisation se classent en de nombreuses catégories. On trouve l'allocation de capacité, l'allocation de routage, l'allocation de capacité et de routage

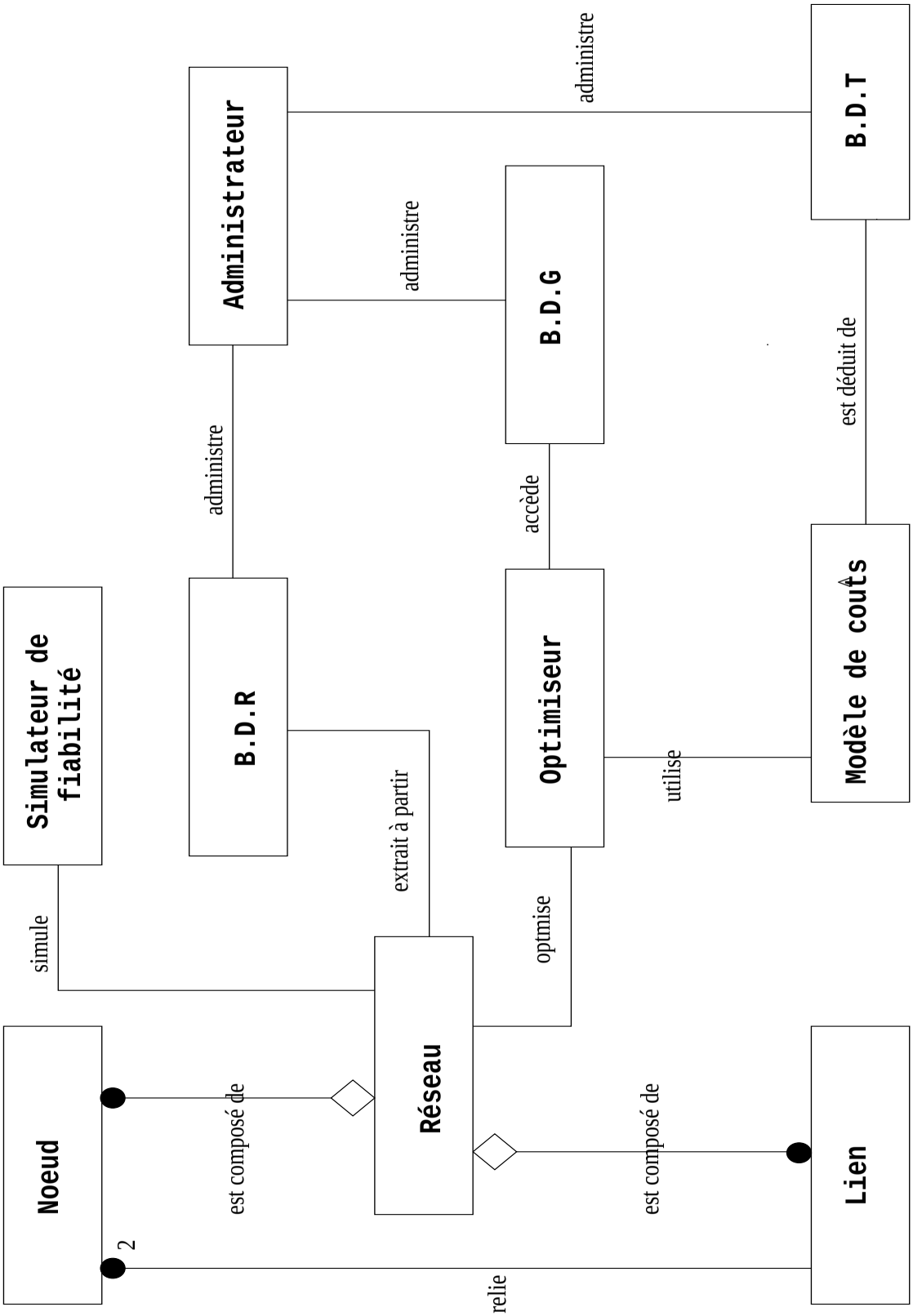


FIG. 2.2 – Diagramme d'objets

0 8 1 0 0 0 0Objets	Attributs	Description
	NomNœud	Le nom du nœud
Nœud	PosX	L'abscisse de la position du nœud
	PosY	L'ordonnée de la position du nœud
	NomLien	Le nom du lien
Lien	ExtremiteDroite	Le nœud de l'extrémité droite
	ExtremiteGauche	Le nœud de l'extrémité gauche
	Longueur	La longueur du lien
	NomReseau	Le nom du réseau
Réseau	Nœuds	L'ensemble de nœuds
	Liens	L'ensemble de liens
	TraficExterne	Matrice du trafic externe
Optimiseur	TypeProblème	Le type du problème d'optimisation
Simul. de fiab.	NomRes	Le nom du réseau à étudier
B.D.G	NomBaseGeo	Le nom de la base de données géog.
B.D.T	NomBaseTarif	Le nom de la base de données tarifaires
Administrateur	Login	Le login de l'administrateur
	MotPasse	Le mot de passe de l'administrateur
Modèle Coût	TypeTech	Le type de la technologie à modéliser
	Fonction	La fonction du calcul de coût

TAB. 2.1 – Les objets et leurs attributs

et l'optimisation de la topologie (maillée, multipoint, en étoile). Ces moteurs héritent de la classe optimiseur des attributs et des opérations communes tels que le réseau à optimiser, certaines contraintes d'optimisation...

Les technologies de communication diffèrent par la procédure de facturation (tarification en fonction du volume de données transférées, en fonction de la distance, en fonction du temps, ...). Il serait, donc, intéressant, pour des raisons d'extensibilité, de distinguer ces classes de modélisation de coûts: classe ModèleLS, classe ModèleX25 et classe ModèleRNIS, et ModèleRTC qui héritent tous de la classe Modèle de coûts.

2.2 Modélisation dynamique de WanDesigner

Le modèle dynamique s'intéresse aux aspects temporels du système ainsi qu'aux objets à l'intérieur de ce système. Il est sans grand intérêt pour un système de données purement statiques. En revanche, il est important pour les systèmes interactifs. Dans notre application, le calcul et l'optimisation est une tâche importante, voire même la tâche principale.

Tout d'abord, il faut préparer des scénarios de séquences typiques. Ensuite, il faut organiser les séquences d'événements et les états dans un diagramme d'états. Enfin, les diagrammes d'états des différents objets sont comparés afin de s'assurer de la correspondance des événements qu'ils s'échangent. Le résultat de cet ensemble de diagrammes d'états forme le modèle dynamique.

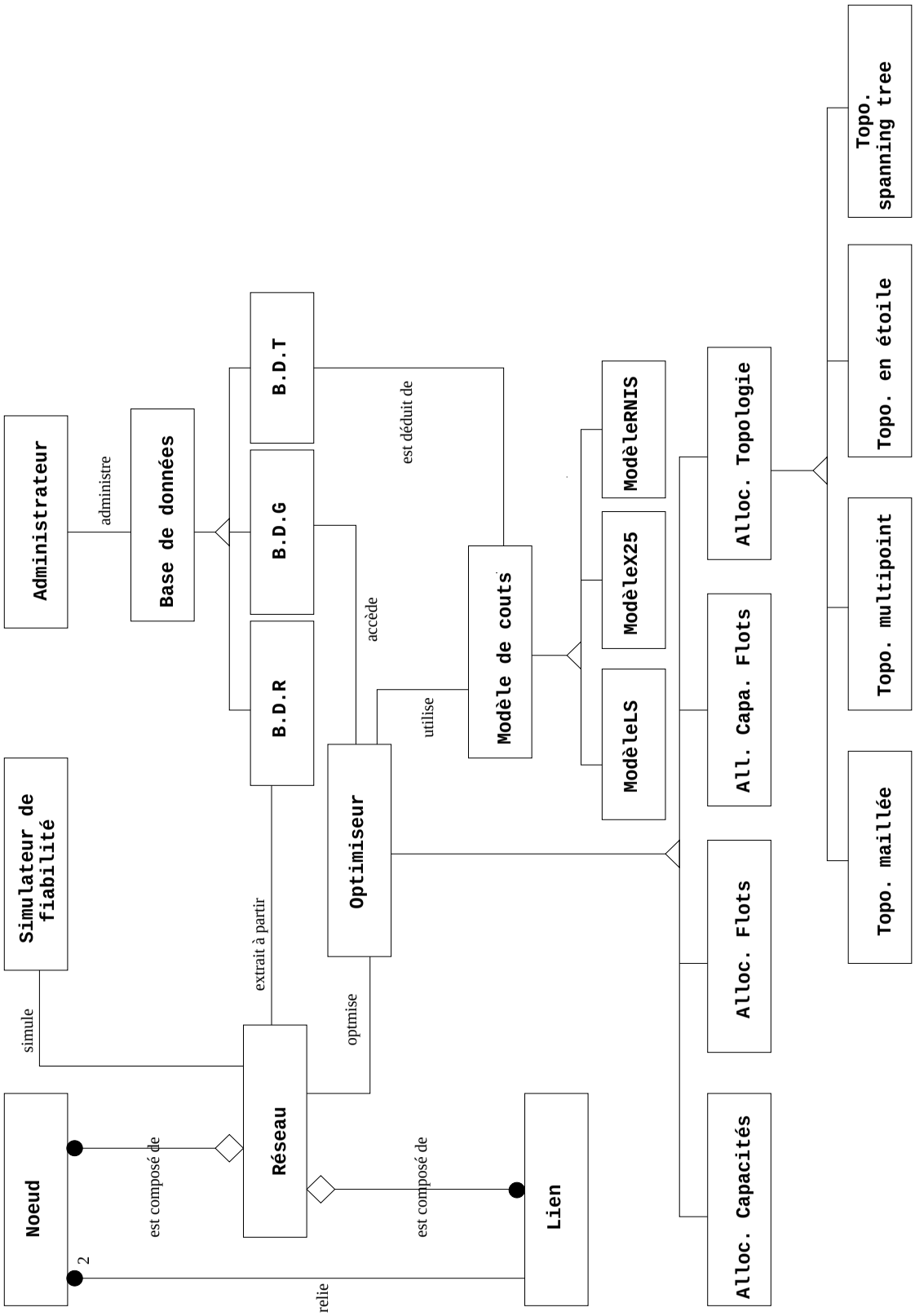


FIG. 2.3 – Diagramme d'objets avec l'héritage

2.2.1 Préparation des scénarios

Il s'agit de préparer un ou plusieurs dialogues typiques entre l'utilisateur et le système afin d'avoir une idée du comportement du système.

Un scénario est une séquence d'événements. Un événement est déclenché dès qu'un échange d'informations est opéré entre un objet et un élément externe (tel que l'utilisateur).

Nous citons dans ce qui suit, une liste non exhaustive des scénarios possibles en commençant tout d'abord par explorer ceux qui décrivent un fonctionnement normal du système puis décrire ceux qui prennent en considération les cas exceptionnels tels que les oublis, les erreurs humaines...

Les scénarios normaux

Scénario 0: *Consultation du Help*

1. L'utilisateur se connecte pour la première fois,
2. Le système lui demande de taper un nouveau mot de passe à enregistrer,
3. L'utilisateur tape et confirme l'enregistrement du mot de passe,
4. L'utilisateur se retrouve devant un nouveau système dont la consultation de l'aide est nécessaire,
5. Le système offre une aide sur l'utilisation des menus, les algorithmes d'optimisation utilisés...
6. L'utilisateur quitte le système après la lecture de l'aide.

Scénario 1: *Conception d'un réseau optimal*

1. Le système demande le mot de passe de l'utilisateur,
2. L'utilisateur tape son mot de passe,
3. Le système vérifie sa validité,
4. Si le mot de passe est valide alors la fenêtre principale est affichée,
5. En sélectionnant l'option appropriée, l'utilisateur demande au système la création d'un nouveau réseau,
6. Un éparpillement aléatoire d'un ensemble de nœuds est exécuté par le système,
7. L'utilisateur peut ensuite modifier le nombre, les emplacements, la topologie, les contraintes...
8. L'utilisateur choisit, parmi les fonctionnalités d'optimisation, celle qui répond à ses besoins,
9. Le système affiche les résultats d'optimisation sous forme de boites de dialogue et le réseau obtenu (topologie optimale, capacités optimales, flot optimal, temps de réponse, coût,...)
10. L'utilisateur peut faire un "zoom" sur un nœud, une liaison, une partie du réseau,...,
11. L'utilisateur enregistre le réseau obtenu dans la base de données réseaux,
12. Après sa confirmation, l'utilisateur quitte le système.

Scénario 2: *Connexion à une base de données*

1. L'utilisateur lance le système en s'authentifiant par son mot de passe,

2. L'utilisateur se connecte à une base de données (de réseaux, tarifaire, géographique) moyennant un protocole de connexion normalisé tels que msql, odbc,...
3. Le système donne la possibilité à l'utilisateur d'effectuer tous les traitements possibles sur la base de données (consultation, mise à jour...),
4. L'utilisateur quitte le système.

Scénario 3: *Étude de fiabilité d'un réseau*

1. L'utilisateur lance le système pour étudier la fiabilité d'un réseau,
2. L'utilisateur précise le réseau à étudier,
3. L'utilisateur précise le type de l'étude à faire soit le degré de connectivité du réseau soit la variation de la probabilité de déconnexion du réseau en fonction de la probabilité de panne d'une ligne,
4. Le système affiche les résultats obtenus,
5. L'utilisateur quitte le système.

Les scénarios anormaux

Scénario 1: *Échec d'authentification*

1. L'utilisateur tape un mot de passe invalide,
2. La boite de dialogue de saisie de mot de passe est affichée de nouveau par le système accompagné d'un message d'erreur,
3. Après trois échecs, le système s'arrête.

Scénario 2: *Échec de connexion à une base de données*

1. L'utilisateur veut se connecter à une base de données particulière,
2. S'il y a une erreur de connexion alors le système doit afficher un message indiquant la cause de l'échec de connexion (hôte hors service, accès refusé...)

2.2.2 Construction du diagramme d'états

Ce diagramme rassemble les diagrammes des différents objets dont le comportement dynamique n'est pas trivial. La construction nécessite l'itération de cette opération, particulièrement, sur les classes ayant des fortes interactions.

Le diagramme d'états représenté dans la figure 2.4 décrit l'aspect d'évolution dynamique du module "*Optimiseuer*". La figure 2.5 résume les transitions pouvant avoir lieu au niveau des bases de données. Quant à l'aspect fiabilité, il est indiqué dans la figure 2.6.

2.3 Modélisation fonctionnelle de WanDesigner

Ce modèle s'intéresse aux traitements des données sans tenir compte du séquencement ni de la structure des objets. Il indique la manière d'obtenir des valeurs, sans s'intéresser à la méthode utilisée pour les obtenir.

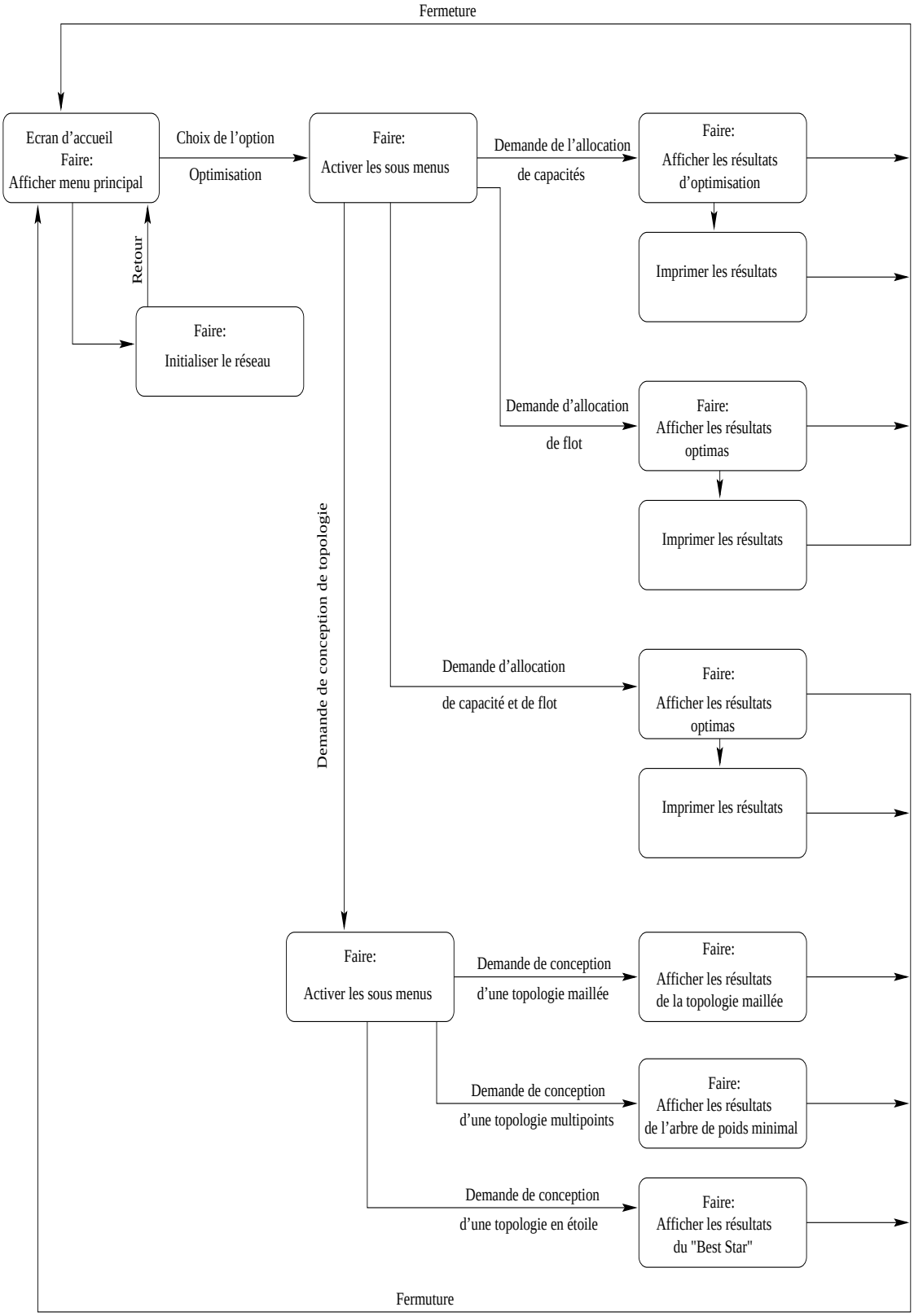


FIG. 2.4 – Diagramme d'états pour la classe "**Optimiseur**"

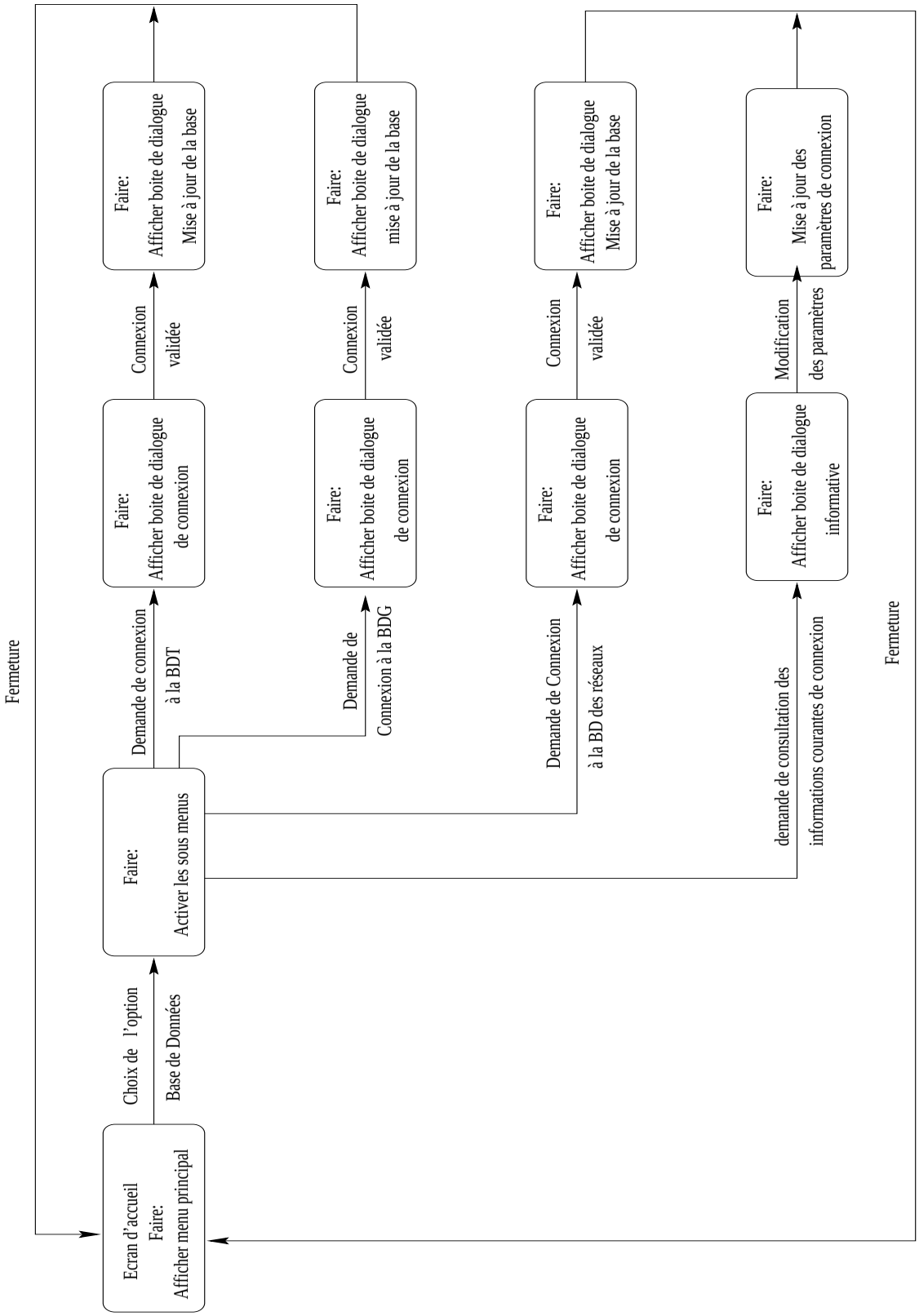


FIG. 2.5 – Diagramme d'états pour la classe abstraite "**Base de Données**"

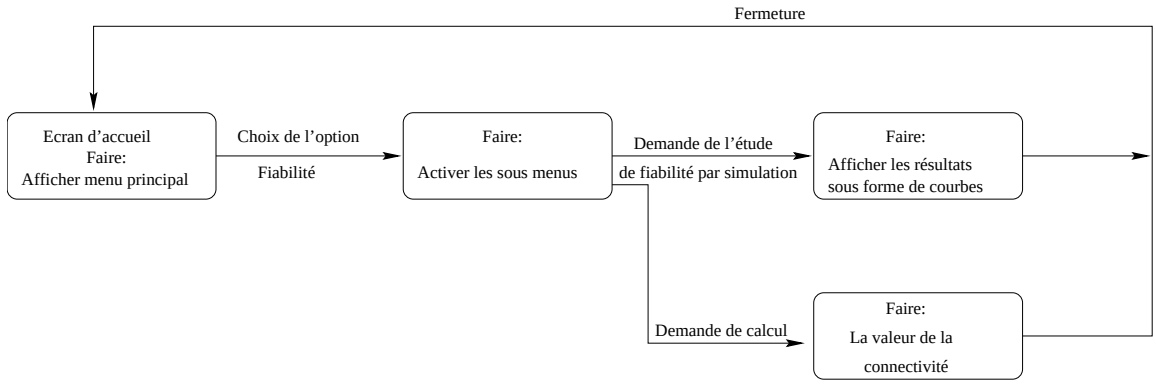


FIG. 2.6 – Diagramme d'états pour la classe "**Simulateur de Fiabilité**"

Dans ce paragraphe, nous décrivons le modèle fonctionnel de l'outil **WanDesigner**. L'outil prend en entrée la description du réseau proposée par le concepteur. Des moteurs intelligents effectuent les calculs d'optimisation nécessaires. Une base de données de services alimente ces algorithmes en données tarifaires. Le résultat en sortie est l'affichage de la topologie optimale. La figure 2.7 montre le diagramme à flots de données du plus haut niveau pour le système **WanDesigner.**

2.4 Conclusion

Dans ce chapitre, nous avons essayé d'extraire et d'analyser les besoins requis par **WanDesigner** et de comprendre son domaine d'application. Le "quoi faire" est, donc, rigoureusement décrit. Moyennant cette analyse nous présentons, dans le chapitre suivant, la démarche à suivre pour la mise en œuvre de **WanDesigner**.

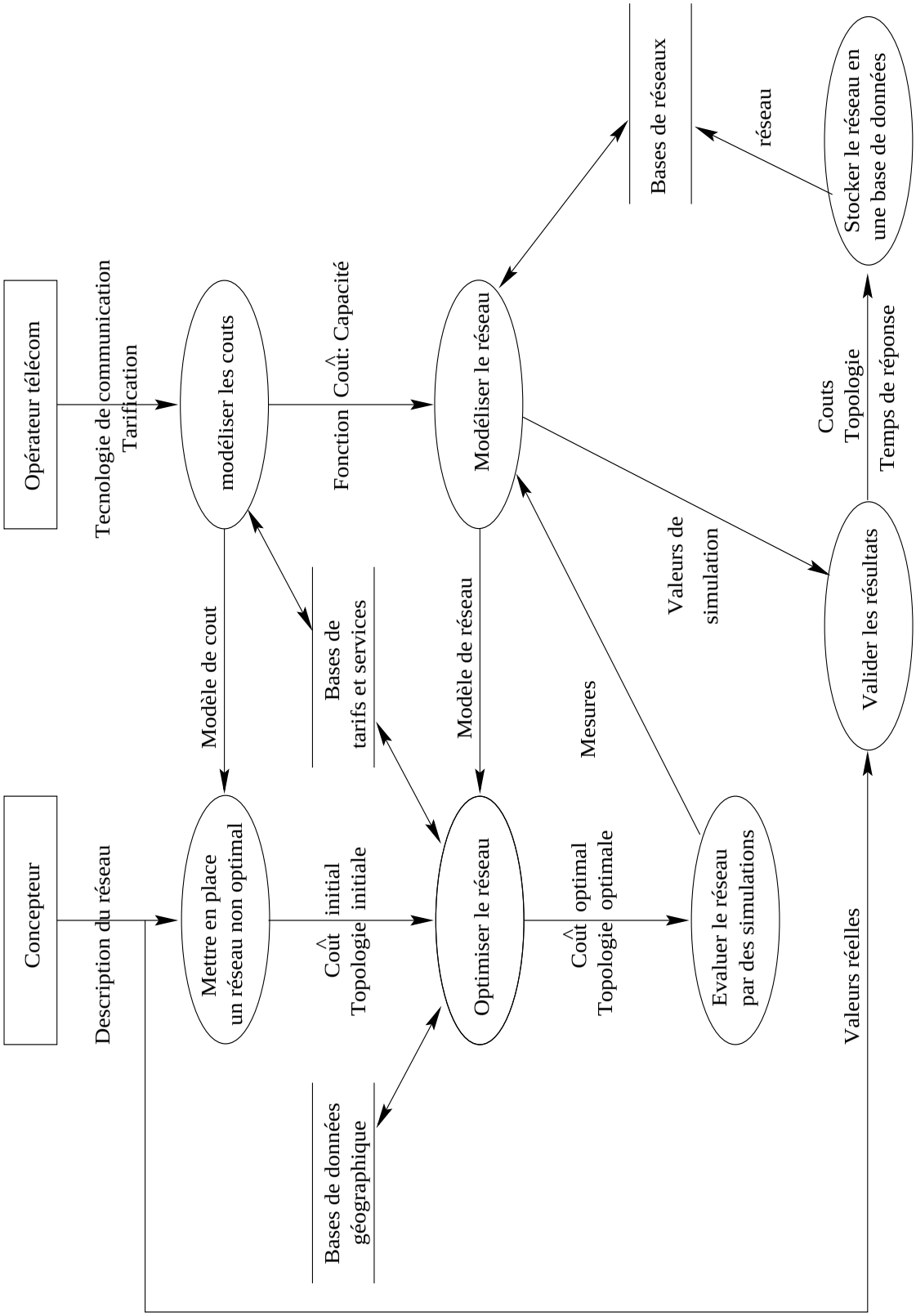


FIG. 2.7 – Modèle fonctionnel de **WanDesigner**

Chapitre 3

Approche adoptée pour la mise en œuvre de WanDesigner

Après avoir analysé les besoins, il nous faut maintenant décider de l'approche à adopter pour la conception de **WanDesigner**. Selon OMT, la conception est guidée par l'organisation du système même en sous-systèmes, allocation de ces sous-systèmes aux composants matériels et logiciels, gestion des réservoirs de données,...

Bien que la taille de notre application ne soit pas assez importante pour parler des sous-systèmes, nous l'avons adopté pour la décomposer, quand même, en trois sous-systèmes indépendants: interface, optimiseurs et gestionnaires des réservoirs de données.

3.1 Architecture de WanDesigner

En terme des fonctionnalités et services rendus, notre système peut être devisé en trois sous-systèmes principaux (cf. figure 3.1):

- L'interface utilisateur.
- Les moteurs intelligents de dimensionnement.
- Le gestionnaire des réservoirs de données.

Le fonctionnement repose sur des moteurs "intelligents". L'application est scindée en deux parties: la phase descriptive et la phase de calcul et d'affichage des résultats:

1. La première étape intègre les étapes de recensement et de description des sites à interconnecter sur un backbone: Topologies, flux, trafic, applications ... Au cours de cette étape s'élabore un scénario décrit lui aussi sous forme de paramètres (applications, équipements, base tarifaire ...).
2. Dans la seconde étape, les données du scénario sont utilisées pour construire et dimensionner le réseau. Durant cette étape, les données du scénario sont utilisées, le logiciel passe ensuite au calcul des coûts proposés, en s'interfaçant avec une base de données. L'utilisateur peut après avoir obtenu les résultats, revenir modifier les paramètres du premier scénario afin de jouer sur les résultats des coûts.

Dans ce qui suit, ces trois composants sont explorés par la description des principaux modules de chacun d'eux. Afin de comprendre le fonctionnement global du système, nous présentons les interfaces de chaque sous-système avec le reste du système.

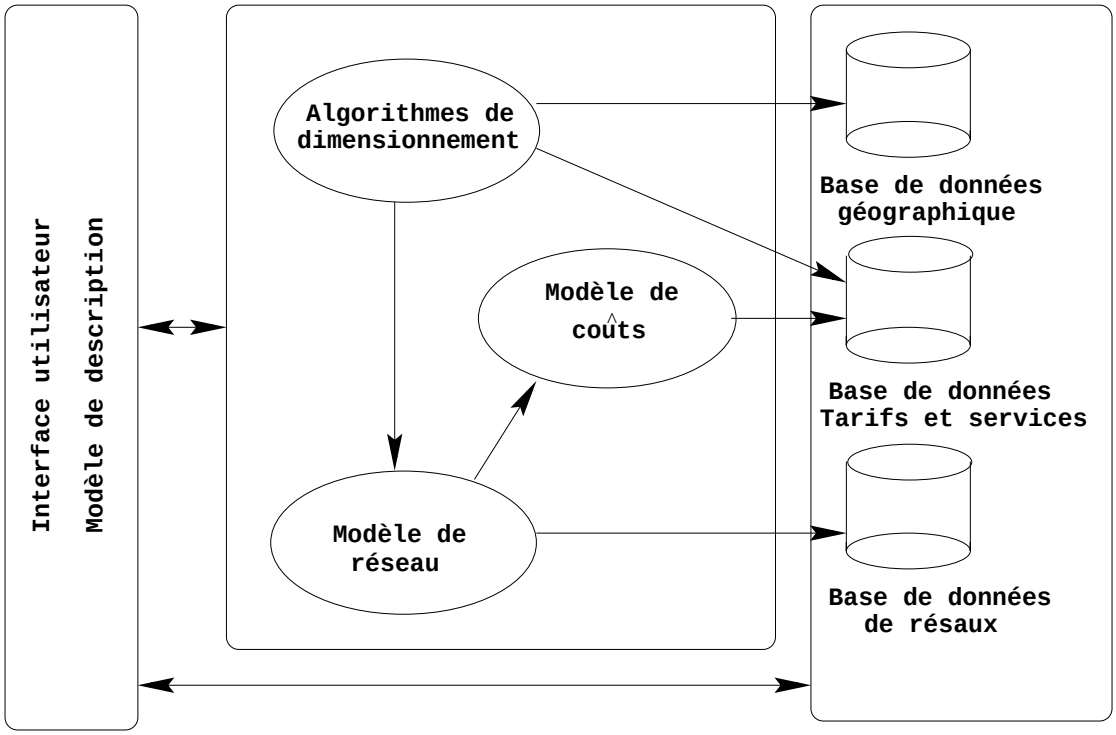


FIG. 3.1 – Interaction des composants du logiciel **WanDesigner**

3.2 L'interface utilisateur

Ce sous-système assure donc l'interaction globale entre l'utilisateur et l'atelier de conception. Les objets de ce sous système sont isolés de ceux qui définissent la sémantique de l'application, ils sont inspirés du modèle dynamique de la figure 2.4. Le contrôle se fait par échanges d'événements entre objets graphiques qui prennent leurs entrées des bases de données ou de sous-système d'optimisation ou directement de l'utilisateur.

L'interface se charge d'offrir les fonctionnalités suivantes:

- La gestion des composants graphiques schématisant le réseau courant. Il doit en particulier gérer la fenêtre d'affichage du réseau et permettre l'augmentation ou la réduction de la zone d'affichage.
- La correspondance entre données sémantiques décrivant les composants du réseau et les objets graphiques schématisant graphiquement le réseau. En particulier, la taille de lignes graphiques doivent différer par les capacités courantes.
- Afin d'alléger la tâche de saisie des données décrivant les caractéristiques physiques du réseau à étudier, ce sous-système doit garantir une souplesse (dans la mesure du possible) dans la représentation des nœuds, des liens et de leurs caractéristiques par l'utilisation des objets graphiques pouvant être manipulés par la souris. Ceci est applicable aussi bien aux composants graphiques qu'aux actions de mise à jour (suppression d'un nœud, déplacement d'un nœud,...).
- La gestion des événements externes (déclenchés par l'utilisateur) et leur traduction en requêtes aux autres sous-systèmes tels que les moteurs d'optimisation ou les bases de données.

– La gestion des erreurs provoquées soit par des données erronées ou par des résultats inattendus.

3.3 Les moteurs d'optimisation

C'est le noyau principal de fonctionnement regroupant les moteurs intelligents d'optimisation. Ce sous-système regroupe trois modules principaux:

3.3.1 Les algorithmes d'optimisation

Ces algorithmes se chargent de la tâche d'optimisation du problème de conception. En plus des données décrivant le réseau à optimiser et qui sont spécifiées par l'utilisateur à travers l'interface graphique. Les algorithmes reçoivent du sous-système de modélisation de coût les formules à utiliser pour calculer les coûts des lignes de communication pour les différentes technologies d'interconnexion.

Il est à signaler à ce stade, que le choix de l'algorithme ou de l'heuristique à utiliser est lié à des contraintes spécifiées par l'utilisateur (exemple routage non alterné) et/ou modèles provenant du module de modélisation de coûts).

Au vu du point précédent, une précaution en besoins de performances semblait être nécessaire. Ainsi, il est utile de prévoir une machine puissante pour l'exécution des moteurs d'optimisation. La communication avec les autres sous-systèmes qui peuvent être déclarés concurrents à ce module peut se faire à travers un modèle **client/serveur**.

3.3.2 Le modèle de coûts

Ce module se charge de la modélisation des coûts des différentes technologies de communication existantes tels que LS, X25, RTC, RNIS (cf. chapitre 3).

La modélisation s'intéresse à l'étude de variation de coût de chaque offre Télécom en fonction de l'une des variables de paiement: capacité, temps, volume de données transférées, services offerts ou une combinaison de certains d'elles.

Cette variation peut être linéaire, concave, discrètes ... Ce module offre les formules de calcul de coûts aux autres modules.

3.3.3 Le modèle de réseau

Ce module regroupe l'ensemble des classes permettant la description globale des différents composants du réseau. Elles offrent les divers fonctionnalités permettant la mise à jour du réseau courant, sa visualisation...

3.4 Gestionnaire des réservoirs de données

Comme réservoir des données, nous avons opté pour l'utilisation des bases de données afin d'assurer les propriétés **ACID** (**A**tomicité, **C**ohérence, **I**solation et **D**isponibilité) des données manipulées.

Le module gestionnaire des réservoirs de données prend en charge les opérations de gestion des trois bases mises en jeux par **WanDesigner**: Base tarifaire, Base géographique

et Base des réseaux. Il permet la connexion à un système de gestion de bases de données distant, tel que SQL Server via un protocole de connexion (cf. annexe B). Toutes les requêtes sont ainsi exécutées selon un modèle **Client/Serveur**. Tout traitement, effectué sur les résultats des requêtes, est exécuté sur la machine cliente (figure 3.2).

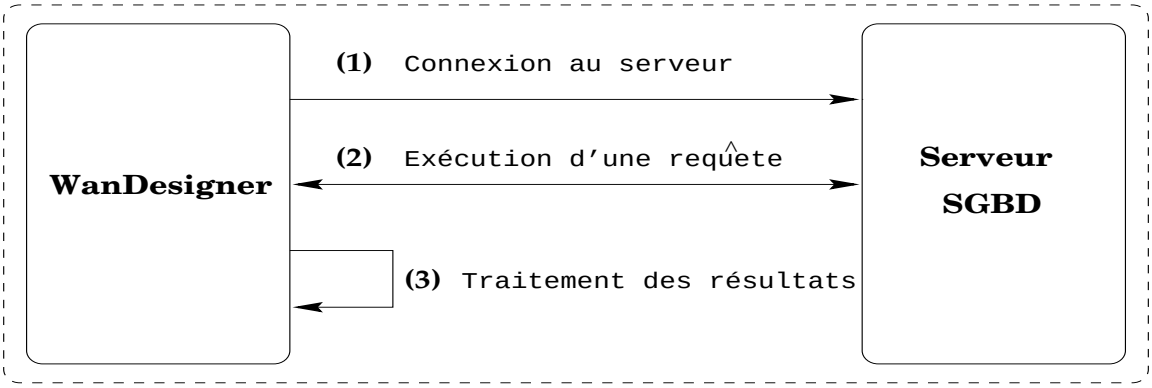


FIG. 3.2 – Module Gestionnaire des réservoirs de données

3.5 Conclusion

Dans ce chapitre, une conception détaillée a été faite. Nous avons dégagé et décrit les sous-systèmes formant l'architecture de **WanDesigner** en spécifiant le rôle de chacun d'eux ainsi que son interaction avec le reste du système.

À la lumière de cette décomposition, nous décrivons de façon complète les structures de données, les classes utilisées dans l'implémentation, les principaux méthodes et attributs des classes jugées importantes. Tous ces points feront l'objet du chapitre suivant.

Chapitre 4

Mise en œuvre de l'atelier WanDesigner

Nous commençons maintenant à déplacer notre angle de vue des spécifications externes vers l'implémentation en machine. Nous ajoutons quelques détails d'implémentation tels que la restructuration des classes pour plus d'efficacité, les structures internes des données et les algorithmes permettant d'implémenter les opérations, les contrôles et les associations.

Dans ce qui suit, nous donnons, pour les objets jugés importants, leurs attributs et leurs méthodes. Toutefois, les algorithmes des procédures internes aux objets (déclarées comme privées) ne seront pas détaillées, nous nous limitons à leur donner une description générale.

4.1 Quelles structures de données?

La performance des algorithmes d'optimisation est fortement liée aux structures de données manipulées. De ce fait, nous devons discuter la structure de données à utiliser pour représenter un réseau informatique.

La première structure, qui vient à l'esprit, est *une matrice d'adjacence N x N* où *N* est le nombre des nœuds dans le réseau. L'élément *(i,j)* est nul s'il n'existe pas un arc joignant les nœuds *i* et *j* et il représente le coût de cet arc dans le cas contraire. Ces coûts seront utiles à la détermination des chemins les plus courts entre chaque paire des nœuds. Cette structure à l'avantage d'être simple, c'est pourquoi nous l'avons adopté pour représenter un objet de la classe *Network*.

L'inconvénient majeur de cette structure est la consommation de l'espace mémoire. Pour le cas de notre application, les réseaux sont en général peu dense pour satisfaire un certain degré de connectivité, généralement, autour de 3. D'où le nombre des éléments significatifs est faible.

4.2 Le sous-sytème Optimiseur

4.2.1 La classe Switch

Cette classe réunit les objets de type nœud. Les nœuds, qui sont les points d'entrée au réseau et sont caractérisés par :

Attributs:

- Name: chaîne. {*nom du Switch*}
- X: réel. {*abscisse graphique du Switch*}
- Y: réel. {*ordonnée géographique du Switch*}
- Identity: entier. {*numéro du Switch*}
- Alpha: réel. {*fraction maximale de trafic délivrée à l'extérieur*}
- Omega: réel. {*fraction maximale reçue de l'extérieure*}
- RoutingTable: matrice. {*table de routage*}

Méthodes:

- distance: calculer la distance entre deux nœuds.

4.2.2 La classe Link

Cette classe réunit les objets de type lien joignant deux nœuds distincts. Un lien est généralement caractérisé par :

Attributs:

- Name : chaîne. {*nom du lien*}
- CapR2L: réel. {*capacité de droite à gauche*}
- CapL2R: réel. {*capacité de gauche à droite*}
- Length: réel. {*longueur du lien*}
- Flow: réel. {*flux passant par ce lien*}
- ResponseTime: réel. {*temps de réponse de ce lien*}
- Type: chaîne. {*type de technologie constituant ce lien*}
- Cost: réel. {*coût du lien*}
- LeftSwitch: Switch. {*extrémité gauche*}
- RightSwitch: Switch. {*extrémité droite*}

Méthodes:

- getEuclidianLength: calculer la distance euclidienne entre les deux extrimités de la ligne.

4.2.3 La classe Network

Cette classe réunit les objets de type réseau. Un réseau est caractérisé par :

Attributs:

- Name: chaîne. {*nom du réseau*}
- SwitchCount: entier. {*nombre de nconstituant le réseau*}
- Switchs: vecteur. {*vecteur contenant les objets de type Switch*}
- Links: vecteur. {*vecteur contenant les objets de type Link*}
- Trafic: matrice. {*matrice de trafic externe}*
- Adjacency: matrice. {*matrice d'adjacence du réseau*}
- ResponseTime: réel. {*temps de réponse moyen du réseau*}
- Cost: réel. {*coût global du réseau*}

Méthodes et Algorithmes :

- **CapacityAssig**: résoudre le problème d'allocation de capacité.

La méthode Network.CapacityAssig

```
CapacityAssig()
Début
 essayer de modéliser la fonction coût du type de technologie
utilisée
 cas technologie dans
  'LS': allocation de capacité pour la technologie LS
  'X25': allocation de capacité pour la technologie X25
  'RTC': allocation de capacité pour la technologie RTC
 fincas
 mettre à jour les capacités
Fin
```

- **StarTopo**: générer aléatoirement une topologie initiale moyennant certaine heuristique.

La méthode Network.StarTopo

```
StartTopo(connect: entier)
Début
 donner un déficit connect pour tout objet Switch du réseau
 répéter
  trouver Switch1 qui a le plus faible déficit
  trouver Switch2 (Switch1) qui a le plus faible déficit (le plus
proche si
  on a plusieurs)
  mettre à jour les déficits
  ajouterLink(Switch1, Switch2)
```

```
 jusqu'à ce que le réseau soit de connectivité "connect"
Fin
```

– **FlowAssig**: résoudre le problème d'allocation de flot.

La méthode Network.FlowAssig

```
FlowAssig(FAorCFA: chaîne)
Début
 si un problème FA alors chercher un flux initial faisable
 sinon prendre le flux courant comme flux initial
 finsi
 répéter
  changer les longueurs des lignes suivant le type du problème
  chercher les flux par les plus courts chemins
  trouver le paramètre optimal de déviation
  calculer la variation de la fonction à optimiser
 jusqu'à ce qu'il n'y ait plus de variation remarquable
Fin
```

– **CFAAssig**: résoudre le problème d'allocation de capacité et de flot.

La méthode Network.CFAAssig

```
CFAAssig()
début
 générer une solution (capacités, flots) faisable
 répéter
  CapacityAssig()
  FlowAssig("CFA")
  calculer la variation de la fonction à optimiser
 jusqu'à qu'il n'y a plus de variation remarquable
Fin
```

– **MailleTopo**: concevoir une topologie maillée optimal.

La méthode Network.MailleTopo

```
MailleTopo()
Début
 répéter
  Génération d'une topologie initiale
  FlowAssig()
  CapacityAssig()
  évaluer le coût du réseau
  si ce réseau est réalisable alors
```

```
      répéter
       générer une variante du réseau courant
       FlowAssign()
       CapacityAssign()
       si cette variante est réalisable et son coût est meilleur alors
        mettre à jour le coût et la topologie
       finsi
      jusqu'à ce qu'il n'y ait plus de variantes de cette topologie ini-
tiale
     finsi
     si le coût courant est inférieur à celui de l'ancien coût alors
      sauvegarder la topologie et le coût
     finsi
    jusqu'à épuisement de tout le budget
   Fin
```

- **BestStar**: concevoir une topologie en etoile optimale.
- **SpanningTree**: concevoir une topologie en arbre optimale.

4.3 Le sous-système Gestionnaire des réservoirs de données: DB-Manager

Cette classe se charge de la gestion des base de données:

Attributs:

- ConnexionProtocol: chaîne. {*protocole de connexion utilisé: msql, odbc, . . .*}
- HostName: chaîne. {*nom de l'hôte distant où tourne le serveur de bases de données*}
- PortNumber: entier. {*numéro du port de connexion*}
- UserName: chaîne. {*nom de l'utilisateur*}
- UserPassword: chaîne. {*mot de passe de l'utilisateur*}
- DBName: chaîne. {*nom de la base de données*}

Méthodes et Algorithmes:

- **Connect**: se conneter à la base de données.

La méthode DBManager.Connect

```
Connect()
Début
 charger le driver de connexion spécifique au SGBD utilisé
 déduire l'URL en combinant les différents attributs
 faire appel à la méthode de connexion du driver
Fin
```

- **Execute**: exécuter une requête.

La méthode DBManager.Execute

```
Execute(type: chaîne, query: chaîne)
Début
 lire la requête
 préparer la requête selon le standard SQL
 faire appel à la méthode ExecuteQuery du driver
 cas type dans
  "Create": remplir la table créée
  "Select": récupérer les résultats de sélection
  "Update": consulter les données modifiées
Fin
```

– **Disconnect**: se déconnecter.

4.4 Le sous-système de l'interface graphique

4.4.1 La classe GUIManager

Cette classe présente la fenêtre principale de **WanDesigner**. Elle s'intéresse à la gestion des événements utilisateurs.

Attributs:

– gpData: general.GeneralPropeties. {*contient les propriétés courantes de WanDesigner*}

Méthodes et Algorithmes:

– **handleEvent**: gérer les événements d'utilisateur et du système

La méthode GUIManager.handleEvent

```
handleEvent(e: Event)
Début
 cas e dans
  "new": créer un réseau aléatoirement
  "open": charger un réseau existant
  "save": sauvegarder le réseau courant
  "Connect": afficher la boîte de dialogue de connexion
  "SpanningTree": Network.SpanningTree()
  "Optimal Flow" : Network.FlowAssig()
Fin
```

– **EndApplication**: terminer l'éxécution de WanDesigner tout en libérant les ressources systèmes utilisées.
– **GetCanvas**: récupérer l'objet GUICanvas qui gère l'affichage de réseaux.
– **DisplayMessageBox**: afficher un message à l'utilisateur suite pour confirmer, alerter, informer…

4.4.2 La classe GUICanvas

Cette classe gère la zone d'affichage du réseau courant. Elle gère et met à jour les paramètres d'affichage du réseau courant.

Attributs:

- editMode: chaîne. {contient le mode d'édition courante: addSwitch, removeSwitch, Zoom,...}
- map: Image. {*contient l'image représentant la carte géographique du réseau*}

Méthodes et Algorithmes:

- **autoScale**: changer l'échelle d'affichage du réseau courant
- **distance**: calculer la distance entre un point donnée (x,y) et le segment de la ligne.
- **translate**: translater le réseau suite à un changement d'échelle.
- **zoomIn**: diminuer la taille du réseau courant.
- **zoomOut**: agrandir la taille du réseau courant.
- **paintNetwork**: dessiner le réseau sur la zone d'affichage courante.

La méthode GUICanvas.paintNetwork

```
paintNetwork(e: Event)
Début
 choisir des paramètres de dessin de n(couleur, taille)
 dessiner tous les ndu réseau
 choisir des paramètres de dessin de liens (couleur, taille)
 dessiner tous les liens du réseau
Fin
```

4.5 Conclusion

Un support détaillé d'implémentation a été fourni dans ce chapitre. Nous nous sommes limités aux classes les plus importantes. La définition de ces classes, des attributs et des méthodes va nous permettre de mettre en œuvre l'outil **WanDesigner**. La description de l'atelier réalisé sera donnée dans le chapitre suivant.

Chapitre 5

Réalisation

Dans ce chapitre, nous commençons par présenter l'environnement matériel et logiciel supportant l'application en argumentant le choix du langage de développement. Ensuite, nous passons en revue sur les tâches réalisées. Enfin, nous terminons par les problèmes rencontrés et le chronogramme décrivant toutes les étapes de mise en œuvre de l'atelier.

5.1 Environnement du travail

5.1.1 Configuration matérielle

Le matériel mis à notre disposition pour la réalisation de ce travail consiste en :

1. Deux micro-ordinateurs
   Marque: Gateway 2000.
   Processeur: pentium 120 MHZ.
   DD:2 GO.
   OS: Deux partitions systèmes Windows 95 et LINUX.
2. Station de travail
   Marque: SUN
   Processeur: RISC.
   DD: 3 GO.
   OS: Solaris 5.5
3. Une imprimante:
   Marque: HP.
   Type: laser.

5.1.2 Configuration logicielle

Cette application a été développée en utilisant le langage Java comme langage de développement. Ce choix découle, en fait, de ses avantages énormes et de ses capacités d'intégrer l'utilisation des réseaux. En effet, dans "The Java langage: A white paper", qui est un document de Sun dérivant le langage, Java est gratifié de la définition suivante:

Java: un langage simple, orienté objet, réparti, interprété, robuste, sûr, indépendant de l'architecture, portable, efficace, mutithread et dynamique.

En d'autres termes, le potentiel révolutionnaire de Java n'est pas dû à une seule fonction du langage, mais à ses quatre principales caractéristiques: *le savoir-faire réseau, la portabilité, la sécurité et l'orientation objet*. Quelques-unes de ces fonctions sont inhérentes au langage, tandis que d'autres sont puisées dans les bibliothèques des classes. Ces fonctions rendent un éventail de services dont on ne pouvait que rêver avant l'arrivée de Java.

Du savoir faire-réseau, nous avons pu faire apparaître l'application **WanDesigner** en plein milieu de document HTML du serveur Web conçu pour être exécuté à distance. Ce savoir a permis aussi l'accès aux différentes bases de données mises en jeu par **WanDesigner** via l'API JDBC de Java (cf. annexe B).

Aussi, grâce à la portabilité de Java, WanDesigner est capable de s'exécuter sur n'importe quel type d'architecture avec n'importe quel type de système d'exploitation sans rien changer dans le code source. C'est d'ailleurs, l'avantage qui permettra de séparer éventuellement les sous-sytèmes concurrents de **WanDesigner** (interface et moteurs d'optimisation) pour les affecter aux ressources matérielles voulues.

Quant à la sécurité, c'est une propriété essentielle assurant le contrôle d'accès aux ressources systèmes quand l'application **WanDesigner** est exécutée via le Web.

Enfin, par l'aspect orienté objet de Java, nous avons pu penser l'atelier en termes de données et de méthodes manipulant ces données, plutôt qu'en terme de procédures. Cet aspect garantira l'extensibilité de l'atelier par l'ajout de nouvelles fonctionnalités jugées utiles.

5.2 Travail réalisé

Dans les limites des quatre mois qui nous ont été confiés et sous l'environnement matériel et logiciel qui a été mis à notre disposition, nous sommes arrivés à réaliser ce travail modeste, qui constitue, en fait, les briques de base d'un projet ne pouvant être achevé qu'à long terme. Les objectifs, que nous avions tracés depuis le commencement de ce projet, étaient de réaliser un outil de conception et d'optimisation de réseaux et ce, compte tenu des besoins de communication d'une part, et des supports et technologies existantes d'autre part.

Du fait de l'importance des coûts de planification des réseaux, ce sujet mérite d'être étudié en détail. Ce n'est pas surprenant que le développement de ce travail a nécessité plus de 10000 lignes de code. En effet, plusieurs aspects ont été traités, ces aspects touchent, entre autres, le calcul, l'interface, la fiabilité, le stockage de données...

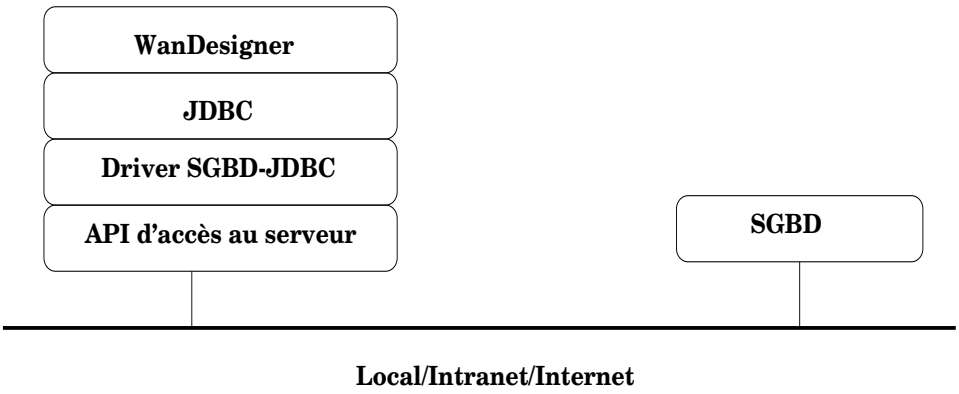
5.2.1 Aspect calcul

Pour la résolution du problème d'optimisation et la résolution des problèmes d'allocation des ressources, nous avons implémenté quelques algorithmes choisis à base de faible complexité et rapidité de convergence. Ces algorithmes d'optimisation s'attachent principalement, du moins compliqué au plus délicat, à : l'allocation de capacité, l'allocation de flot, l'allocation de flot et de capacité et le problème général de conception de topologie.

5.2.2 Aspect stockage de données

Pour des raisons de cohérence et intégrité des données, nous avions prévues 3 bases de données: géographiques, tarifaires, et de réseaux, qui sont accessibles de notre application

java via un driver JDBC[1]. Toutes les opérations de lecture, d'écriture, de consultation des données s'effectuent de façon transparente à l'utilisateur (cf. annexe B).



TAB. 5.1 – L'accès aux bases de données de WanDesigner

5.2.3 Aspect interface

Pour faciliter la communication avec l'utilisateur, et pour mettre en relief les résultats des algorithmes d'optimisation, nous avions consacré énormément de temps pour la conception de l'interface afin de parvenir à l'adapter aux besoins de l'utilisateur et non pas l'inverse. Pour cela, il n'est pas étonnant que le développement des classes de l'interface a nécessité plus de 4000 lignes de code. Notre objectif est d'aider le concepteur à concevoir facilement et rapidement des réseaux étendus.

L'interface conçue a été hiérarchisée en niveaux reflétant la nature des problèmes d'optimisation fréquemment abordés par le concepteur du réseau. Elle permet la visualisation des résultats d'optimisation, topologie du réseau, capacité des lignes, budget minimal, courbes critiques . ..

5.2.4 Réalisation d'un serveur Web

Ce serveur sera mis directement sur les serveurs de l'ENSI et du CCK, **WanDesigner** peut être lancé en mode applet directement à partir de ce site (figure 5.2). Le serveur réalisé contient toutes les informations nécessaires sur le travail développé: code source, aide, rapport, transparents, informations sur les classes implémentées. . . (figure 5.3).

5.3 L'atelier en chiffres

Tout au long du développement, notre souci était d'écrire un code lisible et d'appliquer les règles du génie logiciel. Nous pouvons dire qu'à ce niveau, notre but est atteint grâce aux points suivants:

– la décomposition poussée des classes à l'intérieur d'un package;

1. Pour toute documentation sur les classes JDBC, consulter le site http://splash.javasoft.com/jdbc

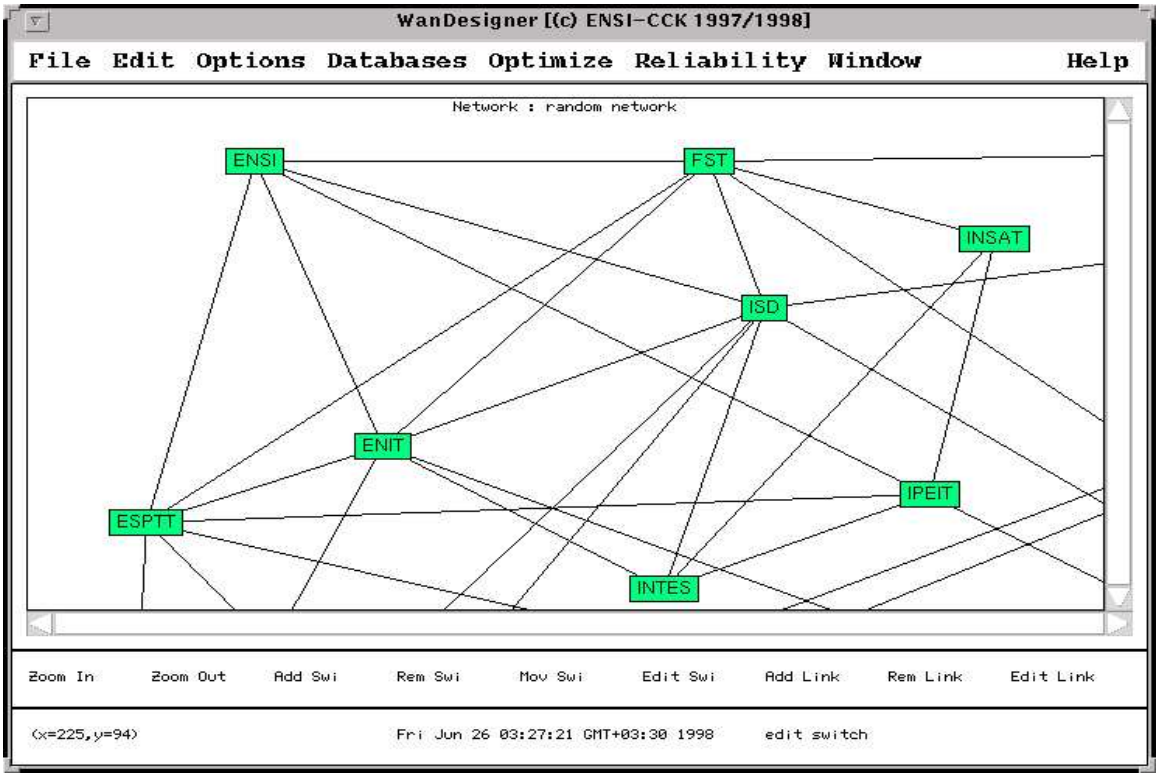


FIG. 5.1 – Fenêtre principale de **WanDesigner**

- les commentaires qui débutent chaque classe, chaque méthode et chaque déclaration des variables;
- les commentaires à l'intérieur des méthodes s'il parait nécessaire;
- les identificateurs significatifs accordés aux fichiers [2], attributs, méthodes et variables.

En terme de chiffres, l'application est l'ensemble de 4 packages et 61 fichiers.

Dans le tableau 5.2, nous donnons une brève description des packages et la correspondance avec les fichiers.

5.4 Difficultés et Problèmes rencontrés

Bien que nous ayons réussi le travail qui nous a été confié, nous tenons quand même à souligner les problèmes rencontrés dont nous avions pu détourner d'une certaine manière.

Tout d'abord, nous signalons que nous avions trouvé des problèmes lors de l'étude des technologies de communication. En effet, la procédure de facturation de l'accès à un médias n'obéit à aucune loi, et faire intégrer ces données tarifaires dans un système de conception n'est pas facile à réaliser. Pour se rapprocher de la réalité, nous avons fait une approximation logarithmique aux fonctions affines par intervalles dégagées de cette étude.

À ce problème, s'ajoute la complexité des algorithmes d'optimisation. L'implémentation de l'algorithme de **déviation de flux** pour la résolution du problème du routage a néces-

2. Dans java, le fichier et la classe doivent avoir le même nom.

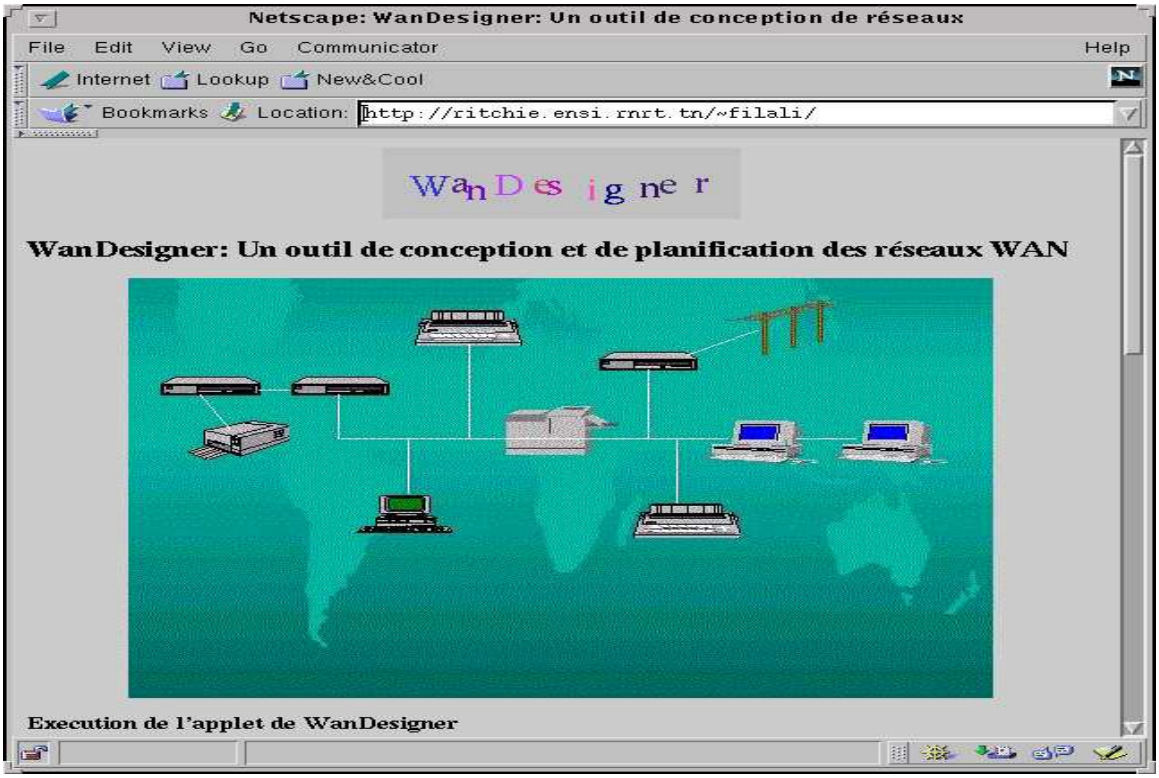


FIG. 5.2 – La page d'accueil du serveur Web de **WanDesigner**

1 8 1 1 0 0 0Nom du package	Description	Classes attachés
gui	C'est le module gérant l'interface utilisateur	GUIManager
		GUICanvas
network	C'est le module regroupant les classes	Network
	d'optimisation	Switch
		Link
		SelectedLink
database	C'est le module responsable de la gestion	DBManager
	des accès aux bases de données	DBTarifs
		DBNetwork
		DBGeographic
general	C'est le module regroupant les classes qui	GeneralProprieties
	ne peuvent pas être mis dans les autres modules	Utility

TAB. 5.2 – Correspondance entre packages et classes

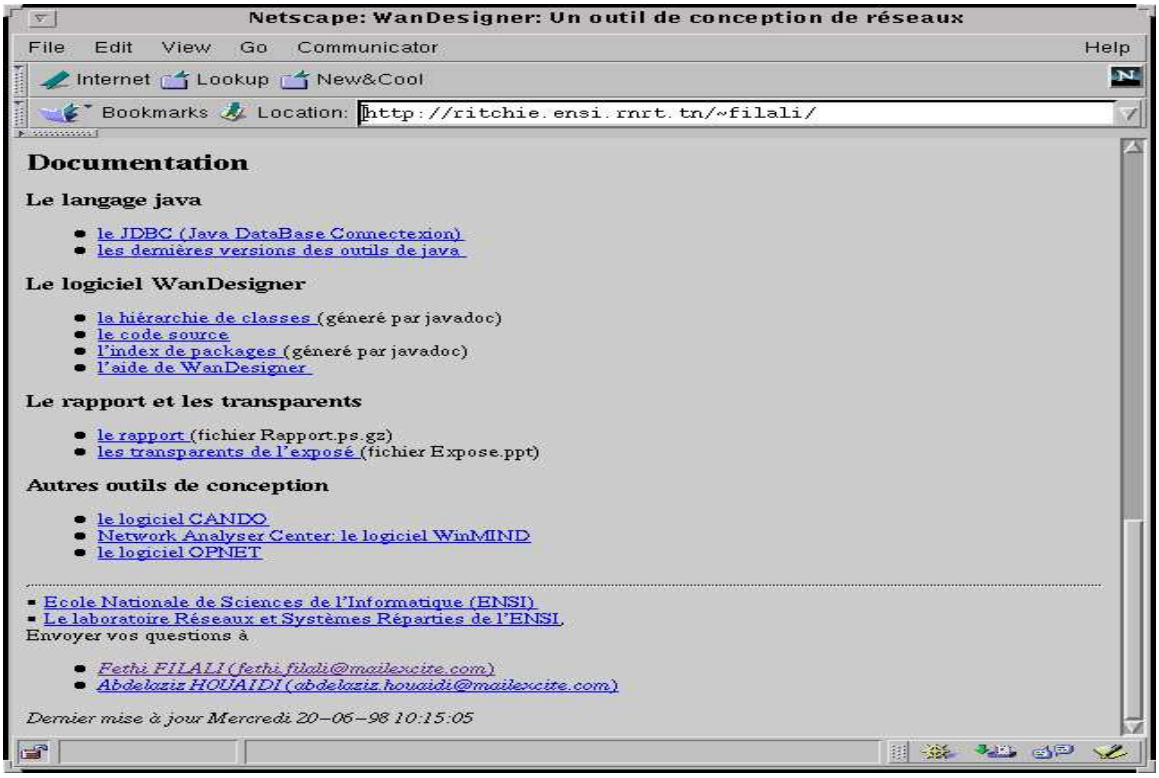


FIG. 5.3 – Les informations fournies par le serveur Web

sité une longue période surtout le choix des valeurs de précision qui affectent beaucoup la convergence de l'algorithme. Aussi, la détermination de la solution initiale semblait d'une importance majeure sans laquelle le problème est déclaré non résoluble. Pour faire face à ce problème, nous avons scindé notre choix des algorithmes sur le facteur temps d'exécution. Nous avons également choisi une plate-forme matérielle puissante de type machine parallèle, Sparc 1000, pour l'exécution de ces calculs.

Le stockage des données tarifaires posait lui aussi un problème majeur: d'une part, il faut cacher le scénario à l'utilisateur, et d'autre part, nous nous trouvons confrontés par une notion aussi nouvelle celle de l'API JDBC de Java (cf. annexe B). Pour comprendre la philosophie, nous avons essayé sur une maquette Java/SQL. Ceci a présenté au début une difficulté puisqu'il nous a fallu aller plus loin dans les concepts de Java. En particulier, nous avons installé le SGBD mSQL[3] en passant par la conception des bases de données relationnelles.

5.5 Chronogramme

Le chronogramme du projet est représenté dans la figure 5.4. Il reflète le cycle de vie proposé par la méthode de conception OMT. Celui-ci se base sur une approche itérative, ce qui explique le chevauchement des étapes de l'avancement de la mise en œuvre.

La documentation s'est prolongée jusqu'à la fin du projet et celle du rapport. En l'occurrence, la dernière partie de la documentation a servi pour l'élaboration des annexes.

3. Ce SGBD est gratuit sur le site http://www.Hughes.com.au/

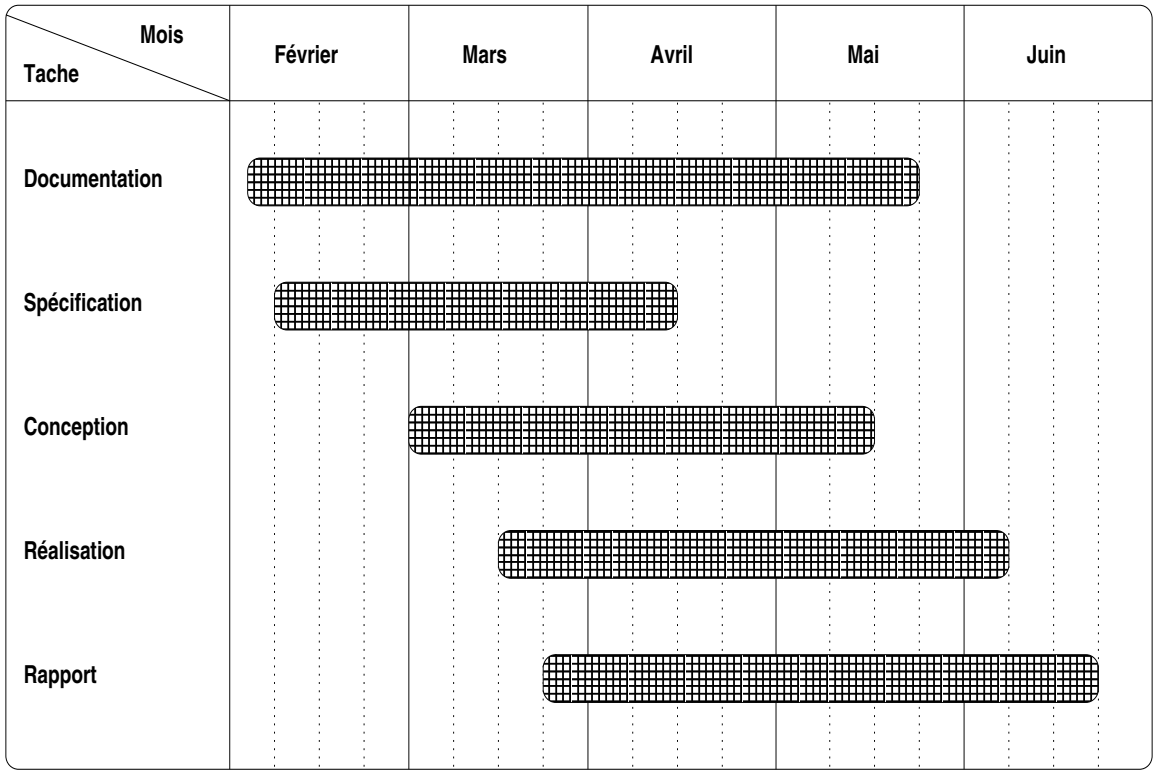


FIG. 5.4 – Chronogramme

La réalisation et la rédaction du rapport ont avancé presque en parallèle. Les résultats expérimentaux inclus dans ce rapport sont été extraits directement de l'exécution des algorithmes d'optimisation implémentés.

Chapitre 6

Conclusion et perspectives

L'objectif escompté de ce travail de fin d'études est de réaliser un atelier de conception et d'optimisation de réseaux étendus, compte tenue des trafics véhiculés par le réseau, des coûts budgétaires disponibles, des temps de réponse et du type de topologie : maillée, multipoint ou en étoile. Ce dernier permet la communication avec des bases de données pour la récupération des tarifs télécom qui caractérisent les différentes technologies d'interconnexion de réseaux informatiques qui seront utilisées pendant la conception et l'optimisation des réseaux.

Rappelons brièvement que dans la première partie de ce rapport, nous avons commencé par introduire le modèle de **Kleinrock** adopté par notre démarche de conception de réseaux. À la suite de cette présentation, il ressort que plusieurs variables et contraintes sont à la base de la définition d'au moins quatre types de problèmes d'allocation de ressources différents.

À cet effet, en se fixant comme finalité la conception optimale de réseaux étendus compte tenue de l'offre télécom d'une part et des besoins de l'utilisateur d'autre part, le troisième chapitre a présenté une étude des technologies de communication existantes sur le marché et ce, dans le but de caractériser la fonction coût-capacité.

À la lumière de cette étude et compte tenu de ce qui précède, il semblait intéressant de traiter séparément les problèmes d'allocation de capacité, de flot, de capacité et de flot et de topologie. Ces problèmes ont fait l'objet respectivement du quatrième, cinquième et septième chapitre.

Pour terminer, nous avons étudié le cas du Réseau National Universitaire. Une description de l'architecture du réseau nous a permis de proposer une topologie particulière faisant intervenir le maillée et le multipoint en même temps.

La deuxième partie de ce rapport a été consacrée aux étapes du processus de mise en œuvre de l'outil, depuis l'analyse jusqu'à la réalisation et implémentation des algorithmes. Nous avons choisi la méthode OMT, comme méthode de conception orientée objet, pour des raisons d'extensibilité et de réutilisabilité.

En achevant ce rapport, il nous est indispensable de signaler les côtés bénéfiques incontestables de ce projet, ne serait ce que:

sur le plan théorique:

- Il nous a permis de découvrir la complexité des problèmes d'allocation optimale des ressources et l'efficacité de certaines heuristiques pour la résolution de certains problèmes aussi compliqués que le problème de la topologie maillée optimale;

- Il nous a permis d'apprendre un nouveau langage de programmation (Java), de mieux maîtriser les concepts orientés objets et de les appliquer dans le développement de notre application;
- C'est une occasion qui nous a permis d'apprendre une nouvelle méthode de conception orienté objet aussi à la mode que "OMT";
- C'est la première fois qu'on aborde une notion aussi nouvelle que celle de JDBC pour l'accès transparent aux bases de données relationnelles via l'API de Java;

Sur le plan professionnel:

- ce stage nous a incité à poursuivre des études approfondies dans la voie de conception de réseaux utilisant différentes technologies d'interconnexion.

Il est certain que plusieurs améliorations peuvent être apportées à ce travail. Ces extensions feront l'objet d'autres P.F.E et elles se résument par les points suivants:

- La parallélisation de certains algorithmes d'optimisation tel que le **Recuit Simulé** pour faire face au temps d'exécution énorme consommé par les problèmes de grandes tailles;
- L'extension de l'application afin de permettre la conception des réseaux LAN par l'ajout des nouveaux modules de dimensionnement;
- Le développement d'autres algorithmes d'optimisation pour la conception des réseaux faisant intervenir des nouvelles technologies (RNIS, ATM, Frame Relay ...);
- Implémentation d'autres algorithmes de recherche globale tels que les algorithmes génétiques, algorithme tabou,...
- La validation des résultats par des simulations;
- La communication avec un Système d'Information Géographique (tel que *ArcInfo* d'ESRI [1]) pour mieux exploiter les données géographiques.

1. ESRI: **E**nvironmental **S**ystem **R**esearch **I**nstitution

Bibliographie

[ALX 97] Alexandre FENYO, F. G., S.T. *"Raccorder son réseau d'entreprise à l'internet"*, Eyrolles 1997.

[AND 94] J. Andrew FINGERHUT, "*Dissertation*", May 1994.

[BER 92] Dimitri BERTSEKAS & Robert GALLAGER, "*Data Networks*", Prentice Hall, 1992.

[CAM 92] Fabien CAMPILLO, *"Optimisation Discrêtes par Recuit Simulé et Machine de Boltzman"*, DESS Mathématique 92/93

[DAR 80] D.W DARIES , D.L.A BARBER, W.L PRICE and C.M SOLOMONIDE, "*Computer Networks and their protocols*", A wiley. Interscience Publication, 1980.

[ESC 96] Newine ESCHEDLY, "*Techniques de communication*", EPT 96.

[FIT 78] Jerry FITZGERALD & Tom S.EASON, "*Fundamentals of data communication*", A Wiley/Hamilton Publication, 1978.

[FLA 96] David FLANAGAN, "*Java in a nutshell*", Editions O'Reilly International Thomson. 1996.

[FOR 92] Ford, L. R . Jr. and D. R. FULKERSON, *"Flows in Networks"*, Princetron, N.J., 1992.

[GER 73] Mario GERLA, "*The design of store-and-forward (s/f) networks for computer communications*", Computer Science Departement, School of Engineering and Applied Science University of California Los Angeles, 1973.

[HAD 64] Hadley, G., *"Non linear and Dynamic Programming"*, Addison-Wisley, Reading, Mass, 1964.

[HAM 97] Graham HAMILTON & rick CATTEL, *"JDBC^{TM}: A java SQL API version 1.2"*, January 10, 97.

[HU 68] HU, T.C., *" A Decomposition Algorithm for Shortest Paths in a Network"*, operations Research, 16:91-102, 1968.

[JAC 57] J. R. JACKSON , *"Networks of waiting lines"*, Operations Research, 5:518-521, 1957.

[KAM 98] F. KAMOUN, "*Cours nouvelles technologies des réseaux (II3/SRI)*", ENSI 1998.

[KLE 64] Leonard KLEINROCK, "*Communication Nets: Stochastic Message Flow and Delay*", McGraw-Hill, New York, 1964.

[KLE 76] Leonard KLEINROCK, "*Queuning systems Volume II: Computer Applications*", A Wiley-Interscience Publication, 1976.

[KRO 95] KRO & Ferguson, "*Le monde internet et windows 95*", Edition O'reilly , 1995.

[KUO 81] Franklin F.KUO, "*Protocls and techniques for data communication networks*", Prentice Hall Inc. Englewood Cliffs, 19981.

[MIN 83] Michel MINOUX, "*Programmation mathématique: théories et algorithmes tome 1*", Collection Dunod, Paris 1983.

[NUS 87] Henri NUSSBAUMER, "*Téléinformatique II: Conception des réseaux, Réseaux, Transport*", Press Polytechniques Romandes, 1987.

[OUR 98] Tchatikpi OURO-TOU et Amor LAZZEZ, "*PFE: Influence de topologie d'architecture parallèle sur les méhodes d'optimisation locales généralisées*", ENSI 1998.

[PRI 96] Christian PRINS, "*Algorithmes de graphes*", Eyrolls Decembre 1996.

[ROM 98] Imed ROMDHANI & Mahrez BEN HASSEN, "*PFE: Mesure et Caractérisation du Trafic Internet du Réseau National Universitaire (RNU)*", ENSI 1998.

[RUN 87] James RUNBAUGA et al., "*OMT 1.Modélisation et conception orientées objet*", Masson, 1987.

[SIA 96] Patrick SIARRY, " *la méthode du récuit simulé, théories et applications*", Ecole Centrale de Paris: Laboratoire d'Electronique et de Physique Appliquée, 1996.

[SUN 96] Sun Microsystems Inc, "*The JDBC API version 1.10*", October 2, 1996

[TAN 81] Andrew S. TANENBAUM, "*Computer Networks*", Prentice Hall International edition (PHI), 1981.

[TEC 96] Hughes TECHNOLOGIES, "*Mini SQL 2.0 User Guide*", Huges TECHNOLOGIES 96.
Web: http://ww.Hughes.com.au/

[TEL 98] Cisco Ssytems, "*Réseaux & Télécoms*", Mars 1998 n° 122.

Glossaire

A

API: *Application Programming Interface:* interface pour langages de programmation, définie de façon formelle, permettant à une application d'accéder à des programmes système pour, par exemple, communiquer (envoi et réception) ou extraire des données.

ATI: *Agence Tunisienne d'Internet:* fournisseur tunisien d'accès Internet.

ATM: *Asynchronous Transfer Mode:* technique avancée de communication rapide qui sera utilisée dans les futures réseaux large-bande.

AWT: *Abstract Window Toolkit:* ensemble des bibliothèques graphiques.

B

BDG: *Base de Données Géographiques:* base de données renfermant de données géographiques à référence spatial.

BDR: *Base de Données de Réseaux:* lieu de sauvegarde de réseaux ayant été objet d'une optimisation.

BDT: *Base de Données Tarifaires:* réservoir de données tarifaires de différentes technologies d'interconnexion existantes.

BE: *Branch Exchange:* méthode utilisée pour la perturbation de réseau au cours d'optimisation de la topologie.

C

CA: *Capacity Assignment:* problème d'allocation de capacité.

CCK: *Center de Calcul de Khawarismi:* founisseur d'accès Internet des institutions universitaires et centres de recherches tunisiens.

CFA: *Capacity and Flow Assignment:* problème d'allocation de capacité et de flot.

F

FA: *Flow Assignment:* problème d'allocation de flot (problème de routage).

FD: *Flow Deviation:* méthode utilisée pour la recherche du routage alterné optimal.

FDDI: *Fiber Distributed Data Interface:* réseau local départemental à haut débit 100 Mbit/s.

FR: *Frame Relay:* réseau public à relayage de trames à débit 2Mbit/s.

G

GAP: *Generalised Assignment Problem:* problème général d'optimisation du réseau, recherche de capacités, flots et topologie optimaux.

J

JDBC: *Java Data Base Connectivity:* ensemble de classes Java offrant un accès à un serveur de système de gestion de base de données.

L

LAN: *Local Area Network:* réseau local ou réseau d'entreprise ne dépassant pas le domaine privé.

LS: *Ligne Spécialisée:* ligne d'interconnexion permanent entre deux points d'accès aux deux réseau différents.

M

mSQL: *mini Standard Query Langage*: système de gestion de bases de données développé par Hughes Technologies.

P

PAD: *Packet Assembler Disassembler:* protocole utilisé dans X25 pour assembler et desassembler les paquets.

PPP: *Point to Point Protocol:* protocole utilisé pour la connexion de deux sites afin d'utiliser les services de TCP/IP.

R

RNIS: *Réseau Numérique à Intégration de Services:* réseau mettant à profit les techniques numérisation pour transporter sur la même infrastructure plusieurs services concernant la voix, les données ou les images.

RNU: *Réseau National Universitaire:* réseau tunisien interconnectant les institutions universitaires et les centres de recherches.

RPC: *Remote Procedure Call:* appel de procédure à distance. Application d'exécution répartie en réseau.

RTC: *Réseau Téléphonique Commuté:* réseau public basé sur la commutation de paquets.

S

SGBD: *Système de Gestion de Base de Données:* système garantissant les propriétés d'atomicité, cohérence, isolation et disponibilité de bases de données.

SQL: *Structured Query Langage:* langage d'interrogation de bases de données.

SLIP: *Serial Link Interface Protocol:* protocole utilisé pour la connexion de deux sites afin d'utiliser les services de TCP/IP.

O

ODMG: *Object Database Management Group:* groupe de normalisation des standards de gestion de bases de données orientée objets.

OMT: *Object Modeling Technique:* méthode de conception orientée objets.

ORD: *Object Request Broker:* technique d'interrogation d'une base de données.

W

WAN: *Wide Area Network:* réseau géographiquement étendu.

Troisième partie

Annexes

Annexe A

Complément théorique

A.1 Loi de Poisson

La loi de **Poisson** est très utilisée dans la théorie des files d'attente, car elle correspond assez bien à la réalité dans un système informatique comportant un grand nombre d'usagers. En effet, cette loi est basée sur les trois hypothèses suivantes:

1. les arrivées des messages sont indépendantes les unes des autres. Cette hypothèse est très bien vérifiée lorsque les messages parviennent d'un grand nombre de sources aléatoires différentes;
2. le processus est stationnaire, c'est-à-dire que ses caractéristiques ne varient pas en fonction de l'origine du temps d'observation;
3. pour un intervalle de temps très faible Δt, la probabilité $\lambda \Delta t$ d'arrivée d'un seul message est beaucoup plus grande que la probabilité d'arrivée de plusieurs messages.

A partir de ces hypothèses, la probabilité $P_n(T)$ d'arrivée de n messages pendant T secondes est de la forme [NUS 87]:

$$P_n(T) = \frac{(\lambda T)^n e^{-\lambda T}}{n!} \tag{A.1}$$

Considérons maintenant la distribution des intervalles de temps τ entre les arrivées successives de messages. Si $f(\tau)$ est la densité de probabilité de τ, la probabilité $f(\tau)d\tau$ pour que l'intervalle de temps entre deux messages successifs soit égale à τ correspond au cas où aucun message reçu pendant le temps τ et où le message est arrivé entre les instants τ et τ+$\Delta\tau$. La densité de probabilité des intervalles entre arrivées est donnée par la loi exponentielle [NUS 87]:

$$f(\tau) = \lambda e^{-\lambda \tau} \tag{A.2}$$

A.2 File d'attente M/M/1

Nous considérons ici la file d'attente la plus simple, équipée d'un seul serveur qui, dans notre cas, sera une ligne dont le débit est égal à C bit/s (cf. figure A.1). La file d'attente est gérée selon une discipline PAPS (Premier Arrivé Premier Servi) et la taille du tampon ainsi

que la population des messages sont supposées infinies. Les messages arrivent dans la file selon une distribution de **Poisson** avec une valeur moyenne de λ messages par seconde, et nous admettrons que la longueur des messages est distribuée exponentiellement avec une moyenne égale à $1/\mu$ bits par message. La densité de probabilité *f(m)* de la longueur des messages est donc donné avec (A.2) par:

$$f(m) = \mu e^{-\mu m} \tag{A.3}$$

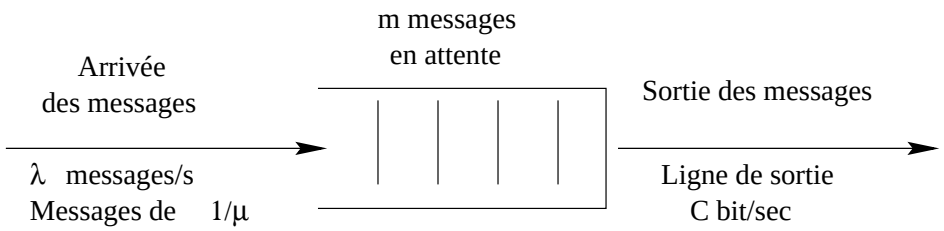


FIG. A.1 – File d'attente M/M/1

Comme la ligne de sortie a un débit fixe égal à C bit/s, l'hypothèse d'une distribution exponentielle de la longueur des messages implique également une distribution exponentielle des temps de service, dont la densité de probabilité *f(t)* est donnée par:

$$f(t) = \mu C e^{-\mu C t} \tag{A.4}$$

avec une valeur moyenne égale à $1/\mu C$. Les lois d'arrivée et de service sont donc exponentielles, et ceci conduit à adopter la notation M/M/1 pour représenter la file d'attente.

Le choix de lois exponentielles pour représenter les processus d'arrivée et de service répond au souci de simplifier les calculs tout en assurant une représentation relativement raisonnable des phénomènes observés. En effet, nous avons vu dans la section précédente, que lorsque les messages sont produits par un grand nombre de sources indépendantes, leur arrivée obéit à une loi de **Poisson** et la distribution des intervalles de temps entre arrivées est exponentielle. Cette hypothèse est donc proche de la réalité.

En ce qui concerne le processus de sortie, l'approximation apportée par le modèle est plus grossière, car la loi de service exponentielle implique que la longueur des messages est elle-même distribuée selon une loi exponentielle, ce qui est en toute rigueur impossible puisque cette loi suppose que les messages puissent avoir une longueur quelconque, alors qu'en pratique ils ont toujours une longueur multiple d'un bit ou d'un caractère. De plus, il n'y a pas a priori de raison pour que les sources produisent des messages dont la longueur est distribuée exponentiellement.

Le temps de résidence moyen $\overline{T}$ dans la file se déduit aisément du nombre moyen de messages en attente, puisque la ligne de sortie débite en moyenne μC messages par seconde. On a donc [NUS 87]:

$$\overline{T} = \frac{\rho}{\mu C(1-\rho)} \tag{A.5}$$

En pratique, l'utilisateur est généralement intéressé par le temps moyen $\overline{T_R}$ de traversée de la file par un message. Ce temps est la somme du temps de résidence et du temps de transmission $1/\mu C$ sur la ligne de sortie, avec

$$\overline{T_R} = \frac{1}{\mu C} + \frac{\rho}{\mu C(1-\rho)} = \frac{1}{\mu C - \lambda} \tag{A.6}$$

Le temps moyen de traversée et le nombre moyen de messages sont liés par la relation simple:

$$\lambda \overline{T_R} = \overline{m} \tag{A.7}$$

Cette relation, qui est appelée **formule de Little**, est tout à fait générale et s'applique dans de nombreux cas de files d'attente, avec des hypothèses beaucoup moins restrictives que celles qui président de la file M/M/1. Cette formule indique que dans un système de files d'attente, le produit du taux moyen d'arrivée des messages par le temps de transfert moyen est égale au nombre de message en attente dans le système. Cette relation sera fréquemment utilisée par la suite.

A.3 CA: Optimisation du temps de réponse du réseau

Donnée:

- Topologie;
- Trafic.

Variables:

- Capacités.

Objectif:

- Minimiser le coût T.

Contraintes:

- $D \leq D_{max}$.

Rappelons que nous sommes toujours sous les mêmes hypothèses de **Kleinrock**. De ce faire, et d'après ce qui a été démontré dans la section 1.1.2, l'expression du temps de réponse moyen T est donnée par 1.6:

$$\overline{T} = \overline{n} \sum_{i=1}^{M} \frac{\lambda_i}{\lambda} \overline{T_i} = \frac{1}{\gamma} \sum_{i=1}^{M} \left(\frac{\lambda_i}{\mu C_i - \lambda_i}_i \right) \tag{A.8}$$

L'objectif $\overline{T}$ est la somme de fonctions séparables convexes donc il est convexe. De plus, l'espace des capacités réalisables est convexe, donc les minima locaux sont aussi des minima globaux [GER 73] et peuvent être déterminés facilement en utilisant la méthode des multiplicateurs de Lagrange [HAD 64].

Tout d'abord nous ignorons la contrainte $C_i \geq \frac{\lambda_i}{\mu}$ et nous vérifions qu'elle est satisfaite par la solution à posteriori. Nous remarquons aussi que la solution optimale est atteinte pour $D = D_{max}$. Dans ces conditions ci, la relation A.8 resterait toujours valable si l'on écrivait le Lagrangien de $\overline{T}$ de la façon suivante :

$$L = \overline{T} + \beta(D - D_{max}) = \frac{1}{\gamma}\sum_{i=1}^{N}(\frac{\lambda_i}{\mu Ci - \lambda_i} + t_i) + \beta(\sum_{i=1}^{M}(diCi + d_{i0}) - D_{max}) \tag{A.9}$$

Où β est le multiplicateur de Lagrange.

Les minima de L sont alors obtenus en cherchant les valeurs de C_i qui annulent les dérivées partielles de L par rapport à C_i, nous obtenons:

$$\frac{\partial L}{\partial C_i} = -\frac{1}{\gamma}\frac{\lambda_i}{(\mu C_i - \lambda_i)^2} + \beta d_i = 0 \tag{A.10}$$

d'où, l'expression optimale de C_i $(i = 1, 2, ..., M)$ est:

$$Ci = \frac{\lambda i}{\mu} + \frac{1}{\sqrt{\beta\gamma\mu}}(\frac{\lambda_i}{d_i})^{1/2} \tag{A.11}$$

Si l'on dispose de la valeur de la constante β, la solution exacte C_i serait déterminée. Pour cela, nous imposons que la contrainte de coût soit satisfaite.

$$\sum_{i=1}^{M}(d_i C_i + d_{i0}) = \sum_{i=1}^{M}(\frac{\lambda_i d_i}{\mu} + d_{i0}) + \frac{1}{\sqrt{\beta\gamma\mu}}\sum_{i=1}^{M}\sqrt{\lambda_i d_i} = D_{max} \tag{A.12}$$

d'où:

$$\frac{1}{\sqrt{\beta\gamma\mu}} = \frac{D_{max} - \sum_{i=1}^{M}(\lambda_i d_i/\mu + d_{i0})}{\sum_{i=1}^{M}\sqrt{\lambda_i d_i}} \tag{A.13}$$

en substituant A.13 dans la relation A.11, nous obtenons:

$$Ci = \frac{\lambda i}{\mu} + (\frac{Dmax - \sum_{i=1}^{M}(\lambda_i d_i/\mu + d_{i0})}{\sum_{i=1}^{M}\sqrt{\lambda_i d_i}})\sqrt{\frac{\lambda_i}{d_i}} \tag{A.14}$$

Remarquons que dans l'équation précédente, la condition de faisabilité est:

$$D_{max} - \sum_{i=1}^{M}(\lambda_i d_i/\mu + d_{i0}) \geq 0 \tag{A.15}$$

C'est à dire le budget d'investissement total D_{max} doit être supérieur ou égal au budget minimal d'allocation de capacités:

$$D = \sum_{i=1}^{M}(\lambda_i d_i/\mu + d_{i0}) \tag{A.16}$$

Donc la condition $C_i \geq \lambda_i/\mu$ est satisfaite.

Si nous exprimons $\overline{T}$ utilisant l'expression de C_i, nous obtenons:

$$\overline{T} = \frac{1}{\gamma}(\sum_{i=1}^{M}(\frac{(\sum_{i=1}^{M}\sqrt{\lambda_i d_i/\mu})^2}{D_{max} - \sum_{i=1}^{M}(\lambda_i d_i/\mu + d_{i0})} + \lambda_i t_i/\mu) \tag{A.17}$$

A.4 CA: Optimisation du coût du réseau

Donnée:

- Topologie,
- Trafic.

Variables:

- Capacités.

Objectif:

- Minimiser le coût D.

Contraintes:

- $T \leq T_{max}$.

Pour les mêmes raisons de convexité que le cas précédent, les optima sont déterminés par la méthode de Lagrange. L est défini dans ce cas par:

$$L = D + \beta(\overline{T} - T_{max}) = \sum_{i=1}^{M}(diCi + d_{i0}) + \beta(\frac{1}{\gamma}\sum(\frac{\lambda_i}{\mu C_i - \lambda_i} + t_i) - T_{max}) \tag{A.18}$$

Les minima de L sont alors obtenus en cherchant les valeurs de C_i qui annulent les dérivées partielles par rapport à C_i, nous obtenons:

$$\frac{\partial L}{\partial C_i} = d_i - \frac{\beta}{\gamma}\frac{\mu}{(\mu C_i - \lambda_i)^2} = 0 \tag{A.19}$$

La solution exacte est obtenue en imposant que la contrainte de temps soit satisfaite. D'où l'expression optimale des C_i:

$$C_i = \frac{\lambda_i}{\mu} + \frac{\sum_{i=1}^{M}\sqrt{d_i\frac{\lambda_i}{\mu}}}{\gamma T_{max} - \sum_{i=1}^{M}\frac{\lambda_i t_i}{\mu}}\sqrt{\frac{\lambda_i}{\mu d_i}} \tag{A.20}$$

et le coût minimal est donné par:

$$D = \sum_{i=1}^{M}(\frac{\lambda_i d_i}{\mu} + d_{i0}) + \frac{\left(\sum_{i=1}^{M}\sqrt{di\frac{\lambda i}{\mu}}\right)^2}{\gamma T_{max} - \sum_{i=1}^{M}\frac{\lambda_i t_i}{\mu}} \tag{A.21}$$

A.5 FA: Espace des solutions

Données:

- Matrice de trafic externe;
- Emplacements géographiques de nœuds.

Variables:

- flux $\{f\}$.

Objectif :

- minimiser $T = \frac{1}{\gamma}\sum_{i=1}^{M}\frac{f_i}{C_i - f_i}$

avec $f = (f_1, f_2, ..., f_M)$

Contraintes

1. f satisfait la matrice de trafic Γ
2. $f_i \leq C_i$

Le problème revient à déterminer l'ensemble F de solutions faisables. Cet ensemble peut être vu comme l'intersection de deux ensembles F_1et F_2 où

$$F_1 = \{f/f\ satisfait\ la\ contrainte\ 1\}$$

$$F_2 = \{f/f\ satisfait\ la\ contrainte\ 2\}$$

Nous pouvons démontrer que l'ensemble F des solutions réalisable est un ensemble convexe. Cet espace n'est autre que l'intersection des deux ensembles F_1 et F_2.

Dans ce qui suit, nous essayons de déterminer les deux ensembles F_1 et F_2 et par la suite l'ensemble F.

A.5.1 Contrainte de flux

Soit $f_{ij}^{(m,n)}$ le flux de l'arc (m,n) dû à la communication des nœuds i et j. Comme nous avons vu dans le deuxième chapitre, la condition de conservation de flux à chaque nœud se traduit par:

$$\sum_{j=1}^{N} f_{ji}^{(m,n)} - \sum_{j=1}^{N} f_{ij}^{(m,n)} = \begin{cases} -r_{mn} & si\ i = m \\ +r_{mn} & si\ i = n \\ 0 & sinon \end{cases} \tag{A.22}$$

avec $f_{ij}^{(m,n)} \geq 0$

Le flux total f est donné par

$$f = \sum_{m=1}^{N}\sum_{n=1}^{N} f^{(m,n)} \tag{A.23}$$

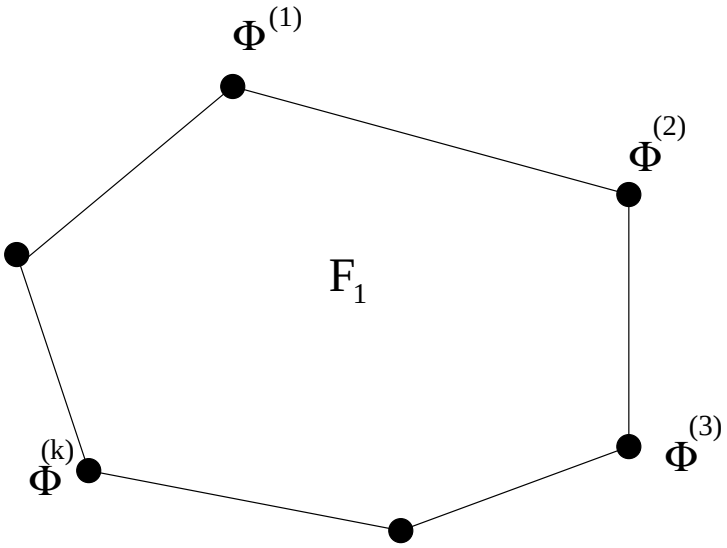


FIG. A.2 – L'ensemble F1 et les flots extrêmes

Où $f^{(m,n)} = (f_1^{(m,n)}, f_2^{(m,n)}, \ldots, f_M^{(m,n)})$ est le vecteur de flux du à la communication de nœuds m est n.

Il a été démontré [HU 68] que l'ensemble F_1 de solutions qui satisfait la contrainte (A.22) est un polygone convexe (cf. figure A.2)

Les points extrêmes du polygone sont les flots obtenus par les chemins les plus courts [FOR 92]. On peut aussi démontrer que tous les flots de chemins les plus courts correspondent à ces points.

A.5.2 Contrainte de capacité

L'ensemble F_2 est donné par :
$F_2 = \{f/f < C\}$ qui est un ensemble convexe.

En conclusion, $F = F_1 \cap F_2$ est aussi convexe (cf. figure A.3). Une observation importante liée à la fonction objective à minimiser va restreindre l'ensemble de solutions. En effet, nous remarquons que:

$$\lim_{f_i \mapsto C_i} T(f) = +\infty \tag{A.24}$$

D'où, durant l'application de la technique d'optimisation (minimiser le temps de réponse moyen T), on est sûr que le flux f va être limité par les bords de l'ensemble F_2 et par la suite, on garant la condition de faisabilité.

Donc, étant donné une solution initiale faisable f_0, on peut se débarrasser de la contrainte de capacité et le problème d'allocation de flot devient un problème sans contraintes.

A.6 Conception de topologie optimale multipoint

Dans beaucoup de cas, le trafic est relativement faible, et il est alors avantageux de relier ces nœuds au nœud central par l'intermédiaire des lignes multipoint. Le problème de

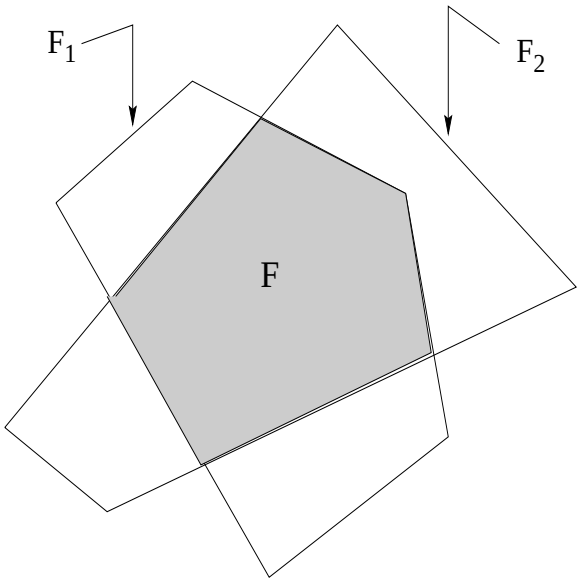


FIG. A.3 – L'ensemble $F = F_1 \cap F_2$

l'optimisation globale d'un réseau WAN comportant des lignes multipoint n'est pas encore résolu de façon satisfaisante à ce jour. Nous nous intéressons ici au problème de connexion optimum d'un certain nombre de concentrateurs au nœud d'entrée au réseau (cf. figure A.4).

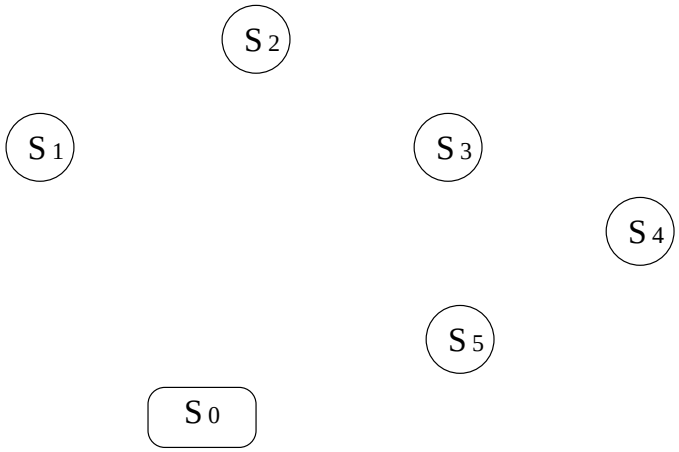


FIG. A.4 – Carte des stations à relier au ncentral par des lignes multipoint

Pour simplifier nous supposons que toutes les lignes du réseau ont même capacité C, que nous supposons égal à 2400 bits/s. Le trafic produit par les sites Si à destination du nœud central S0 est lamdai/i bits/sec. (cf. tableau A.1) et les coûts de connexion sont définis par une matrice de coûts f_{ij}donnée dans ce cas par le tableau A.2.

Il s'agit maintenant de déterminer l'arbre de coût minimum qui permet de relier les nœuds au nœud central. S'il n'y a pas de contrainte de débits maximum, l'arbre de coût minimum peut être déterminée facilement par l'algorithme de **Kruskal**:

Algorithme de Kruskal

---

0 8 1 0 0 0 0Station Si	S_0	S_1	S_2	S_3	S_4	S_5
Trafic produit par les stations (bits /s)	/	600	900	1000	200	1600

TAB. A.1 – Trafic produit par les stations S_i

0 8 1 0 0 0 0Station	S_0	S_1	S_2	S_3	S_4	S_5
S_0	/	5	7	8	2	3
S_1	5	/	5	3	6	7
S_2	7	5	/	5	16	8
S_3	8	3	5	/	5	3
S_4	2	6	10	5	/	3
S_5	3	7	8	3	3	/

TAB. A.2 – Matrice des coûts des liaisons entre stations

– `Tant qu'il y a des nnon interconnectés Faire`

1. `Choisir un ndont le coût de connexion au ncentral est minimum et vérifier qu'il ne forme pas un cycle avec le réseau déjà construit;`
2. `Vérifier que la contrainte de capacité est vérifiée;`
3. `aller à 1.`

– `Fin Tant que`

Le résultat de l'arbre de coût minimum sans contraintes est représenté par la figure A.5

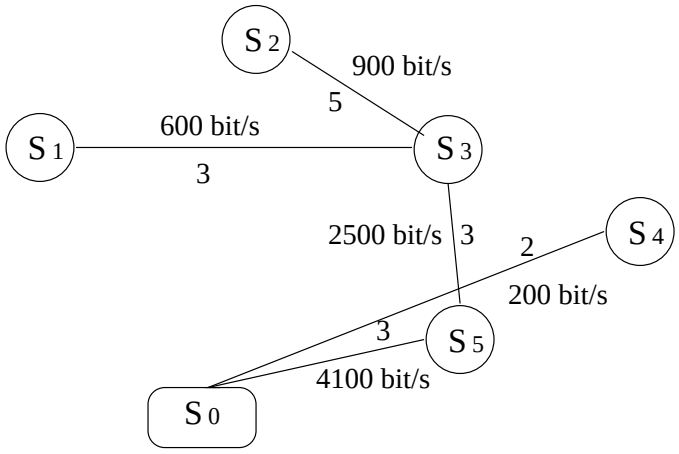


FIG. A.5 – Réseau multipoint optimum sans contraintes de débit

Complexité de l'algorithme de Kruskal

D'après la définition d'un arbre, l'ajout d'un arc créerait alors un cycle et un seul. Tester est alors en O(log N) et pour le choix d'arc est de O(M). D'ou la complexité est de O (M logN).

Annexe B

L'API JDBC de JAVA

L'informatique a toujours été un fantastique outils de manipulation de données. Lorsque ces dernières deviennent nombreuses et qu'elles ont une structure bien particulière, on les stocke dans de bases de données. L'accès aux informations contenues dans une base de données s'effectue au travers d'un logiciel prévu à cet effet..

Aujourd'hui, la plupart des outils de développement offrent des facilités permettant un accès aux bases de données. Cette évolution atteint même des outils de développement grand public comme Delphi ou Visual Basic. Qu'en est-il pour Java?

B.1 La philosophie de JDBC

Les JDBC ou Java DataBase Connectivity, désignent un ensemble de classes qui permet à un programme Java d'avoir accès à une base de données relationnelle.

Java permet de réaliser des programmes dont le code source ne tient compte ni du système d'exploitation, ni du matériel sur lesquels ces programmes seront exécutés.

Cet esprit d'indépendance caractéristique à Java se devait d'être perpétué dans les JDBC. Ainsi, tout programme effectuant ses accès à des bases de données au travers des JDBC pourra être appliqué à n'importe quelle base de données à la condition qu'il existe un driver JDBC pour la base de donnés concernée.

Comment cela est-il possible? La situation est un peu similaire à celle de l'AWT (Abstract Window Toolkit).

Le programmeur n'effectue dans son code que des appels à des méthodes ou des classes appartenant aux JDBC. La plupart des méthodes offertes par Sun dans les JDBC sont abstraits. D'où, elles dépendent de la base de données utilisée. La portion de code se chargeant d'implémenter ces méthodes abstraites est nommée driver JDBC. On aura besoin d'un driver spécifique pour chaque SGBD auquel on souhaite accéder.

Ce driver est en général fourni par l'entreprise commercialisant le SGBD. Les entreprises distribuent généralement leur SGBD avec une bibliothèque offrant une API permettant d'accéder à leur SGBD. Lorsqu'elle est disponible pour le langage Java, le driver SGBD repose en général sur la bibliothèque.

Ainsi la bibliothèque Java permettant d'accéder à une base de données mSQL se nomme MsqlJava (cf. figure B.1). Le driver JDBC pour le SGBD mSQL se nomme mSQL-JDBC et utilise la bibliothèque MsqlJava. Le programme utilisateur écrit en Java ne verra que les

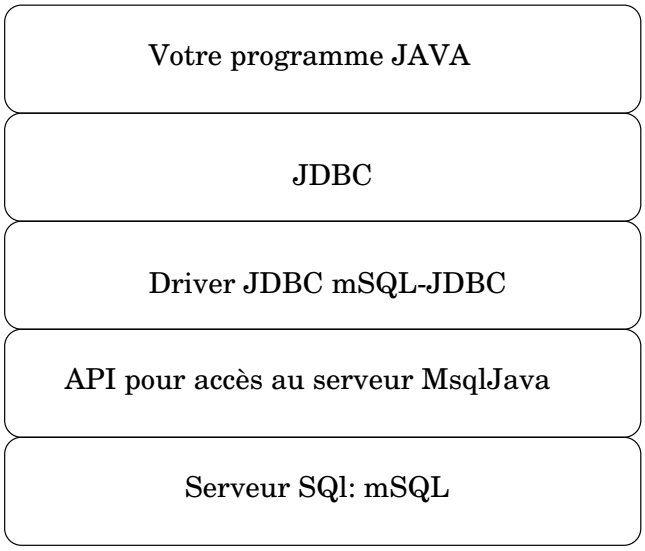


FIG. B.1 – Couches intervenant dans l'accès à un SGBD

JDBC et sera ainsi totalement indépendant des couches se trouvant en dessous des JDBC, à savoir du driver, de la bibliothèque d'API pour chaque SGBD.

B.1.1 Avantages de l'API JDBC

Cette API offre de nombreux avantages:

- Rationalisation du temps d'apprentissage: les programmeurs se trouvant toujours devant le même ensemble de classes (JDBC), ils n'auront pas à apprendre à utiliser une bibliothèque d'API pour chaque SGBD auquel ils souhaiteraient accéder.
- Réutilisation de code: toute portion de code pourra être réutilisée quel que soit le SGBD pour lequel il avait originellement été écrit..
- Evolution des besoins: comme dans nombreux autres domaines (systèmes d'exploitation, plate-forme, etc...), il n'y a pas de SGBD idéal. Tout dépend de l'utilisation souhaitée. Il arrive qu'avec l'évolution de l'entreprise, la qualité de données à manipuler prenne des proportions dépassant les capacités du SGBD installé. Un changement de SGBD peut alors s'imposer. Mais qu'en est-il des applications écrites pour l'ancien SGBD? Si ces dernières ont été écrites en Java en utilisant les JDBC, elles pourront exécuter avec le nouveau SGBD.
- Indépendance vis-à-vis d'un vendeur: comme nous venons le voir, la réalisation d'applications en Java utilisant les JDBC rend le changement du SGBD bien plus aisé. Ce qui limite la dépendance de l'utilisateur vis-à-vis de la société produisant le SGBD qu'il utilise. Si le client n'est plus satisfait des services offerts par l'entreprise commercialisant son SGBD, il pourra plus facilement en changer. Cette concurrence renforcée diminue l'éventualité d'un quasi-monopole. Ce qui est toujours un bénéfice de l'utilisateur.

B.1.2 Bases de données orientées objet

L'utilisation de bases de données orientées objet offre des perspectives particulièrement intéressantes puisqu'il serait possible d'avoir une correspondance exacte entre les objets manipulés par un programme en Java et ceux stockés dans la base de données. Ainsi, contrairement aux SGBD relationnels, l'accès aux SGBD orientés objet depuis Java ne nécessiterait pas la maîtrise d'une API, comme celle offerte par les classes JDBC, et d'un langage de manipulation relationnel tel que SQL.

A l'heure actuelle, un standard pour l'accès aux bases de données orientées objet depuis Java est en cours d'élaboration par le ODMG (Object Database Management Group) [SUN 96].

B.2 Description de l'API de JDBC

L'API de JDBC est en fait une série d'interfaces abstraites Java qui permettent aux applications Java d'ouvrir des connexions vers des bases de données particulières, d'exécuter des requêtes SQL et de traiter les résultats. Cette API offre différentes interfaces dont les plus importantes sont (cf. figure B.2):

- java.sql.DriverManager: cette interface exécute le chargement de driver et donne un support pour la création des nouvelles connexions à des bases de données,
- java.sql.Connexion: cette interface permet la connexion à une base de données particulière.
- java.sql.Statement: c'est la requête à exécuter,
- java.sql.ResultSet: cette interface contrôle l'accès aux résultats d'une requête (Statement).

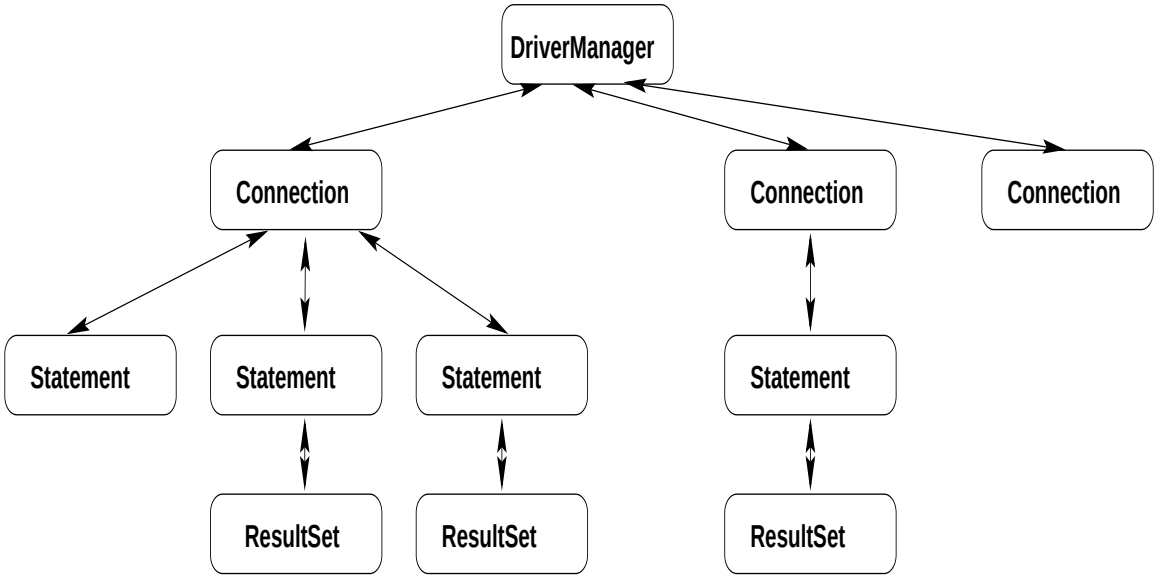


FIG. B.2 – Les principales interfaces de l'API JDBC de Java

B.3 Les scénarios d'utilisation de JDBC

B.3.1 Les Applets

La grande utilisation actuelle de Java est dans l'implémentation des applets qui peuvent être téléchargés à travers un navigateur comme une partie des documents Web. Ces applets peuvent entre autres ouvrir des connexions permanentes avec des SGBD à travers une API JDBC (cf. figure B.3).

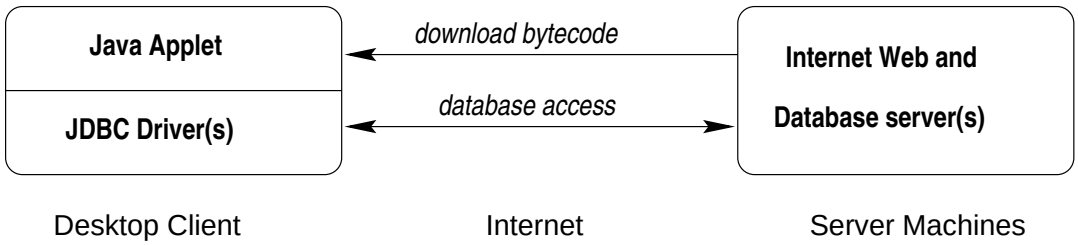


FIG. B.3 – L'accès à une base de données via une Applet Java

Les applets diffèrent des applications Java par plusieurs points:

- les applets "*Untrustred*" sont souvent limitées par les opérations qui peuvent les exécuter. En particulier, elles ne sont pas autorisées d'accéder aux fichiers locaux et l'ouverture des connexions avec des hôtes arbitraires,
- les applets "Internet" présentent de nouveaux problèmes,
- des considérations de performances pour l'implémentation des connexions aux bases de données si la base de données est locale ou elle est de l'autre partie du monde.

B.3.2 Les Applications

Java est utilisé aussi dans le développement des applications traditionnelles (cf. figure B.4). Dans ce cas l'application est confiée et elle est autorisée de lire et d'écrire dans des fichiers après l'ouverture de connexions réseaux...

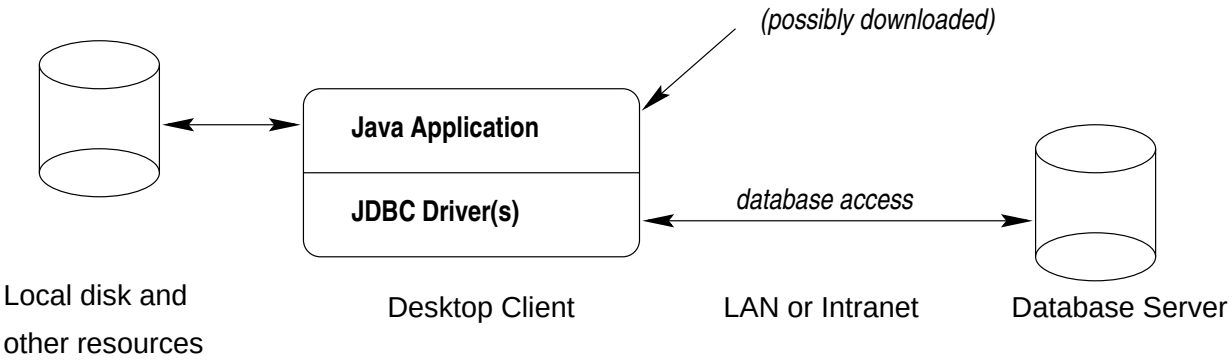


FIG. B.4 – L'accès à une base de données via une application Java

B.3.3 L'accès par un intermédiaire (Three-Tier)

Ce scénario est utilisé par les applications Client/Serveur pour accéder aux bases de données a travers des GUIs (Graphical User Interfaces). Les applications voulant accéder à une base de données font un appel à un *'Middle tier'* qui à son tour effectue les opérations demandées avec les bases de données (cf. figure B.5). Ces appels peuvent être fait soit à travers RPC (Remote Procedure Call) ou par ORD (Object Request Broker).

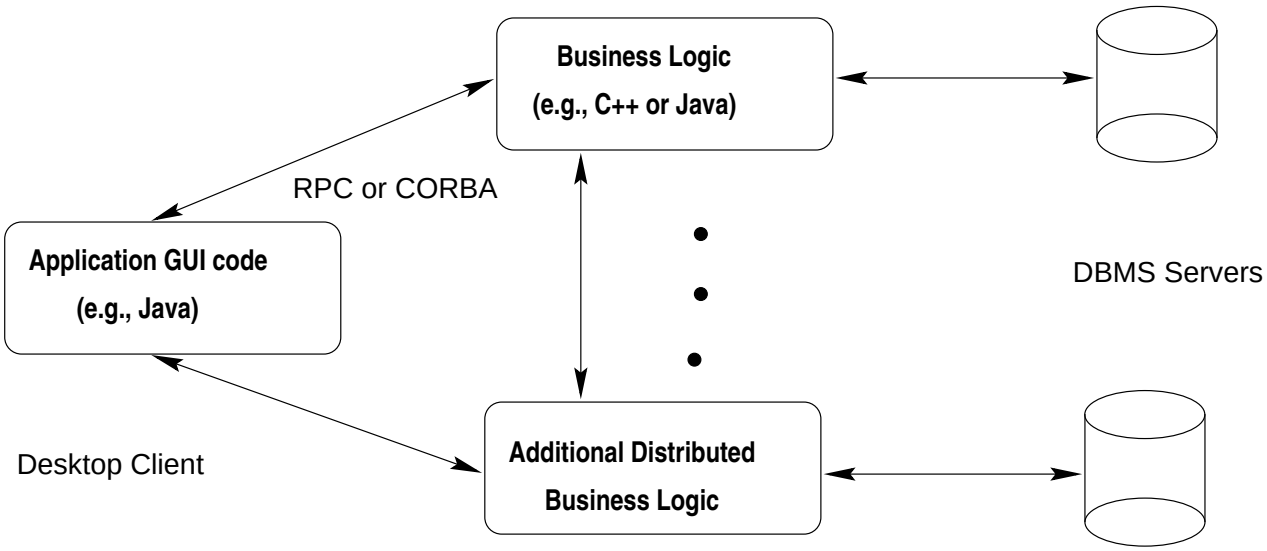


FIG. B.5 – L'accès à une base de données via un Middle tier [SUN 96]

Annexe C

Manuel d'utilisation

C.1 Lancer WanDesigner

Cet outil peut être lancé sous deux modes: mode **application** et mode **applet**. En mode application, il est nécessaire d'avoir l'interpréteur Java correspondant à votre machine.

Vous pouvez récupérer WanDesigner soit par téléchargement à partir du serveur Web, soit à partir de la disquette fournie avec ce rapport. Dans les deux cas, vous trouverez l'outil sous un format compressé. Pour le décompresser, il suffit d'exécuter les deux commandes suivantes:

```
gzip -d WanDesigner.tar.gz

tar xvf WanDesigner.tar
```

Quatre paquetages seront générés: *gui, network, databases* et *general*.

Ainsi, pour exécuter **WanDesigner** il suffit de taper la commande:

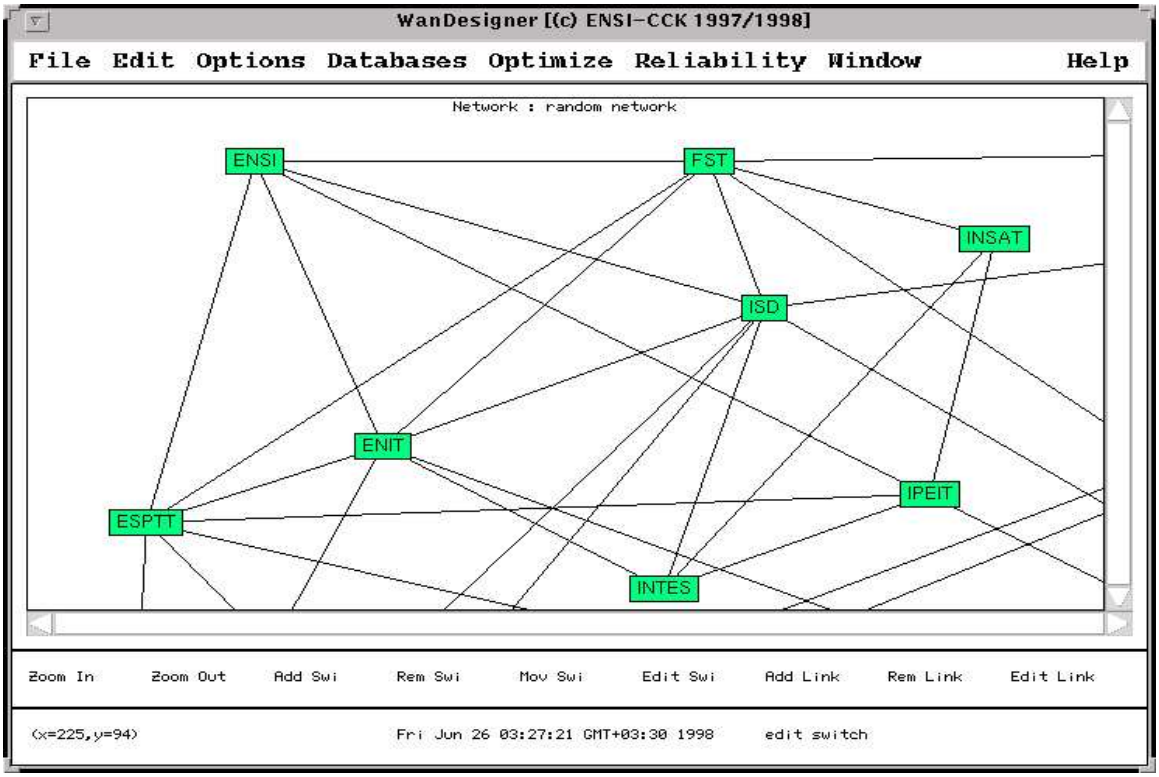
```
java WanDesigner
```

WanDesigner est, en fait, le nom de la classe principale (se trouvant dans le paquetage *gui*) qui vérifie la validité du mot de passe et réagit en conséquence.

En mode applet, vous pouvez lancer WanDesigner à partir du serveur Web de WanDesigner. Le browser doit supporter l'exécution des applets Java ce qui est le cas pour la plupart des navigateurs actuels.

Pour pouvoir accéder aux bases de données de réseaux, tarifaires et géographiques, il est nécessaire de connaître leurs localisations et les protocoles de connexion utilisés.

C.2 Les menus de WanDesigner

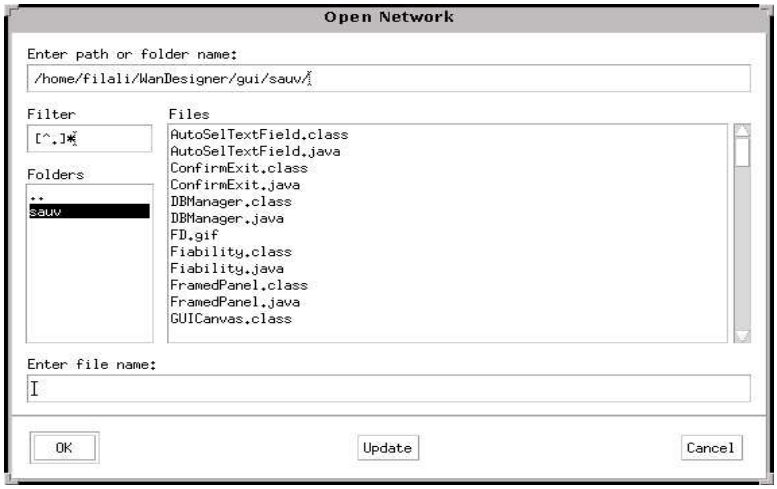


C.2.1 File

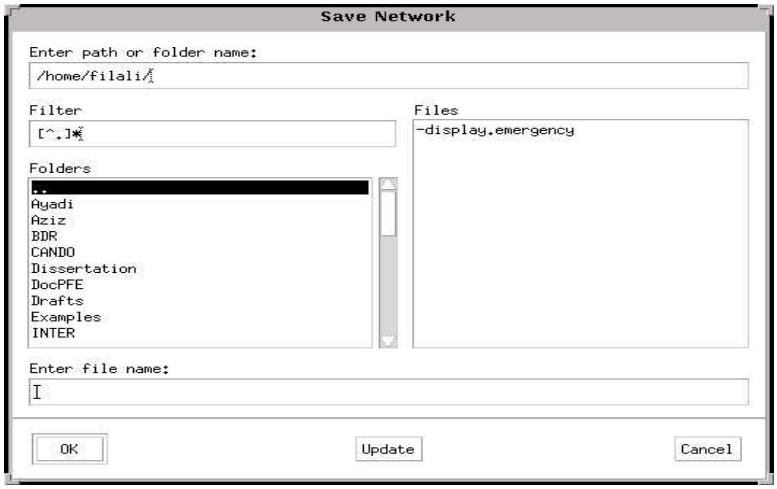
Ce menu regroupe les commandes de manipulation de fichiers et de sécurité de **WanDesigner**. A cause des restrictions de sécurité dans Java, certaines commandes sont non autorisées dans le mode Applet.

New: créer un réseau ayant N nœuds. Les nœuds sont aléatoirement places dans un rectangle de taille (0,0,400,400) pour le mode Applet et (0,0,800,400) dans le mode application. Les éléments de la matrice de trafic sont distribués uniformément entre [0,1kbits/sec]. Il est à noter que cette commande efface le réseau courant.

Open: ouvrir un réseau enregistré dans un fichier (ou base de données). Une boite de dialogue est affichée à l'utilisateur pour sélectionner le fichier à ouvrir.



Save: sauvegarder le réseau courant dans un fichier. Cette commande n'est pas permise dans le mode Applet pour des raisons de sécurité. Le format du fichier est la suivante: *//description //switches //links //constraints*



Print: imprimer le réseau courant. L'utilisateur a le choix de l'imprimante, de nombre de copies...

Passwd: changer le mot de passe de l'administrateur du logiciel en s'authentifiant par le mot de passe courant. Cette commande n'est pas permise dans le mode Applet.

Exit: quitter **WanDesigner**.

C.2.2 Edit

Ce menu offre les commandes utiles pour la modification du réseau courant. La majorité de ces commandes sont exécutées par de simples cliques avec la souris.

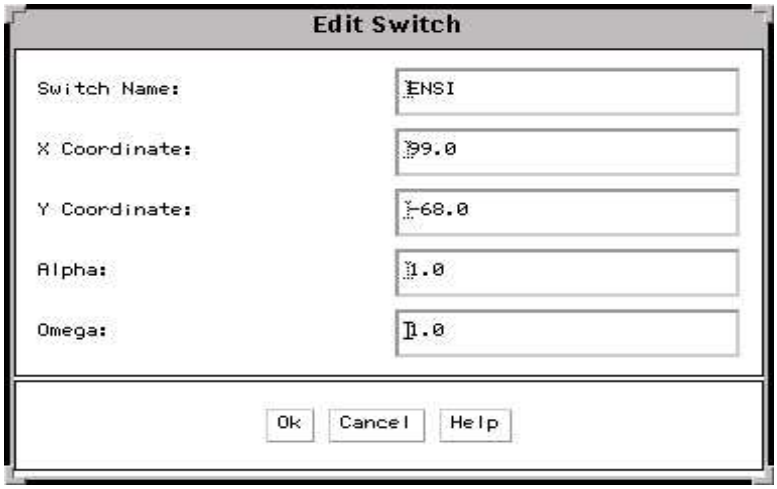
Switch: ajouter, supprimer, déplacer, éditer et effacer les nœuds.

Add Switch: ajouter un nœud en cliquant sur la position d'ajout. Le trafic vers tous les autres nœuds sera génère aléatoirement et vice versa. L'utilisateur peut changer ces trafics en ajoutant une contrainte de trafic.

Remove Switch: supprimer un nœud du réseau courant en cliquant sur le nœud a supprimer. Tous les liens vers les autres nœuds seront automatiquement éliminés.

Move Switch: déplacer un nœud en le retirant par la pointe de la souris vers la nouvelle position. Tous les liens auquel ils sont lies seront automatiquement mis a jour.

Edit Switch: éditer les caractéristiques d'un nœud en cliquant dessus. L'utilisateur peut modifier certains paramètres tels que: le nom, la position (abscisse et ordonnée).



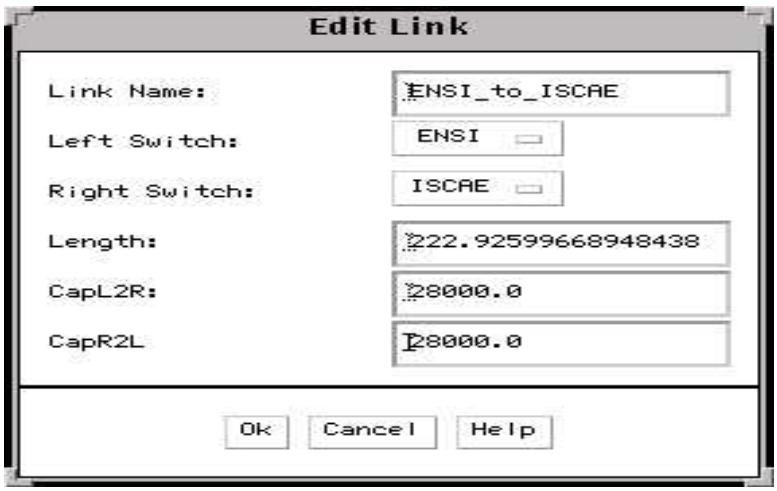
Clear All: effacer tous les nœuds, ceci provoque aussi la suppression de tous les liens. On aura ainsi un nouveau réseau.

Link: ajouter, supprimer, déplacer, éditer et effacer les liens.

Add Link: ajouter un lien en cliquant sur les deux nœuds a interconnecter. Le nouveau lien sera automatiquement crée.

Remove Link: supprimer un lien du réseau courant en cliquant sur le lien a supprimer.

Edit Link: éditer les caractéristiques d'un nœud en cliquant dessus. L'utilisateur peut modifier certains paramètres tels que: le nom, le nœud de gauche, le nœud de droite, la longueur (non géographique).

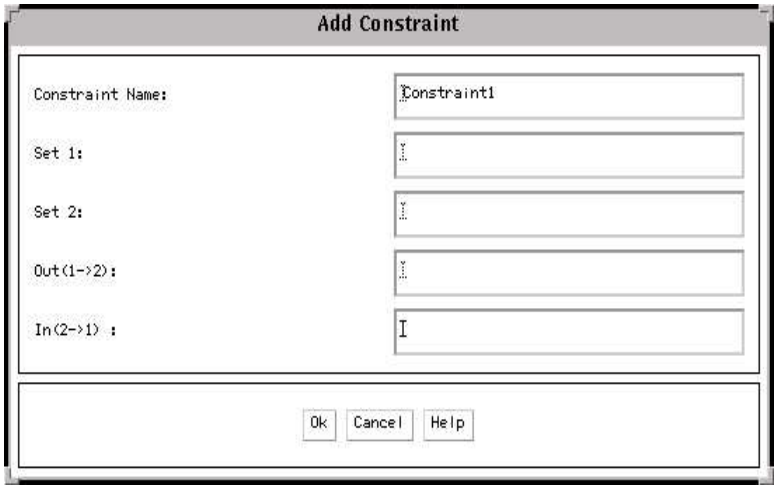


Complete Graph: créer un graphe complet, on aura n * (n -1) / 2 liens dans le réseau courant où n est le nombre de nœuds.

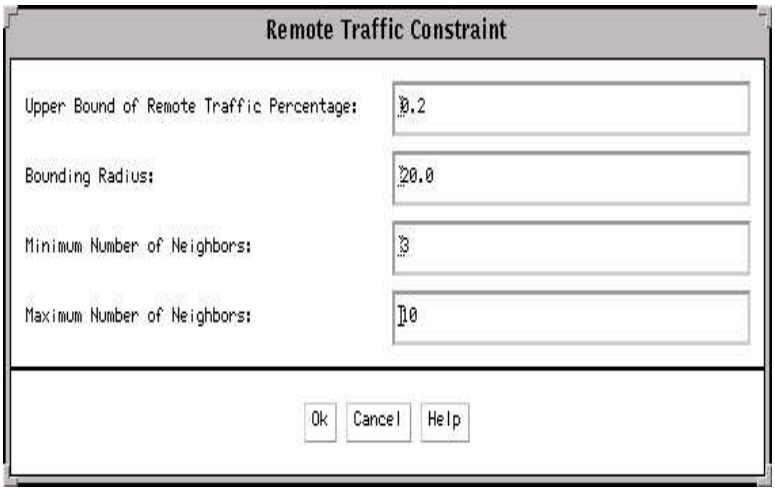
Clear All: effacer tous les liens du réseau courant.

Constraint: ajouter, supprimer, éditer une contrainte de trafic. Cette contrainte spécifié le trafic maximal entre deux nœuds du réseau.

Add Constaint: ajouter une contrainte de trafic



Remove Constraints: supprimer une contrainte de trafic.



C.2.3 Options

Ce menu regroupe les fonctionnalités permettant la gestion de la taille de la zone d'affichage du réseau courant.

Scale: changer l'échelle de fenêtre. Une valeur large d'échelle représente une magnifique vue.

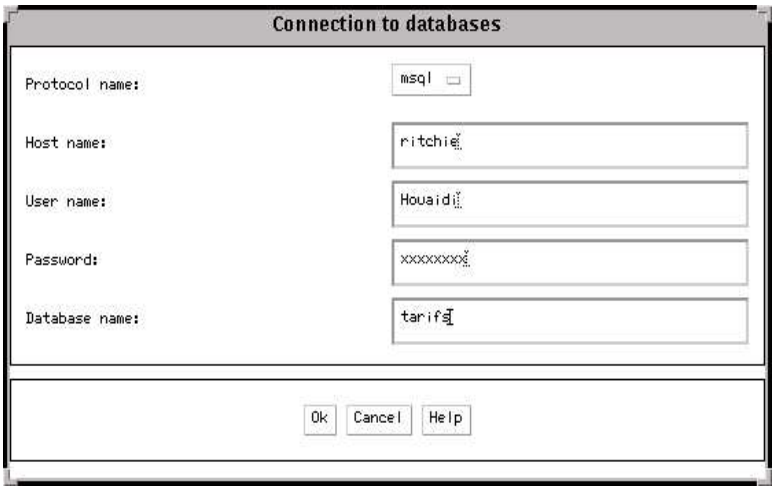
Zoom Out: agrandir la fenêtre d'apparition du réseau. En cliquant avec la souris sur une position de la fenêtre, celle ci s'agrandira en taille d'un facteur paramétrable par l'utilisateur. Si le réseau sort de la fenêtre principale, deux barres de défilement seront affichées (l'une horizontale et l'autre verticale).

Zoom In: diminuer la fenêtre d'apparition du réseau. En cliquant avec la souris sur une position de la fenêtre, celle ci se diminuera en taille d'un facteur paramétrable par l'utilisateur. Les barres de défilement seront illuminées automatiquement dès que le réseau soit en entier dans la fenêtre principale.

C.2.4 Databases

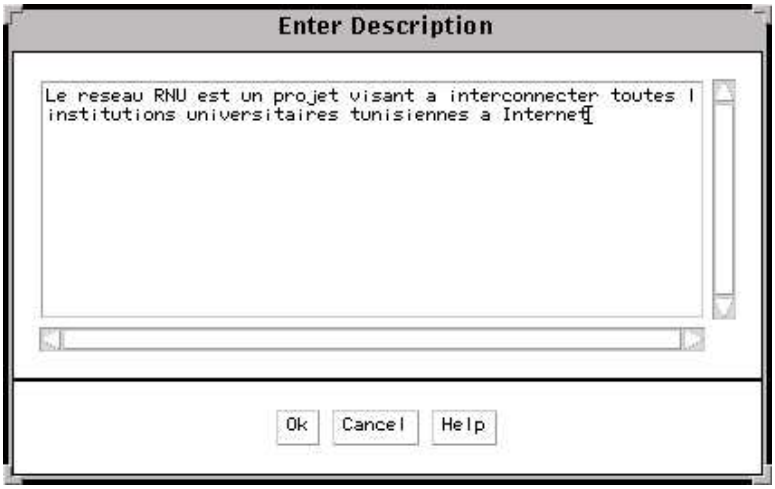
A l'aide de ce menu l'utilisateur peut se communiquer avec un SGBD pour effectuer des manipulations sur les bases de données de l'application.

Connexion: se connecter à un serveur de base de données. Le concepteur doit spécifier le protocole d'accès au SGBD (msql, odbc...), le port de connexion utilisé (souvent c'est 1114), le nom du hôte (ou son adresse IP), le nom et mot de passe d'utilisateur, les noms des bases de données (tarifaires, de réseau et géographique).



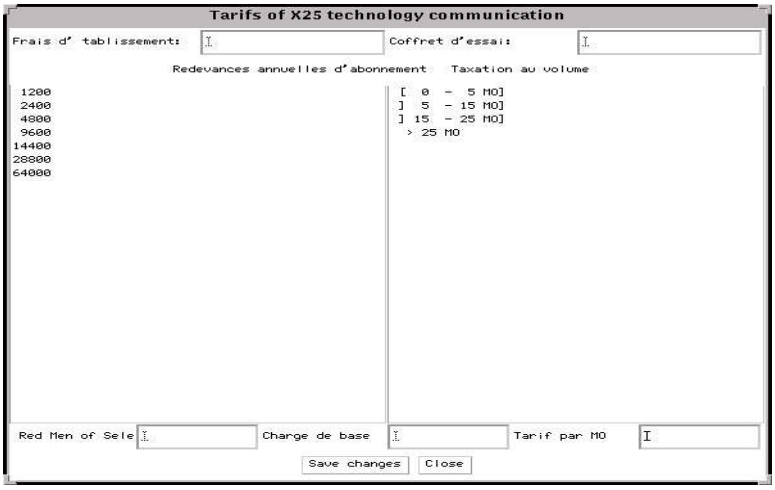
Network:

Description: entrer la description du réseau. Cette description sera sauvegardée dans l'entête du fichier.



Tarifs:

Tarifs X25: mettre à jour la base de données de tarifs X25.



Tarifs LS: mettre à jour la base de données de tarifs LS.

Tarifs RTC: mettre à jour la base de données de tarifs RTC.

Tarifs RNIS: mettre à jour la base de données de tarifs RNIS.

Tarifs ATM: mettre à jour la base de données de tarifs ATM.

Tarifs FR: mettre à jour la base de données de tarifs FR.

Geogarphic: mettre à jour la base de données géographique. Cette base de données doit se retrouver en fait sur un système de gestion de base de données géographiques (SGBDG) tel que ArcView. Dans sa version actuelle, ce menu ne permet que la mise à jour des distances entre les nœuds du réseau. Elles seront extraites de la carte géographique si on dispose d'un SGBDG.

C.2.5 Optimize

Capacity Assignment: allouer les capacités optimales du réseau courant. Après l'exécution de l'allocation, les épaisseurs des lignes seront mises à jour, elles seront proportionnelles aux capacités optimales trouvées.

Flow Assignment: allouer les flots optima. Cette allocation permet la résolution du problème de routage. L'utilisateur est invité à choisir entre un routage fixe alterné ou non.

Capacity and Flow Assignment: allouer les capacités et flots optima.

Topology Assignment: chercher une topologie optimale, l'utilisateur choisit la nature de la topologie cherchée:

Maillée: créer une topologie maillée "optimale". L'exécution de cette fonctionnalité demande de temps d'exécution énorme surtout si la taille du réseau est importante. Donc, l'utilisateur doit patienter particulièrement s'il dispose d'une machine pas assez puissante.

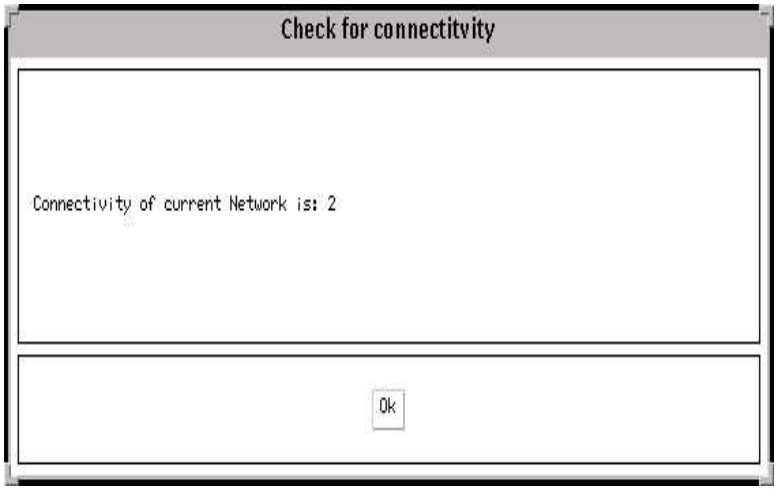
Best Star: créer le meilleur réseau en étoile. Ce réseau est la meilleure configuration étoilée qui minimise le coût de lignes.

Minimum Spanning Tree: créer un arbre de recouvrement minimal.

C.2.6 Reliability

Ce menu regroupe des fonctionnalités permettant l'étude de la fiabilité du réseau afin de prévoir toute déconnexion possible.

Check Connectivity: déterminer la connectivité du réseau courant.

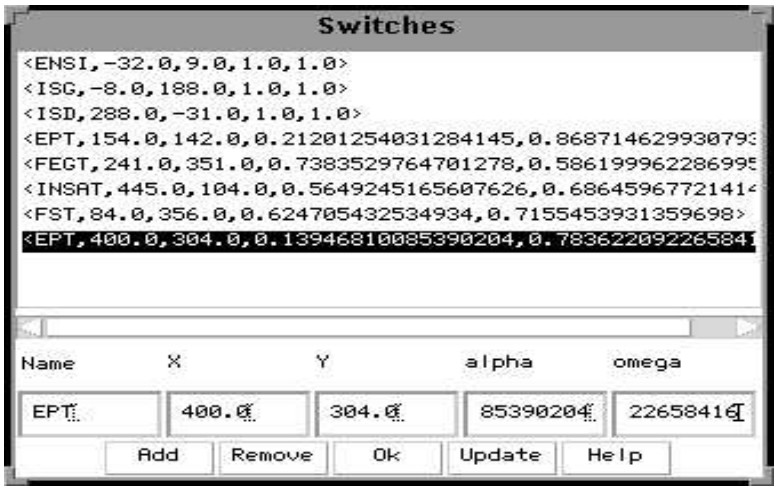


Simulate: simuler la probabilité de déconnexion du réseau en se basant sur des simulations. En résultat, ce menu fournit une courbe graphique donnant la variation de la probabilité de déconnexion du réseau en fonction de la probabilité de la panne d'une ligne. Aussi, il donne la fraction des nœuds qui seront isolés du reste du réseau.

C.2.7 Window

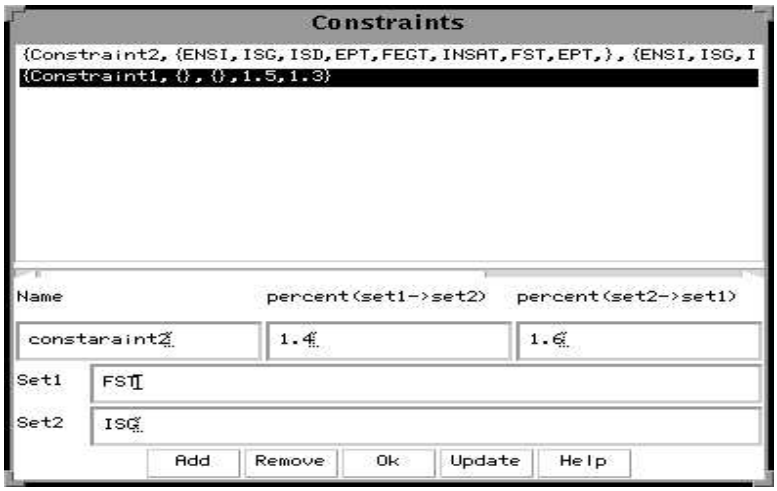
regroupe une vue globale sur le réseau courant.

Switches: afficher une boîte de dialogue de nœuds. À partir de ce menu vous pouvez exécuter tous les commandes du menu **Switch**.



Links: afficher une boîte de dialogue. Chaque élément de la liste est un lien du réseau. Son format est [*Non de lien, le nœud droit, le nœud gauche, la longueur, la capacité, le flot*]. À partir de ce menu vous pouvez exécuter tous les commandes possibles dans le menu **Link**.

Constraints: afficher une boîte de dialogue de contraintes. Toutes les contraintes seront affichées dans la partie supérieure de la boîte. À partir de ce menu vous pouvez exécuter tous les commandes du menu **Constraint**.



C.2.8 Help

Des informations concernant **WanDesigner**.

About: afficher des informations à propos de cadre de réalisation de cet atelier et de groupe de développement.

Using WanDesigner: afficher l'aide d'utilisation de l'outil. Cet aide est disponible à l'adresse: *http://www.ritchie.ensi.rnrt.tn./~WanDesigner/help/help.html*

Algorithms Documentation: donner des informations concernant les algorithmes développés dans cet atelier. Ces algorithmes sont disponibles à l'adresse: *http://www.ritchie.ensi.rnrt.tn/~WanDesigner/help/algorithms.html*