Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Contribution a la conception de systèmes d'informatique distribuée

Persoons, Eliane; Hemmerling, Marc

Award date:
1982

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 23. Jun. 2020

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

RUE GRANDGAGNAGE, 21, B - 5000 NAMUR (BELGIUM)

FMB 16/1982/5 I

FACULTES
UNIVERSITAIRES
N.-D. DE LA PAIX
NAMUR

Bibliothèque
FMB
16/1982/5 I

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX - NAMUR (BELGIQUE),

INSTITUT D'INFORMATIQUE

CONTRIBUTION A LA CONCEPTION DE

SYSTEMES D'INFORMATIQUE DISTRIBUEE

Promoteur : Ph. Van Bastelaer.

Eliane PERSOONS.
Marc HEMMERLING.

Mémoire présenté en vue
de l'obtention du grade de

LICENCIE ET MAITRE EN INFORMATIQUE

ANNEE ACADEMIQUE 1981 - 1982.

Nous tenons tout d'abord à remercier Monsieur Philippe Van Bastelaer qui a accepté de diriger ce mémoire et qui nous a permis, par ses conseils et critiques, de mener ce travail à son terme.

Nous exprimons également toute notre gratitude à Mademoiselle Claudie Chappuis et à Monsieur Pierre Masson qui nous ont très bien accueillis à la compagnie CII-Honeywell Bull à Paris et dont les critiques constructives nous ont permis d'aboutir à ce mémoire. Nous remercions aussi tout particulièrement Monsieur Michel Colas sans l'intervention duquel ce travail n'aurait pu se réaliser.

Enfin, notre plus vive reconnaissance va à tous les membres de l'Institut d'Informatique, à tous les employés de CII-Honeywell Bull (Paris) et CII-Honeywell Bull (Bruxelles) qui, d'une manière ou d'une autre, nous ont aidés au cours de cette année.

Ce volume constitue la première partie du travail réalisé dans le cadre du mémoire.

La deuxième partie, comprenant la base théorique et le dossier de programmation de l'outil automatisé et qui constitue notre apport personnel, reste la propriété de l'entreprise qui nous a accueillis.

TABLE DES MATIERES

PAGE

INTRODUCTION 1

PARTIE 1 : LES SYSTEMES D'INFORMATIQUE DISTRIBUEE 4

INTRODUCTION 5

CHAPITRE 1 : EVOLUTION VERS LES SYSTEMES D'INFORMATIQUE DISTRIBUEE 6

1.1. Introduction 6

1.2. Evolution 6

1.2.1. Etape 1 : connexion directe des terminaux à l'ordinateur central 6

1.2.2. Etape 2 : apparition des concentrateurs 8

1.2.3. Etape 3 : apparition des frontaux 9

1.2.4. Etape 4 : réseaux généraux d'ordinateurs 11

CHAPITRE 2 : ASPECTS ECONOMIQUES ET ORGANISATIONNELS DES SYSTEMES D'INFORMATIQUE DISTRIBUEE 13

2.1. Introduction 13

2.1. Centralisation et décentralisation 13

2.3. Motivations à la décentralisation 14

CHAPITRE 3 : EXEMPLES DE SYSTEMES D'INFORMATIQUE DISTRIBUEE 16

3.1. Introduction 16

3.2. Système d'enregistrement de commandes 16

3.3. Système de réservation de places d'avion 18

3.4. Système bancaire 19

3.5. Sytsèmes bancaires évolués 20

CHAPITRE 4 : CLASSIFICATION ET CARACTERISTIQUES FONDAMENTALES DES SYSTEMES D'INFORMATIQUE DISTRIBUEE 23

4.1. Introduction 23

4.2. Classifications des sustèmes d'informatique distribuée ... 24

4.2.1. Classification par type 24

PAGE

4.2.2. Classification par nature intrinsèque 25

4.2.2.1. le degré de couplage entre deux composants 25
4.2.2.2. la structure d'interconnexion 26

1. structures à interconnexions directes 26
2. interconnexions indirectes entre les composants 27

4.2.2.3. interdépendance des composants 28
4.2.2.4. synchronisation entre composants 28

4.3. Caractéristiques des systèmes d'informatique distribuée .. 28

CHAPITRE 5 : STRUCTURES FONDAMENTALES DES SYSTEMES D'INFORMATIQUE DISTRIBUEE 29

5.1. Introduction 29

5.2. Structure de traitement distribué 30

5.2.1. Traitement distribué horizontalement 30
5.2.2. Traitement distribué hiérarchiquement 31
5.2.3. Traitement distribué de manière hybride 32

5.3. Structure de stockage distribué 33

5.3.1. Bases de données partitionnées 33

5.3.1.1. bases de données partitionnées géographiquement 34
5.3.1.2. bases de données partitionnées hiérarchiquement 34
5.3.1.3. bases de données partitionnées selon un critère de sécurité et de confidentialité 35

5.3.2. Bases de données répétées 35

5.3.2.1. bases de données répétées hiérarchiquement 36
5.3.2.2. bases de données répétées horizontalement 37

5.3.3. Combinaison des bases de données répétées et partitionnées 37

5.4. Structure de transmission distribuée 38

5.4.1. Réseaux hiérarchiques 38
5.4.2. Réseaux maillés 39

CHAPITRE 6 : AVANTAGES ET DESAVANTAGES DES SYSTEMES DISTRIBUES 42

6.1. Introduction 42

6.2. Avantages 42

6.2.1. Meilleures performances et temps de réponse par rapport au coût 42
6.2.2. Meilleure disponibilité ou survie du système 43
6.2.3. Coûts de communication décroissants 43
6.2.4. Flexibilité et adaptabilité 43

PAGE

6.3. Désavantages 44

6.3.1. Complexité structurale du système 44
6.3.2. Problèmes techniques 44
6.3.3. Multiplication 44

PARTIE 2 : SYSTEMES D'INFORMATIQUE DISTRIBUEE ET ARCHITECTURES DE RESEAUX 45

INTRODUCTION 46

CHAPITRE 1 : NORME ISO 47

1.1. Introduction 47

1.2. Objectifs de l'OSI 47

1.3. Principes généraux de la structure en couches 48

1.4. Les 7 couches de l'architecture OSI 49

1.4.1. Couche des applications 50
1.4.2. Couche de présentation 50
1.4.3. Couche de session 50
1.4.4. Couche de transport 51
1.4.5. Couche d'acheminement 51
1.4.6. Couche de liaison 52
1.4.7. Couche de liaison physique 52

1.5. Les protocoles 53

1.5.1. Pour la couche de liaison physique 53
1.5.2. Pour la couche de liaison 53
1.5.3. Pour la couche d'acheminement 53
1.5.4. Pour la couche de transport 53
1.5.5. Pour la couche de session 53
1.5.6. Pour la couche de présentation 53
1.5.7. Pour la couche des applications 54

CHAPITRE 2 : CII-HB : DSA 55

2.1. Introduction 55

2.2. Concepts de base 55

2.3. Les couches de DSA 56

2.3.1. Gestion des applications 57
2.3.2. Gestion des messages 57
2.3.3. Gestion des communications 57

PAGE

2.4. Les protocoles 60

2.4.1. HDLC 60
2.4.2. X25 60
2.4.3. X21 60
2.4.4. Transfert de fichiers 60
2.4.5. Initialisation de processus à distance 60
2.4.6. Entrée de données interactive 61

2.5. Mise en oeuvre de DSA : composition d'un réseau 61

2.6. Administration de réseau 61

CHAPITRE 3 : AUTRES ARCHITECTURES 63

3.1. Introduction 63

3.2. SNA : IBM 63

3.2.1. Concepts de base 63
3.2.2. La structure en couches 64

3.3. DECNET : DEC 66

3.3.1. Concepts de base 66
3.3.2. La structure en couches 66

3.4. CNA : NCR 68

3.5. DCA : UNIVAC 69

3.5.1. Structure 69
3.5.2. Protocoles 71

PARTIE 3 : METHODES DE CONCEPTION DE SYSTEMES D'INFORMATIQUE DISTRIBUEE 72

INTRODUCTION 73

CHAPITRE 1 : DECISIONS DE CONCEPTION. UNE TAXONOMIE POUR LES SYSTEMES D'INFORMATIQUE DISTRIBUEE 74

1.1. Introduction 74

1.2. Points de départ 74

1.3. Les décisions de conception, la taxonomie 75

1.4. Appréciations 78

CHAPITRE 2 : METHODE DE CONCEPTION PROPOSEE PAR BOOTH 79

2.1. Introduction 79

PAGE

2.2. Etapes principales 79

2.2.1. Conception d'essai 79

2.2.1.1. sélectionner une structure de traitement . 80
2.2.1.2. sélectionner une structure de bases de données 80
2.2.1.3. sélectionner les composants de traitement de l'information 80
2.2.1.4. allouer les fonctions de traitement aux composants 80
2.2.1.5. allouer les fonctions de stockage aux composants 81
2.2.1.6. sélectionner une structure de réseau 81

2.2.2. Evaluation de la conception d'essai 81

2.2.2.1. analyse de coût 82
2.2.2.2. temps de réponse, fiabilité, flexibilité . 82

2.2.3. Réévaluations 83

CHAPITRE 3 : METHODE DE CONCEPTION PROPOSEE PAR MARTIN 84

3.1. Introduction 84

3.2. Description de la méthode 84

3.3. Appréciation 88

3.4. Comparaison des méthodes de conception présentées par Booth et Martin 88

CHAPITRE 4 : METHODE GENERALE DE CONCEPTION DE SYSTEMES REPARTIS .. 90

4.1. Introduction 90

4.2. Etablissement du cahier des charges 91

4.2.1. Introduction 91
4.2.2. Première étape : la définition des objectifs du système 91

4.2.2.1. l'analyse de la situation 91
4.2.2.2. l'analyse des besoins 92
4.2.2.3. l'analyse des contraintes 93
4.2.2.4. étude de faisabilité 93

4.2.3. Deuxième étape : la détermination des critères de choix 94

4.3. Recherche d'une solution 95

4.3.1. Introduction 95
4.3.2. Définition de l'architecture du système 96

4.3.2.1. introduction 96
4.3.2.2. distribution géographique des postes de travail 96

PAGE

4.3.2.3. répartition du traitement et du stockage . 96
4.3.2.4. définition de l'architecture du système de gestion 96

4.3.3. Choix des composants du système 96

4.3.3.1. introduction 96
4.3.3.2. choix des terminaux 97
4.3.3.3. choix des moyens de traitement et de stockage 97
4.3.3.4. choix des moyens de transport et optimalisation du réseau 97

4.3.4. Vérification de la solution 99

CONCLUSION 100

PARTIE 4 : OUTIL PRAGMATIQUE D'AIDE A LA CONCEPTION DE SYSTEMES D'INFORMATIQUE DISTRIBUEE 101

INTRODUCTION 102

CHAPITRE 1 : INTRODUCTION A L'OUTIL PRAGMATIQUE ET AUX OBJECTIFS POURSUIVIS 103

1.1. Introduction 103

1.2. Objectifs 104

1.3. Outil automatisé 105

CHAPITRE 2 : ETUDES PRELIMINAIRES REQUISES 106

2.1. Plan des études 106

2.2. Données fondamentales 106

2.2.1. Etude des comportements des équipements 106
2.2.2. Méthodes de mesure de performances 107

2.2.2.1 première catégorie : les simulations 108
2.2.2.2. deuxième catégorie : les stimulateurs 109

a) stimulateurs intégrés 109
b) stimulateurs intégrés sur un processeur interne 110
c) stimulateurs externes 111

2.2.2.3. troisième catégorie : les moniteurs 111

a) moniteurs hardware 112
b) moniteurs software 113
c) comparaison des deux types de moniteurs 113

PAGE

2.2.3. Analyse des mesures de performances 114

2.2.3.1. intérêt de la régression 115
2.2.3.2. exemple 115

2.2.4 Conclusion 118

2.3. Les données variables ou "utilisateur" 118

2.3.1 Etude de l'ensemble des "applications utilisateur" . 118

2.3.1.1. description des unités de travail, les types de transactions 118
2.3.1.2. description des utilisateurs terminaux ... 120
2.3.1.3. origine de ces informations 121

2.3.2. Etude de la topologie du réseau 121

2.3.2.1. classes de sites 121
2.3.2.2. relations inter-classes 122
2.3.2.3. relations intra-classes 124
2.3.2.4. exemple de relations 124
2.3.2.5. origine de ces informations 125

CHAPITRE 3 : CALCULS DE DIMENSIONNEMENT ENVISAGEABLES 126

3.1. Point de départ 126

3.2. Trafic potentiel dans le système distribué 126

3.2.1. Localisation des transactions 126
3.2.2. Problématique des liens physiques 128

3.2.2.1. mode de communication entre classes de sites "H" ou hiérarchique à 100 % 129
3.2.2.2. mode de communication entre classes de sites "E" ou équidistribué à 100 % 130
3.2.2.3. mode de communication "E" et "H" 130

3.2.3. Evaluation de l'activité des classes de sites 131

3.3. Evaluation du trafic inter-classes 131

3.4. Synthèse du trafic inter-classes 133

3.4.1. Evaluation du trafic sur les chemins de données individuels 133
3.4.2. Passage du point de vue logique à la réalisation physique des liens 135

3.5. Dimensionnement des équipements de transport 136

3.6. Résumé schématique 137

PAGE

CHAPITRE 4 : AUTOMATISATION DE L'OUTIL 138

4.1. Introduction 138

4.2. Description succincte de l'outil 138

4.2.1. Fonctions de base 138
4.2.2. Fonctions "utilisateur" 139
4.2.3. Structure du système automatisé 140
4.2.4. Problème des sorties 141

CHAPITRE 5 : PROCEDE ALTERNATIF DE MODELISATION DE RESEAUX 143

5.1. Introduction 143

5.2. Description de la méthode 143

5.2.1. Collecte des données 143
5.2.2. La topologie du système distribué, la matrice "émetteur-récepteur" 144

5.2.2.1. étude des éléments de la matrice 144
5.2.2.2. exemple 145
5.2.2.3. remarque 147

5.2.3. Dimensionnement 148

5.2.3.1. dimensionnement des liens 148
5.2.3.2. dimensionnement des stations de transport des données 148

5.3. Appréciation - conclusion 149

CONCLUSION 150

BIBLIOGRAPHIE 153

INTRODUCTION

Les systèmes d'informatique distribuée constituent un des plus importants développements de l'histoire de l'informatique. Les progrès antérieurs dépendaient presque tous des motivations technologiques.

Les systèmes d'informatique distribuée reflètent une approche rationnelle de l'informatique et de ses utilisations. En fait elle est l'expression d'un mode d'organisation entièrement nouveau, d'une maturité.

L'informatique distribuée met l'accent sur la distribution des fonctions plutôt que sur les moyens. Il n'est plus question de réformer l'organisation de l'entreprise pour l'adapter au schéma des ressources informatiques.

Aujourd'hui l'informatique peut de ce point de vue être neutre, transparente. Tous les schémas d'informatisation peuvent être réalisés, permettant tous les modes d'organisation du travail, jusqu'à la convialité la plus totale.

La généralisation du phénomène "informatique distribuée" aura sans doute des conséquences dont on mesure encore mal l'ampleur.

Le mémoire comporte quatre parties :

La première partie est consacrée à une étude des systèmes d'informatique distribuée. Un premier chapitre retrace l'évolution des SID (systèmes d'informatique distribuée). Le deuxième chapitre passe en revue les aspects économiques et organisationnels des SID. Le troisième chapitre fournit quelques exemples typiques d'utilisation de SID. Au terme de ces trois chapitres nous essayons de dégager les caractéristiques fondamentales des SID (chapitre 4). Les trois fonctions fondamentales découvertes dans les SID sont étudiées au chapitre 5. En guise de complétude nous présentons quelques avantages et désavantages majeurs des SID (chapitre 6).

La deuxième partie est consacrée à l'étude des relations entre l'architecture de réseau et le système d'informatique distribuée. Cette partie étudie la fonction de transport dans les SID et analyse certains produits d'architecture de système distribué et en particulier DSA (distributed systems architecture) de CII-Honeywell Bull qui est proche du système de référence pour l'interconnexion de systèmes ouverts (ISO) que nous exposons succinctement au début de cette partie.

La troisième partie a pour but d'étudier les problèmes rencontrés lors de la conception de SID, de dégager les éléments à prendre en compte et de trouver des méthodes adéquates de conception de SID.

La quatrième partie présente des outils d'aide à la conception de systèmes d'informatique distribuée, et en particulier un outil qui, en partant d'une analyse conceptuelle et d'un ensemble d'applications, permet de dimensionner le réseau sur lequel elle doit être mise en oeuvre. Nous avons étudié ce type d'outil lors de notre stage chez CII-HB à Paris. Cette étude a débouché sur la conception d'un outil automatisé.

PARTIE 1 : LES SYSTEMES D'INFORMATIQUE DISTRIBUEE

INTRODUCTION

Les systèmes d'informatique distribuée ou répartie (1) sont basés sur le principe de la distribution fonctionnelle. Ceci implique que les trois fonctions fondamentales, à savoir les fonctions de traitement, de stockage (et de mouvement de données) sont réparties sur différents composants du système distribué. Ces composants peuvent être rapprochés des utilisateurs.

Cette répartition des fonctions s'oppose à leur centralisation, où la capacité de traitement est concentrée au niveau d'un seul élément du système informatique.

Pour comprendre les systèmes d'informatique distribuée, il est utile de retracer brièvement l'évolution des systèmes informatiques vers des systèmes distribués. Cette évolution de plus en plus rapide possède une composante économique et organisationnelle qu'il ne faut jamais perdre de vue.

Une analyse des systèmes distribués existants montre qu'ils sont de nature multiple. Une classification de ces différents systèmes devrait faciliter la compréhension des problèmes rencontrés lors de leur étude.

---

(1) Certains auteurs (L2) font la distinction entre les concepts d'informatique répartie et d'informatique distribuée. Le premier concept concerne les systèmes où on se limite aux échanges de lots de données en différé avec le système central. Le deuxième vise les systèmes où on distribue davantage l'intelligence machine de façon à permettre des traitements locaux. Dans la suite nous utiliserons indifféremment ces deux termes.

CHAPITRE 1 : EVOLUTION VERS LES SYSTEMES D'INFORMATIQUE DISTRIBUEE

1.1. INTRODUCTION

Nous allons retracer les grandes lignes de l'évolution des systèmes informatiques vers les systèmes d'informatique distribuée que l'on connaît actuellement.

La plupart des auteurs ((M1), (M2), (V2)) discernent quatre étapes essentielles dans cette évolution :

1. connexion directe des terminaux à l'ordinateur central
2. apparition des concentrateurs
3. apparition des frontaux
4. réseaux "généraux" d'ordinateurs

Nous commenterons brièvement ces quatre étapes. Le lecteur intéressé pourra trouver un examen succinct dans (V2) et (A2), et des détails dans (M1).

1.2. EVOLUTION

Les systèmes d'informatique distribuée ont fait leur apparition suite à une série de développements organisationnels logiquement distincts et non à cause d'une génération spontanée d'une certaine capacité de traitement.

1.2.1. étape 1 : connexion directe des terminaux à l'ordinateur central

Les premiers systèmes informatiques ont été développés pour traiter des problèmes scientifiques. Les données du problème étaient introduites dans le système par l'intermédiaire de cartes perforées. Après le traitement (les calculs), les résultats étaient imprimés à la fin de la "tâche" (job). La nature de ces problèmes impliquait l'utilisation d'un système centralisé.

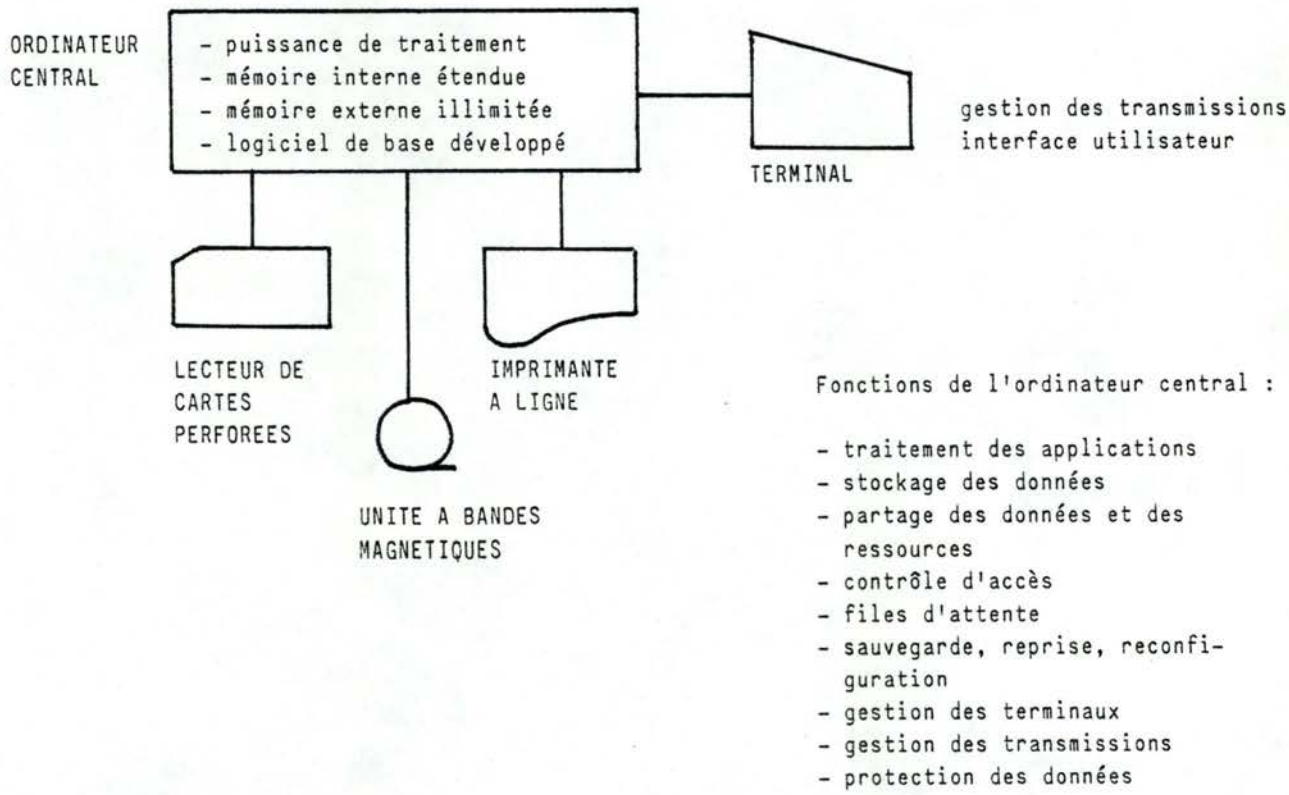
Lorsque ces ordinateurs évoluèrent pour fournir des services de traitement d'information plus complexes, les techniques traditionnelles d'interaction avec l'ordinateur devinrent insuffisantes pour certaines classes d'utilisateurs. En effet, beaucoup d'environnements commerciaux nécessitent des systèmes de traitement avec lesquels l'utilisateur peut dialoguer.

La réponse à ces besoins fut la connexion directe d'un terminal à l'ordinateur par des liaisons de données. Cette connexion avait le même statut que celle qui reliait les lecteurs de cartes, les unités de bandes magnétiques, etc ... à l'ordinateur (cfr figure 1). Désormais l'utilisateur pouvait demander des services de traitement d'information sans devoir passer par l'intermédiaire de cartes perforées. Au terme du traitement, la réponse à la demande de l'utilisateur pouvait être lue directement à partir du terminal "interactif".

La distribution des fonctions au sein d'un tel système informatique était simple :

- le maximum dans le système central
- le minimum près du terminal

figure 1.



1.2.2. étape 2 : apparition des concentrateurs

Grâce à l'utilisation de plus en plus importante des lignes téléphoniques dans les systèmes informatiques, les utilisateurs éloignés purent profiter des services interactifs offerts par le système.

Le nombre de terminaux augmentant, la connexion directe des terminaux à l'ordinateur central devint problématique :

- d'une part il y a les coûts résultant :

  * du matériel nécessaire à la connexion d'un terminal à l'ordinateur central

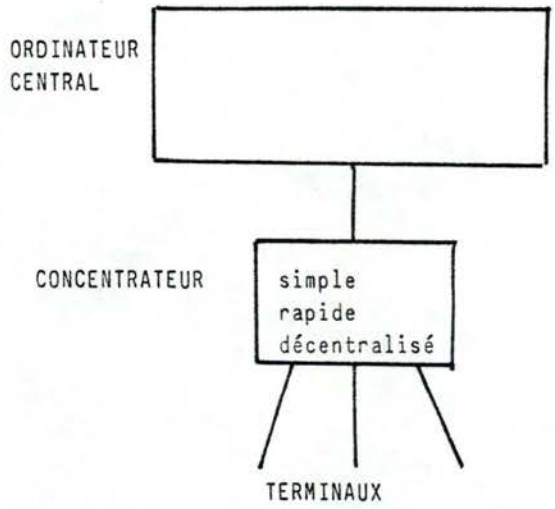
  * de la monopolisation de la puissance de l'ordinateur central

  * de la faible utilisation des lignes de transmission

- d'autre part la complexité d'une telle organisation entraîne une forte dégradation du temps de réponse du système.

Le problème de la faible utilisation des lignes a déjà été partiellement résolu à l'étape 1 par l'utilisation de liaisons multipoints, malheureusement au détriment de l'ordinateur central qui a vu sa charge accrue par la gestion de ces liaisons.

L'introduction des concentrateurs, la plupart du temps des miniordinateurs orientés vers les transmissions donc programmables et situés à proximité des terminaux, a permis d'apporter une réponse satisfaisante à ces deux points. Les concentrateurs assurent au moindre coût les fonctions de communication et de transport. On a tiré parti des concentrateurs pour leur faire remplir certaines fonctions liées aux applications à côté des fonctions de concentration du trafic.



Fonctions de l'ordinateur central :

- traitements des applications
- stockage des données
- partage des ressources et des données
- contrôle d'accès
- files d'attente
- sauvegardes, reprises
- allègement de la gestion des terminaux

Fonctions des concentrateurs :

- gestion des transmissions
- gestion des terminaux
- prétraitements
- stockage des données et traitements
- sauvegardes et reprises
- transmission de fichiers

1.2.3. étape 3 : apparition des frontaux

Comme nous l'avons vu dans les deux premières étapes, l'ordinateur central reste chargé des fonctions de gestion des transmissions avec les terminaux (étape 1) ou avec les concentrateurs (étape 2) au moyen d'un contrôleur de communications câblé.

L'apparition des miniordinateurs et l'expérience acquise avec les concentrateurs ont conduit à remplacer les contrôleurs de communication câblés par des contrôleurs programmés, dits processeurs frontaux (front-end processors FEP).

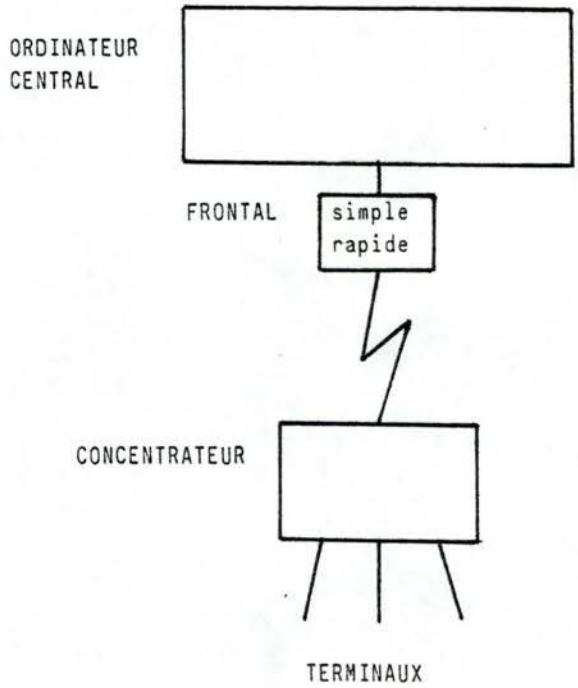
Ces frontaux, intimement reliés à l'ordinateur central par une interface rapide, le déchargent de fonctions telles que la gestion des transmissions.

Ils réalisent les fonctions de contrôle du réseau de télécommunication. Ils gèrent automatiquement les ordres d'entrées/sorties vers les différentes stations du réseau.

Les programmes fournis par le système pour gérer les échanges avec les terminaux et rendre les programmes d'application indépendants de caractéristiques physiques des appareils sont exécutés dans le frontal. Celui-ci gère également les incidents liés aux transmissions.

Un processeur frontal constitue une extension du central dans lequel les opérations de gestion sont décentralisées. L'utilisation d'un frontal accroît la performance globale du système et autorise une souplesse dont on ne dispose pas avec un contrôleur câblé.

Le fait que le frontal soit programmable permet de modifier les procédures de transmission en fonction des caractéristiques du réseau sans avoir à modifier le logiciel implanté au central.



Fonctions de l'ordinateur central :

- traitement des applications
- stockage des données
- partage des ressources et des données
- contrôle d'accès
- files d'attente
- sauvegardes, reprises
- gestion des terminaux

Fonctions du frontal :

- gestion des transmissions
- fonctions d'un concentrateur
- gestion de files d'attente

1.2.4. étape 4 : réseaux généraux d'ordinateurs

Dans les trois premières étapes, le système "téléinformatique" a une structure étoilée autour de l'ordinateur central, ce qui est caractéristique d'un système initialement centralisé qui se décentralise lentement. Dans cette dernière étape, nous allons prendre en compte deux phénomènes qui ont permis de généraliser la notion de système d'informatique distribuée.

L'avènement des micro-ordinateurs et des terminaux intelligents a créé des possibilités d'utiliser des ordinateurs dans les environnements les plus divers, tout en assurant une exploitation économique. Maintenant les utilisateurs peuvent mettre en oeuvre un certain nombre de fonctions, à des prix intéressants, à tous les niveaux d'une organisation.

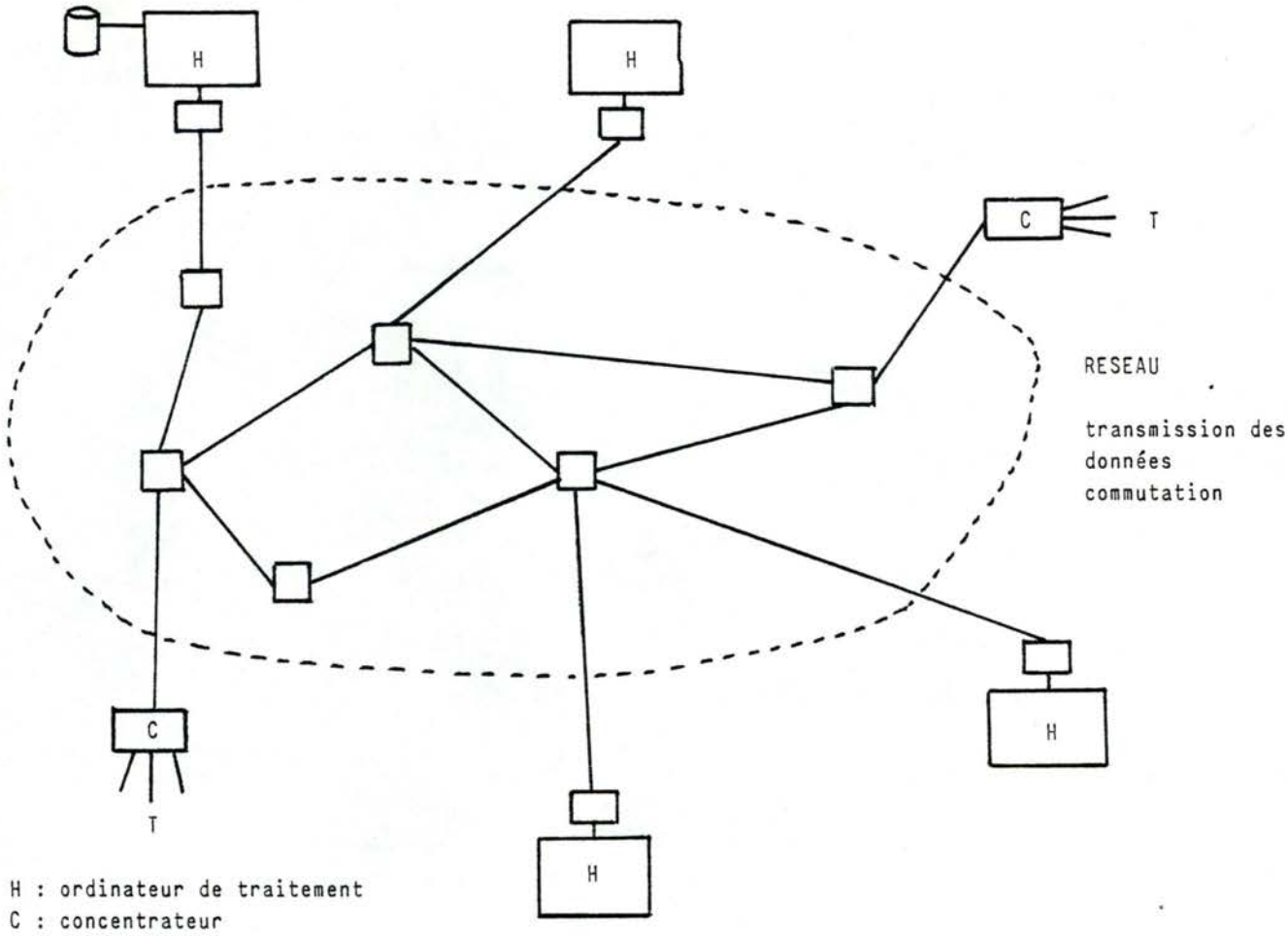
Cette possibilté d'installer l'ordinateur à proximité des utilisateurs a entraîné rapidement une période de décentralisation. Cette décentralisation a pour conséquence que des entreprises se retrouvent avec quelques gros systèmes (tels qu'on les a vus aux étapes 1 à 3), auxquels s'ajoutent les nouveaux mini et micro-ordinateurs souvent de constructeurs différents.

La gestion des entreprises réclame un contrôle centralisé, et le fait de permettre aux mini et micro-ordinateurs d'accéder à la base de données centralisée de l'organisation exerce une pression pour connecter les miniordinateurs à l'ordinateur central.

Dans les réseaux généraux d'ordinateurs :

- un terminal peut accéder à différents ordinateurs pour bénéficier d'une variété de services

- les communications entre ordinateurs permettent de réaliser des applications d'informatique distribuée



L'idée de la répartition des fonctions n'est pas sans incidence sur l'architecture du réseau, qui peut elle-même avoir un caractère centralisé ou décentralisé.

- l'architecture hiérarchique centralisée permet au calculateur frontal d'optimiser le trafic total de messages qu'un réseau de lignes à débit donné peut supporter, ce qui donne l'occasion à l'utilisateur de faire des économies en coût de location des lignes spécialisées.

- l'architecture décentralisée maillée (cfr exemple schéma ci-dessus) ne permet pas une telle optimisation car aucun contrôleur de noeud de réseau ne peut prévoir les messages qui lui seront envoyés par d'autres contrôleurs de noeud. Par contre ces réseaux offrent en général une ou plusieurs voies possibles pour le routage adaptatif des messages, permettant aux contrôleurs de noeud d'équilibrer le trafic sur le réseau entier.

CHAPITRE 2 : ASPECTS ECONOMIQUES ET ORGANISATIONNELS DES SYSTEMES D'INFORMATIQUE DISTRIBUEE

2.1. INTRODUCTION

Contrairement à ce qu'on vient de voir, ce ne sont pas uniquement les progrès constants de la technologie qui ont contribué à l'essor des systèmes d'informatique distribuée, mais c'est l'apparition d'un mode nouveau d'organisation des entreprises qui, en utilisant les nouvelles technologies puissantes, explique l'évolution rapide de l'informatique distribuée.

2.2. CENTRALISATION ET DECENTRALISATION

Dans un système centralisé, les moyens de traitement et de stockage se trouvent situés dans un même lieu. Des terminaux géographiquement dispersés dialoguent avec le système central.

Dans un tel contexte, l'ordinateur central doit être pourvu d'un logiciel de base très élaboré. En éffet, il doit, d'une part, assurer le contrôle du partage des ressources du système et, d'autre part, supporter des défaillances partielles en mettant en oeuvre des mécanismes de reconfiguration automatique. Ceci explique que, de plus en plus, les chefs d'entreprise se sont plaints de la lourdeur de leur système d'information et de la lenteur qu'implique toute modification de procédure.

Comme alternative à de telles structures apparaît actuellement la tendance à la décentralisation. Celle-ci repose sur le principe que toute cellule de travail d'une entreprise doit avoir les moyens d'assurer sa gestion, moyens qu'elle ne partage pas avec une autre unité.

Dans de telles architectures le traitement et le stockage de l'information, eux-même placés plus près des utilisateurs, sont éclatés sur plusieurs ordinateurs plus petits ou sur des terminaux intelligents qui peuvent échanger des données entre eux par l'intermédiaire de moyens de communication. La disparition du système central introduit un autre type de complexité dans la mesure où il faut malgré tout veiller à assurer au système global sa cohérence et sa cohésion, mais les services offerts sont probablement meilleurs.

Certains auteurs (V2) considèrent les systèmes semi-centralisés comme une solution intermédiaire entre les deux extrêmes évoqués ci-dessus. Les caractéristiques essentielles sont la saisie autonome de l'information, un premier traitement éventuel sur le site distant, le transfert depuis ce site vers le système central où l'information sera à nouveau traitée et/ou exploitée par d'autres utilisateurs.

2.3. MOTIVATIONS A LA DECENTRALISATION

Des considérations d'ordre économique et organisationnel ont favorisé le partage des capacités d'un système central parmi un nombre important d'utilisateurs.

Actuellement les progrès technologiques, et en particulier la percée des microprocesseurs, permettent d'atteindre une capacité de traitement et une intelligence inespérées avec des organes de prix très peu élevé qu'on peut donc répartir en de nombreux points d'une organisation ou rendre accessibles à de très petits budgets.

Les progrès technologiques récents contribuent à une disponibilité croissante des moyens de communication et de télétraitement, et au développement d'une nouvelle organisation du logiciel. Ainsi purent être jetées les bases de la distribution de l'informatique.

Conjointement à cette avance technologique, on peut observer dans beaucoup d'entreprises le désir ou la nécessité de décentraliser l'organisation. Le phénomène de décentralisation demande une description succincte de l'environnement socio-économique des entreprises.

D'après F. Dalle (D1), aujourd'hui "faire de la décentralisation des organisations" est un objectif prioritaire.

Actuellement un plus grand nombre d'employés d'une entreprise peuvent, veulent, doivent être des décideurs, c'est-à-dire qu'ils peuvent, veulent et doivent s'exprimer, s'impliquer et se réaliser plus complètement dans leurs tâches quotidiennes.

- Premièrement, aujourd'hui un plus grand nombre de membres de l'entreprise peuvent être des décideurs. Ceci est une conséquence de l'élévation généralisée du niveau d'éducation des gens.

- Deuxièmement, aujourd'hui un nombre croissant de membres de l'entreprise veulent donc être des décideurs, mais il faut reconnaître que l'entreprise se révèle souvent incapable de leur offrir des responsabilités à la mesure de leurs capacités.

- Troisièmement, aujourd'hui il se trouve que l'entreprise a intérêt à remettre en cause les principes tayloriens d'organisation et à faire en sorte qu'un plus grand nombre de ses membres soient réellement des décideurs. Il s'avère que les grandes organisations centralisées sont beaucoup plus vulnérables que les petites aux turbulences qui dominent désormais l'environnement des entreprises. Or, dans le contexte économique actuel, il devient essentiel que l'entreprise fasse preuve de souplesse et de rapidité de réaction, de flexibilité pour s'adapter aux turbulences, et se montrer créative et innovatrice pour anticiper les changements et survivre. Les schémas traditionnels (tayloriens) ont fait la preuve de leur efficacité dans la phase de croissance relativement facile et rapide que nous avons connue depuis la seconde guerre mondiale ; s'ils ont donné aux entreprises une organisation pour aller vite, loin et tout droit, ils se sont montrés insuffisants face à l'évolution économique actuelle.

Pour remédier à cette situation, il faut prendre le contre-pied des méthodes d'organisation qui ont prévalu jusqu'ici. Il faut faire du taylorisme à l'envers, c'est-à-dire qu'il faut entre autres :

- démassifier les structures de l'entreprise

- la décentraliser réellement ; c'est-à-dire créer des cellules plus autonomes et de taille humaine qui rendent responsables le plus grand nombre possible d'hommes dans l'entreprise.

- faire en sorte que toute la richesse informative qui existe dans l'écorce de l'entreprise, à la périphérie, vienne irriguer par induction les organes chargés de la stratégie.

La technologie informatique actuelle est belle et bien capable de fournir les supports matériels nécessaires pour réaliser ces réorganisations des entreprises.

On ne peut cependant pas généraliser ces nouvelles structures d'organisation, et, dans certains cas, le maintien d'une volonté de centralisation peut être un frein au choix de systèmes répartis qui menaceraient de remettre en question la structure des pouvoirs en place.

La principale crainte dans un tel environnement est celle d'une décentralisation des décisions qui peut conduire à une anarchie et à une perte de contrôle par la direction. Pour garantir la cohérence des décisions, l'entreprise doit se doter de procédures claires et strictes. A cette crainte nous ajoutons celle de la perte de contrôle par la direction de l'informatique face à l'élaboration de logiciels de gestion parallèles et non compatibles les uns avec les autres et face à un parc d'équipements de plus en plus hétérogène. Ici aussi l'entreprise doit élaborer des standards afin de maintenir la cohérence souhaitée.

CHAPITRE 3 : EXEMPLES DE SYSTEMES D'INFORMATIQUE DISTRIBUEE

3.1. INTRODUCTION

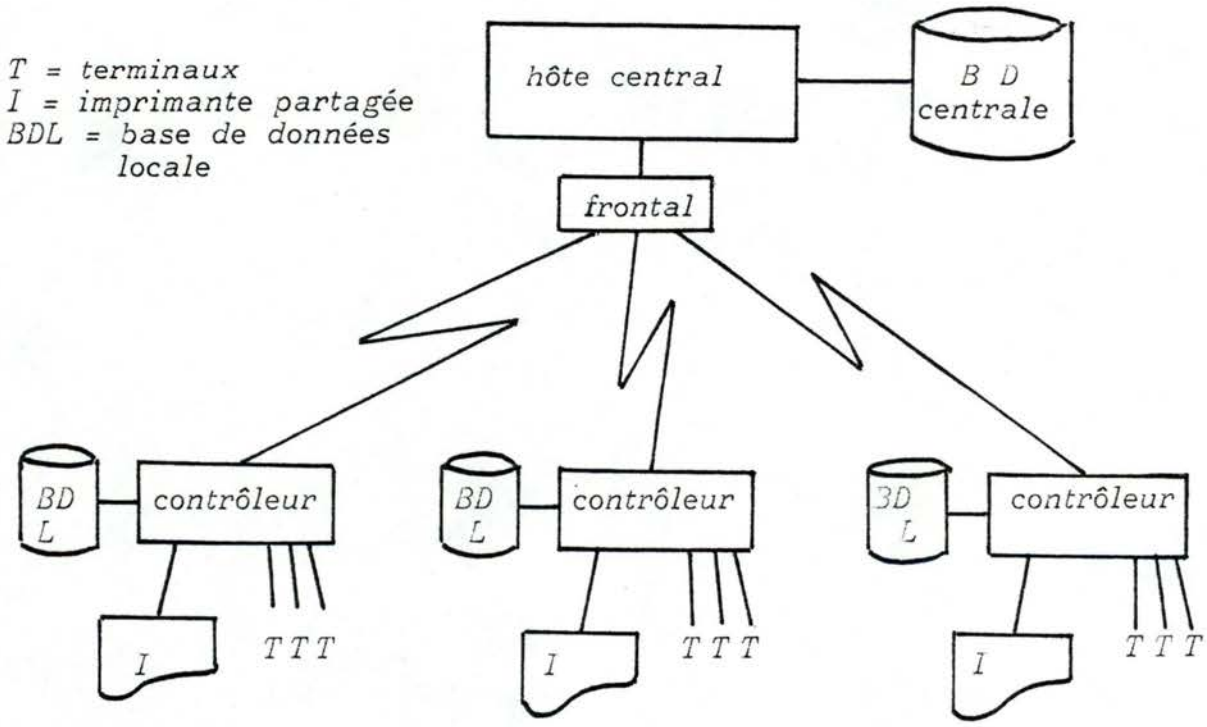
Dans ce chapitre nous décrivons un certain nombre de systèmes typiques d'informatique distribuée, présentant chacun un certain degré de distribution fonctionnelle.

Les exemples se basent sur des systèmes réels et sont présentés avec un minimum de détails pour que la généralité des exemples ne soit pas perdue.

Lors de l'analyse des exemples de systèmes d'informatique distribuée nous avons constaté que ces systèmes sont essentiellement inévitables. D'ailleurs la majorité des responsables de l'informatique ne se posent plus la question : "Faut-il installer des systèmes d'informatique distribuée ?", mais avant tout : "Comment réaliser la meilleure installation de ces systèmes ?".

3.2. SYSTEME D'ENREGISTREMENT DE COMMANDES

Dans le système présenté ci-dessous par (B1), peu de fonctions ont été distribuées.



Dans ce système, un contrôleur de terminaux intelligent (mini/micro) est localisé au niveau de chaque grappe de terminaux. Ces terminaux relativement simples sont constitués par un clavier et un écran (CRT). Dans chaque grappe il y a un terminal à imprimante (hardcopy), partagé entre les terminaux à écran.

Les contrôleurs sont programmés pour réaliser certaines fonctions d'entrée de données, notamment pour réaliser les formattages des différents écrans. L'utilisateur doit simplement remplir les zones non protégées. Le contrôleur assure le contrôle de validité des données introduites et signale immédiatement toute erreur à l'utilisateur.

Chaque transaction correcte est envoyée à l'ordinateur hôte. Chaque réponse de l'ordinateur hôte est formattée par le contrôleur de terminaux et sortie à l'écran approprié et/ou sur l'imprimante partagée.

Si la connexion avec l'ordinateur hôte est perdue (panne d'ordinateur, problèmes de lignes), le contrôleur peut stocker localement des transactions sur son disque souple, tout en informant l'utilisateur du terminal du statut temporaire "off-line".

Ces transactions accumulées sont envoyées vers l'ordinateur hôte dès qu'un échange devient possible. Le contrôleur des terminaux sait changer entre différents modes (off-line/local, on-line) selon les circonstances.

Cet exemple est une illustration réaliste d'un système d'informatique distribuée qu'on retrouve dans beaucoup d'entreprises.

Un volume considérable de traitements a été enlevé de l'ordinateur hôte, et les fonctions qu'on a distribuées (contrôle des terminaux, éditeur) correspondent le moins aux capacités de l'ordinateur hôte, et le plus aux capacités des mini et micro-ordinateurs utilisés comme contrôleurs de terminaux.

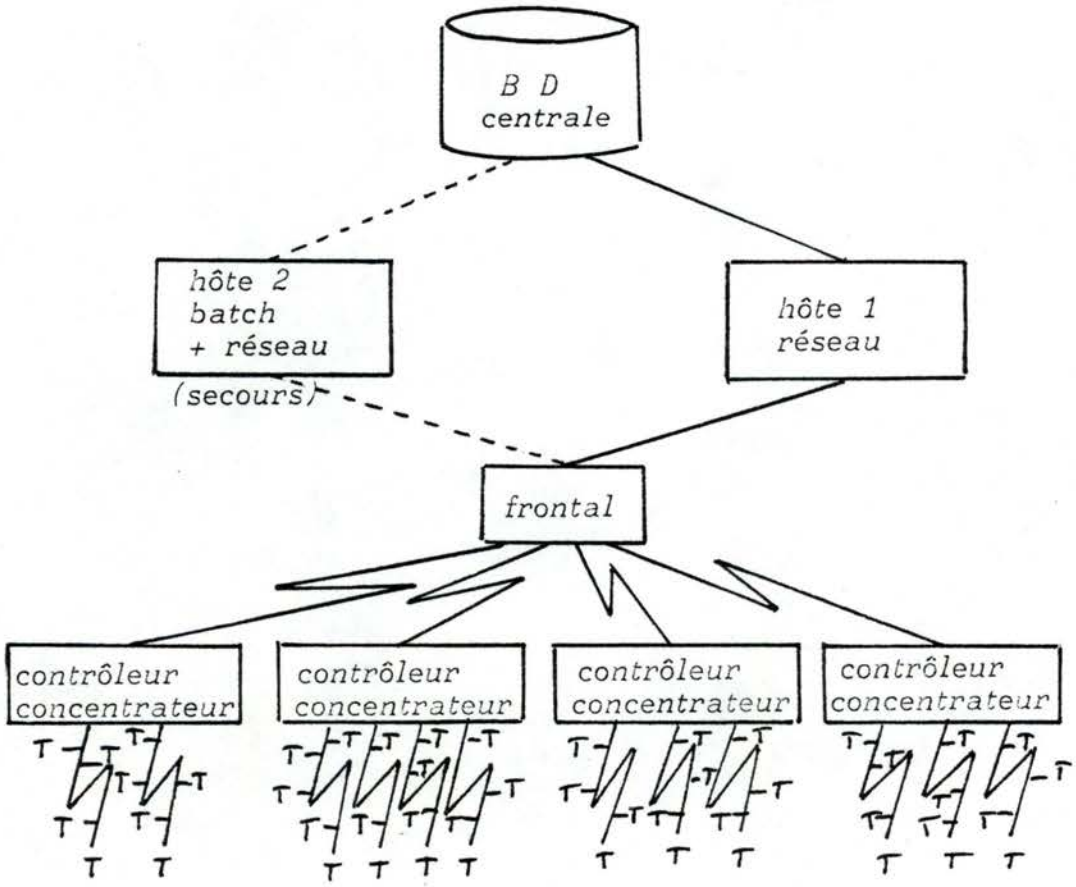
Ce type de système constitue un premier pas vers une distribution fonctionnelle plus importante. Le pas suivant serait que les contrôleurs traitent des éléments de l'application.

3.3. SYSTEME DE RESERVATION DE PLACES D'AVION

Dans la majorité des systèmes de réservation de places d'avion et dans certains systèmes informatiques d'organismes bancaires (ex : Banque Internationale à Luxembourg, le Credito Italiano), on découvre un schéma de distribution hiérarchique.

Tout le traitement de l'information est centralisé sur un site unique, d'habitude sur un gros ordinateur hôte secondé par un ordinateur en réserve. Ce deuxième ordinateur réalise normalement le traitement par lots, mais sert d'ordinateur de secours au premier, qui réalise les réservations proprement dites. Dans le cas d'une panne du premier ordinateur, le deuxième arrête ses traitements par lots et reprend les réservations.

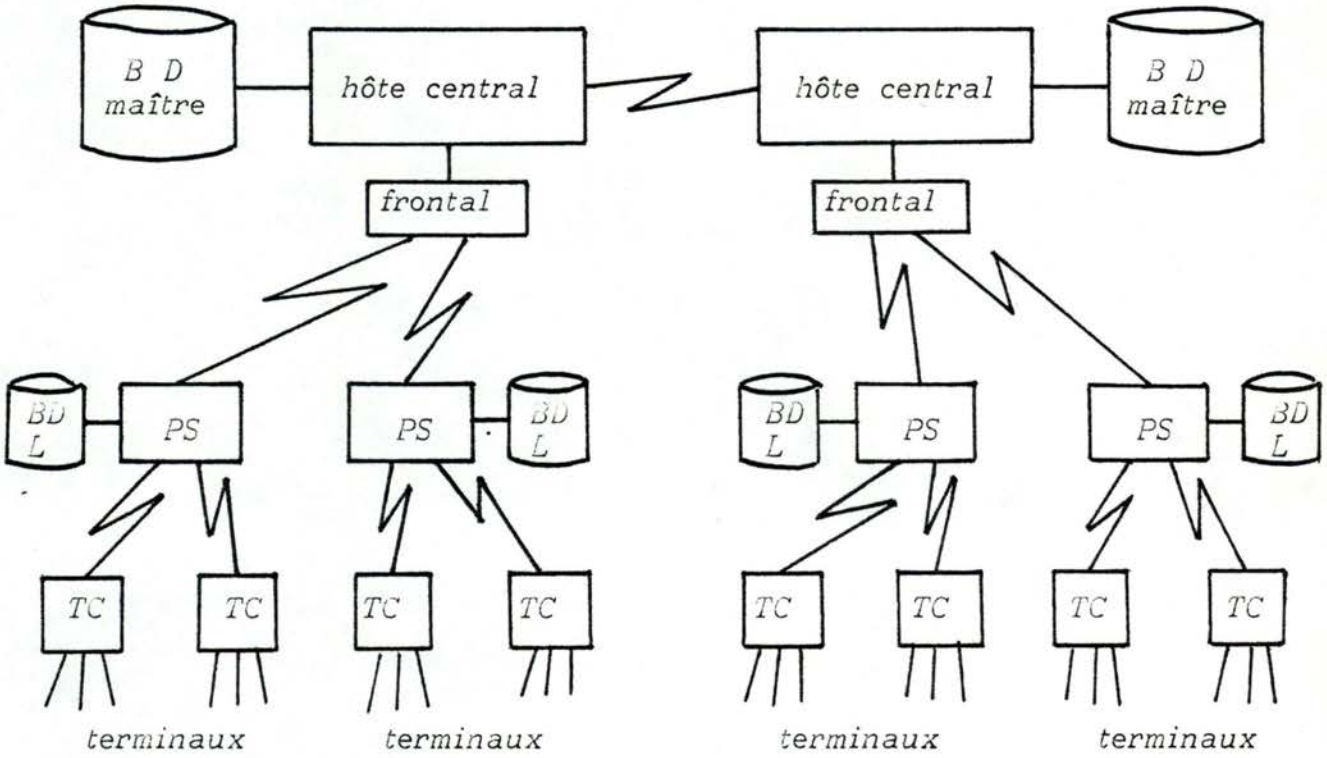
La distribution fonctionnelle concerne uniquement les fonctions de traitement du réseau. Le contrôle des terminaux (généralement un terminal non intelligent à CRT + clavier + imprimante) et des liens associés se fait dans les concentrateurs. Les terminaux sont connectés sur des lignes multipoint (multidrop), c'est-à-dire que plus d'un terminal est connecté à un seul lien afin de réduire le coût des liens. Les concentrateurs jouent aussi le rôle de multiplexeurs pour le trafic entre l'ordinateur hôte et les terminaux, pour réduire les coûts des liens de transmission et aussi pour réduire la manipulation des communications au niveau de l'ordinateur hôte.



On pourrait être tenté de qualifier cet exemple de "centralisé" plutôt que de "distribué". Cependant il s'agit bien d'un système distribué : certaines fonctions logiques ont été distribuées sur les concentrateurs.

3.4. SYSTEME BANCAIRE

Le système bancaire schématisé ci-dessous est un système présentant à la fois une distribution horizontale et hiérarchique des fonctions.



BDL = base de données locale - copie
TC = contrôleur
PS = processeur satellite

Le champ d'action de la banque est divisé en deux régions, chacune étant desservie par un ordinateur hôte de grosse taille. Chacun de ces hôtes se trouve à la racine d'une structure hiérarchique constituée par des processeurs satellites, des contrôleurs de terminaux et des terminaux.

Chaque ordinateur satellite rend des services à une filiale ou à plusieurs filiales plus petites, physiquement proches les unes des autres. Chaque processeur satellite possède sa propre base de données locale contenant des informations sur les comptes des clients locaux réguliers. Ces données représentent une copie d'une partie des informations contenues dans la base de données principale maintenue sur les ordinateurs hôtes.

Les contrôleurs de terminaux réalisent également des fonctions relatives aux applications telles que le formattage des écrans, l'édition des données collectées avant de les envoyer à l'ordinateur satellite.

Chaque satellite est équipé de programmes d'application capables de traiter toutes les transactions bancaires "normales" (par exemple : dépôt et retrait d'argent) entrées par les terminaux locaux. Comme les autres transactions bancaires sont beaucoup plus complexes (demande de prêt, ... etc), ces transactions sont traitées au niveau de l'ordinateur hôte.

Ce système tire profit des structures d'utilisation de l'organisation par les clients, et il est adapté à cette structure. Par exemple le fait que les clients travaillent d'habitude avec la filiale la plus proche de leur domicile permet de distribuer la base de données sur les filiales.

Ce système tire également profit d'une structure naturelle des transactions. Un nombre élevé de transactions demandent un traitement peu compliqué et ainsi ce traitement est assuré par les ordinateurs satellites. Le nombre de transactions nécessitant un traitement plus complexe est beaucoup plus petit, ces traitements sont assurés par les ordinateurs hôtes.

Trouver des structures de transactions et les exploiter durant la phase de conception des fonctions de traitement et de gestion des données est la base d'un système d'informatique distribuée performant.

3.5. SYSTEMES BANCAIRES EVOLUES

Le système informatique bancaire présenté ci-après est celui de la Bank of Louiville, Kentucky (USA), par (E1).

Pour le traitement des données, la banque utilise un ordinateur IBM 370 auquel sont connectés des terminaux qui se trouvent dans les divers services du siège central. Les filiales qui désirent obtenir des informations relatives aux états des comptes accèdent à l'ordinateur central par des unités "audio response".

Pour améliorer les services à la clientèle, notamment en ce qui concerne les prêts, la direction a décidé d'installer des micro-ordinateurs qui réduisent en même temps la dépendance des filiales de l'ordinateur central.

La banque a acheté en tout 65 micro-ordinateurs Apple II + machine à écrire IBM modifiée, répartis dans les différentes filiales et départements. Ces ordinateurs réalisent actuellement quatre fonctions principales qui, avant, étaient réalisées par des calculatrices programmables ou de façon manuelle.

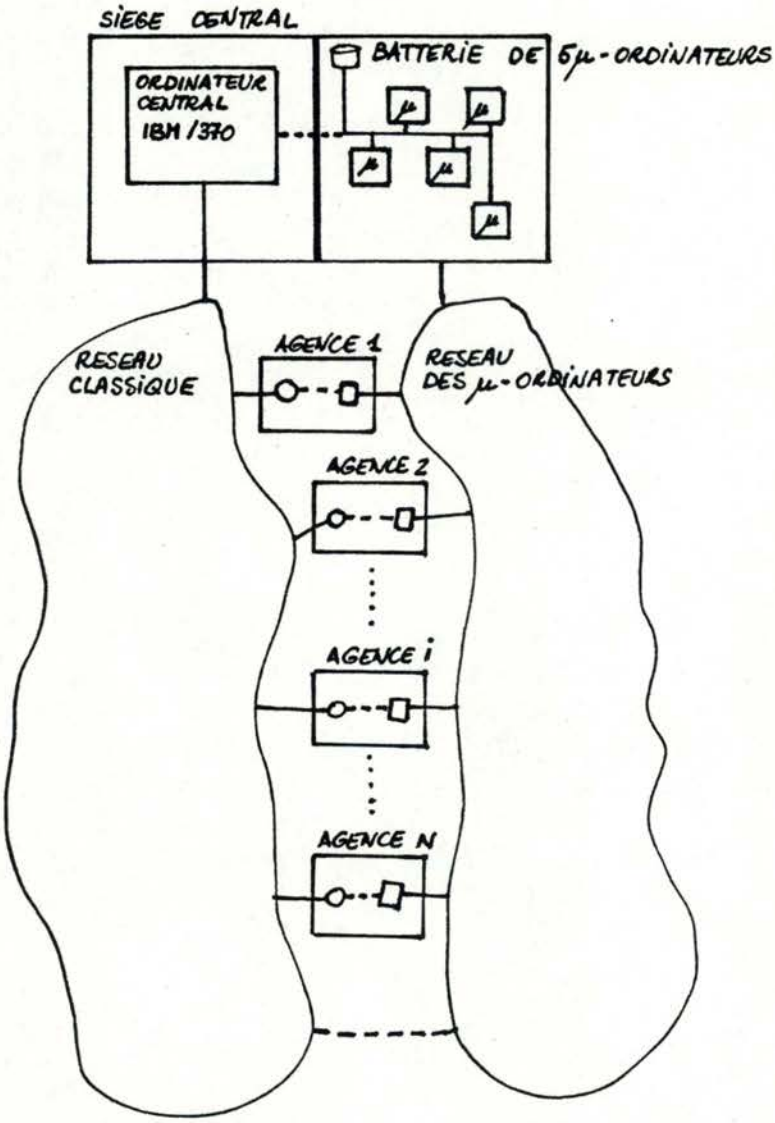
La première fonction assurée par les micro-ordinateurs concerne les calculs de paiement de prêts. Les micro-ordinateurs ont avantageusement remplacé les calculatrices programmables parce qu'ils offrent une plus grande flexibilité aux utilisateurs.

La deuxième fonction concerne l'édition de textes légaux grâce à des progiciels de traitement de texte tournant sur les Apple II. Les dactylos disposent ainsi d'un outil très puissant.

Les micro-ordinateurs aident les clients à la rédaction des demandes d'obtention de prêts, ce qui constitue la troisième fonction.

La quatrième fonction concerne le calcul des taux d'intérêt. Les données générées lors de l'aide à la rédaction des demandes d'obtention de prêts sont envoyées au site central, où une batterie de 5 Apple II les reçoivent et les stockent sur disque souple. Les employés examinent les demandes et renvoient leurs décisions dans un délai très court.

En face de cette structure flexible, le nombre d'applications nouvelles s'accroît rapidement. En fait ce système permet une introduction aisée de la bureautique. Il y a la possibilité d'interconnecter le réseau classique et le réseau des micro-ordinateurs, ce qui permet de faire tourner de nouvelles applications sur les micro-ordinateurs, de réduire en même temps la charge du réseau classique et de concevoir un système d'informatique distribuée original.



O := TERMINAL CONNECTÉ AU RÉSEAU CLASSIQUE (IBM/370)

□ := µ-ORDINATEUR (APPLE II)

---- := INTERCONNEXION DES 2 RÉSEAUX

CHAPITRE 4 : CLASSIFICATION ET CARACTERISTIQUES FONDAMENTALES DES SYSTEMES D'INFORMATIQUE DISTRIBUEE

4.1. INTRODUCTION - LE CONCEPT FONDAMENTAL

Au chapitre précédent, nous avons présenté quelques systèmes d'informatique distribuée (SID). Etant donné la grande variété de SID, notre étude ne nous a pas permis d'aboutir à une définition rigoureuse pour ces systèmes. En effet, toute définition était ou bien trop restrictive ou bien trop vaste.

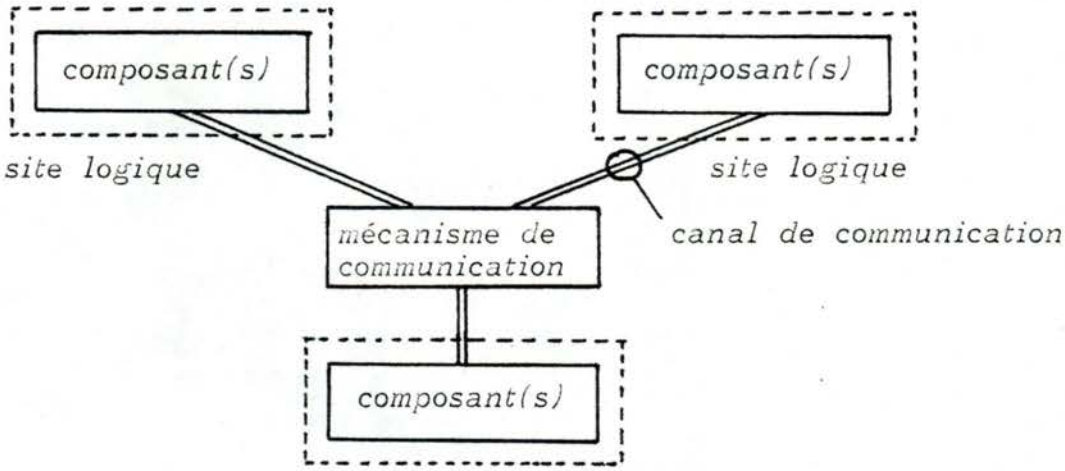
Cependant nous avons pu dégager l'idée de base des SID, qui est celle de la distribution fonctionnelle. Les fonctions de transport, de traitement et de stockage des données sont distribuées sur les différents composants du système. Ces composants sont localisés dans divers sites logiques interconnectés (cfr figure).

La notion de distribution n'implique pas nécessairement la notion de distance géographique. Un site logique est un endroit conceptuel dans la structure du système. Certains protocoles et des données à traiter y sont associés. Le concept de site logique permet de définir une structure logique relativement indépendante des structures physiques environnantes.

La distinction faite entre systèmes logiques et physiques est d'une importance capitale. En effet, le développement d'un système logique est une réponse aux besoins organisationnels des entreprises, face à des changements perpétuels de leur environnement et de leur propre structure.

La mise en oeuvre d'un SID à partir de ce schéma conceptuel est réalisée grâce à des mappings physiques en accord avec les contraintes technologiques actuelles.

figure



4.2. CLASSIFICATIONS DES SYSTEMES D'INFORMATIQUE DISTRIBUEE

Nous allons proposer deux classifications auxquelles on peut aboutir. La première classification est déduite d'une analyse superficielle des SID et se base avant tout sur la notion de types de systèmes. La deuxième nécessite une étude plus approfondie des SID.

4.2.1. Classification par type

La classification que nous présentons est reprise par la majorité des auteurs que nous avons consultés : (B1), (E1), (E2), (T2).

On peut distinguer trois types fondamentaux de systèmes d'informatique distribuée :

- les systèmes hiérarchisés
- les réseaux constitués de processeurs ou de stations de travail qui coopèrent
- les systèmes départementaux "stand-alone" (indépendants) qui peuvent communiquer les uns avec les autres

Dans un système du type hiérarchique, dont la structure logique est arborescente, un ou plusieurs ordinateurs ayant une responsabilité complète et générale sont situés au niveau de la racine de l'arbre. Ces ordinateurs reçoivent des messages en provenance de tous les autres ordinateurs du système. Aux feuilles de l'arbre logique nous retrouvons des composants à responsabilité limitée. Entre ces deux extrêmes se situent des composants avec un degré de responsabilité variable.

Dans les systèmes de processeurs ou stations de travail coopérant, tous les processeurs se trouvent au même niveau logique. Chaque processeur peut solliciter des données ou des processus des autres processeurs. Chaque processeur dispose de certaines données et certains modules de traitement uniques.

L'avènement des miniordinateurs durant les années 60 permit la localisation économique d'ordinateurs auprès des utilisateurs des services informatiques. Ainsi les organisations optant pour cette possibilité étaient décentralisées et on se trouvait en face d'une multitude de petits centres de calcul. Des entreprises se retrouvèrent avec différents types d'ordinateurs de différents constructeurs.

La nécessité de contrôle de ces centres de calcul et la possibilté de leur permettre l'échange de données a conduit vers des réseaux hétérogènes. Aujourd'hui les systèmes départementaux actuels résultent de décisions organisationnelles et doivent être indépendants tout en offrant la possibilté de communication entre les divers départements, y compris le centre de l'entreprise.

4.2.2. Classification par nature intrinsèque

Cette classification est basée sur les quatre critères suivants :

- le degré de couplage entre les composants
- la structure des interconnexions
- l'interdépendance des composants
- la synchronisation entre les composants

Ces critères sont appliqués au modèle de système vu (point 4.1/fig)

4.2.2.1. le degré de couplage entre deux composants

Ce degré de couplage peut être défini comme le rapport entre les données échangées et le volume des données traitées localement. Ainsi on peut déterminer trois types de systèmes :

1. systèmes à couplage faible entre les composants

   Ces systèmes utilisent des canaux de communication qui peuvent atteindre une capacité de quelques Kbits par seconde

2. systèmes à couplage prononcé

   Ici le canal de communication présente une capacité comparable à celle des taux de transfert des unités de mémoire secondaire (systèmes localement distribués)

3. systèmes à couplage fort

   Ces systèmes sont avant tout des systèmes multiprocesseurs et des systèmes virtuellement distribués

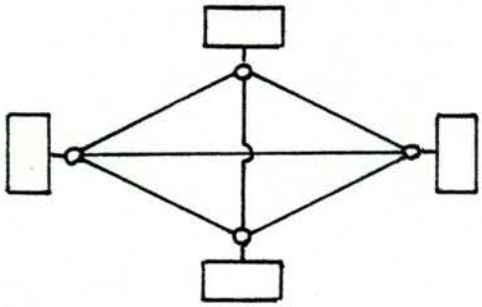
4.2.2.2. la structure d'interconnexion (anglais : shape)

Ce critère est basé sur la structure logique du système, et en particulier sur les stratégies d'adressage et de routage.

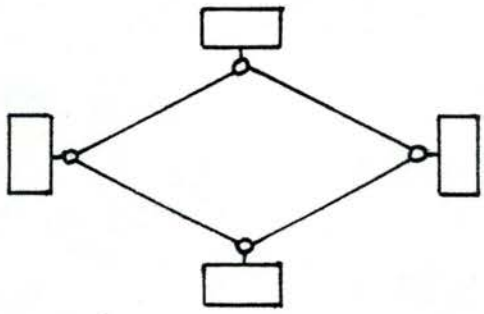
1. structures à interconnexions directes

1.1. Chaque paire de composants communiquant possède un moyen de communication dédié.

exemples : * structure à interconnexion totale (graphe total)

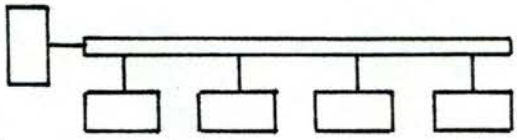


* structures en boucles



1.2. Les moyens de communication sont partagés parmi tous les composants.

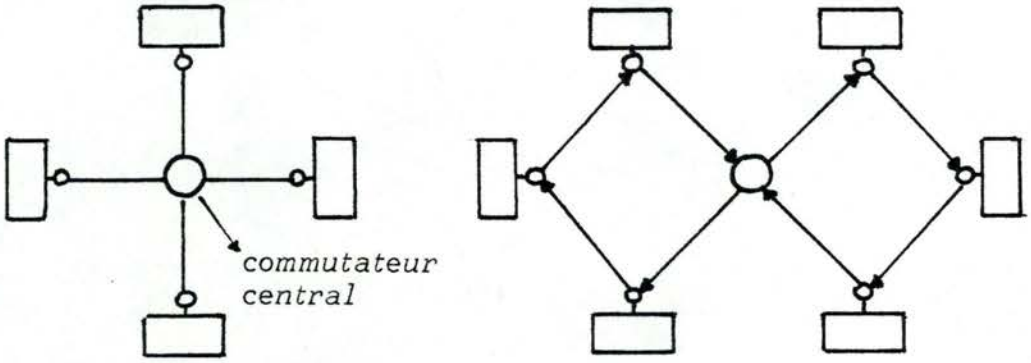
exemple : structure locale en bus (anglais : peer systems)



2. interconnexions indirectes entre les composants

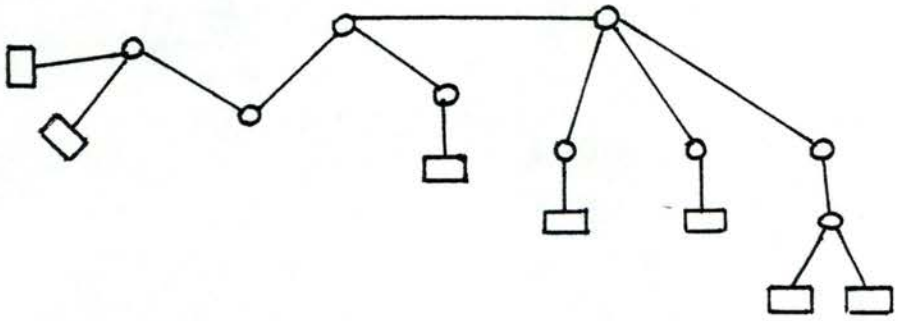
2.1. Routage central des messages

exemples typiques : les réseaux en forme d'étoile avec un commutateur central. Remarquons que chaque branche peut être une boucle.

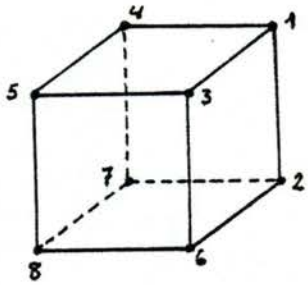


2.2. Routage non centralisé

* il y a les réseaux avec un chemin possible et unique par paire de composants communiquant
  exemple : les réseaux arborescents



* il y a les réseaux où plusieurs chemins de communication sont possibles entre deux composants
  exemple : structure d'interconnexion n-cubique (boolean cube)



$n = 8$

réseaux irréguliers

4.2.2.3. interdépendance des composants

Les composants sont fortement interdépendants si l'opération de l'un d'entre eux dépend de l'opération correcte et réussie de l'autre (des autres).

Les composants sont faiblement interdépendants si la panne de l'un des composants ne met pas en question les opérations des autres. Un tel comportement de systèmes est atteint au moyen de redondances. Ainsi on aboutit à des systèmes hautement disponibles avec des dégradations légères lors de pannes partielles.

4.2.2.4. synchronisation entre composants

La plupart des systèmes d'informatique distribuée sont asynchrones, dans la mesure où chaque composant travaille à son propre rythme. La vitesse du système global est déterminée par la vitesse du composant le plus lent. Dans un tel système, le mécanisme de communication doit fournir les moyens pour corréler les différentes vitesses des divers composants.

D'autres systèmes sont synchrones, dans la mesure où il y a des relations fixes entre les composants. La synchronisation est possible et est maintenue par une horloge unique.

4.3. CARACTERISTIQUES DES SYSTEMES D'INFORMATIQUE DISTRIBUEE

Dans cette partie nous allons rappeler de façon succincte les caractéristiques principales des SID qui sont énumérées par la plupart des auteurs ((L1), (B2), (V2)).

En général, un système d'informatique distribuée

- est constitué d'un ensemble de composants présentant chacun une certaine capacité de traitement et un certain degré d'intelligence.

- présente une image plus ou moins unique, c'est-à-dire l'architecture du système est plus ou moins transparente pour l'utilisateur.

- Les composants sont électroniquement interconnectés selon différentes structures par des lignes téléphoniques, lignes spécialisées ou par voie hertzienne. Ces interconnexions sont plus ou moins flexibles selon la modularité du système.

- Ces composants sont interdépendants et les interactions entre eux sont significatives.

- Lors de la conception d'un SID, l'utilisateur est en présence d'un nombre élevé d'alternatives.

CHAPITRE 5 : STRUCTURES FONDAMENTALES DES SYSTEMES D'INFORMATIQUE DISTRIBUEE

5.1. INTRODUCTION - LES TROIS FONCTIONS FONDAMENTALES

Tout système distribué comporte trois fonctions : traitement, stockage et mouvement des données. Toutefois, chaque système peut avoir une combinaison différente de centralisation ou de distribution de ces fonctions. Par exemple, le traitement et le stockage des données peuvent rester centralisés, tandis que la transmission est distribuée.

Après quelques exemples de combinaisons de ces fonctions, nous verrons dans les autres parties de ce chapitre comment distribuer ces trois fonctions.

Exemples de combinaisons de ces trois fonctions :

- traitement et stockage centralisés, mouvement distribué

  Le système central réalise les traitements, et la base de données qui y est localisée est commune à toute l'entreprise. Les points d'accès sont décentralisés.

  avantages : * contrôles bien définis
  * bonne communication des informations

  inconvénients : * centres éloignés sous la dépendance du siège central
  * coûts de communication élevés
  * besoins régionaux difficilement couverts

- on délègue certaines fonctions de traitement dans les centres régionaux

  avantages : décharger le siège central de certains travaux et permettre ainsi de faire face à une expansion de l'activité

  inconvénients : * problème de contrôle des informations
  * problème de synchronisation des traitements (si deux processus situés sur des sites différents coopèrent)
  * problème de développement de logiciel (certains sites pouvant être autonomes pour le développement de programmes, cela cause des problèmes d'uniformité du logiciel)

- le traitement et le stockage sont effectués sur les centres régionaux. Le siège central assure la coordination mais de nombreuses fonctions sont exécutées localement.

  Les avantages et inconvénients du point précédent sont d'autant plus applicables ici.

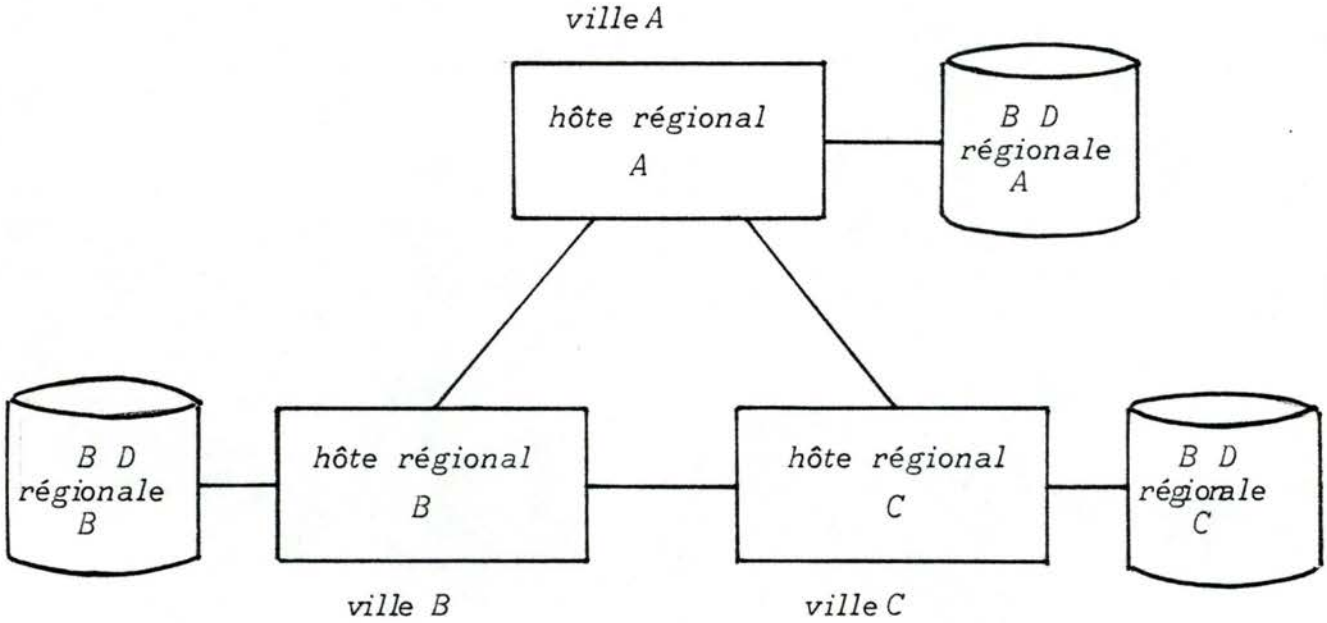
5.2. STRUCTURE DE TRAITEMENT DISTRIBUE

Il y a deux structures de base de traitement distribué : distribution horizontale et hiérarchique. Celles-ci peuvent également être combinées.

5.2.1. Traitement distribué horizontalement

La distribution horizontale du traitement implique l'interconnexion d'au moins deux composants qui sont équivalents d'un point de vue logique. Point de vue logique ne veut pas dire égal physiquement, en capacité, ou en fonctionnalités, mais cela veut dire qu'aucun composant n'exerce un contrôle sur aucun autre.

Exemple : trois sites régionaux



Les ordinateurs hôtes sont interconnectés pour :

- l'échange de données :

  produire des rapports sur l'ensemble du système, nécessitant donc des données de tous les hôtes

- la répartition de la charge de travail :

  distribuer un travail sur un certain nombre de processeurs, afin d'avoir la meilleure utilisation possible des ressources de traitement disponibles

- le partage des ressources :

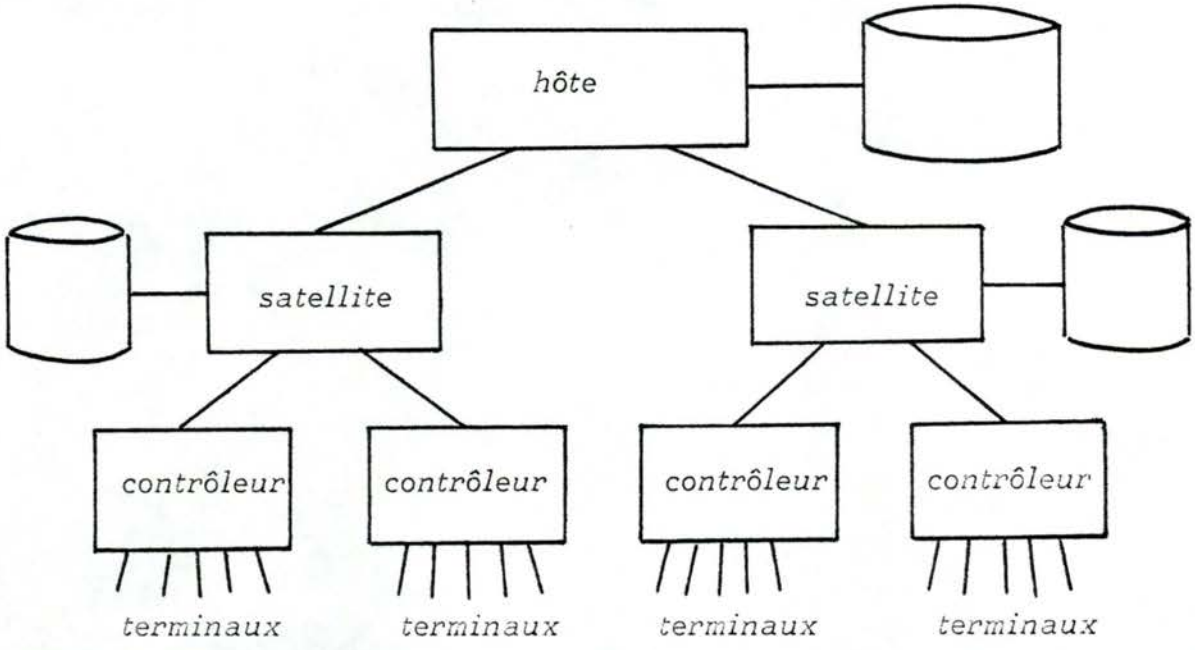
  des ressources non présentes sur tous les hôtes peuvent ainsi être utilisées par les autres (par exemple les bases de données régionales, ou un appareil coûteux comme une imprimante à laser)

Les systèmes horizontaux résultent souvent de l'interconnexion d'ordinateurs jadis indépendants (stand-alone). Les hôtes connectés peuvent être de différents modèles ou contructeurs. Leur capacité de coopérer pour une répartition de la charge de travail peut donc être limitée par leurs incompatibilités.

5.2.2. Traitement distribué hiérarchiquement

Comme son nom l'indique, la structure des relations entre les composants du système distribué forme une hiérarchie. Cela implique une distribution verticale des fonctions de traitement parmi une grande variété de composants : gros ordinateurs, mini ou micro-ordinateurs, ainsi que des terminaux. On remplace une augmentation des capacités de traitement de l'ordinateur hôte par des terminaux intelligents ou des processeurs satellites plus proches des utilisateurs.

Exemple :



Dans cet exemple, la structure hiérarchique a trois niveaux d'intelligence.

Une règle de base pour la conception de ces systèmes est de répartir le traitement dans la hiérarchie de manière à avoir le meilleur rapport coût/performances :

- les fonctions qui sont souvent employées et qui demandent une réponse rapide seront aussi proches que possible de l'utilisateur

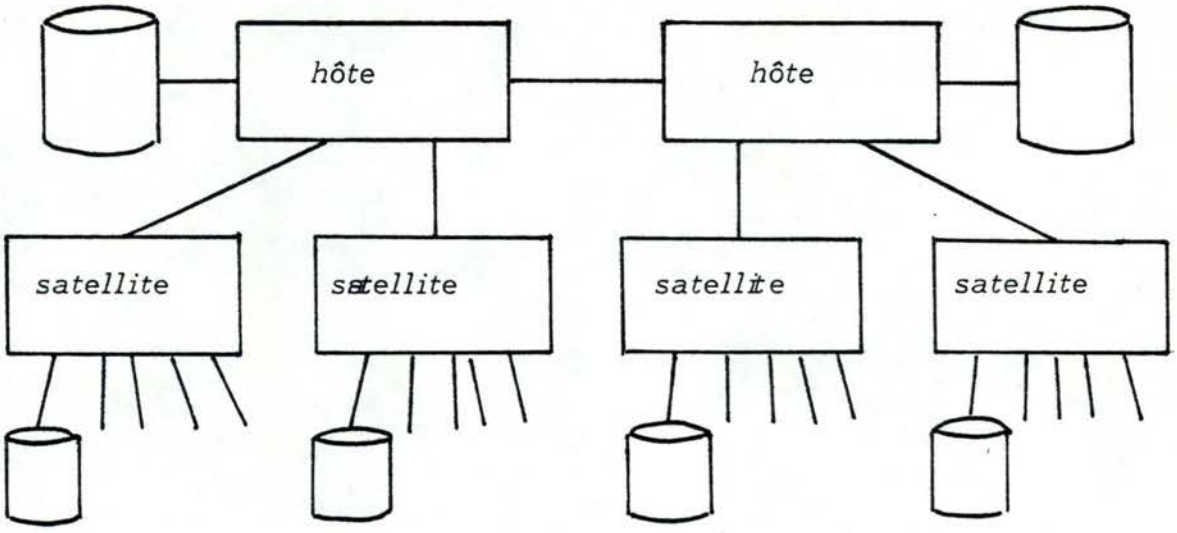
- les fonctions moins souvent demandées et/ou avec une réponse moins rapide seront plus proches du centre

Cette allocation de fonctions dans un système distribué de manière hiérarchique conduit à l'observation que plus on descend dans la hiérarchie, plus les composants sont spécialisés.

5.2.3. Traitement distribué de manière hybride

Certains systèmes distribués complexes combinent les distributions horizontale et hiérarchique des fonctions de traitement.

Exemple :



Qu'un système distribué comporte une ou plusieurs hiérarchies peut dépendre de la dispersion géographique des activités, des applications et/ou de l'organisation de l'entreprise.

Si les activités de l'entreprise sont géographiquement rassemblées, le système distribué comportera généralement une seule hiérarchie. Par contre, des activités géographiquement dispersées peuvent créer des hiérarchies multiples, une par site. L'interconnexion de ces hiérarchies permettra alors l'échange de données, la répartition de la charge de travail et/ou le partage des ressources.

Si l'application supportée par le système distribué est relativement ou entièrement isolée des autres applications de l'entreprise, alors elle sera supportée par une seule hiérarchie. Cependant des applications dépendant d'autres applications informatisées sont mieux servies par des hiérarchies distinctes liées.

Si l'organisation de l'entreprise est centralisée, le système comportera une seule hiérarchie. Si elle a de multiples sous-divisions semi-indépendantes, chacune peut consister en une hiérarchie. Celles-ci seront interconnectées si c'est nécessaire.

Cette flexibilité de la structure de traitement, en opposition avec les structures rigides et centralisées les plus utilisées jusqu'ici, rend le système d'information le plus proche possible de l'utilisateur.

5.3. STRUCTURE DE STOCKAGE DISTRIBUE

La distribution des fonctions de traitement des données entraîne souvent la distribution du stockage de ces données.

Il y a deux structures de base, proposées par (B1), pour la distribution des bases de données : structure partitionnée et répétée.

5.3.1. Bases de données partitionnées

Une base de données conceptuelle est une collection de toutes les données qui intéressent une entreprise. Une base de données partitionnée est créée quand la base de données conceptuelle est divisée en parties non redondantes, et répartie sur plusieurs processeurs d'information.

Une base de données partitionnée consiste donc en des données non redondantes, chaque partition étant stockée à un seul endroit.

Les méthodes selon lesquelles une base de données est partitionnée sont liées à la structure de traitement de l'information. Dans beaucoup de cas, les fonctions de traitement sont situées aussi près que possible de l'utilisateur, ce qui entraîne souvent le mouvement des bases de données qui supportent ces fonctions.

Les bases de données peuvent être partitionnées de plusieurs manières : géographiquement, hiérarchiquement ou selon un but de sécurité et de confidentialité.

5.3.1.1. bases de données partitionnées géographiquement

Cette méthode est utilisée pour les structures de traitement distribué de manière horizontale. A chaque hôte équivalent (dans le sens vu plus haut) est associée une base de données.

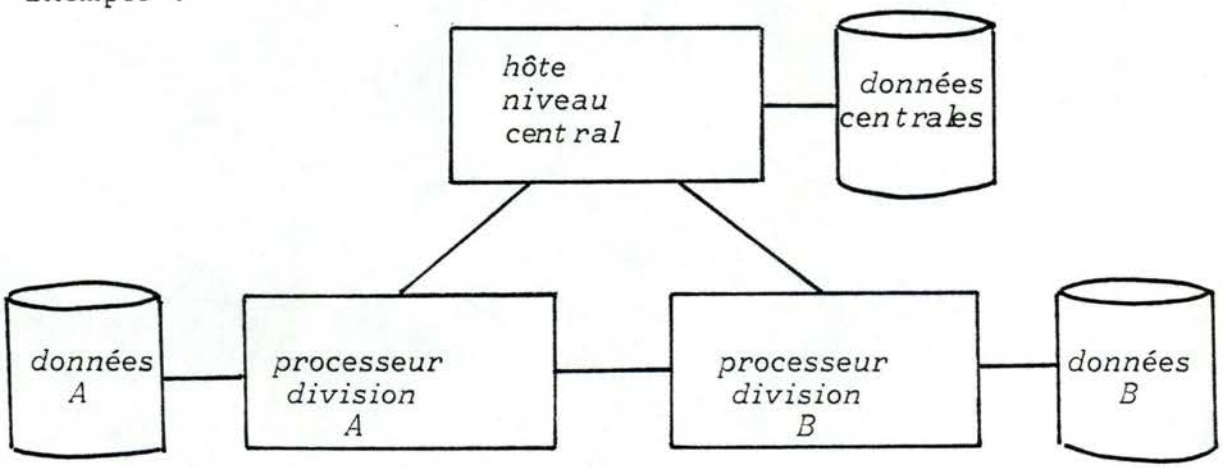
Mais elle n'est adéquate que s'il y a un regroupement géographique des accès à la base de données, la majorité des accès doivent provenir du site même où elle est située. Si ce n'est pas le cas, donc si l'accès à la base de données provient de manière équivalente de tout point géographique, alors cette partition va entraîner un trafic trop important entre les hôtes.

5.3.1.2. bases de données partitionnées hiérarchiquement

Il est également possible de distribuer les données selon la hiérarchie de traitement. Une partie de la base de données est rattachée à l'ordinateur hôte. Chaque satellite a sa propre partition de la base de données. Il n'y a aucune répétition d'information dans ces partitions. Toutes ces bases de données partitionnées rassemblées forment la base de données conceptuelle.

Les relations entre les sites permettent à chaque processeur d'accéder aux données des autres sites, sous réserve de contrôles de sécurité et de confidentialité.

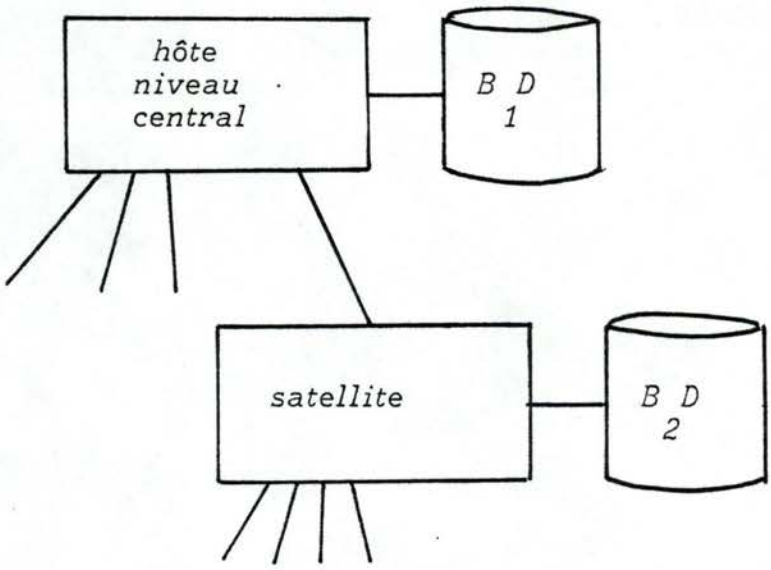
Exemple :



Chaque division agit comme une entreprise indépendante. Elles envoient chacune des données au niveau central, mais il n'y a pas d'échange entre divisions. Les données ne sont échangées que verticalement, et pas horizontalement. Il n'y a aucune raison technique à ce manque d'échanges de manière horizontale, et c'est très concevable qu'il y ait des échanges entre divisions. Mais pour des raisons de sécurité et de confidentialité, chaque division veut protéger ses informations et les partager uniquement avec le niveau central.

5.3.1.3. bases de données partitionnées selon un critère de sécurité et de confidentialité

Exemple :



La base de données est divisée en deux parties, une située à l'ordinateur hôte et l'autre au satellite. Ce système évite des accès des utilisateurs du satellite à la base de données 1, et limite donc les problèmes de protection et de sécurité.

5.3.2. Bases de données répétées

L'autre méthode pour créer des bases de données distribuées est de placer des copies de toute ou une partie de la base de données à plusieurs endroits. De même que pour la méthode de partition, le but de la méthode par répétitions est de rapprocher les données de l'endroit d'où viennent les demandes d'accès. La différence est qu'ici, seul un double est rapproché, et non pas l'original.

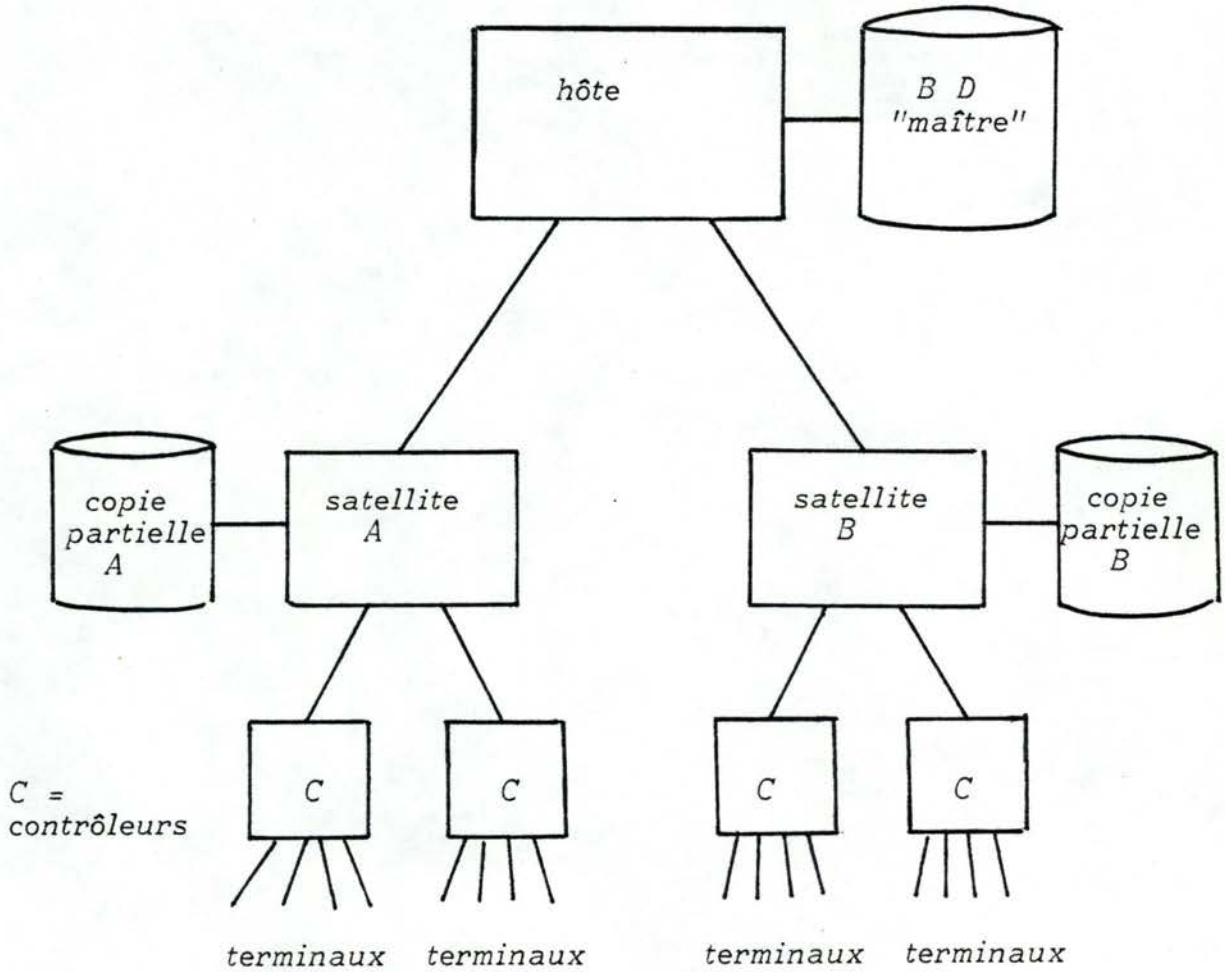
L'avantage des répétitions est qu'elles servent de copie de sécurité. L'inconvénient est que ces copies peuvent toutes se trouver à différentes étapes de mise à jour; il y a un problème de synchronisation des mises à jour.

On peut ainsi accroître la confidentialité des données : seuls les éléments "non confidentiels" sont répétés et accessibles. Les données "confidentielles" sont stockées dans une seule base de données à laquelle l'accès est protégé.

On retrouve les deux méthodes pour former ces bases de données répétées : hiérarchiquement et horizontalement.

5.3.2.1. bases de données répétées hiérarchiquement

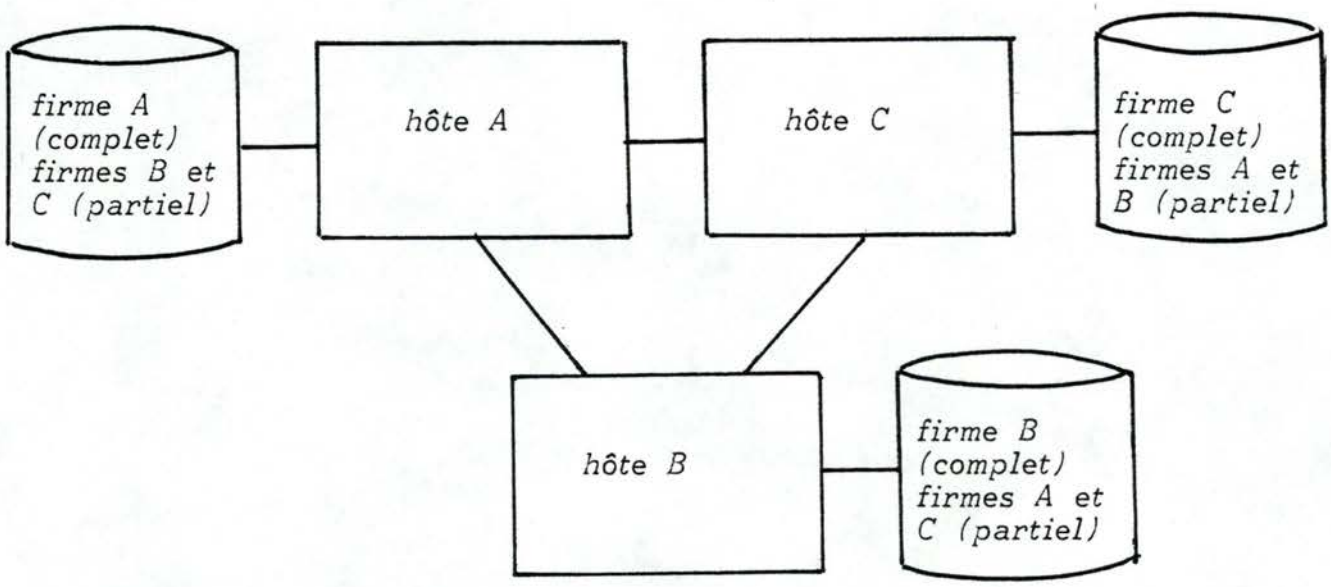
Exemple :



L'ordinateur hôte possède la base de données entière. Chaque satellite dispose d'une copie d'une partie de cette base de données "maître", contenant uniquement les informations pertinentes pour les terminaux qui sont sous le contrôle du satellite. Donc chaque copie est différente des autres.

5.3.2.2. bases de données répétées horizontalement

Exemple :



Chaque hôte conserve les informations qui lui sont propres, concernant la firme à laquelle il est associé. Et il a une copie partielle des bases de données des autres, reprenant les informations qui lui sont nécessaires.

Conclusion

Les bases de données répétées sont plus souvent utilisées dans les systèmes hiérarchiquement distribués. Les structures horizontales emploient plus souvent les bases de données partitionnées, ou bien une base de données séparée, indépendante, à chaque hôte.

5.3.3. Combinaison des bases de données répétées et partitionnées

Ces deux techniques peuvent être utilisées de manière combinée, pour arriver à une structure de base de données distribuée optimale.

Les données statiques (c'est-à-dire qu'on ne modifie pas) sont répétées pour faciliter l'accès de tous les sites. Les données plus actives, pour lesquelles la répétition poserait de graves problèmes de synchronisation, sont partitionnées. Chaque partition est située à l'endroit où l'activité la concernant est la plus grande.

5.4. STRUCTURE DE TRANSMISSION DISTRIBUEE

La structure logique d'un système distribué est déterminée par les relations entre les éléments (ordinateurs hôtes, satellites, terminaux, ...) qui forment le système. La tâche de la fonction de transmission est de fournir un réseau de communication, formé d'éléments matériels et logiciels qui supporteront ces relations logiques.

Les éléments "matériel de transmission" incluent les éléments suivants :

- éléments facilitant la transmission (lignes téléphoniques, ...)
- éléments facilitant le couplage (modems, ...)
- multiplexeurs
- noeuds de réseau (concentrateurs, commutateurs, frontaux)

Les éléments logiciels contrôlent les noeuds de réseau et fournissent des fonctions comme la concentration, la gestion des liens, le routage.

Les éléments matériels et logiciels forment le réseau. Le réseau a trois buts :

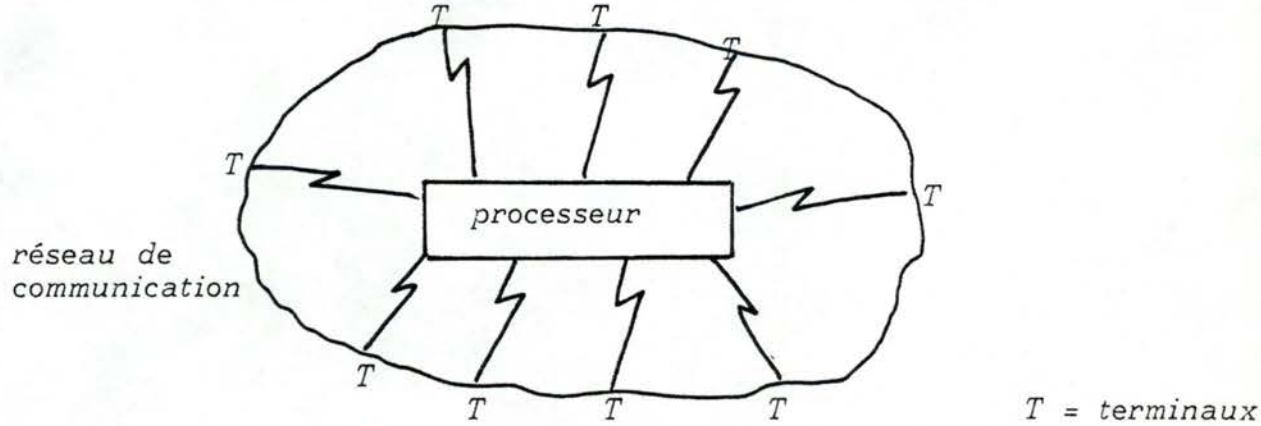
- fournir les chemins de données nécessaires entre les éléments
- supporter la réponse du système dans un certain temps
- minimiser le coût de la transmission. Celui-ci dépend de la vitesse de la ligne. Il faut donc trouver un rapport optimal entre le coût et le service fourni.

Il y a deux grandes catégories de réseaux : réseaux hiérarchiques et réseaux maillés.

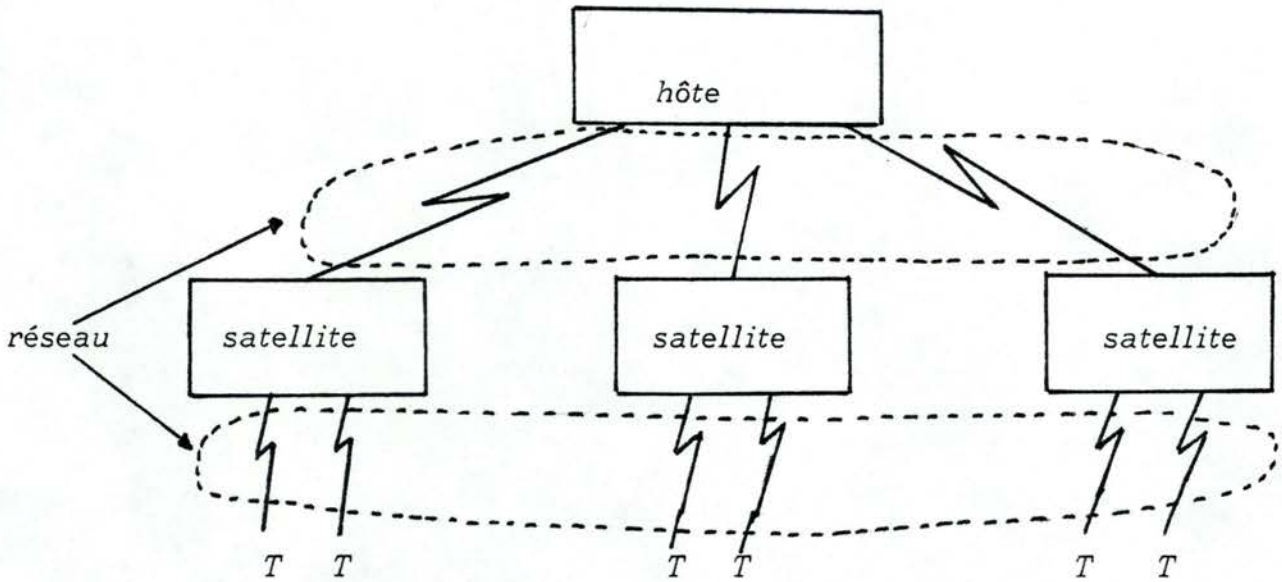
5.4.1. Réseaux hiérarchiques

La structure de réseau la plus commune est la hiérarchie, dans laquelle les facilités de communication forment un arbre.

La forme la plus simple de réseau hiérarchique est le réseau en étoile :



Autre exemple de réseau hiérarchique :



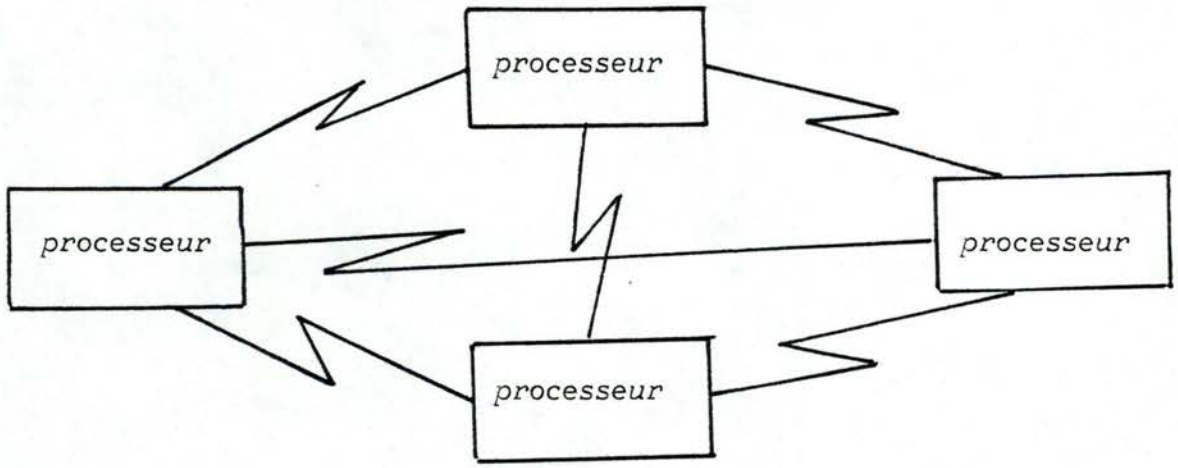
La structure physique du réseau reflète exactement les relations logiques entre les éléments.

Les multiplexeurs ou concentrateurs sont souvent utilisés à cause du coût des liens. Il est plus intéressant d'utiliser un petit nombre de liens à grande vitesse qu'un grand nombre de liens à faible vitesse.

5.4.2. Réseaux maillés (anglais : meshed)

Un réseau maillé permet l'interconnexion directe de tous les éléments (ou du moins tous les éléments intelligents) d'un système distribué.

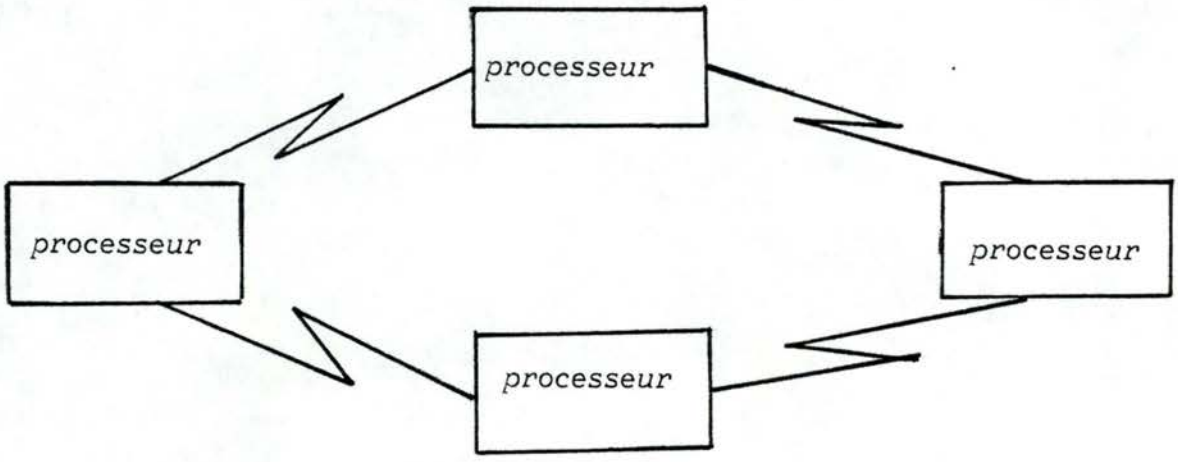
Exemple le plus simple : réseau complètement connecté :



Cette structure, où tous les éléments sont connectés à tous les autres, demande beaucoup trop de liens.

Dans un réseau maillé tel qu'on le conçoit habituellement, les terminaux sont connectés au réseau via l'ordinateur hôte le plus proche. Cela évite ce grand nombre de connexions nécessaires dans un réseau complètement connecté. En général, un terminal est connecté à un ordinateur hôte, et accède aux autres via cet ordinateur hôte.

Exemple : liens minimaux :



Pour éviter que le trafic qui ne ferait que passer par un ordinateur hôte n'encombre celui-ci, les fonctions de traitement de réseau peuvent être implémentées dans des frontaux indépendants.

La structure maillée a un avantage sur la structure hiérarchique : elle fournit des chemins de données redondants. Ainsi la perte d'un lien n'isolera pas nécessairement un composant.

La forme la plus commune de réseau maillé se situe entre un réseau où on a des liens minimaux et un réseau complètement connecté.

Les réseaux par paquets utilisent une de ces deux méthodes d'acheminement : datagram ou circuit virtuel.

Par le service datagram, chaque paquet comporte sa destination et son numéro dans la séquence. Cela autorise une complète liberté pour acheminer chaque paquet. Ils peuvent emprunter chacun une route différente, et seront réunis à la sortie du réseau. Il faut donc ajouter au réseau une logique de réassemblage de paquets.

Le service par circuits virtuels établit une connexion logique entre deux points terminaux et garde cette connexion durant tout l'échange. Cela évite l'arrivée des paquets dans un ordre différent de celui à l'origine, ainsi que l'adressage de chaque paquet. Une connexion logique donnée peut impliquer une connexion physique fixe ou peut autoriser une flexibilité de routage.

CHAPITRE 6 : AVANTAGES ET DESAVANTAGES DES SYSTEMES DISTRIBUES

6.1. INTRODUCTION

Dans ce chapitre, nous exposons de manière simplifiée les avantages et inconvénients des systèmes distribués, présentés par (B1).

6.2. AVANTAGES

- meilleures performances et temps de réponse par rapport au coût
- meilleure disponibilité ou survie du système
- coûts de communication plus faibles
- flexibilité et adaptabilité

Chacun de ces avantages va être discuté un peu plus en détail.

6.2.1. Meilleures performances et temps de réponse par rapport au coût

Cette affirmation contredit la notion d'économie d'échelle, qui a conduit à la centralisation de l'informatique. En fait, pendant longtemps, des systèmes d'ordinateurs de plus en plus grands ont fourni des performances accrues avec une augmentation de coût relativement basse. Il était apparu que, plus grand était l'ordinateur, moins coûteuses étaient les fonctions qu'il exécutait.

A présent, les économies d'échelle ne sont plus si importantes. Le coût des composants devenant de plus en plus bas, il est plus réaliste de distribuer certains composants plutôt que de les centraliser. De plus, un nombre croissant de terminaux sont connectés, ce qui devient de plus en plus difficile à gérer avec un système centralisé.

Afin d'éviter au maximum les délais de communication, certaines fonctions sont rapprochées des utilisateurs, comme nous l'avons vu au chapitre 5. Les mini et micro-ordinateurs peuvent facilement être spécialisés pour certaines fonctions, comme le contrôle des terminaux, le dialogue avec le réseau. Ils fournissent ainsi de meilleurs rapports prix/performances qu'un seul ordinateur hôte de grande échelle qui traiterait ces fonctions.

6.2.2. Meilleure disponibilité ou survie du système

Dans de grands systèmes apparaît la notion de vulnérabilité. La défaillance d'un système centralisé affecte tous les utilisateurs de ce système. Si le nombre d'utilisateurs augmente, une défaillance devient de moins en moins acceptable.

Au contraire, dans un système distribué, la défaillance d'un composant n'affecte généralement qu'un petit nombre d'utilisateurs.

Un autre aspect de la disponibilité est le fait d'éviter les défaillances grâce à l'utilisation d'éléments redondants. Comme chaque composant d'un système distribué est relativement peu cher, ceux qui sont vitaux peuvent être doublés à des fins de secours. Pour plus de détails, nous faisons référence à (M1) chapitre 11.

6.2.3. Coûts de communication décroissants

Dans un système distribué, une partie ou la totalité du traitement est exécuté près du terminal. Cela évite donc certaines communications entre éléments du système.

La structure d'un système distribué fournit une concentration des chemins entre le terminal et l'ordinateur central. Les coûts des liens sont souvent réduits de manière significative par rapport à ceux d'un système centralisé comportant le même nombre d'utilisateurs.

6.2.4. Flexibilité et adaptabilité

La modularité, inhérente aux systèmes distribués, permet une spécialisation fonctionnelle. Le système d'exploitation, la configuration du système et l'utilisation de ces miniordinateurs spécialisés sont plus simples que ceux d'un système centralisé capable de traiter toutes ces fonctions de façon concurrentielle.

Généralement, la modularité augmente la flexibilité. Par exemple, des changements peuvent être introduits à un site sans affecter les autres sites, ce qui ne serait pas le cas dans un système centralisé.

Il est plus facile d'étendre la capacité d'un système distribué en ajoutant de nouveaux modules, que d'étendre celle d'un système centralisé.

6.3. DESAVANTAGES

- complexité structurale du système
- problèmes techniques
- matériel et logiciel multipliés

6.3.1. Complexité structurale du système

Il y a une tendance à croire que, comme les éléments du système (les miniordinateurs par exemple) sont simples et faciles à programmer, le système construit sur ces composants sera également simple. Il y a aussi une tendance à croire que la modularité physique du matériel implique la modularité logique du logiciel et du système. Ces deux croyances ne sont pas nécessairement justifiées.

La complexité résulte du fait que ces composants soi-disant simples sont interconnectés. Tous les éléments d'un système distribué doivent travailler d'une manière coordonnée, ce qui demande une bonne conception du système. Le contrôle d'accès aux bases de données distribuées peut être très compliqué. Ces fonctions de contrôle et de coordination existent aussi dans les systèmes centralisés, mais elles sont simplifiées à cause du fait qu'elles prennent place dans un seul ordinateur, plutôt que parmi de multiples ordinateurs.

Le traitement des défaillances dans un sytème distribué demande de la prudence. Si un composant tombe en panne, cela peut provoquer la défaillance des autres, selon le "domino syndrome" rendu familier par les pannes de courant de ces dernières années aux Etats-Unis.

6.3.2. Problèmes techniques

Il y a toujours des problèmes pour lesquels la recherche technique n'a pas toujours fourni de solutions applicables de manière générale.

Ces problèmes concernent la détection et la correction des erreurs sur tout le système, les méthodes de reprise, la sécurité et la confidentialité, les méthodes de conception de bases de données distribuées, la conception de systèmes, et la prévision des performances.

6.3.3. Multiplication

Les systèmes distribués demandent des matériels et logiciels multipliés, et des opérateurs supplémentaires pour s'en occuper.

PARTIE 2 : SYSTEMES D'INFORMATIQUE DISTRIBUEE ET ARCHITECTURES DE RESEAUX

INTRODUCTION

Le terme architecture désigne un cadre ou une structure fournissant les instructions de construction d'un objet. Une architecture d'un système d'information définit :

1. la modularité

2. les interfaces de communication entre les modules

Les concepts architecturaux sont extrêmement importants dans le contexte des systèmes répartis. Comme ces systèmes sont complexes, une structure logiquement organisée est requise pour ramener cette complexité à des proportions acceptables.

Cette partie explore les concepts architecturaux rencontrés dans les systèmes d'informatique distribuée, le concept fondamental étant le concept des fonctions en couches. Nous explorons d'abord le modèle de référence "Open Systems Interconnexion" de l'ISO, et ensuite nous analysons succinctement quelques architectures de réseaux ou de systèmes répartis. Pour une étude plus détaillée, cfr (M4).

CHAPITRE 1 : NORME ISO

1.1. INTRODUCTION

Les premiers réseaux d'ordinateurs se sont développés de manière expérimentale, tels que les réseaux Arpanet (développé par le département de la défense américain), ou Cyclades (développé par le comité de recherche en informatique). Immédiatement après, chaque constructeur a créé ses propres conventions pour interconnecter ses propres équipements, et donc sa propre architecture.

Bien que ces architectures soient fonctionnellement équivalentes, elles ne permettent pas la construction de systèmes hétérogènes. C'est pourquoi l'ISO (International Standard Organization) a créé un sous-comité, l'OSI (Open Systems Interconnection), ayant comme objectif de concevoir un modèle standard d'architecture de réseau. Le terme "open" a été choisi pour mettre l'accent sur le fait qu'en se conformant à ces standards internationaux, un système sera ouvert à tous les autres systèmes obéissant à ces standards.

Dans ce chapitre, nous exposons les objectifs de l'OSI, et quelques principes et notions de base de ce standard.

1.2. OBJECTIFS DE L'OSI

Le standard OSI n'a pas comme buts de :

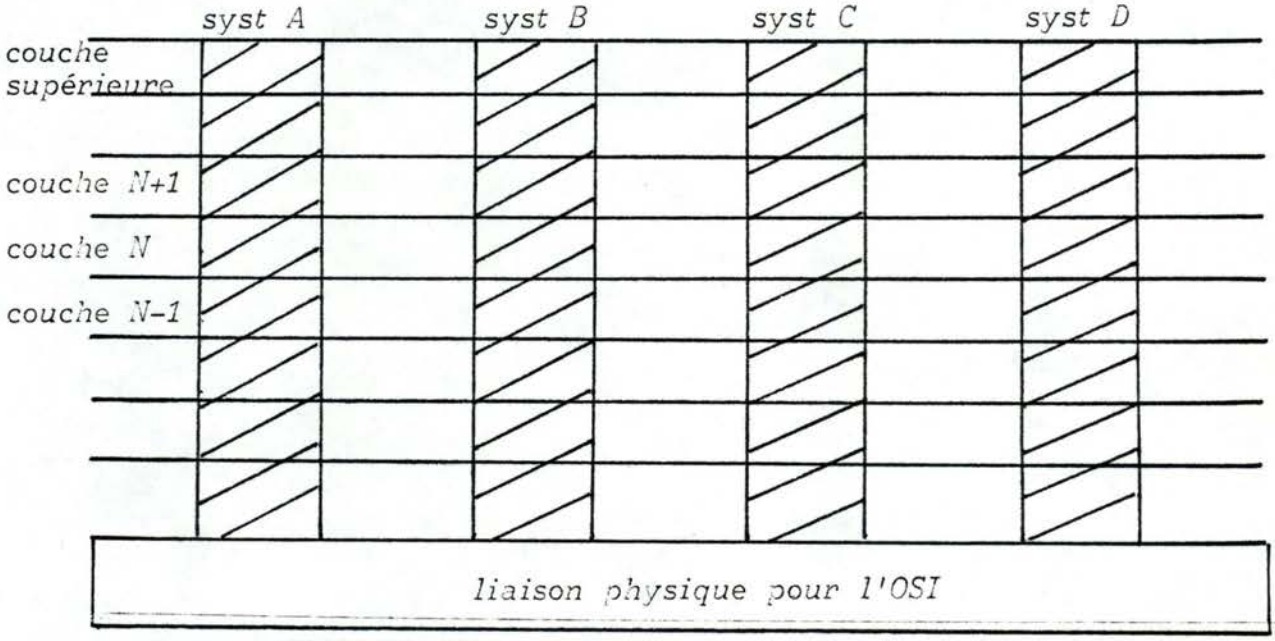
- servir de spécification d'implémentation
- servir de base pour estimer la conformité des implémentations réelles
- fournir un niveau de détail suffisant pour définir précisément les services et protocoles de l'architecture

L'objectif de l'OSI est de standardiser les règles d'interaction entre les systèmes interconnectés. Seul le comportement externe des systèmes ouverts doit être conforme à l'architecture OSI. L'organisation interne et le fonctionnement de chaque système ouvert ne sont pas du ressort des standards de l'OSI, puisqu'ils ne sont pas visibles pour les autres systèmes auxquels il est interconnecté.

1.3. PRINCIPES GENERAUX DE LA STRUCTURE EN COUCHES

La structure en couches est une technique de structuration permettant de voir le système comme étant logiquement composé d'une succession de couches, chacune contenant les précédentes, et les isolant des suivantes. Les couches sont représentées par commodité dans une séquence verticale.

Exemple de structure en couches de quatre systèmes coopérant :



Les couches adjacentes communiquent à travers leur interface commune. Une couche est composée d'une ou plusieurs entités. Les entités d'une même couche sont appelées "paires".

L'idée de base du découpage en couches est que chaque couche ajoute des services à ceux fournis par les couches précédentes. Chacune fournit un service à la couche qui est au-dessus d'elle, en utilisant les services de la couche qui est au-dessous d'elle.

Un autre principe de base est d'assurer l'indépendance des couches en définissant les services fournis par une couche à celle juste supérieure, indépendamment de la manière dont ces services sont fournis.

Une couche demande des services à la couche inférieure en respectant les règles normalisées pour l'échange des données et des signaux. Ces règles déterminent l'interface entre les couches. Cela permet de changer la manière dont une couche opère, pourvu qu'elle offre toujours le même service aux couches supérieures à travers la même interface.

Les services de la Nième couche sont fournis aux entités de la (N+1)ième couche par les Nièmes SAPs (service access points), c'est-à-dire les Nièmes points d'accès. Ces SAPs représentent les interfaces logiques entre les entités de la Nième couche et celles de la (N+1)ième. La fin d'une Nième connexion à un Nième SAP est appelée un Nième point terminal de connexion (CEP = connection end point). Une connexion avec plus de deux CEPs est appelée connexion multipoint. Une connexion entre deux CEPs est appelée connexion point-à-point.

1.4. LES 7 COUCHES DE L'ARCHITECTURE OSI

applications
présentation
session
transport
acheminement
liaison
liaison physique

1.4.1. Couche des applications

C'est la couche supérieure de l'architecture OSI. Elle sert de fenêtre entre les utilisateurs qui communiquent dans l'environnement OSI, et qui, grâce à lui, échangent des informations. L'utilisateur est représenté par une application-entité à ses pairs.

Tous les paramètres à spécifier pour une communication sont fournis à l'environnement OSI (et aux mécanismes implémentant cet environnement) par la couche des applications.

Etant la seule couche dans l'environnement OSI qui fournit des services directement aux utilisateurs, celle-ci fournit nécessairement tous les services qui sont directement compréhensibles par les utilisateurs.

Voici quelques exemples de services qu'elle fournit :

- identification des partenaires dans la communication (nom, adresse, ...)
- détermination de la qualité acceptable du service (temps de réponse, taux d'erreur tolérable)
- sélection de la discipline de dialogue

1.4.2. Couche de présentation

Le but de cette couche est de représenter l'information d'une manière qui préserve le sens tout en résolvant les différences de syntaxe.

Exemples de fonctions :

- transformation des données (conversion de code)
- formattage des données
- sélection de la syntaxe utilisée

1.4.3. Couche de session

Le but de cette couche est de fournir les moyens nécessaires aux entités qui coopèrent (au niveau de la couche de présentation) pour organiser et synchroniser leur dialogue et diriger leur échange de données.

Exemples de fonctions :

- établissement de la connexion (de session), libération de cette connexion

- échange normal de données

- échange de données accéléré (ce service fournit un traitement accéléré d'unités de données. Une restriction est placée sur la taille de ces unités de données)

- synchronisation de la connexion de session (ce service autorise les entités de présentation à marquer et reconnaître des points de synchronisation identifiables, et à reprendre la connexion à un point défini)

1.4.4. Couche de transport

Cette couche fournit un service de transport, en association avec les services fournis par les couches inférieures. Elle fournit un transfert de données transparent entre les entités (de session). Elle décharge les utilisateurs de tout souci concernant la manière détaillée dont les données sont transférées.

Elle optimise l'utilisation des ressources de communication disponibles pour effectuer le transfert à un coût minimal.

L'existence de chaque connexion (de transport) est indépendante de toutes les autres connexions, excepté pour les limitations imposées par les ressources disponibles limitées.

Exemples de fonctions :

- multiplexage des connexions de transport sur les connexions d'acheminement

- établissement et terminaison des connexions de transport

- détection des erreurs de bout en bout et contrôle de la qualité du service (séquencement des unités de données)

- contrôle de flux de bout en bout sur les connexions individuelles

1.4.5. Couche d'acheminement (ou couche de réseau)

Cette couche fournit les moyens d'établir, maintenir et terminer les connexions (d'acheminement) entre les systèmes contenant des entités qui communiquent, ainsi que les moyens fonctionnels et procéduraux pour échanger des unités de données entre deux entités (de transport).

Elle rend les entités (de transport) indépendantes des considérations de routage et de commutation. Elle contient des fonctions qui masquent les différentes caractéristiques des différentes technologies de transmission.

Exemples de fonctions :

- routage et commutation
- multiplexage de connexions d'acheminement
- segmentation et regroupement en blocs, pour faciliter le transfert
- contrôle de flux

1.4.6. Couche de liaison

Cette couche fournit les moyens fonctionnels et procéduraux pour établir, maintenir et relâcher les connexions (de liaison) entre les entités (d'acheminement). Une connexion est construite sur une ou plusieurs connexions physiques.

L'objectif de cette couche est de détecter et corriger si possible les erreurs qui peuvent se produire à la couche de liaison physique.

Exemples de fonctions :

- activation et désactivation de la connexion (de liaison)
- multiplexage de la connexion (de liaison) sur plusieurs connexions physiques
- contrôle de séquence
- détection des erreurs (de transmission, de format)
- contrôle de flux

1.4.7. Couche de liaison physique

Cette couche fournit les moyens mécaniques, électriques, fonctionnels et procéduraux pour activer, maintenir et désactiver les connexions physiques pour la transmission de bits.

Exemples de fonctions :

- activation et désactivation de la connexion physique
- transmission d'unités de données physiques (cette transmission peut être synchrone ou asynchrone)

1.5. PROTOCOLES

Le modèle de l'architecture OSI définit les services fournis par chaque couche à la couche supérieure, et offre des concepts qui peuvent être utilisés pour spécifier comment chaque couche exécute ses fonctions spécifiques. Le fonctionnement détaillé de chaque couche est défini par les protocoles spécifiques à la couche.

1.5.1. Pour la couche de liaison physique

Des standards existent au CCITT, définissant :

- les interfaces avec les moyens physiques
- les protocoles pour établir, contrôler et relâcher les circuits de données commutés

Par exemple : X21, V24, V35, ...

1.5.2. Pour la couche de liaison

Le protocole le plus populaire est HDLC, proposé par l'ISO, et ses protocoles dérivés (BDLC, SDLC, etc ...)

1.5.3. Pour la couche d'acheminement

Une base importante est le niveau 3 de l'interface X25, définie par le CCITT.

1.5.4. Pour la couche de transport

Aucun standard n'existe à présent pour cette couche. La proposition la plus connue est le protocole de transport proposé par IFIP-TC-6, et connu sous le nom de INWG 96.1 (proposal for an internetwork end-to-end transport protocol), qui pourrait servir de base à la définition d'un standard international.

1.5.5. Pour la couche de session

Aucun standard et aucune propostion n'existent. Dans la plupart des réseaux, les fonctions de session sont souvent considérées comme étant une partie des fonctions de la couche supérieure.

1.5.6. Pour la couche de présentation

"Virtual terminal protocols" (VTP) et une partie de "virtual file protocols" sont les protocoles les plus urgents à développer pour cette couche.

Un certain nombre de VTPs sont disponibles, ils sont assez semblables et il serait assez facile de dériver un standard de ces propositions ("IFIP-WG6.1 proposal for a standard virtual terminal protocol", et "data entry virtual terminal protocol for Euronet"). ISO a développé des standards pour "extended control characters for I/O imaging devices", ainsi que "file transfer protocols".

1.5.7. Pour la couche des applications

Peu de choses ont été faites pour cette couche. Il est trop tôt pour donner des indications sur le travail de l'ISO dans ce domaine.

CHAPITRE 2 : CII-HB : DSA

2.1. INTRODUCTION

DSA (distributed systems architecture) est l'architecture de systèmes distribués développée par HIS (Honeywell Information systems) et CII-HB. Le plan directeur dans lequel Honeywell présente les possibilités des systèmes distribués est DSE (distributed systems environment). Les règles architecturales gouvernant l'implémentation de ces possibilités sont connues sous le nom de DSA.

DSE définit un certain nombre de grands principes qui gouvernent le traitement distribué :

- protéger l'environnement
- être adaptable
- rester flexible
- optimiser les performances à un coût raisonnable
- assurer que l'utilisateur garde le contrôle du système
- respecter les normes internationales
- assurer la compatibilité de la ligne des produits

Nous allons exposer les grandes lignes de DSA, celle-ci étant à la base de l'outil automatisé que nous avons développé.

2.2. CONCEPTS DE BASE

- application :

  Une application est un ensemble d'activités

- activité

  Une activité représente un groupe de tâches et les ressources nécessaires pour accomplir ces tâches. Une activité peut être manuelle, automatisée ou partiellement manuelle et partiellement automatisée. Dans DSA, les activités automatisées consistent en des procédures (programmes), les ressources requises pour exécuter ces procédures et les fichiers ou bases de données devant être accédés et peut-être mis à jour par ces procédures.

- processus :

Un processus est une occurence d'une exécution d'une procédure. Une activité automatisée ne peut accomplir son travail que si le processus associé à cette activité est initialisé. Une activité peut avoir un ou plusieurs processus en exécution à tout moment.

- points terminaux :

Les points terminaux sont les frontières entre les activités et le réseau. Chaque point terminal a un nom unique, qui est utilisé quand l'activité veut communiquer avec une autre. Une activité a un ou plusieurs points terminaux. Ce concept de point terminal réalise un des objectifs majeurs de DSA : transparence de la configuration du réseau aux utilisateurs et programmes d'application. Toutes les entités qui peuvent être adressées sont connues par leurs noms de points terminaux logiques. Le logiciel associe le nom logique avec la localisation physique.

- connexions logiques :

Les connexions logiques mettent en relation deux points terminaux qui communiquent. Une connexion logique peut être établie entre deux activités (avec deux points terminaux) localisées au même endroit ou éloignées. Les processus ne doivent pas savoir s'ils sont proches ou éloignés, le logiciel rendant cela transparent.

- chemins :

Les chemins sont les réalisations physiques des connexions logiques.

- site :

Un site est un centre de traitement situé dans un lieu géographique donné.

2.3. LES COUCHES DE DSA

La structure en couches de DSA suit les recommandations de l'ISO pour "l'interconnexion des systèmes ouverts", que nous avons exposées au chapitre 1 de cette partie.

Cette architecture satisfait deux objectifs :

- définir les conventions gouvernant l'échange des données entre deux points terminaux

- dans le cas où les points terminaux se trouvent dans des sites différents, mettre à leur disposition un service de transport

Les couches peuvent être réparties en trois catégories :

- gestion des applications
- gestion des messages
- gestion des communications

2.3.1. Gestion des applications

Cette catégorie contient les processus du système et les applications de l'utilisateur. Les entités de la couche des applications échangent des messages (il est commode de donner un nom aux blocs de données échangés entre les couches).

2.3.2. Gestion des messages

Cette catégorie assure le formattage des messages pour que les processus puissent se comprendre, et le mécanisme de dialogue permettant la synchronisation de leurs actions. Elle est utilisée, que les activités soient locales ou éloignées.

Elle comporte deux couches :

- la couche de présentation

  Cette couche fournit un ensemble de services qui assurent une compréhension mutuelle. Elle rend les communications indépendantes des caractéristiques matériel et logiciel. Les entités de cette couche échangent des segments.

- la couche de session

  Cette couche fournit les services d'un mécanisme de dialogue qui permet à deux processus de coopérer. Elle établit une connexion logique entre les boîtes aux lettres (qui sont les points d'accès des activités au service de transport), surveille et maintient le dialogue sur cette connexion logique. Elle libère la connexion dès que le dialogue est terminé. Les entités de cette couche échangent des lettres.

2.3.3. Gestion des communications

Quand deux points terminaux qui communiquent sont localisés dans des sites différents, la gestion des communications fournit un service de transport entre ces deux sites. Elle masque la nature du moyen de communication sous-jacent.

Elle comporte trois ou quatre couches (selon le protocole de cheminement) :

- la couche de transport

  Cette couche multiplexe les connexions logiques sur le même chemin. Pour chaque connexion logique, elle fournit à la gestion des messages un ensemble de services indépendants du réseau, qui garantit une transmission sans erreur et l'avertit si la transmission est interrompue. Les entités de cette couche échangent des fragments.

- la couche d'acheminement

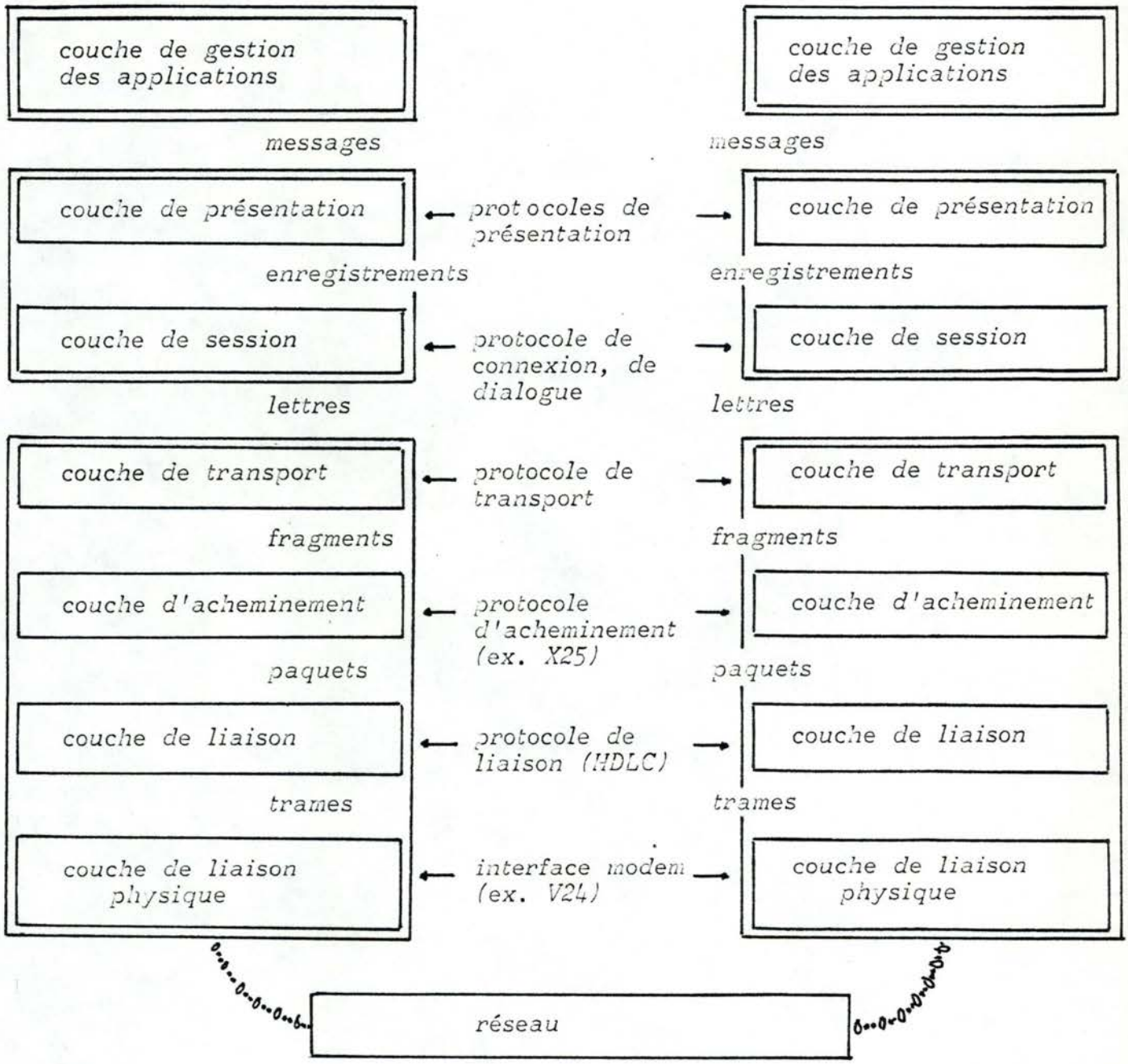
  Cette couche ne sera pas utilisée si l'échange des données se fait par une ligne louée. Elle fournit les services nécessaires pour acheminer les données à la destination correcte. Les entités de cette couche utilisant les techniques de communication par paquet échangent des paquets.

- la couche de liaison

  Cette couche transfère les données entre deux noeuds adjacents sans perte ni erreur. Les entités de cette couche échangent des trames.

- la couche de liaison physique

  Cette couche est la plus basse, et définit l'interface avec le moyen de communication. Elle transporte les données sous forme de séquences de bits.


couche de gestion des applications
couche de gestion des applications
messages
messages
couche de présentation
protocoles de présentation
couche de présentation
enregistrements
enregistrements
couche de session
protocole de connexion, de dialogue
couche de session
lettres
lettres
couche de transport
protocole de transport
couche de transport
fragments
fragments
couche d'acheminement
protocole d'acheminement (ex. X25)
couche d'acheminement
paquets
paquets
couche de liaison
protocole de liaison (HDLC)
couche de liaison
trames
trames
couche de liaison physique
interface modem (ex. V24)
couche de liaison physique
réseau

2.4. LES PROTOCOLES

Comme toutes les structures en couches, DSA supporte des protocoles et interfaces à chaque niveau de l'architecture.

Les protocoles principaux qui sont supportés sont :

- HDLC (high level data link control)
- X25
- X21

2.4.1. HDLC

HDLC a été défini par l'ISO comme protocole orienté bits pour interconnecter un ordinateur et un terminal intelligent, dans un mode point-à-point ou multipoint.

2.4.2. X25

Une des forces de DSA est l'interface qu'elle fournit aux réseaux publics utilisant les définitions d'interfaces X21 ou X25 du CCITT. X25 est l'interface pour la commutation de paquets. On appelle cette technique la technique de "circuit virtuel commuté". Le service de circuit virtuel est offert sur un réseau de commutation de paquets, et assure que les données de l'usager seront remises par le réseau dans l'ordre dans lequel elles ont été reçues par le réseau.

2.4.3. X21

X21 est l'interface pour les réseaux de commutation de circuits. La demande d'établissement d'une connexion établit entre les deux correspondants une liaison physique continue par l'intermédiaire de laquelle ils peuvent échanger des données.

2.4.4. Transfert de fichiers

Ces protocoles sont utilisés pour la transmission de bases de données.

2.4.5. Initialisation de processus à distance

Ces protocoles autorisent de petits systèmes à agir comme stations de travail éloignées pour des ordinateurs de moyenne ou grande échelle.

2.4.6. Entrée de données interactive

Ces protocoles autorisent des données à être entrées via des contrôleurs de terminaux intelligents et traitées de manière interactive sur un processeur hôte ou sur un satellite donné.

2.5. MISE EN OEUVRE DE DSA : COMPOSITION D'UN RESEAU

Un réseau peut comporter :

- des ordinateurs hôtes,

  qui offrent des services aux utilisateurs dans leur réseau satellite local, et aussi aux autres systèmes dans le réseau primaire. Ils peuvent être des CII-HB DPS8, 66/DPS, DPS7, 64/DPS.

- un réseau primaire,

  qui est un ensemble de sites interconnectés dans un réseau DSA.

- un ou plusieurs réseaux secondaires,

  qui sont des ensembles de terminaux et applications sous le contrôle d'un site.

- des systèmes satellites,

  qui offrent des services aux utilisateurs dans leur réseau secondaire, et communiquent avec l'ordinateur hôte et les autres satellites par le réseau primaire.

- des processeurs de réseau,

  qui sont des ordinateurs dédiés à la gestion des communications. Ils peuvent remplir trois rôles : processeur frontal, concentrateur, et commutateur. Ils peuvent être des DN 7100, ou MINI/6.

2.6. ADMINISTRATION DE RESEAU

La mise en oeuvre d'un système distribué nécessite l'utilisation de fonctions administratives qui permettent l'utilisation efficace du système. Les fonctions administratives sont intégrées dans l'architecture.

Les fonctions d'administration comprennent :

- le contrôle de la configuration du réseau
- l'initialisation des éléments du réseau
- la surveillance continue de l'état de ces éléments
- la création de statistiques sur l'état et l'activité de chaque élément du réseau
- le lancement de tests pour vérifier les opérations du réseau

Trois modules fournissent ces services :

- un administrateur de noeud de réseau, qui est présent dans chaque noeud (NAD)
- un ou plusieurs modules administratifs d'archivage réseau (NASF)
- un ou plusieurs modules d'interface opérateur réseau (NOI)

CHAPITRE 3 : AUTRES ARCHITECTURES

3.1. INTRODUCTION

Il est impossible de décrire de manière détaillée toutes les architectures de réseaux utilisées aujourd'hui. Nous allons donner dans ce chapitre un aperçu des architectures principales, c'est-à-dire SNA (d'IBM), DECNET (de DEC), CNA (de NCR), et DCA (de Univac).

Nous n'approfondirons donc pas l'étude de ces architectures ; pour plus de précisions nous faisons référence à (M4).

3.2. SNA : IBM (Systems Network Architecture)

3.2.1. Concepts de base

* réseau d'information

Un réseau d'information est une collection d'unités adressables de réseau (NAU = network-addressable units). Un réseau SNA relie les NAUs les unes aux autres via le sous-système de transmission, et fournit une interface à chaque NAU sous forme d'interface de sous-système d'information (TSI = transmission subsystem interface).

* unité adressable de réseau

Une NAU est une localisation à laquelle des données peuvent être envoyées. La TSI est l'interface qui rend le réseau de communication transparent. La NAU est un concept général, qui est constitué de trois entités logiques : unités physiques, unités logiques, et le point de contrôle des services du système.

* unités physiques

Les unités physiques (PU) sont des ensembles de services qui gèrent les ressources d'un noeud, en réponse aux demandes des points de contrôle des services du système. L'unité physique d'un contrôleur, par exemple, est responsable pour activer les autres noeuds du réseau.

* unités logiques

Les unités logiques (LU) sont les ports par lesquels les applications peuvent communiquer entre elles. C'est l'unité logique qui permet à l'application d'être un point terminal adressable.

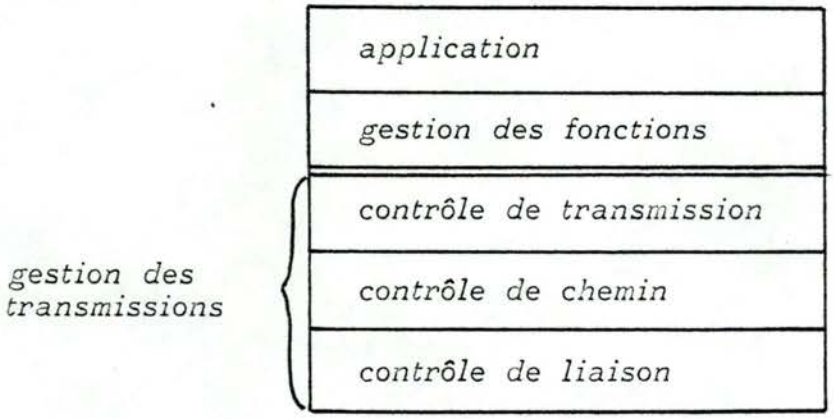
* points de contrôle des services du système

Les points de contrôle des services du système (SSCP = system services control points) fournissent un ensemble de services nécessaires pour les opérations du réseau. Par exemple ils établissent les connexions logiques entre paires de NAUs. Ils sont aussi responsables de certaines fonctions de gestion des erreurs.

3.2.2. La structure en couches

SNA est constitué de trois catégories de couches :

- application
- gestion des fonctions
- gestion des transmissions
  - * contrôle de transmission
  - * contrôle de chemin
  - * contrôle de liaison



- application

Cette couche consiste en programmes d'application demandant des services de traitement de l'information. SNA fait référence à ces sources ou destinations d'information en termes d'utilisateurs terminaux (end-users).

- gestion des fonctions

Cette couche s'occupe du format et de l'adressage des messages (au contraire, les fonctions inférieures ne transmettent plus des messages mais des bits). Elle inclut les facilités de présentation, qui fournissent une indépendance par rapport aux caractéristiques des points terminaux (conversions de code, protocoles, ...).

- gestion des transmissions

La gestion des transmissions consiste en trois couches fonctionnelles :

* contrôle de transmission,

  qui s'occupe de la gestion totale des sessions entre NAUs : l'établissement des sessions, leur terminaison, la gestion des erreurs, le contrôle de flux.

* contrôle de chemin,

  qui analyse l'adresse de destination et sélectionne le chemin correct à travers le réseau. Cela inclut la notion de chemin virtuel, qui est un chemin logique entre deux points terminaux. Ce chemin virtuel peut être implémenté sur plusieurs chemins physiques, ce qui rend le système moins vulnérable aux défaillances de liens. Pour fournir ces services, le protocole SDLC (synchronous data link control) a été développé. SDLC est un protocole orienté bits, qui maintient l'intégrité des unités de données transmises sur des lignes point-à-point ou multipoint. Il est similaire à HDLC en ce qui concerne la structure générale et les possibilités, mais pas identique.

* contrôle de liaison,

  qui gère les liens physiques entre les composants du réseau. Cette couche livre les données au noeud suivant sélectionné par le contrôle de chemin.

3.3. DECNET : DEC (ou DNA : Digital Network Architecture)

3.3.1. Concepts de base

* objets

Les objets sont les ressources du système distribué. Exemples d'objets : fichier, base de données, programme

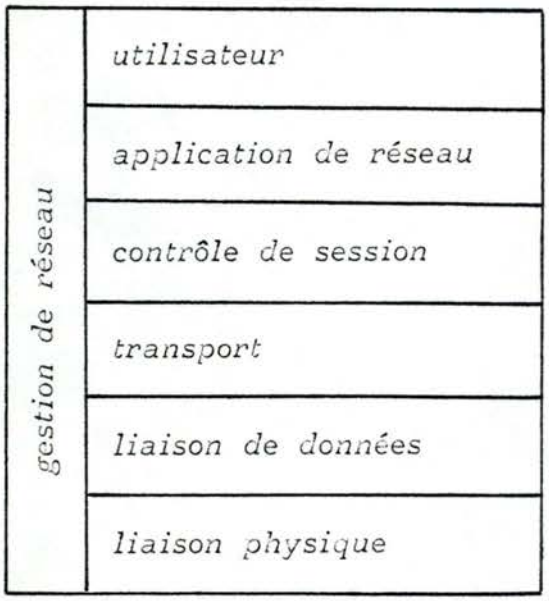
* descripteur de type d'objet global

Le descripteur de type d'objet global fournit les informations nécessaires sur les attributs et la localisation d'un objet. Ces descripteurs forment un catalogue.

* dialogue

Le dialogue est la communication entre deux objets. Le but de base de DECNET est de fournir des moyens de communication indépendants de la localisation entre deux objets.

3.3.2. Structure en couches



* utilisateur

Cette couche inclut toutes les applications et les fonctions fournies à l'utilisateur. C'est dans cette couche que le message est généré, par un programme d'application ou par l'utilisateur terminal.

* gestion de réseau

Cette couche fournit des sous-routines qui définissent les fonctions utilisées par les opérateurs et les programmes pour contrôler et maintenir les opérations du réseau. A la différence des autres couches, celle-ci n'a pas seulement une interface avec les couches adjacentes, mais avec toutes les couches de l'architecture. Les services fournis par cette couche sont fournis par le protocole NICE (network information and control exchange).

* application de réseau

Cette couche fournit des services tels que le transfert de fichiers et l'accès aux fichiers éloignés. Le protocole utilisé est le protocole DAP (data access protocol), qui fournit un format indépendant de la machine pour l'échange de données. DAP inclut un processus de négociation pour déterminer si les deux systèmes échangeant des données utilisent le même format. Si ce n'est pas le cas, un des deux systèmes ou les deux devra transformer les données en un format standard.

* contrôle de session

Cette couche établit et gère les liens logiques (ou dialogues) entre paires d'objets, permettant à un programme ou à un utilisateur dans un noeud de communiquer avec un programme ou un utilisateur dans un autre noeud. Le protocole utilisé est le protocole NSP (network services protocol), qui garantit la livraison du message d'un point terminal à l'autre, et le bon séquencement des messages à l'arrivée. Il fournit aussi le contrôle de flux.

* transport

Cette couche traite l'adressage du message. Elle tire de sa table des paramètres l'adresse et l'information de routage nécessaires pour acheminer le message par le réseau au noeud de destination.

* liaison de données

Cette couche assure que le message est transmis sans erreur au noeud adjacent. A la différence des autres constructeurs, DEC n'a pas choisi un protocole orienté bits, mais a développé un protocole orienté caractères, le DDCMP (digital data communication message protocol).

* liaison physique

Cette couche détermine le mode d'interconnexion physique. DECNET supporte des interfaces tels que V24, RS232, etc... Cette couche interprète les signaux électriques qui permettent aux appareils de communication d'être connectés.

En plus, il y a un module X25, qui fournit une interface aux réseaux publics conformes à la définition X25 du CCITT. Ce module X25 inclut le protocole HDLC, et est accédé directement par la couche de transport.

3.4. CNA : NCR (Communication Network Architecture)

Structure en couches

correspondant
méthode d'accès
services de communication
gestion du routage
contrôle de liaison

* correspondant

Cette couche est utilisée pour décrire les applications, les entités qui peuvent communiquer via CNA.

* méthode d'accès

La couche TAM (telecommunication access method) est une facilité sur l'ordinateur hôte, qui fournit des interfaces aux applications.

* services de communication

La couche CSS (communications system services) fournit la fragmentation des messages en paquets à la localisation qui envoie, et le réassemblage à celle qui reçoit. Le protocole qui autorise cette communication est un protocole de bout en bout.

* gestion de routage

Cette couche détermine quel chemin utiliser pour acheminer les paquets par le réseau. Les liaisons peuvent être point-à-point, multipoint, ainsi que par un réseau X25.

* contrôle de liaison

Cette couche utilise le protocole de liaison pour contrôler le transfert de données. Ce protocole est orienté bits et est basé sur X25.

3.5. DCA : UNIVAC (Distributed Communications Architecture)

3.5.1. Structure

Pour DCA, un système distribué consiste en ces entités logiques :

- un réseau de transport
- système de terminaison
- système de communications
- utilisateurs du système de communications
- utilisateurs terminaux

* réseau de transport

Le réseau de transport (TN) inclut toutes les facilités de communication : liens privés, liens commutés, réseaux publics.

* système de terminaison

Le système de terminaison (TS) consiste en facilités de logiciel qui fournissent des fonctions de gestion de réseau telles que la fragmentation des messages, le réassemblage, le contrôle de flux.

* système de communications

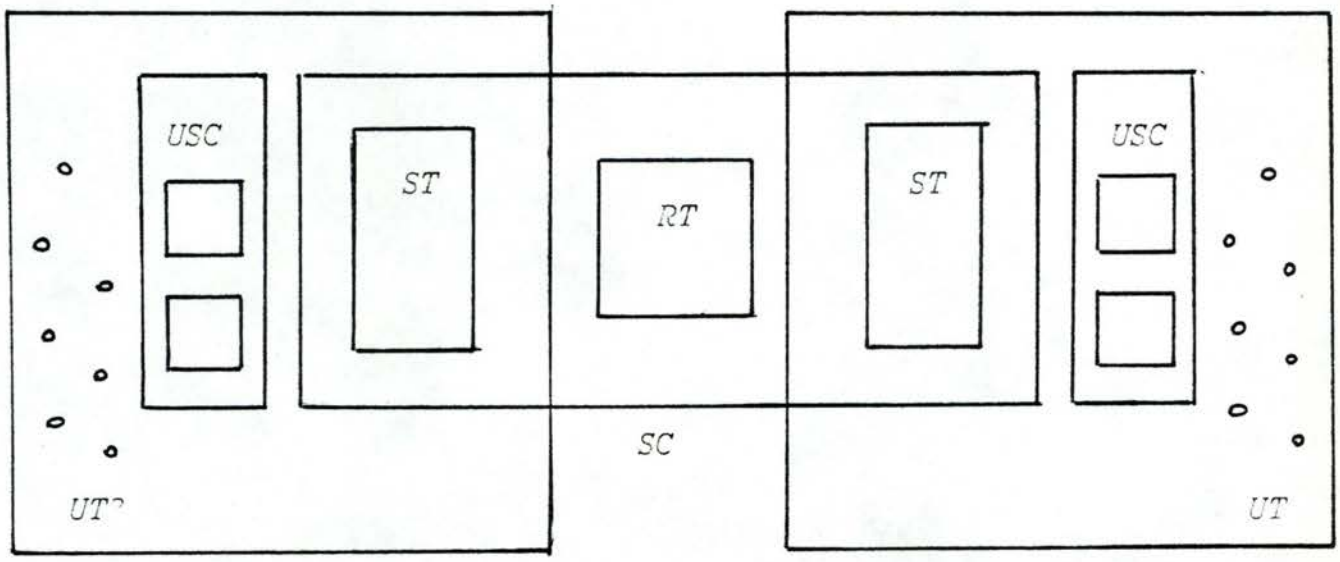
Le système de communications (CS) entoure le réseau de transport et le système de terminaison.

* utilisateurs du système de communications

Les utilisateurs du système de terminaison (CSU) sont des progiciels qui fournissent des services aux utilisateurs terminaux. Ils font l'interface entre les utilisateurs terminaux et le système de communications, et aussi avec les services de traitement de l'ordinateur hôte.

* utilisateurs terminaux

Les utilisateurs terminaux incluent les utilisateurs aux terminaux et les programmes d'application.



localisation A *localisation B*

RT = réseau de transport
ST = système de terminaison
SC = système de communications
USC = utilisateurs du système de communications
UT = utilisateurs terminaux

3.5.2. Protocoles

DCA inclut un protocole orienté bits, appelé UDLC (universal data link control). UDLC est décrit comme incluant toutes les variations de HDLC (ISO), ADCCP (ANSI), SDLC (IBM). Il autorise aussi une extension pour traiter des conditions spéciales dûes au site ou à l'application. Par exemple, il peut être utilisé pour réaliser l'interconnexion entre un réseau DCA et un réseau utilisant un autre protocole.

PARTIE 3 : METHODES DE CONCEPTION DE SYSTEMES D'INFORMATIQUE DISTRIBUEE

INTRODUCTION

Les premiers systèmes d'informatique distribuée ont été conçus et mis en place il y a une bonne quinzaine d'années. Ces réseaux, réalisés au fur et à mesure des besoins de l'utilisateur et sans vue globale, ont vite fait apparaître des carences, tant sur le plan des performances que sur celui de la facilité d'exploitation et d'extension.

Nous avons analysé un certain nombre de lignes de conduite distinctes à suivre lors de la conception d'un système réparti :

- La première ligne de conduite se base sur une formalisation du processus de conception. Ce processus de conception, qui procède par niveaux successifs de décisions, permet d'aboutir à une taxonomie des systèmes d'ordinateurs interconnectés.

- La deuxième méthode propose les étapes fondamentales à franchir lors de la conception totale d'un système. Cette méthode permet déjà de dégager les éléments importants à prendre en compte.

- La troisième méthode est constituée d'un processus itératif où un certain nombre d'étapes, relatives à un problème de conception précis, sont à franchir. Cette méthode est en principe utile lors de la conception de n'importe quel type de système d'informatique distribuée.

- La quatrième méthode est la méthode la plus complète et la plus détaillée. Elle reprend tous les éléments économiques, organisationnels et informatiques qui peuvent intervenir lors de la conception de systèmes répartis. Cependant l'application d'une telle méthode ne se justifie pas pour un petit réseau. Vu son caractère général, elle peut être adaptée aux besoins d'un concepteur.

CHAPITRE 1 : DECISIONS DE CONCEPTION. UNE TAXONOMIE POUR LES SYSTEMES D'INFORMATIQUE DISTRIBUEE

1.1. INTRODUCTION

Actuellement, la conception de systèmes informatiques répartis constitue une des activités les plus importantes en informatique. Une absence presque totale de publications relatives à la description de méthodes de conception et à la comparaison des résultats obtenus par ces méthodes alourdit cette tâche complexe. Ce phénomène peut être expliqué en partie par l'absence d'une nomenclature unique pour identifier les systèmes.

Nous allons présenter succinctement les travaux menés par Anderson et Jensen, qui ont pour but de proposer une taxonomie pour identifier divers systèmes d'ordinateurs interconnectés, facilitant ainsi le processus de conception.

1.2. POINTS DE DEPART

La taxonomie proposée par Anderson et Jensen repose sur deux restrictions principales :

- la première concerne la nature des équipements matériels pris en compte : on se limite à interconnecter des unités matérielles dans lesquelles peuvent s'exécuter des processus ; ces unités sont les éléments de traitement (processing elements PE).

- la deuxième restriction concerne la structure d'interconnexion du système : on ne s'intéresse qu'aux systèmes dans lesquels n'importe quelle unité peut communiquer avec n'importe quelle autre unité via le mécanisme d'interconnexion.

Tout d'abord, il faut distinguer les éléments de traitement qui participent dans le transfert d'information. Ensuite, il faut distinguer deux unités fonctionnelles dans la structure d'interconnexion :

- les chemins

- les éléments de commutation

Un chemin est le moyen par lequel un message est transféré entre les éléments de traitement et de commutation. Exemples de chemins : lignes téléphoniques, voies hertziennes, ...

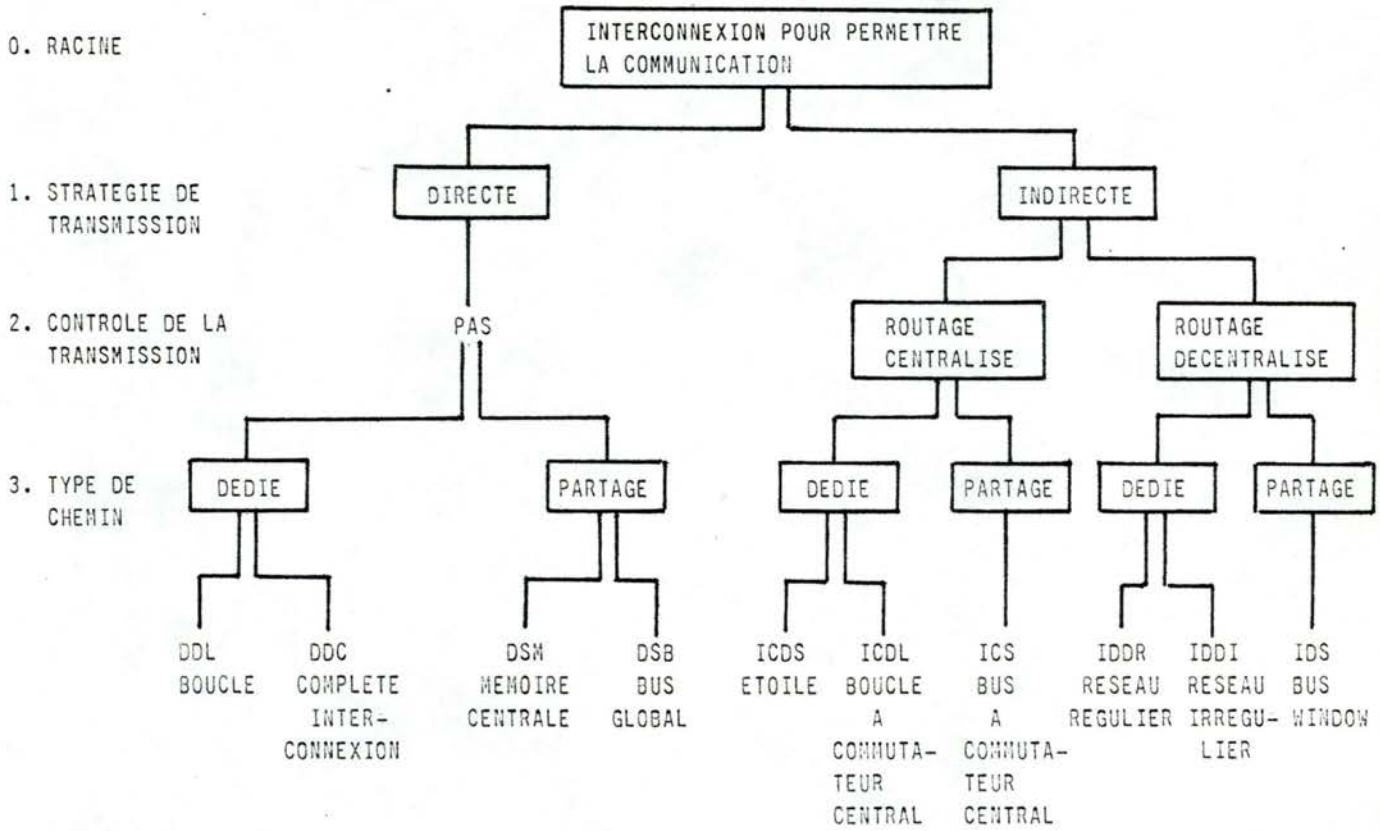
Un élément de commutation est une entité qui peut être représentée par une "intelligence intervenant" entre l'émetteur et le récepteur d'un message.

Les notions de message, chemin et commutateur sont à la base de la taxonomie proposée.

1.3. LES DECISIONS DE CONCEPTION, LA TAXONOMIE

Selon Anderson et Jensen, un système interconnecté résulte d'un certain nombre de décisions de conception. L'espace de décision peut être représenté par un arbre.

Le modèle arborescent pour la procédure de conception présente quatre niveaux de décisions.



La racine de l'arbre est constituée par la décision d'interconnecter complètement un certain nombre d'ordinateurs.

En-dessous de la racine, on retrouve les niveaux de décision relatifs au choix de la stratégie de transfert des messages, à la méthode de contrôle des transferts, et au choix du type de chemin utilisé lors du transfert.

Les deux premiers niveaux concernent la stratégie, tandis que le troisième et le quatrième niveau concernent la tactique de la conception (implémentation).

- La première décision stratégique doit être faite entre la transmission directe de messages et la transmission indirecte de messages. Dans cette dernière solution, une opération intermédiaire de commutation est requise.

- Si on choisit une communication indirecte, on doit prendre une décision concernant la méthode de commutation. Cette décision se situe au deuxième niveau de l'arborescence. Les alternatives sont la centralisation, où une seule unité commute tous les messages, et la décentralisation, où plusieurs commutateurs interviennent.

- Au troisième niveau, le concepteur décide de la nature du chemin, c'est-à-dire s'il est dédié ou partagé. Par chemin partagé, on entend un chemin qui est accessible à partir de plus de deux points. En réalité, il y a trois possibilités à distinguer : des chemins unidirectionnels point-à-point, des chemins bidirectionnels point-à-point, et des chemins bidirectionnels point-à-point qui relient plus de deux points. Les deux premiers types sont considérés comme dédiés, parce qu'il y a peu ou pas de contention, le troisième type est considéré comme partagé.

- Au quatrième niveau de l'arbre taxonomique se trouvent les feuilles représentant les systèmes spécifiques :

  * DDL (Direct transmissions, Dedicated path, Loop)

    réalisation : architecture sous forme de boucle

  * DDC (Direct transmissions, Dedicated path, Complete interconnection)

    réalisation : architecture où les éléments de commutation et de traitement sont complètement interconnectés

* DSM (Direct transmissions, Shared path, Memory)

  réalisation : architecture multiprocesseurs, où les processeurs communiquent à travers une mémoire commune

* DSB (Direct transmissions, Shared path, Bus)

  réalisation : bus global

* ICDS (Indirect transmissions, Centralized routing, Dedicated path, Star)

  réalisation : architecture "classique" sous forme d'étoile

* ICDL (Indirect transmissions, Centralized routing, Dedicated path, Loop)

  réalisation : architecture sous forme de boucle à commutateur central

* ICS (Indirect transmissions, Centralized routing, Switch)
  = ICDS sauf que les processeurs sont connectés au commutateur via le bus

  réalisation : architecture en bus à commutateur central

* IDDR (Indirect transmissions, Decentralized routing, Dedicated path, Regular network)

  réalisation : réseau régulier

* IDDI (Indirect transmissions, Decentralized routing, Dedicated path, Irregular network)

  réalisation : réseau irrégulier

* IDS (Indirect transmissions, Decentralized routing, Shared path)

  réalisation : architecture appelée "bus window" : dans ce type d'architecture, l'accès aux commutateurs se fait via un chemin partagé par plusieurs éléments de traitement. La commutation est réalisée par plusieurs commutateurs et les messages peuvent être retransmis sur le même chemin ou sur un autre.

Les lecteurs intéressés trouveront plus de détails dans (A1).

1.4. APPRECIATIONS

La formalisation des processus de décision intervenant dans la conception de réseaux proposée par Anderson et Jensen, permettant d'aboutir à une nomenclature des systèmes informatiques répartis, présente avant tout un intérêt théorique. En effet, la restriction relative à l'interconnexion totale des éléments de traitement entre eux a sérieusement limité l'utilisation de cette méthode, la majorité des réseaux n'offrant pas cette possibilité.

Cependant les travaux de Anderson et Jensen ont dégagé un outil de conception de réseau intéressant. En effet, cette nomenclature pourrait aider les concepteurs dans l'accomplissement de leur tâche, dans la mesure où une nomenclature universellement acceptée permettrait des comparaisons et évaluations des différentes possibilités qui se présentent. Une telle nomenclature constituerait également une bonne base pour le développement d'outils automatisés d'aide à la conception de réseaux.

CHAPITRE 2 : METHODE DE CONCEPTION PROPOSEE PAR BOOTH

2.1. INTRODUCTION

Dans ce chapitre, nous allons présenter une méthode de conception totale d'un système distribué, proposée par Grayce Booth (B1). Cette méthode ne sera pas approfondie ici, nous en donnons les grandes lignes.

Cette méthode, par un certain nombre d'étapes à franchir, met en évidence les éléments à prendre en considération lors de la conception d'un système distribué.

2.2. ETAPES PRINCIPALES

Le processus de conception d'un système distribué est itératif, et consiste en les étapes suivantes :

2.2.1. formuler la conception "d'essai".

2.2.2. analyser cette conception, de manière à déterminer si le coût est acceptable et si les caractéristiques fonctionnelles, de performances et de disponibilité sont adéquates.

2.2.3. si tous les paramètres ne sont pas satisfaisants, modifier la conception d'essai et répéter l'étape 2.

2.2.4. si une conception d'essai ne peut pas être modifiée de manière satisfaisante, retourner à l'étape 1 et essayer une conception d'essai différente.

Cette itération se fait jusqu'à ce qu'une conception satisfaisante soit trouvée.

2.2.1. Conception d'essai

La formulation d'une conception d'essai consiste en les étapes suivantes :

1. sélectionner une structure de traitement de l'information (2.2.1.1.)

2. sélectionner une structure de base de données (2.2.1.2.)

3. sélectionner les types de composants utilisés pour le traitement et le stockage des données (2.2.1.3.)

4. allouer les fonctions de traitement de l'information aux composants (2.2.1.4.)

5. allouer les fonctions de stockage de l'information aux composants (2.2.1.5.)

6. sélectionner une structure et les composants pour interconnecter le réseau (2.2.1.6.)

Il peut être nécessaire de répéter une ou plusieurs de ces étapes avant de procéder à l'évaluation de la conception. Bien que chaque étape soit présentée comme étant indépendante, elles sont complètement interreliées.

2.2.1.1. sélectionner une structure de traitement

Choisir une structure hiérarchique, horizontale, ou hybride. Ce problème a été discuté dans la partie 1 / chapitre 5.

2.2.1.2. sélectionner une structure de base de données

Déterminer si la base de données est distribuée, et de quelle manière. Ce problème a été discuté dans la partie 1 / chapitre 5.

2.2.1.3. sélectionner les composants de traitement de l'information

Si un système entièrement nouveau doit être conçu, la meilleure approche est de faire un appel d'offres. Chaque vendeur proposera un ensemble d'équipements optimal pour une ligne de produits particulière.

Mais souvent, le nouveau système distribué est une extension, une évolution d'un ou plusieurs systèmes existants. Ceci laisse au concepteur un certain nombre de choix, moins importants toutefois que si tout le système était à concevoir.

2.2.1.4. allouer les fonctions de traitement aux composants

Cette étape, ainsi que la précédente et la suivante (où les fonctions de stockage sont allouées aux composants) ne peuvent être prises séparément.

Pour cela, il faudra choisir entre l'allocation et le partage des ressources statiques ou dynamiques.

Dans le mode statique, à chaque composant est assigné un ensemble de fonctions, qui peut changer dynamiquement quand le système se dégrade, mais qui ne peut pas le faire pour des raisons telles qu'un déséquilibre de la charge.

Cette allocation des fonctions repose sur des critères plus techniques en ce qui concerne la faisabilité technique d'un choix de conception, et sur des critères organisationnels et économiques.

2.2.1.5. allouer les fonctions de stockage aux composants

Si la base de données est distribuée, les parties ou copies de cette base de données doivent être allouées aux composants.

2.2.1.6. sélectionner une structure de réseau

Il faut sélectionner la structure pour la communication de données. Une possibilité est d'interconnecter les localisations par un réseau public. Dans ce cas, la structure ne devra pas être conçue. Si un réseau public n'est pas approprié (pour des raisons de coût ou de confidentialité par exemple), une structure de réseau devra être sélectionnée.

Les catégories principales de réseaux sont le réseau en étoile, hiérarchique et maillé. Ce problème a été discuté dans la partie 1 / chapitre 5.

2.2.2. Evaluation de la conception d'essai

Une conception d'essai consiste en un ensemble d'éléments matériel et logiciel, interconnectés par une structure de réseau proposée, où chaque composant exécute un ensemble de fonctions défini.

L'évaluation de cette conception consiste en les étapes suivantes :

1. analyse de coût (2.2.2.1.)

2. temps de réponse, fiabilité, flexibilité (2.2.2.2.)

3. réévaluations (2.2.2.3.)

2.2.2.1. analyse de coût

* coûts du réseau

évaluer le coût des facilités de communication

* coût du matériel et du logiciel pour le traitement et le stockage

évaluer les coûts initiaux (d'acquisition) et les coût récurrents (maintenance et coûts de location ou leasing)

* coût du personnel

* coûts divers

coûts n'entrant pas dans les trois premières catégories. Ils incluent les coûts des fournitures (papier, rubans, disques,...), et de préparation éventuelle des sites (climatisation, ...)

2.2.2.2. temps de réponse, fiabilité, flexibilité

* temps de réponse

temps séparant la fin de l'envoi d'une transaction au point où elle sera traitée, et le début de la réception de la réponse.

* fiabilité

déterminer si le système apparaît comme étant fiable aux utilisateurs, et les effets des défaillances. Les effets de chaque type de défaillance seront analysés (défaillance de la ligne, du terminal, du contrôleur, ...). On déterminera également la probabilité que chaque type de défaillance arrive, les possibilités de minimisation des effets de chaque type de défaillance, ainsi que le coût de minimisation.

* flexibilité

elle est evaluée en termes de séries de questions : "que se passe-t-il si "
En voici quelques exemples :
- que se passe-t-il si le volume des transactions
  . diminue de 20 %
  . augmente de 30 %
- que se passe-t-il si les profils des transactions changent
- que se passe-t-il si les profils des utilisateurs changent
  . s'ils changent de localisation
  . si leur profil d'initialisation des transactions change
  . s'ils sont moins expérimentés et utilisent fréqemment les systèmes d'aide

- que se passe-t-il si le nombre d'utilisateurs
  - . augmente de 50 % avec une augmentation correspondante du volume des transactions
  - . diminue de 20 % avec le même volume des transactions

2.2.3. Réévaluations

A ce point, il est rare qu'une conception d'essai d'un système complexe reste inchangée. Une série de réévaluations doit souvent être faite pour arriver à une conception satisfaisante.

Critères pour qu'une conception soit satisfaisante :

- fournit-elle les fonctions requises ?
- traite-t-elle le volume requis ?
- rencontre-t-elle les exigences de temps de réponse ?
- fournit-elle le niveau minimal requis de fiabilité ?
- le coût du système est-il dans les limites ?

Si les réponses à ces cinq questions sont positives, la conception peut être acceptée, mais cela ne veut pas dire qu'elle est la meilleure.

Si les réponses ne sont pas toutes positives, on passe à l'étape 3 en essayant de remplir les critères qui ne l'étaient pas.

CHAPITRE 3 : METHODE DE CONCEPTION PROPOSEE PAR MARTIN

3.1. INTRODUCTION

Dans ce chapitre, nous allons proposer une méthode de conception de système réparti, proposée par James Martin (M3). Le processus pragmatique de conception repose sur un modèle qui, comme tous les modèles, n'est pas toujours exact.

Un certain nombre d'hypothèses doivent être posées pour simplifier le modèle, car tous les faits ne sont pas connus avec précision.

Cependant les modèles permettent de situer la problématique et de construire des réseaux compatibles avec les critères de performances et de coût minimal de l'utilisateur.

3.2. DESCRIPTION DE LA METHODE

Une longue séquence d'étapes est nécessaire pour la conception d'un système de transmission de données. La plupart du temps, la procédure de conception est un processus itératif.

Tout d'abord, une estimation d'un système possible de transmission de données sera faite. Des calculs seront faits pour voir s'il remplit les exigences. Si ce n'est pas le cas, le système sera ajusté jusqu'à ce qu'il remplisse ces exigences. Si c'est le cas, on esaie de minimiser le coût du réseau. Une fois qu'une forme particulière de réseau a été sélectionnée, divers algorithmes sont disponibles pour établir une configuration optimale de ce réseau.

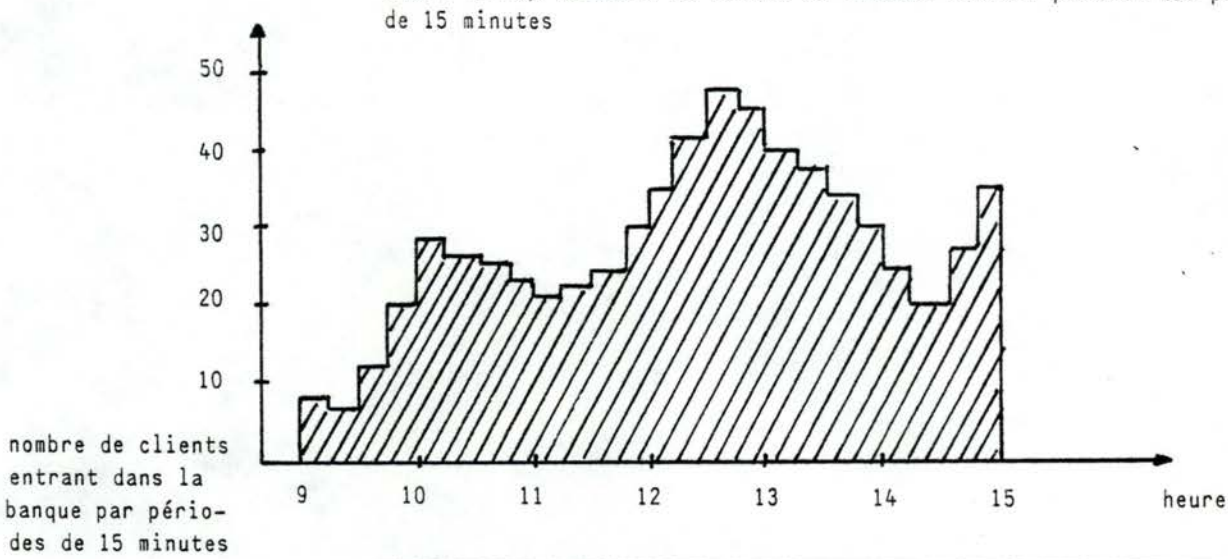
Voici une séquence possible d'étapes, préconisée par James Martin, pour concevoir le réseau de communication. Mais la conception de ce réseau peut n'être qu'une étape dans un processus plus large de conception d'un système en temps réel.

1. déterminer les types de messages
2. déterminer le trafic
3. établir le critère du temps de réponse souhaité
4. déterminer les localisations des terminaux
5. établir les caractéristiques des terminaux et choisir les types adéquats
6. déterminer le nombre de terminaux par localisation
7. la connexion terminal-ordinateur se fait-elle par commutation manuelle
8. déterminer en détail l'interface homme-machine et la structure de conversation
9. déterminer le nombre de caractères transmis
10. déterminer les structures de réseau possibles (multiplexeurs, concentrateurs, ...)
11. déterminer les procédures de contrôle de ligne possibles (séquences de "polling", priorités, half ou full duplex)
12. sélectionner les modems et les autres équipements se rapportant aux lignes
13. construire des modèles pour un chemin de communication (théorie des files d'attente ou simulation)
14. établir une table des taux de trafic
15. établir un réseau de lignes louées à un coût minimal
16. faire des raffinements au réseau (maintenance par exemple)

étapes 1 et 2

Tout d'abord, il faut déterminer les types de messages qui seront transmis, et, pour chaque type, le trafic. Ce trafic varie selon l'heure et le jour. Des pointes de trafic seront établies comme dans la figure ci-dessous.

Variation journalière des entrées des clients dans une banque. Les chiffres représentent le nombre de clients entrant pendant des périodes de 15 minutes



Il faudra décider quel niveau de trafic le système devra réellement traiter, compte tenu de ces pointes. Faut-il concevoir un système qui absorbera toutes les pointes, et donc qui sera sérieusement inutilisé à d'autres moments ?

étape 4

Déterminer où seront localisés les terminaux. Cette localisation donne lieu à des calculs économiques (de rentabilité par exemple).

étape 5

Déterminer les caractéristiques des terminaux. Les caractéristiques affectant le temps de réponse sont la vitesse "de sortie" (écran, imprimante), la taille des tampons s'il y en a, si ces terminaux sont prévus pour des lignes "full" ou "half duplex", les caractères de contrôle nécessaires, si on peut avoir une grappe de terminaux sur une ligne, et dans ce cas s'ils répondent au "polling" et combien de temps il faut pour "basculer" la direction de transmission (turn around time).

étape 6

Déterminer le nombre de terminaux nécessaires à chaque localisation.

étape 7

Déterminer si l'établissement de la connexion du terminal à l'ordinateur se fait par commutation manuelle. Si c'est le cas, il est nécessaire de déterminer la probabilité d'un échec lors de l'établissement de la connexion. Si le délai entraîné par la commuation est trop important, il faut trouver une autre solution.

étapes 8 et 9

Déterminer les longueurs de messages. La structure de la conversation homme-machine doit être connue. Souvent cette structure change pendant l'implémentation du système, ce qui provoque presque toujours une augmentation du nombre total de caractères transmis.

Dans ce cas, il est nécessaire de changer le réseau de communication. Quand les longueurs des messages sont connues, il peut être nécessaire de réitérer le processus, le coût du réseau pouvant être trop élevé. On voudra peut-être réduire les exigences en temps de réponse ou changer les équipements terminaux.

étape 10

Déterminer les appareils à utiliser, tels que les multiplexeurs, concentrateurs, commutateurs. Pour cela, il existe diverses méthodes approximatives de calcul.

étape 12

Choisir les modems, en tenant compte de leur temps de basculement entre le "polling" et le "selecting", aussi bien que de la vitesse de transmission de la ligne multipoint.

étape 13

Des modèles pour un chemin de communication de terminal à ordinateur peuvent être construits, indiquant le temps de réponse. On pourra voir comment celui-ci varie, et avec quel écart-type, selon le trafic. Ces modèles peuvent montrer les effets de changements dans l'organisation des lignes et dans les caractéristiques des terminaux, modems, etc ...

étape 14

Des modèles sont construits, utilisant la théorie des files d'attente ou la simulation, produisant une table des taux de trafic. Celle-ci indique les charges maximales de trafic qui sont admises selon des nombres différents de terminaux sur les lignes.

étape 15

La réalisation géographique du réseau peut être envisagée. L'utilisateur dispose d'un certain nombre d'outils d'aide à la conception physique du réseau.

3.3. APPRECIATION

La méthode que nous venons d'exposer présente un certain intérêt, étant donné son orientation "utilisateur". Cependant elle repose sur un certain nombre de procédures qui demandent des connaissances mathématiques telles que la théorie des files d'attente, ... etc, économiques telles que le calcul de rentabilité, ..., techniques telles que le temps de basculement des modems, etc ...

3.4. COMPARAISON DES METHODES DE CONCEPTION PRESENTEES PAR BOOTH ET MARTIN

A première vue, les deux méthodes semblent être équivalentes. Cependant une étude plus approfondie fait apparaître un certain nombre de différences.

Ces différences résultent de deux faits. Le premier fait est la date de publication de ces méthodes. Celle de Martin a été proposée au début des années 1970, tandis que celle de Booth a été présentée en 1981. En une décennie, les connaissances sur l'informatique distribuée ont fortement évolué.

la méthode de Martin reste à un niveau plus théorique et très technique. En effet, l'auteur met beaucoup d'importance sur l'utilisation de modèles mathématiques adéquats aux réseaux "classiques" (réseau en étoile, lignes multipoint). L'évaluation d'une solution se fait principalement au niveau des modèles.

La conception de systèmes d'informatique distribuée, dont l'essor a vraiment débuté vers la fin des années 1970, se base sur les connaissances acquises lors de la mise en oeuvre de systèmes. La méthode proposée par Booth rend compte de ce phénomène. Il y a une prise en considération quasi totale de l'environnement dans lequel le système d'informatique distribuée doit fonctionner. La méthode étant fortement pragmatique, elle utilise des installations réelles pour aboutir par tâtonnements successifs à une solution acceptable. En fait, actuellement la méthode proposée par Martin représente une étape de la méthode présentée par Booth (la troisième).

CHAPITRE 4 : METHODE GENERALE DE CONCEPTION DE SYSTEMES REPARTIS

4.1. INTRODUCTION

Cette méthode, que nous considérons comme une des meilleures, a été proposée par Macchi et Guilbert dans (M1). Nous nous limitons à reproduire succinctement les grandes lignes de cette méthode.

On peut généralement distinguer quatre étapes :

- 1ère étape : la définition des objectifs du réseau
- 2ème étape : la détermination des critères de choix
- 3ème étape : la définition de l'architecture du système
- 4ème étape : le choix des composants du système

La première et la deuxième étape correspondent à l'établissement d'un cahier des charges du système, c'est-à-dire la description qualitative et quantitative externe vue par l'usager comme par l'exploitant.

La troisième et la quatrième étape correspondent à une recherche d'une solution, c'est-à-dire à la spécification d'un système répondant au cahier des charges.

La première étape est essentielle dans la mesure où les résultats constituent les données de base pour la conception du réseau, et elle nécessite la coopération des usagers avec le concepteur.

La deuxième étape a pour but de quantifier un certain nombre de critères qui permettent d'effectuer des choix au cours des étapes suivantes de la conception. Il faut définir et chiffrer les constantes, techniques ou non, que doit satisfaire le réseau.

La troisième étape a pour but de déterminer une philosophie de réseau répondant au mieux au cahier des charges.

La quatrième étape a pour objet de fixer les équipements intervenant dans la mise en place du réseau.

A la fin de la conception, en reprenant certaines étapes, la solution obtenue lors des quatre étapes peut être affinée.

4.2. ETABLISSEMENT DU CAHIER DES CHARGES

4.2.1. Introduction

En principe, l'établissement du cahier des charges pour un système réparti ne se distingue pas beaucoup de l'établissement de celui d'un système classique.

Cependant dans un système réparti, l'usager a directement accès au système, ce qui donne une autre dimension aux problèmes relatifs aux moyens de communication avec le système, au temps de réponse et à la disponibilité du système.

4.2.2. Première étape : la définition des objectifs du système

Cette analyse donne lieu à quatre types différents d'études :

- l'analyse de la situation
- l'analyse des besoins
- l'analyse des contraintes
- l'étude de faisabilité

4.2.2.1. l'analyse de la situation

Cette analyse aboutit à la définition des objectifs généraux du système.

La conception de nouveaux systèmes doit faire l'objet d'études approfondies tenant compte de la réalité de l'entreprise et de son évolution possible ; la nature et l'importance de l'entreprise pouvant orienter le choix des solutions.

L'analyse de la situation commence par une prise de connaissance de l'entreprise, qui permet d'obtenir une description de

- la structure de l'entreprise
- l'environnement du système

L'environnement est défini par un certain nombre de paramètres tels que

- l'implantation géographique de l'entreprise (implantation des divers sites)

- la nature du traitement existant dans l'entreprise

- etc ...

L'organigramme de gestion fournit généralement une représentation symbolique du réseau d'acheminement des informations (diagramme des flux). La structure fonctionnelle qui met en évidence les sites générateurs et utilisateurs d'informations sera précisée en fonction de l'implantation géographique de l'entreprise.

Après cette prise de connaissance, il faut étudier les applications téléinformatiques existantes, et leurs liaisons avec le système doivent être analysées des points de vue :

- grandes fonctions
- trafic
- services offerts
- etc ...

4.2.2.2. l'analyse des besoins

Cette analyse, qui demande la coopération des usagers avec le concepteur, permet une description des services attendus par les usagers du système. Elle vise à élaborer la solution sans entrer dans les détails internes du fonctionnement du système, ni définir ses composants matériels et logiciels. La description du système comprend :

- la description de l'environnement du système

- l'analyse fonctionnelle ou conceptuelle du système

- le schéma général fonctionnel du système

a) description de l'environnement du système

Il faut approfondir les connaissances sur l'environnement du système. A la fin de cette étape, on aboutit à une description précise de l'environnement fonctionnel du système, c'est-à-dire :

- à une description des tâches effectuées au sein de l'entreprise en amont et en aval
- à une description des problèmes organisationnels
- à une description des limites du système

b) analyse fonctionnelle ou conceptuelle du système

L'analyse conceptuelle a pour but d'étudier qualitativement et quantitativement :

- la structure des traitements mis en oeuvre
- la structure des données manipulées

Nous n'allons pas entrer dans les détails d'une analyse fonctionnelle, toutefois nous prions le lecteur intéressé de consulter (C13), (M1), (B3).

c) schéma général fonctionnel

L'analyse fonctionnelle se concrétise par un document reflétant le consensus au niveau fonctionnel entre le concepteur, les usagers et les exploitants du système. Ce document décrit avec précision les fonctions assumées par le système.

4.2.2.3. l'analyse des contraintes

Lors de l'analyse de la situation et de l'analyse des besoins, on peut mettre en évidence un certain nombre de contraintes relatives à la mise en place du système au sein de l'environnement défini. Ces contraintes, qui peuvent être externes ou internes à l'entreprise, ont un impact sur la conception du système réparti, dans la mesure où elles apportent un certain nombre de restrictions au domaine des choix possibles.

Ces contraintes sont de plusieurs natures :

- contraintes techniques
- contraintes d'exploitation
- contraintes budgétaires
- contraintes de délai de réalisation
- contraintes externes (matériel et logiciel)
- contraintes relatives au personnel
- contraintes juridiques
- contraintes psychologiques et sociologiques (réactions des usagers)

4.2.2.4. étude de faisabilité

Cette étude vérifie que les objectifs et contraintes sont envisageables, c'est-à-dire que les divers éléments quantitatifs et qualitatifs représentatifs du système ne conduisent pas à une conception irréalisable. Il y a possibilité d'une remise en cause éventuelle de ces objectifs et contraintes.

Quatre types d'études peuvent être envisagés :

a) étude de faisabilité fonctionnelle

Cette étude justifie les besoins exprimés par les usagers. L'objectif essentiel est de vérifier le bien-fondé de ces besoins et de voir si on ne peut pas satisfaire des besoins analogues par une légère adaptation du système.

b) étude de faisabilité technique

Cette étude doit démontrer, indépendamment des aspects financiers du problème, qu'il existe au moins une solution technique envisageable, tant sur le plan du matériel, que des logiciels.

c) étude de faisabilité opérationnelle

Cette étude analyse les diverses possibilités de gestion et d'exploitation du système au sein de l'environnement défini par les études précédentes.

d) étude de faisabilité financière

Cette étude tient compte du coût de réalisation, de mise en place et d'exploitation. Elle vérifie que ces coûts entrent dans le cadre des contraintes budgétaires ou sont compétitifs avec la solution existante ou d'autres solutions envisageables.

4.2.3. Deuxième étape : la détermination des critères de choix

La solution recherchée doit répondre à un certain nombre de critères de choix relatifs au système. On peut distinguer plusieurs catégories de critères :

- les critères de performances

  par exemple : . temps de réponse
  . stabilité du système
  . disponibilité, etc ...

- les critères techniques

  concernant les équipements en matériel et logiciel

par exemple : pour le matériel : . conformité aux normes
. compatibilité
. fiabilité
etc ...

pour le logiciel : . fiabilité
. outils de production
etc ...

- les critères d'exploitation

par exemple : . simplicité d'exploitation
. reprise de l'existant
. vulnérabilité aux pannes
etc ...

- les critères relatifs au personnel

concernant les différentes catégories de personnel que nécessitent la mise en oeuvre du système et son utilisation ; on s'intéresse à la qualification, la stabilité et les politiques de recrutement et de formation de ce personnel

- les critères géographiques

- les critères économiques

- les critères politiques

La pondération de chacun de ces critères permet au concepteur d'orienter ses choix au moment de la recherche d'une solution.

4.3. RECHERCHE D'UNE SOLUTION

4.3.1. Introduction

Le travail de recherche consiste en :

- la définition de l'architecture du système

- le choix des composants

- l'optimalisation du système conforme au cahier des charges

4.3.2. Définition de l'architecture du système

4.3.2.1. introduction

A ce niveau, l'expérience pratique et les connaissances du concepteur sont nécessaires pour choisir une solution. Son choix peut être facilité par l'utilisation de méthodes automatiques. En tous cas, le concepteur utilise les critères de choix donnés par le cahier des charges.

L'étude que le concepteur doit entamer est à la fois qualitative et quantitative, et porte sur les différents éléments constitutifs du système :

- points d'accès au système
- centre de traitement et de stockage de l'information
- le réseau de transport pour acheminer les informations

A ce niveau, il est conseillé de définir également la maintenance et l'exploitation du système.

4.3.2.2. distribution géographique des postes de travail

En nous basant sur les données de l'analyse conceptuelle, nous sommes capables de déterminer le nombre de terminaux, de manière à satisfaire les contraintes de temps de réponse, de disponibilité, etc ... au niveau de chaque poste de travail.

4.3.2.3. répartition du traitement et du stockage

Ce problème a été discuté dans la partie 1 / chapitre 5 de ce mémoire.

4.3.2.4. définition de l'architecture du système de gestion

Lorsque l'architecture du système a pris sa forme, il est nécessaire d'envisager sa gestion. Le travail consiste à définir les fonctions de gestion et de contrôle à réaliser.

4.3.3. Choix des composants du système

4.3.3.1. introduction

Cette étape a pour objet de déterminer les divers éléments qui vont être assemblés pour constituer le système dont l'architecture vient d'être définie.

Ce choix concerne trois ensembles de composants :

- les terminaux
- les moyens de traitement et de stockage
- les moyens de transport et optimalisation du réseau

4.3.3.2. choix des terminaux

Le terminal est le moyen d'accès au système téléinformatique qui assure la saisie et la restitution de l'information. Les types de terminaux envisageables pour le système ont été définis lors de l'étude des postes de travail (analyse fonctionnelle).

Le choix des terminaux est non seulement fonction de la philosophie du système, mais aussi de leur facilité d'utilisation et de leurs performances (terminaux plus ou moins intelligents, plus ou moins autonomes).

4.3.3.3. choix des moyens de traitement et de stockage

Le choix du système de traitement se fait après la prise de connaissance des charges entre les terminaux et les systèmes centraux. La connaissance des volumes d'informations échangées, des types et du nombre de terminaux par lieu géographique permet de déterminer les caractéristiques de ces moyens de traitement et de stockage.

4.3.3.4. choix des moyens de transport et optimalisation du réseau

L'étude du réseau comporte :

- le choix d'un ou plusieurs services de transmission de données proposés par les services publics de télécommunication (RTT)
- le choix du matériel de transmission de données
- l'organisation de ces divers moyens dans un réseau de transport plus ou moins complexe

Les objectifs à respecter sont :

- la satisfaction des exigences de performances (débit efficace, temps de réponse, etc ...) et de sécurité qui découlent du cahier des charges
- la diminution du coût du réseau (par exemple : par la concentration du trafic, l'utilisation d'un réseau public)

L'utilisation des moyens de transport, en particulier des lignes spécialisées, peut être optimalisée par l'adjonction d'équipements qui regroupent plusieurs flux de données sur un circuit de données. Ceci est réalisé par des techniques de multiplexage ou de concentration.

Les besoins et les solutions peuvent être dégagés en analysant le :

- taux de connexion

  $E = N * T / 3600$ Erlang

  où N = nombre de périodes d'activité par heure de durée T(s)
  (si E = 1 : connexion permanente)

- taux d'activité

  $\theta = n * l / T * D$

  où n = nombre de transmissions de longueur l (caract) par période d'activité
  D = débit en caract / sec

- débit efficace

  $D_m = N * n * l / 3600$ caract / sec

Décisions qu'on peut prendre à partir des valeurs de E et de θ

E	θ	*types de trafic ou d'application*	*concentration possible*
voisin de 1	*voisin de 1*	*télémesure, conduite de processus*	*aucune (point-à-point)*
	faible	*système en temps réel*	*concentration par liaison multipoint* *- concentrateur* *- commutation par paquets*
faible	*voisin de 1*	*transmission par lots*	*concentration par commutation de circuit*
	faible	*conversationnel par courtes périodes*	*double concentration* *- de circuits* *- de messages*

4.3.4. Vérification de la solution

C'est essentiellement un contrôle de faisabilité. Il s'agit d'évaluer le coût de la configuration en fonction des performances attendues.

Cette étape tient compte des critères donnés par le cahier des charges, dont les plus importants sont le débit du système, le temps de réponse, la stabilité, l'extensibilité, la disponibilité et le coût.

CONCLUSION

La conception d'un système d'informatique distribuée est une entreprise difficile qui n'est pas régie par des règles strictes.

Cependant toutes les méthodes nécessitent des connaissances diverses en ce qui concerne :

- le matériel et logiciel pour le traitement des données
- le matériel et logiciel pour la transmission des données
- la théorie des files d'attente
- la théorie des graphes
- le calcul de fiabilité
- l'économie
- les moyens de transport
- l'exploitation des grands systèmes
- les statistiques

Cependant le concepteur peut être aidé dans sa tâche difficile par des outils divers d'aide à la conception, qui peuvent être du type mathématique ou autre. Dans la partie suivante, nous allons présenter un outil pragmatique d'aide à la conception, qui complète la quatrième méthode de conception présentée ci-dessus.

Dans les chapitres précédents, nous avons présenté quatre lignes de conduite lors de la conception d'un sytème d'informatique distribuée. A priori on ne peut pas recommander la meilleure solution. En effet, à chaque système réparti à développer correspond un problème de conception particulier, et ainsi il faut dégager la méthode de conception adéquate. Ou bien une des quatre méthodes est applicable, ou bien le concepteur doit développer une nouvelle ligne de conduite adaptée aux besoins spécifiques de son système.

PARTIE 4 : OUTIL PRAGMATIQUE D'AIDE A LA CONCEPTION DE SYSTEMES D'INFORMATIQUE DISTRIBUEE

INTRODUCTION

Dans la partie précédente, nous avons présenté la problématique de la conception de systèmes d'informatique distribuée. Après avoir présenté les éléments majeurs à prendre en compte lors de la conception, nous avons proposé un certain nombre de méthodes de conception possibles.

Dans cette quatrième partie, nous allons présenter en toute généralité un outil pragmatique d'aide à la conception de réseaux informatiques complexes. Ce type d'outil se base plutôt sur des informations concernant les applications utilisateur, et sur des observations, que sur des modèles mathématiques rigoureux.

L'utilisation de cet outil est particulièrement intéressante lors de la vérification d'une solution. En effet, l'outil permet un contrôle de faisabilité simple à utiliser.

Lors de notre stage chez CII-Honeywell Bull à Paris, nous avons pu nous familiariser avec ce type d'outil. Les études que nous avons faites ont été clôturées par la confection d'un système automatisé d'aide à la conception de réseaux DSA complexes.

CHAPITRE 1 : INTRODUCTION A L'OUTIL PRAGMATIQUE ET AUX OBJECTIFS POURSUIVIS

1.1. INTRODUCTION

L'outil que nous présentons dans les chapitres suivants fait partie d'un ensemble d'instruments d'aide à la conception de systèmes d'informatique distribuée.

La conception de tels systèmes est un processus présentant essentiellement trois phases :

1. phase d'avant-vente de matériel informatique de réseau
2. phase de réglage (anglais : tuning) et d'optimisation de systèmes d'informatique distribuée
3. phase de modifications du système

Les méthodes de conception présentées dans la partie 3 reprennent uniquement les phases d'avant-vente et parfois de modification du système. Ainsi notre outil, qui peut être utilisé dans une méthode générale, ne s'applique qu'aux phases 1 et 3.

Pour chacune de ces trois phases, il existe des outils spécialisés :

- Pour la première et la troisième phase, on peut appliquer des outils pragmatiques analogues à celui que nous avons étudié et automatisé. Ce type d'outil est principalement orienté vers les utilisateurs du système sous étude. Bien que les formalisations requises pour l'utilisation de ces instruments ne permettent que d'aboutir à des résultats approchés, ces outils donnent une bonne idée générale de la structure, des performances et du coût du système.

- Pour la deuxième phase, on dispose d'outils plus sophistiqués qui sont basés, dans la plupart des cas, sur des modèles mathématiques tels que la théorie des graphes ou la théorie des files d'attente. Vu la précision des résultats obtenus, ces outils permettent d'optimiser l'utilisation et le fonctionnement du système. La qualité de ces optimisations dépend fortement de la qualité des modèles utilisés, dont l'élaboration constitue un travail compliqué et minutieux, et de celle des données d'entrée.

1.2. OBJECTIFS

Dans la première partie du mémoire, nous avons décrit les trois fonctions principales assumées par les sytèmes d'informatique distribuée, à savoir :

- la fonction de traitement des données
- la fonction de stockage des données
- la fonction de transport des données

L'outil que nous avons étudié s'occupe uniquement de la fonction de transport des données à travers le réseau.

La structure de cet outil est relativement simple : à partir de plusieurs types de données, il permet de dimensionner le réseau sous étude.

Les données à fournir sont de deux types différents. Il y a d'une part des données fondamentales, relatives aux comportements des divers équipements constituant le réseau face à un trafic de données. Il y a d'autre part les données variables ou "utilisateur", qui proviennent d'une analyse fonctionnelle de l'ensemble des applications de l'utilisateur à mettre en oeuvre sur le réseau.

Les résultats auxquels l'outil permet d'aboutir sont de plusieurs natures. Il y a d'abord des "résultats-résumés" présentant de façon synthétique les données collectées concernant

- les transactions
- les agents
- les sites

Ensuite, il dimensionne les liens entre les divers sites, en donnant le nombre de lignes physiques, leur vitesse, leur taux d'utilisation, ... Finalement, l'outil dimensionne les équipements tels que les concentrateurs, commutateurs, contrôleurs de terminaux, ...

Lors de notre étude, nous nous sommes limités à la charge CPU de ces équipements. Celle-ci peut être présentée de façon globale, ou elle peut être présentée par fonction assumée par l'équipement sous étude.

Remarque : chez CII-HB, l'outil dimensionne l'ordinateur de réseau "Datanet DN 7100", et présente la charge CPU de cet équipement :

- globalement
- pour la connexion du datanet à un ordinateur hôte
- pour le réseau primaire
- pour le réseau secondaire

1.3. OUTIL AUTOMATISE

Un lecteur initié à la problématique des systèmes d'informatique distribuée peut entrevoir la complexité des manipulations de données à effectuer pour aboutir aux résultats souhaités.

Après avoir présenté les grandes lignes de la "théorie de l'outil" développé par Claudie Chappuis (CII-HB), nous présentons brièvement le programme que nous avons développé. Ce prototype répond aux spécifications exprimées dans la base théorique, et dans un but de minimiser les sources d'erreurs et d'augmenter la facilité d'utilisation, nous avons ajouté un certain nombre de fonctionnalités.

CHAPITRE 2 : ETUDES PRELIMINAIRE REQUISES

2.1. PLAN DES ETUDES

Dans ce chapitre, nous exposons les études que l'utilisateur doit faire avant d'utiliser l'outil de dimensionnement. Ces études vont, d'une part, permettre la paramétrisation de l'outil, et, d'autre part, elles vont déboucher sur une collecte de données pour laquelle l'utilisation de supports tels que des formulaires est vivement conseillée.

Les études se situent à deux niveaux :

- Au premier niveau, on retrouve les études concernant les données dites fondamentales. On y trouve les études des comportements ou les mesures de performances des équipements impliqués dans l'architecture de réseau. Sans cette information, on n'arrive pas à dimensionner le réseau. Il convient de remarquer que ce niveau est très technique, et que dans la majorité des cas, seuls les constructeurs sont capables de fournir ces données. Au paragraphe 2.2., nous allons passer en revue un certain nombre de techniques qui permettent d'aboutir à des formules de calcul telles que par exemple le temps CPU du processeur frontal en fonction des trames HDLC traitées.

- Au deuxième niveau, on retrouve les études relatives aux données variables, c'est-à-dire les données sur l'ensemble des applications. Ces données sont divisées en deux catégories :

  * la première catégorie concerne les données "applications". Les applications de l'utilisateur sont analysées dans une optique de réseau, c'est-à-dire qu'on s'intéresse aux transactions générées et aux agents qui initialisent ces transactions.

  * la deuxième catégorie concerne les données "réseau". L'utilisateur définit la topologie de son réseau, c'est-à-dire qu'il définit les sites et il détermine les liens entre ces sites.

2.2. DONNEES FONDAMENTALES

2.2.1. Etude des comportements des équipements

Les comportements qui peuvent intéresser l'utilisateur lors de la conception de systèmes d'informatique distribuée sont :

- la charge CPU des divers composants du système, face à un trafic donné

- le temps de réponse
- le temps de transit
- les taux d'utilisation des ressources
- les besoins en place mémoire dans les divers équipements
- l'efficacité des protocoles utilisés

L'intérêt de ces mesures est :

- la possibilité d'obtenir des formules de calcul permettant le calibrage d'un réseau. C'est ce point qui nous intéresse le plus. En effet, pour assurer un contrôle de faisabilité d'une architecture de réseau, l'outil automatisé doit connaître les performances des divers équipements matériels vis-à-vis d'une certaine charge. Ces performances ne varient pas uniquement avec la charge, mais aussi avec les protocoles utilisés. Ainsi, avant même d'utiliser l'outil, il faut connaître par exemple le temps CPU d'un frontal pour traiter n paquets par message. La connaissance de ces informations permet "d'initialiser" les procédures de calcul du programme.

- la possibilité d'obtenir des indications pour optimiser une installation informatique.

La qualité de l'outil dépend fortement de la précision et de la validité des mesures qui ont été effectuées.

2.2.2. Méthodes de mesure de performances

Ces formules de calcul déduites des mesures constituent la base de tout le dimensionnement. Comme nous n'avons pas obtenu des informations précises sur les mesures effectuées chez CII-HB, nous jugeons important de passer en revue les principales méthodes de mesures utilisées. Nous avons découvert trois catégories distinctes de mesures :

- La première catégorie regroupe les méthodes basées sur des simulations. Ces simulations sont fondées sur des modèles de files d'attente. Les mesures de ce genre donnent une idée globale du fonctionnement du système (composant ou ensemble de composants).

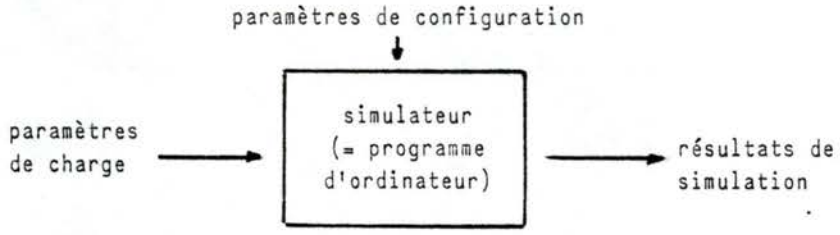
- La deuxième catégorie de mesures est basée sur l'utilisation de stimulateurs (ou émulateurs). Ces mesures sont faites sur le matériel réel. Ce qu'on simule sont les charges de travail. Cette méthode permet de découvrir des goulots d'étranglement.

- La troisième catégorie de méthodes fait appel aux techniques du "monitoring". Les moniteurs matériel et logiciel permettent d'analyser de façon précise le système sous étude, et ils permettent le réglage (tuning) et l'optimisation de l'installation.

2.2.2.1. première catégorie : les simulations

Dans le contexte une telle méthode est rarement utilisée, cependant elle permet en principe d'arriver à des résultats globaux.

Le schéma suivant présente la procédure d'évaluation des performances d'un système informatique. Le simulateur simule le comportement de ce système faisant face à une charge donnée.



* problèmes

  - l'équipement dont on veut connaître une performance précise doit être modélisé de façon précise

  - la charge qu'on simule doit être aussi réaliste que possible

* avantage

  - rapidité des mesures

* désavantage

  - la qualité des résultats dépend fortement de la qualité du modèle sur lequel repose le simulateur

* remarque

  - IBM par exemple utilise cette méthode (SNAPSHOT)

2.2.2.2. deuxième catégorie : les stimulateurs

Les stimulateurs ou émulateurs (de l'anglais to emulate : faire aussi bien que ...) peuvent être classés dans trois groupes que nous allons brièvement présenter. Chacun de ces stimulateurs repose sur les mêmes principes :

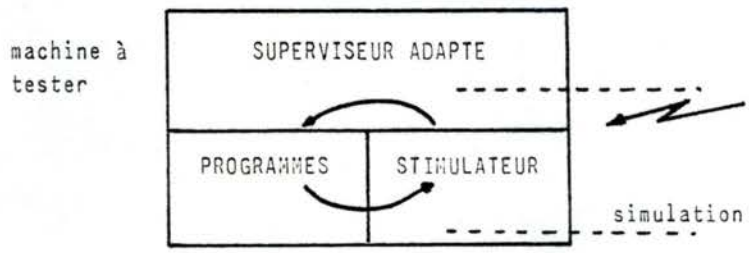
- Les mesures sont faites sur un nombre restreint de cas, considérés comme représentatifs

- Dans le cas de mesures de processeurs frontaux, il y a établissement d'un flux continu de données. La condition de continuité est nécessaire pour se trouver dans un régime stable de fonctionnement, et pour éviter les effets de bord dûs à l'intialisation et à la saturation. Ce flux doit être maintenu pendant une période assez longue (environ 1 à 2 heures).

- On stimule le système sous étude selon des scénarios de charge prédéfinis, qui doivent être réalistes.

a) stimulateurs intégrés

Cette méthode est utilisée pour les mesures de comportement d'un ordinateur face à une charge précise de travail. Le schéma suivant montre le principe de cette méthode



Les tests se font sur la machine réelle. Le stimulateur ou émulateur est un programme qui se trouve en mémoire centrale de l'ordinateur, et il simule n terminaux asynchrones ou autres.

* avantage

  - réalisation aisée des mesures

* désavantage

  - cette méthode présente un désavantage considérable : du fait que l'émulateur est un programme s'exécutant lui-même dans le système qu'il doit mesurer, il y a création d'une charge supplémentaire pour le processeur central de l'ordinateur, ainsi que pour les processeurs canaux.

* remarques

- cette méthode ne convient pas pour mesurer les performances d'un processeur frontal où les capacités de traitement et de stockage sont limitées.

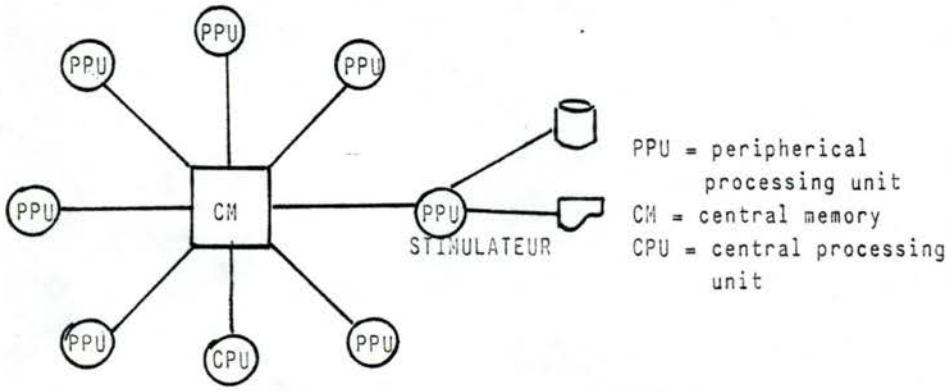
- les stimulateurs collectent eux-même les résultats des stimulations.
  Par exemple Siemens a utilisé cette méthode à la fin des années 70, à l'aide du stimulateur "Einstein" tournant sous BS 2000.

- une telle méthode peut être appliquée au cas des processeurs de réseau intégrés, comme par exemple les "Datanets DN 7100" de CII-HB qui ont des possibilités de traitement considérables.

b) stimulateurs intégrés sur un processeur interne

Cette méthode est uniquement applicable à des systèmes informatiques multiprocesseurs.

Schéma :



Dans cette méthode, le stimulateur n'a plus besoin du CPU pour s'exécuter, mais il utilise un processeur périphérique qui ne sera pas présent dans la configuration vendue. Ici le stimulateur remplace n lignes asynchrones ou autres.

* remarques

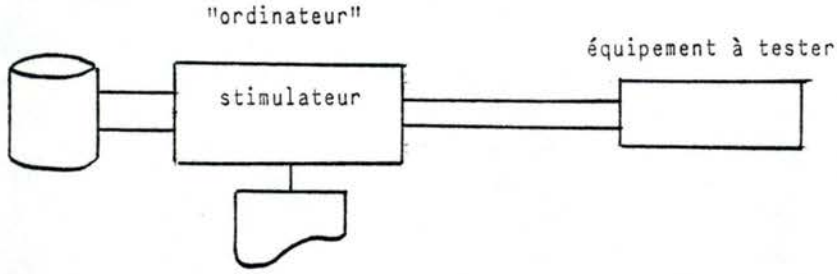
- cette méthode est inadéquate pour le cas des processeurs frontaux et autres équipements de réseau, puisque ces machines ont généralement une structure monoprocesseur.

- cette méthode est surtout utilisée par CDC.

c) stimulateurs externes

Cette méthode est fréquemment utilisée, et semble être une solution réaliste au problème des mesures.

Schéma :



Dans le cas de processeurs frontaux ou d'équipements analogues, on connecte l'appareil sous étude à un ordinateur dans lequel s'exécute le stimulateur. Le programme responsable de l'émulation des terminaux ou autres équipements est la plupart du temps un générateur de charge créant un flux spécifique de données. Ce flux peut être paramétré, et est constant pendant la durée des mesures (par exemple : utilisé dans les mesures de l'ARPANET, PRNET, SATNET). Plusieurs scénarios de charge sont ainsi soumis à l'équipement.

* avantages

- coût modéré des mesures
- proche de la réalité
- flexible

* remarque

- méthode utilisée par exemple par DEC, CII-HONEYWELL BULL et UNIVAC

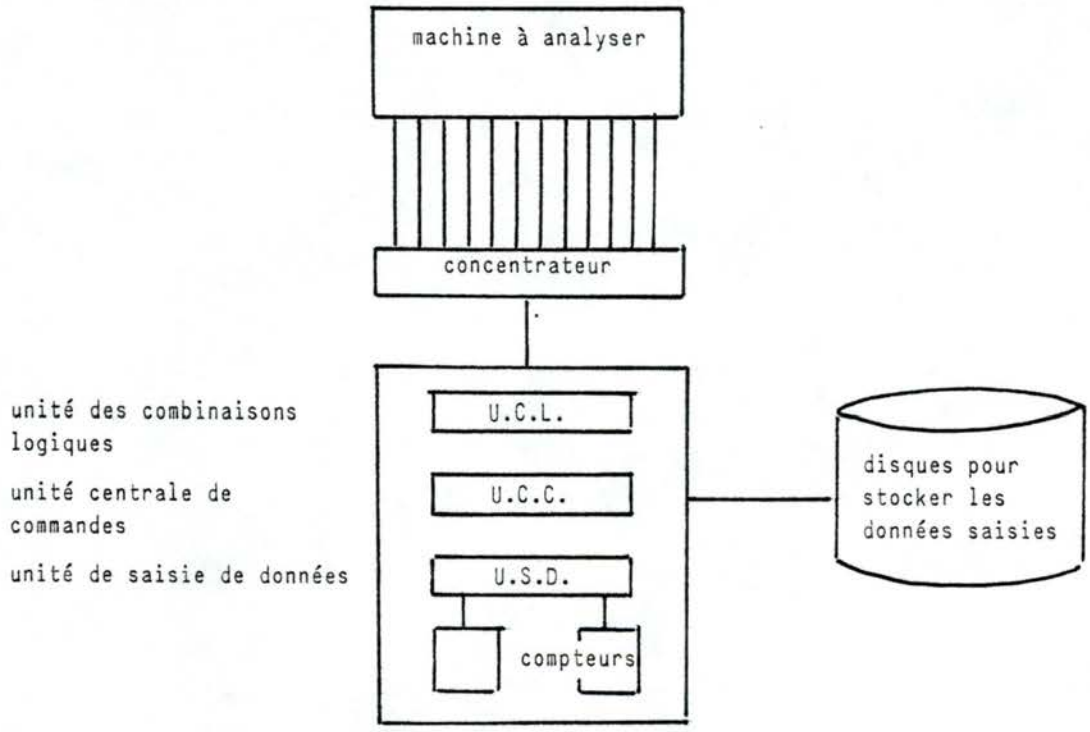
2.2.2.3. troisième catégorie : les moniteurs

Il y a deux catégories principales de moniteurs : les moniteurs "hardware" et les moniteurs "software". Les moniteurs "hardware" sont réalisés à partir de matériel ordinateurs, mini, micro, ... etc, tandis que les moniteurs "sofware" sont réalisés par des programmes.

a) moniteurs hardware

Le moniteur est un processeur capable d'enregistrer, à l'aide de sondes, des signaux électriques en provenance d'une unité centrale. L'utilisation d'un tel moniteur nécessite la présence de points de connexion pour les sondes dans les divers éléments constituant le système d'informatique sous étude. (le moniteur hardware utilisé par Siemens permet par exemple de placer 144 sondes).

Schéma général d'un moniteur hardware :



Les signaux sont enregistrés sur disque magnétique au fur et à mesure de leur apparition. L'expérimentateur a aussi la possibilité d'enregistrer la durée et la fréquence d'apparition de ces signaux. Les mesures terminées, les données sur disque seront analysées par un programme d'analyse approprié.

Les sondes sont connectées à un concentrateur qui réalise d'une part la concentration et d'autre part une adaptation des lignes. Dans l'unité assurant les combinaisons logiques, il y a réalisation de diverses connexions logiques

câblées (exemple : on analyse l'état : ((canal 1 et canal 2) actifs et CPU idle). Après avoir transité par l'unité de contrôle et de commande, les données entrent dans une unité de saisie de données. Dans cette unité, elles sont analysées et comptées. Les résultats de cette analyse sont enregistrés sur disque magnétique.

L'avantage de ces moniteurs peut être vu dans la vitesse élevée des mesures effectuées.

b) moniteurs software

Les moniteurs sont des programmes de mesures, qui enregistrent certains états d'un système de traitement de données. A l'aide de ces programmes, qui s'exécutent sous le contrôle du système d'exploitation, on interroge, contrôle, analyse le contenu des divers registres de travail. Il est important que le programme soit très court, afin que l'influence de son exécution sur le fonctionnement du système global reste négligeable.

Principalement il y a deux techniques fondamentales. Il y a, d'une part, la méthode de l'échantillonnage (anglais : sampling), et, d'autre part, la méthode du "tracing". Dans la méthode de l'échantillonnage, on analyse et enregistre, à des intervalles réguliers, les états de certains registres de travail. Dans la méthode du tracing, ce sont les éléments eux-mêmes qui déclenchent la procédure de mesure.

Ces types de moniteurs sont fréquemment utilisés.

c) comparaison des deux types de moniteurs

* caractéristiques

moniteur hardware	moniteur software
- pas d'intervention dans le système	- accès aux adresses, tables et registres de travail
- pas d'occupation de la place mémoire	- saisie du contenu des files d'attente devant les différentes ressources
- pas d'overhead	- analyse de l'état du système d'exploitation
- mesure en parallèle de plusieurs points	- prix assez bas
- haute précision des mesures	
- indépendant de la machine et de son système d'exploitation	
- accès aux registres	

* utilisations des moniteurs hardware et software

critère	moniteur hardware	moniteur software
- saisie de données hardware	- facile	- difficile
- saisie de données système d'exploitation	- avec des moyens énormes	- facile
- saisie des données programmes utilisateur	- difficile	- facile
- influence du moniteur	- aucune	- petite et réglable
- précision	- grande	- petite : peu d'influence - grande : beaucoup d'influence
- portabilité	- possible	- modification du moniteur nécessaire
- facilité d'utilisation	- compliquée	- facile
- coût par utilisation	- élevé	- bas

2.2.3. Analyse des mesures de performances

Les résultats d'une émulation doivent être analysés, afin de dégager des comportements des équipements. Nous souhaitons aboutir à des formules de calcul qui permettent de faire des prévisions utiles lors du dimensionnement d'un réseau.

Nous avons à notre disposition les résultats de n expériences, où nous connaissons les valeurs prises par p+1 variables quantitatives (nombre de messages/sec, nombre de paquets, temps CPU, ...). Nous allons essayer "d'expliquer" les valeurs prises par une de ces variables, appelée "variable à expliquer", et notée y, en fonction des valeurs prises par les autres variables en notre possession.

Ici, cette explication de y prendra la forme d'une estimation de cette variable, connaissant les valeurs prises par les autres.

Il s'agira donc d'établir une relation fonctionnelle du type $y = f(x_1, x_2, \ldots x_p)$, qui donnera une estimation de y, connaissant $x_1 \ldots x_p$.

Nous étudions brièvement un outil d'estimation : la régression.

2.2.3.1. intérêt de la régresssion

Le but de la régression est de pouvoir estimer (et donc prédire, sous certaines conditions) la valeur d'une variable y, valeur que l'on peut connaître directement, en appliquant au modèle $y = f(x_1 \ldots x_p)$ les valeurs correspondant à chacune des variables x_i.

Le problème consiste à déterminer quelle est la meilleure structure, la meilleure forme pour ce modèle, et donc à définir f. Parmi l'infinité possible de formes de f, on utilise généralement la combinaison linéaire entre les variables, c'est-à-dire que le modèle s'écrit :

$$y = \sum_{i=1}^{p} a_i x_i + b$$

La prédominance de ce modèle tient à trois raisons principales :

1. cette relation linéaire est la plus simple parmi l'infinité de formes possibles, tant au niveau de sa détermination, qu'au niveau de son interprétation

2. il est possible de s'assurer de la validité des résultats d'estimation, ce qui est particulièrement intéressant dans le cas de prévisions

3. on peut facilement transformer le modèle

Il faut remarquer que les relations ainsi trouvées entre y et les x_i sont des relations d'existence simultanées et non de causalité.

2.2.3.2. exemple

Nous voulons estimer le temps CPU d'un processeur frontal, en fonction du nombre "d'enclosures" (fragments, messages, trames, paquets, ...), et en particulier en fonction du nombre de paquets.

Le modèle qui convient le plus, et qui est le plus simple, est un modèle linéaire du type :

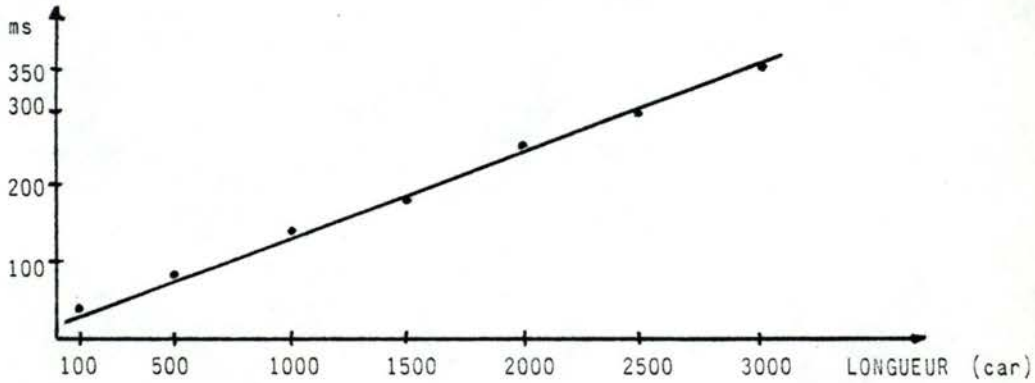
$t = L + Mx$

L = coût de prise en charge de message
unité : ms
M = coût de prise en charge d'un enclosure (par exemple : paquet)
unité : ms
x = nombre d'enclosures

Une expérience de mesures a fourni des valeurs pour t pour des messages de taille différente. Nous devons estimer L et M. Pour cette expérience, nous considérons que l'enclosure = le paquet.

tableau des mesures pour t :

longueur message (caract)	temps CPU (unité reçue) (ms)
100	40
500	80
1000	135
1500	180
2000	250
2500	290
3000	350



Puisque c'est le nombre de paquets qui nous intéresse, les longueurs de lettres seront converties en nombre de paquets, au moyen de la formule :

$$p = \text{arrondi}\left(\frac{\text{longueur lettre} + 14}{128}\right)$$

Il faut estimer L et M dans la formule :

$t = L + Mp$

p = nombre de paquets

On peut calculer l'équation de la droite par la méthode des moindres carrés.

méthode pratique

1. Créer un tableau ayant des colonnes P (nombre de paquets par message), et T (temps de traitement du message correspondant).

2. Dans la colonne P^2, calculer les carrés des P ; dans la colonne P * T, calculer les produits de P et T.

3. Former les sommes des diverses colonnes. Diviser par le nombre de lignes du tableau pour obtenir la moyenne de chaque ligne.

	P	T	P^2	P * T
	1	40	1	40
	4	80	16	320
	8	135	64	1080
	12	180	144	2160
	16	250	256	4000
	20	290	400	5800
	24	350	576	8400
Σ	85	1325	1457	21800
MOY	12.14	189.29	208.14	3114.29
$(\bar{P})^2$			147.45	
$\bar{P} * \bar{T}$				2298.47

4. Former les différences :

$D_1 = (\overline{P^2}) - (\bar{P})^2 = 60.69$

$D_2 = (\overline{P * T}) - (\bar{P}) * (\bar{T}) = 815.82$

pente de la droite : $M = \frac{D_2}{D_1} = \frac{815.82}{60.69} = 13.44$

5. La droite passe par le centre de gravité du nuage des 7 points, c'est-à-dire qu'il passe par la moyenne des variables P et T. L'équation de la droite est données par :

$t - \bar{T} = M\ (p - \bar{P})$ $\qquad (Y-Y_1) = \frac{Y_2-Y_1}{X_2-X_1}(X-X_1)$

$$t = 13.44\ p - 13.44 * 12.14 + 189.29$$

$$= 13.44\ p + 26.13$$

Cette formule nous donne donc le temps CPU en ms pour traiter p paquets.

2.2.4. Conclusion

Ainsi nous pouvons nous donner des "outils" de prévision qui permettent, selon leur nature, d'évaluer le comportement futur des divers équipements face à un trafic donné. Ces formules de calcul permettent ainsi de dimensionner le système d'informatique distribuée.

2.3. LES DONNEES VARIABLES OU "UTILISATEUR"

2.3.1. Etude de l'ensemble des "applications utilisateurs"

Dans les paragraphes suivants, nous allons répondre aux questions suivantes :

- "Ce réseau, que transporte-t-il ?", c'est-à-dire que nous devons étudier les unités de travail manipulées par la fonction de transport dans les systèmes d'informatique distribuée.

- "Qui génère la charge de travail ?" Nous allons nous occuper des utilisateurs terminaux du système.

Toutes les informations nécessaires sont retenues.

2.3.1.1. description des unités de travail, les types de transactions

L'utilisateur doit d'abord décrire ce que le réseau doit acheminer. Il décrit les types de transactions, c'est-à-dire les unités de travail, en termes de types de processus utilisés et de messages échangés, dans une vue d'utilisateur-terminal.

Pour chaque type de transaction identifié, l'utilisateur détermine tous les types de processus nécessaires au traitement de la transaction. Chaque type de processus est indistribuable, c'est-à-dire que le processus ne peut pas être implanté partiellement sur un site et partiellement sur un autre.

Pour chaque type de transaction, il faut déterminer les transmissions, c'est-à-dire qu'il faut déterminer le nombre de messages, et leur longueur, transmis entre agent initialisant la transaction et type(s) de processus, entre type(s) de processus, et entre type(s) de processus et agent initialisant.

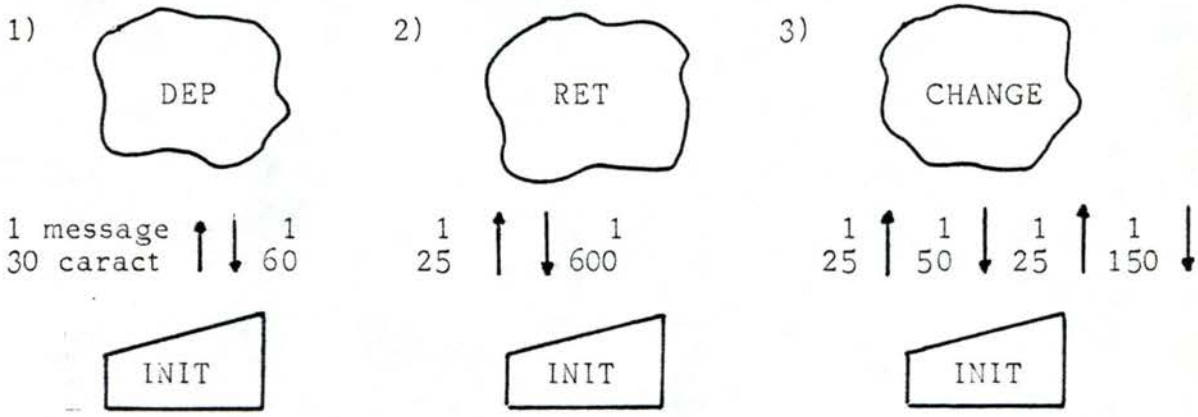
Exemple

Soit une application bancaire. Une analyse a déterminé 3 unités de travail, c'est-à-dire 3 types de transactions :

1. dépôt d'argent
2. retrait d'argent
3. change

L'analyse nous fait en même temps découvrir les processus utilisés pour traiter ces transactions.

Nous devons donc connaître les informations suivantes :



Celles-ci peuvent être visualisées par le tableau suivant :

TYPES DE TRANSACTIONS					
TYPE DE TRANSACTION	TYPE DE PROCESSUS UTILISE	TRANSMISSIONS			
		TYPE DE PROC ENVOYANT	TYPE DE PROC RECEVANT	NBRE DE MESSAGES	LONGUEUR (CARACT)
DEPOT	DEP	INIT	DEP	1	30
		DEP	INIT	1	60
RETRAIT	RET	INIT	RET	1	25
		RET	INIT	1	600
CHANGE	CHANGE	INTI	CHANGE	2	25
		CHANGE	INIT	1	50
				1	150

2.3.1.2. description des utilisateurs-terminaux

L'utilisateur décrit celui qui initialise les transactions. Il détermine les types d'agents vus par l'analyse fonctionnelle des applications, et par l'organisation de l'entreprise. Cette description des types d'agents n'est que quantitative.

Le travail fait par ces agents peut varier selon différentes heures, en termes de transactions initialisées. Ainsi, chaque type d'agents est caractérisé par son (ses) profil(s) d'initialisation de transactions.

Par profil, nous entendons l'ensemble des types de transactions que l'agent initialise durant une certaine période de la journée de travail. Le nombre nécessaire de profils peut être limité à trois : le premier profil étant relatif aux heures dites "normales", le deuxième et le troisième pour les heures dites "de pointe" et pour les événements saisonniers.

Pour chaque type d'agent et pour chaque profil utilisé, il faut déterminer les types de transactions initialisées, et indiquer pour chaque type la part qu'il représente dans l'ensemble des types de transactions initialisées.

Exemple

Pour une application bancaire, on a distingué 3 types d'agents :

1. le type "caissier"
2. le type "guichetier"
3. le type "bureau"

Le tableau suivant donne un exemple visualisant l'utilisation des profils : (les profils 2 et 3 sont optionnels)

TYPES D'AGENTS				
TYPE D'AGENT ID	TRANSACTIONS : PROFILS D'INITIALISATION			
	TRANSACTIONS	PROFIL 1 %	PROFIL 2 %	PROFIL 3 %
CAISSIER	DEPOT	40		10
	RETRAIT	50		70
	CHANGE	10		20
BUREAU	DEPOT	100		0
GUICHET	DEPOT	10		0
	RETRAIT	90		100

2.3.1.3. origine de ces informations

Toutes ces informations peuvent être trouvées dans le dossier d'une analyse conceptuelle approfondie. Ceci a pour conséquences que l'utilisateur n'a pas besoin de se familiariser avec un vocabulaire spécialisé, et qu'il ne doit pas refaire une analyse spéciale de ses applications.

2.3.2. Etude de la topologie du réseau

Par topologie du réseau, nous entendons l'ensemble des sites du réseau, ainsi que les liens qui les interconnectent. L'entreprise désirant un réseau pour faire tourner ses applications doit donc être vue comme un ensemble structuré de sites.

Chaque site est un centre de traitement situé dans un lieu géographique donné. Le nombre de sites peut être très élevé, ce qui alourdit l'étude de la topologie du réseau. Mais, comme la plupart des systèmes d'informatique distribuée ont une structure hiérarchique en forme d'arbre ou d'étoile, on peut introduire la notion de classe de sites, qui permet de réduire considérablement le volume des données à collecter.

2.3.2.1. classes de sites

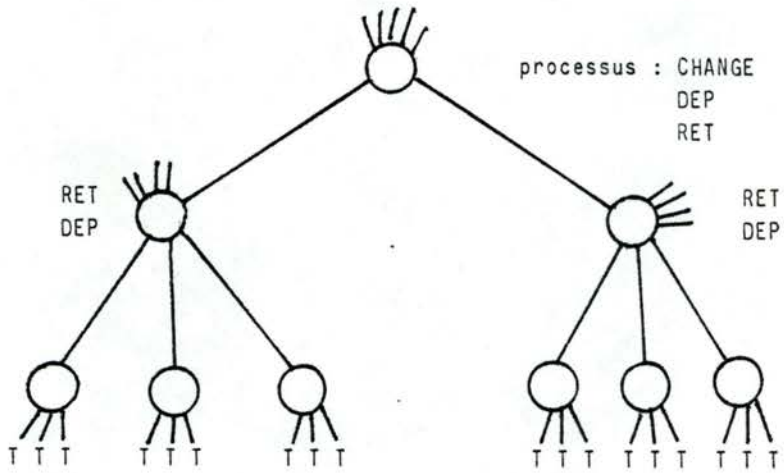
La notion de classe sera donc utilisée pour décrire un ensemble de sites qui sont identiques (excepté pour la localisation géographique). Tous les sites d'une même classe présentent :

- la même configuration hardware
- les mêmes processus
- les mêmes types d'agents en nombre, en intensité (qui est la moyenne de travail exprimée en transactions par heure, sans distinguer les différents types de transactions), le même protocole de connexion
- les mêmes liens avec d'autres sites (situés dans la même classe ou dans d'autres classes)

Il est donc évident que si une classe contient plusieurs éléments, les calculs et spécifications sont suffisants pour un élément de la classe. En fait, un site est équivalent à une classe contenant un élément. Dans le cas le plus défavorable d'utilisation de cet outil, chaque classe est formée par un seul élément.

Pour chaque classe, il faut déterminer le nombre de sites dans cette classe. Par classe de sites, il faut ensuite préciser la configuration hardware, c'est-à-dire les équipements. Par classe de sites, il faut déterminer les processus implantés. Les processus peuvent uniquement être implantés sur un ordinateur hôte.

Exemple de la répartition des processus dans le réseau :



A chaque profil utilisé correspond un problème, et donc une solution globale. C'est pourquoi il faut définir les classes de sites autant de fois qu'il y a de profils, car il est probable que l'activité aux sites change avec le profil.

2.3.2.2. relations inter-classes

Les processus responsables du traitement d'une transaction doivent communiquer entre eux. Cette communication va donner naissance à un trafic dans le réseau.

Les processus sont implantés sur des sites qui, dans la majorité des cas réels, sont arrangés dans une structure logique arborescente. Cette structure peut être différente de la topologie du réseau. Typiquement, tous les sites se trouvant à un même niveau d'une structure arborescente forment une classe.

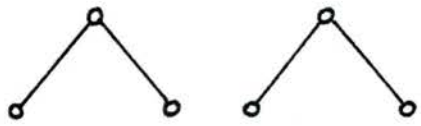
Nous analysons plus en détail les possibilités de modélisation de l'outil sur le type de réseau qui est le plus utilisé actuellement : le réseau organisé en forme d'arborescence. La structure d'arbre est alors décrite par une relation hiérarchique entre deux niveaux (classes), aussi allons nous parler d'un mode de transmission hiérarchique (H). (ces relations ou modes sont de nature logique)

Dans ce cas, il est suffisant d'analyser les configurations, le trafic, etc ... pour deux sites communiquant, chacun appartenant à une classe distincte.

Exemple :

classe 1 :

classe 2 :



Il existe un autre mode de transmission, à savoir <u>le mode équidistribué.</u>

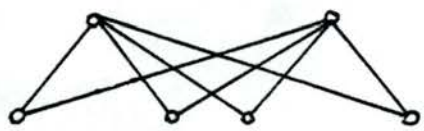
Ce mode définit une relation différente entre deux classes. Dans ce cas précis, n'importe quel site d'une classe peut communiquer avec n'importe quel site de l'autre classe.

L'analyse se limite de nouveau à un couple de sites.

Exemple :

classe 1 :

classe 2 :



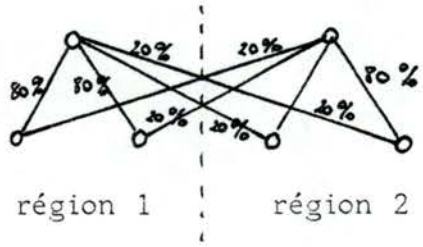
Il existe finalement un troisième mode de transmission, <u>le mode mixte</u>.

Ce mode est un mélange des deux modes précédents. Il est probablement plus commun que le mode équidistribué à 100 %. Normalement, la majorité des transmissions entre classes se font en mode hiérarchique, alors qu'un trafic équidistribué significatif subsiste.

Exemple : exemple bancaire :

site régional

agences



2.3.2.3. relations intra-classes

Dans une même classe, on peut avoir des transmissions. On peut distinguer deux modes :

Le mode identique définit une relation d'un site à lui-même, c'est-à-dire que le trafic provient du "réseau secondaire" attaché à ce site. En fait, l'activateur et le serveur de la transaction se trouvent sur le même site (physique).

Il faut aussi prendre en compte les relations qui existent entre les éléments d'une même classe. Ce mode est dit "mixte intra-classe", c'est-à-dire identique et équidistribué. De nouveau, ce mode semble être plus commun que le mode identique à 100 %.

2.3.2.4. exemple de relations

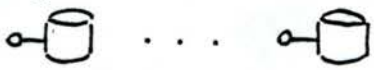
Cet exemple vise à visualiser les concepts exposés aux paragraphes précédents.

Prenons une organisation possédant des sites régionaux, supportant chacun une base de données "régionale", qui peuvent être interrogées par une classe de sites "agences", dispersés dans le pays et regroupés en régions.

La plupart du temps, le travail d'interrogation d'une base de données par une agence se fait au site "régional" auquel l'agence est rattachée. Toutefois, il est concevable que cette agence fasse aussi des interrogations sur les autres bases de données.

Schématiquement

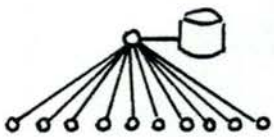
- classe des sites "régionaux" : 5 sites



- classe des sites "agences" : 50 sites rattachés par groupes de 10 aux sites régionaux

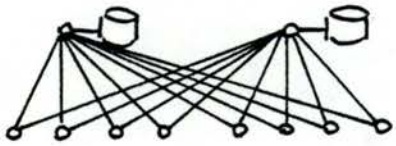
Le problème consiste à spécifier quelle agence consulte quelle base de données.

solution 1 : mode hiérarchique



chaque site "régional" est consulté par son groupe de 10 agences

solution 2 : mode équidistribué



n'importe quel site "agence" peut consulter n'importe quelle base de données "régionale"

solution 3 : mode mixte

analogue à la solution 2 sauf que la majorité des consultations des bases de données "régionales" se fait selon le mode hiérarchique.

2.3.2.5. origine de ces informations

Toutes ces informations peuvent être tirées de l'organigramme définissant la structure de base de l'entreprise et du dossier d'analyse fonctionnelle. La spécification de la configuration est cependant moins évidente, et une assistance de la part du fournisseur du matériel est souhaitable.

CHAPITRE 3 : CALCULS DE DIMENSIONNEMENT ENVISAGEABLES

3.1. POINT DE DEPART

Avant de commencer le dimensionnement proprement dit d'un réseau, nous devons collecter l'ensemble des données dont nous avons présenté la nature au chapitre précédent. La collecte des données peut être facilitée si elle est réalisée à l'aide de formulaires définissant clairement les besoins informationnels du dimensionnement. Cependant, le fait d'utiliser des formulaires est indépendant de l'utilisation d'un outil automatisé de dimensionnement.

Comme nous venons de le voir, les informations requises sont disponibles sans exception chez l'utilisateur du réseau, sous condition d'avoir procédé à une analyse conceptuelle et organisationnelle de ses applications.

Une première série de données servant de base aux calculs concerne les types de transactions, les types d'agents et les classes de sites. Cette notion est utilisée pour réduire le volume des données à saisir, mais elle est aussi utile lors des calculs. En effet, on n'effectue l'évaluation du trafic qu'entre classes de sites, et non pas entre sites. C'est uniquement pour la présentation des résultats que nous nous ramenons au niveau du site.

La deuxième série de données concerne des informations supplémentaires relatives à des résultats partiels générés à partir des données appartenant à la première série.

Pour un exemple, nous faisons référence à la base théorique et au manuel d'utilisation, qui sont en annexe.

3.2. TRAFIC POTENTIEL DANS LE SYSTEME DISTRIBUE

3.2.1. Localisation des transactions

Les données collectées ne donnent pas le trafic généré dans le réseau. Cependant, il est possible de synthétiser cette information à partir des informations dont nous disposons.

Le point de départ pour évaluer le trafic sont les types de transactions. Par type de transaction, nous connaissons les processus nécessaires au traitement. Ces processus sont localisés à des sites (classes de sites) précis. Par la définition des types d'agents, nous sommes en mesure de nommer les initialisateurs des diverses transactions. Par la définition des classes de sites, nous connaissons la localisation (logique si pas physique) de ces agents.

En résumé, ce que nous pouvons synthétiser à partir des données de base, c'est, pour chaque type de transaction, la localisation du processus initialisant, et celle du processus traitant. (cette localisation se fait au niveau des classes).

Le rôle de la phase ci-décrite est de découvrir tous les liens logiquement possibles entre les classes de sites, de sélectionner et de définir les liens requis par l'analyse fonctionnelle de l'application.

Quelques remarques utiles :

- La classe de site émetteur, c'est-à-dire le lieu du "processus d'initialisation de la transaction" doit évidemment supporter un agent dont le profil d'initialisation de transactions contient la transaction qu'on est en train de localiser.

- D'habitude, un site (une classe de sites) apparaîtra au moins deux fois dans la spécification de la transaction : une fois comme émetteur, et une fois comme récepteur.

Après avoir localisé la transaction, c'est-à-dire après avoir déterminé un lieu de traitement entre deux classes de sites, nous devons maintenant spécifier le mode de communication. Ceci est très important si la classe contient plusieurs sites.

Rappelons rapidement les modes possibles :

a) pour la communication inter-classes

- équidistribué (E)
- hiérarchique (H)
- mixte (c'est-à-dire un mélange de E et de H)

b) pour la communication intra-classe

- identique (I)
- équidistribué (E)
- mixte (c'est-à-dire un mélange de I et de E)

Pour chaque mode sélectionné, il faut indiquer la proportion. Une catégorie de données directement rattachée à celle de localisation est le nombre et la longueur des messages échangés.

Une fois le lien inter ou intra-classes sélectionné, on doit spécifier la nature du lien physique qu'on désire utiliser. En effet, lors de la localisation des transactions, on se situait à un niveau purement logique. Cette spécification devient nécessaire car, selon le lien choisi, le volume de caractères variera fortement, par exemple point-à-point par rapport à X25 public.

Le problème de passage du point de vue logique au point de vue physique entraînera quelques difficultés que nous allons exposer ci-dessous.

Pour la définition de la nature du lien, on peut envisager l'utilisation de réseaux de commutation par paquets publics (par exemple transpac ou datex-p ...), ou privés, qui présentent un attrait pour les grandes entreprises ; un autre type de lien envisageable est une liaison point-à-point.

3.2.2. Problématique des liens physiques

Pour limiter la discussion sur ce problème, nous supposons que l'utilisateur peut choisir entre trois liens physiques distincts :

- lien point-à-point
- lien via X25 public
- lien via X25 privé

L'utilisateur vient donc de localiser les transactions. Chacune d'elles est initialisée dans une classe de sites, et traitée dans une classe de sites (la même ou une autre). Cette localisation donne naissance à un trafic de données sur le réseau. Or le trafic spécifié est toujours un trafic logique qui doit être acheminé physiquement par un lien physique.

Le fait d'utiliser des concepts logiques entraîne un certain nombre de problèmes quant au dimensionnement du (des) lien(s) physique(s) existant entre un site appartenant à une classe et un autre site appartenant à la même classe ou à une autre. Ce problème se pose avant tout si par exemple des processeurs frontaux réalisent des fonctions de commutation (switching).

En passant en revue les trois types de liens physiques envisageables, on voit que le problème se situe au niveau du lien X25 privé. Pourquoi ?

Lorsque l'utilisateur choisit un lien point-à-point, il n'y a pas de commutation, car, par définition, une communication point-à-point est une communication entre un site émetteur et un site récepteur.

Dans le cas d'un réseau public à commutation par paquets, le problème est analogue. Ici, la commutation est inutile au niveau des sites de l'utilisateur, car elle est réalisée par le réseau public. En fait, une communication via un tel réseau est logiquement analogue à une communication du type point-à-point entre un site émetteur et un site récepteur.

Le problème de la commutation se pose pour l'utilisation d'un réseau privé à commutation par paquets, car, contrairement à ce qu'on a vu dans le cas précédent, ce sont les ordinateurs du réseau de l'utilisateur qui doivent assurer la commutation, ce qui entraîne une augmentation de leur charge de travail qui peut être importante.

Selon le mode de communication, le problème est plus ou moins complexe. Nous allons illustrer cette problématique à l'aide de quelques exemples :

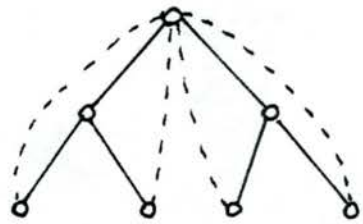
3.2.2.1. mode de communication entre classes de sites "H" ou hiérarchique à 100 %

Schéma :

classe A

classe B

classe C



L'utilisateur prévoit un mode de communication hiérarchique entre la classe C et la classe A, mais les messages sont commutés dans la classe B. Dans le cas du trafic hiérarchique entre sites, la commutation ne se fait pas dans la classe "récepteur".

La localisation des transactions doit tenir compte de cette possibilité, et la localisation initiale de la transaction de la classe C vers la classe A doit être découpée en une localisation de C vers B et de B vers A (évidemment avec des retours s'il y en a). Le seul problème est d'indiquer que les messages arrivant à la classe B vont nécessiter un traitement de commutation.

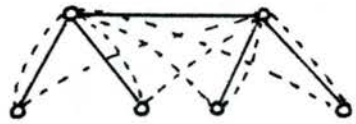
Les résultats qui vont être générés pour les liens physiques gardent toute leur valeur.

3.2.2.2. mode de communication entre classes de sites "E" ou équidistribué à 100 %

Exemple :

classe A

classe B

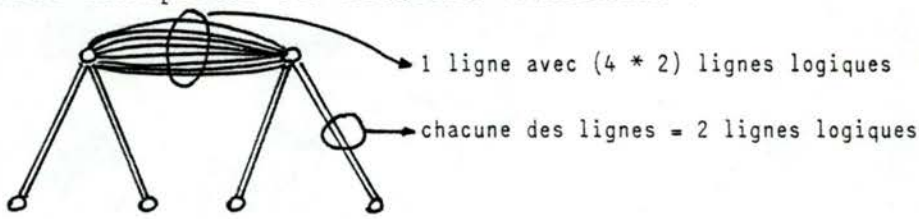


L'utilisateur a prévu une commutation au niveau de la classe A. La communication équidistribuée de B vers A doit être découpée en deux phases : communication de B vers A et de A vers A. Le problème qui se pose à ce niveau est celui d'indiquer la commutation au niveau de la classe A.

Par cette méthode de découpage, la charge de commutation sera prise en compte.

L'interprétation des résultats des liens physiques demande cependant une attention particulière : dans le cas de trafic équidistribué, la charge d'une ligne physique est égale à la charge d'une ligne logique. Or, dans le cas de la commutation, il faut interpréter ces résultats autrement :

Exemple :

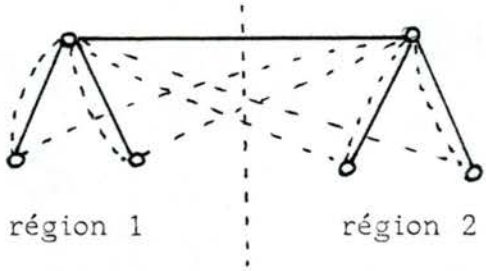


3.2.2.3. mode de communication "E" et "H"

Pour le mode "E" ou équidistribué, le problème est analogue au point précédent, mais seulement un certain pourcentage des messages échangés se fait par ce mode.

Pour le mode "H" ou hiérarchique, le problème reste le même qu'au point 3.2.2.1. Cependant il convient de remarquer que, si la classe de sites "récepteur" est aussi la classe de sites "commutateur", le trafic en mode H n'est pas commuté.

Exemple :



3.2.3. Evaluation de l'activité des classes de sites

Comme nous voulons estimer le trafic entre classes, il faut connaître l'activité globale d'une classe pour un type de transactions. Pour chaque site d'une classe, nous connaissons les types d'agents qui y sont localisés, leur nombre et leur activité en nombre de transactions (le type n'est pas important) par heure.

Le calcul à faire se résume à la formule suivante :

$$NA_{t,a} = p_{t,a} * i_a * n_a$$

où $NA_{t,a}$ = nombre d'activations de la transaction du type t (par heure) par le type d'agent a

$p_{t,a}$ = proportion d'activations de la transaction du type t notée dans le profil du type d'agent a

i_a = intensité de travail du type d'agent a sur le site

n_a = nombre d'agents du type a localisé au site

Pour un type de transaction, on additionne les divers $NA_{t,a}$ pour chaque type d'agent.

Cette somme est appelée activité du site pour la transaction concernée. En multipliant cette somme par le nombre de sites contenus dans la classe, on obtient l'activité de la classe pour la transaction sous étude.

Cette activité s'obtient donc en faisant :

$$N_t = \text{nbre de sites} * \sum_{i}^{n} NA_{t,a_i}$$

3.3. EVALUATION DU TRAFIC INTER-CLASSES

Pour chaque type de transactions initialisé par les différentes classes, il faut calculer les flux de données générés. Ceci sera réalisé en appliquant les fréquences d'activations de transactions aux informations concernées dans l'ensemble des informations relatives à la localisation des transactions.

A une localisation d'une transaction sont rattachés le nombre et la longueur des messages résultant du traitement.

On connaît aussi la nature du lien physique, ainsi il est possible de connaître le volume de données, non pas en nombre de messages, mais en nombre de fragments, ou de paquets ... selon le choix.

Cette conversion est assez simple et peut être résumée par les formules suivantes :

Pour le nombre de fragments :

$$F = N * \text{arrondi} \left(\frac{L + OF}{LF} \right)$$

où N = nombre de messages

L = longueur du message, en caractères

OF = en-têtes diverses, en caractères, dépendant des architectures choisies

LF = longueur du fragment, en caractères

Pour le nombre de paquets :

$$X = N * \text{arrondi} \left(\frac{L + OX}{LX} \right)$$

où N = nombre de messages

L = longueur du message, en caractères

OX = en-têtes diverses, en caractères, dépendant des architectures choisies

LX = longueur du paquet, en caractères

Pour une localisation de transaction entre classes, nous connaissons le mode et la proportion des transmissions qui se font dans ce mode. Nous connaissons également l'activité de la classe initialisant la transaction. Ainsi, nous pouvons calculer :

- le nombre de messages entre classe initialisant et classe recevant :

  nombre de messages = P * A * N

- le volume de caractères = P * A * N * L

- le nombre de fragments point-à-point = P * A * F
  ou
  le nombre de paquets = P * A * X

où P = proportion du mode utilisé
A = activité de la classe
N = nombre de messages
L = longueur du message, en caractères
F = nombre de fragments point-à-point
X = nombre de paquets X25

Si le mode de communication est identique, il n'y aura pas de trafic sur la partie primaire du réseau.

3.4. SYNTHESE DU TRAFIC INTER-CLASSES

Nous avons maintenant à notre disposition toutes les informations nécessaires pour dimensionner les chemins de données et les composants responsables du transport des données. Cependant, l'information n'est pas présentée sous une forme convenable.

Le but de cette étape est de regrouper toutes les informations relatives à un chemin de données intersite (concept logique).

Comme base de notre synthèse, nous prenons l'ensemble des informations relatives à la localisation des transactions. Pour chaque occurence d'un couple de classes (émettrice, réceptrice), il faut relever les transmissions par mode de communication, car le trafic sur un chemin de données "hiérarchique" est différent de celui sur un chemin de données "équidistribué". Par mode de communication nous faisons la somme des messages échangés, du volume de caractères, et la somme des paquets, fragments ou autre unité de transport.

3.4.1. Evaluation du trafic sur les chemins de données individuels

La synthèse du trafic intersite nous permet d'aboutir au flux de données en bits par seconde sur un chemin de données individuel entre deux sites appartenant à deux classes distinctes.

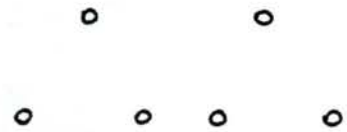
Il faut d'abord sommer par mode les résultats obtenus pour deux classes, où chaque classe apparaît comme émetteur et récepteur de messages. Afin d'évaluer correctement le trafic, il faut ajouter l'overhead résultant des protocoles de transmission. Ensuite il faut convertir les résultats obtenus en bits par seconde. Remarquons que ce trafic est le trafic moyen observé entre deux classes de sites.

Comment faut-il traiter ce trafic pour aboutir au flux de données entre deux sites appartenant à ces deux classes ?

Considérons deux classes de sites communiquant A et B :

classe B Nb = 2

classe A Na = 4

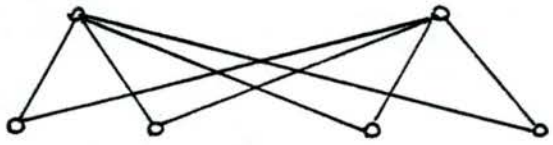


En général, on peut avoir Na * Nb chemins de données entre les sites de A et de B (= 8 chemins dans notre exemple).

- Si le mode de communication est hiérarchique par "région"

on a besoin de Na chemins de données individuels

- Si le mode de communication est équidistribué



on a besoin de Na * Nb chemins de données

Pour passer maintenant au trafic en bps par chemin de données individuel, il faut appliquer les formules suivantes :

- Si le mode est hiérarchique à 100 % :

$$\text{trafic sur chemin individuel} = \frac{\text{trafic inter-classes}}{\max(Na\ ;\ Nb)}$$

- Si le mode est équidistribué à 100 % intra-classe :

$$\text{trafic sur un chemin individuel} = \frac{\text{trafic intra-classe}}{\frac{Na\,(Na - 1)}{2}}$$

- Si le mode est équidistribué à 100 % inter-classes :

$$\text{trafic sur un chemin individuel} = \frac{\text{trafic inter-classes}}{Na * Nb}$$

- Si les modes E et H existent entre deux classes, il faut distinguer les chemins individuels où il y a du trafic uniquement en mode "E", et les chemins qui supportent le trafic en mode "H" et "E".

3.4.2. Passage du point de vue logique à la réalisation physique des liens

Dans l'état actuel, l'outil propose un dimensionnement physique des liens logiques, ce qui ne nous semble pas toujours réaliste. Cependant, la mise en oeuvre physique pose un certain nombre de problèmes qui sont loin d'être tous résolus, et qui demandent une étude plus approfondie.

Si le lien physique choisi est une ligne point-à-point ou une connexion via un réseau public de commutation par paquets (par exemple : transpac), les résultats fournis, soit :

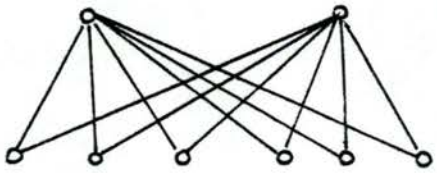
- volume des messages, de caractères
- vitesse de la ligne requise, nombre de lignes (en gardant un temps de service acceptable)
- taux d'utilisation, qui ne devrait pas dépasser les 60 à 65 %

sont valables.

Le problème, rappelons le, se pose si l'utilisateur veut construire son propre réseau de commutation par paquets.

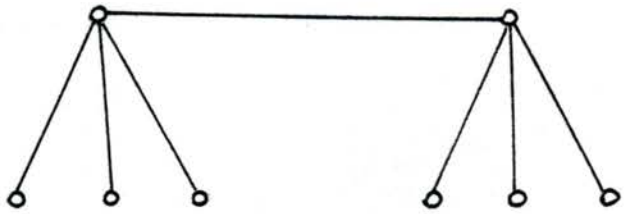
Une réalisation physique 1 à 1 des liens logiques entre deux classes semble peu réaliste.

Exemple :



liens logiques

La réalisation physique suivante semble plus raisonnable :



où les liens logiques "équidistribués" sont physiquement commutés par les voies hiérarchiques verticales et horizontales.

3.5. DIMENSIONNEMENT DES EQUIPEMENTS DE TRANSPORT

Au chapitre 2, nous avons modélisé et formalisé le comportement des équipements de transport des données. Ces formules permettent de prédire la performance de ces machines en face d'une charge de travail que nous venons d'évaluer aux paragraphes précédents.

En effet, en connaissant le trafic sur un lien intersite, on connaît le volume de caractères, messages, paquets, ... etc que l'équipement doit traiter. En sommant tout le trafic, on peut estimer si l'équipement (par exemple un processeur frontal) est suffisant pour faire face à la charge de travail. Le critère de cette estimation peut être la charge CPU de l'équipement...

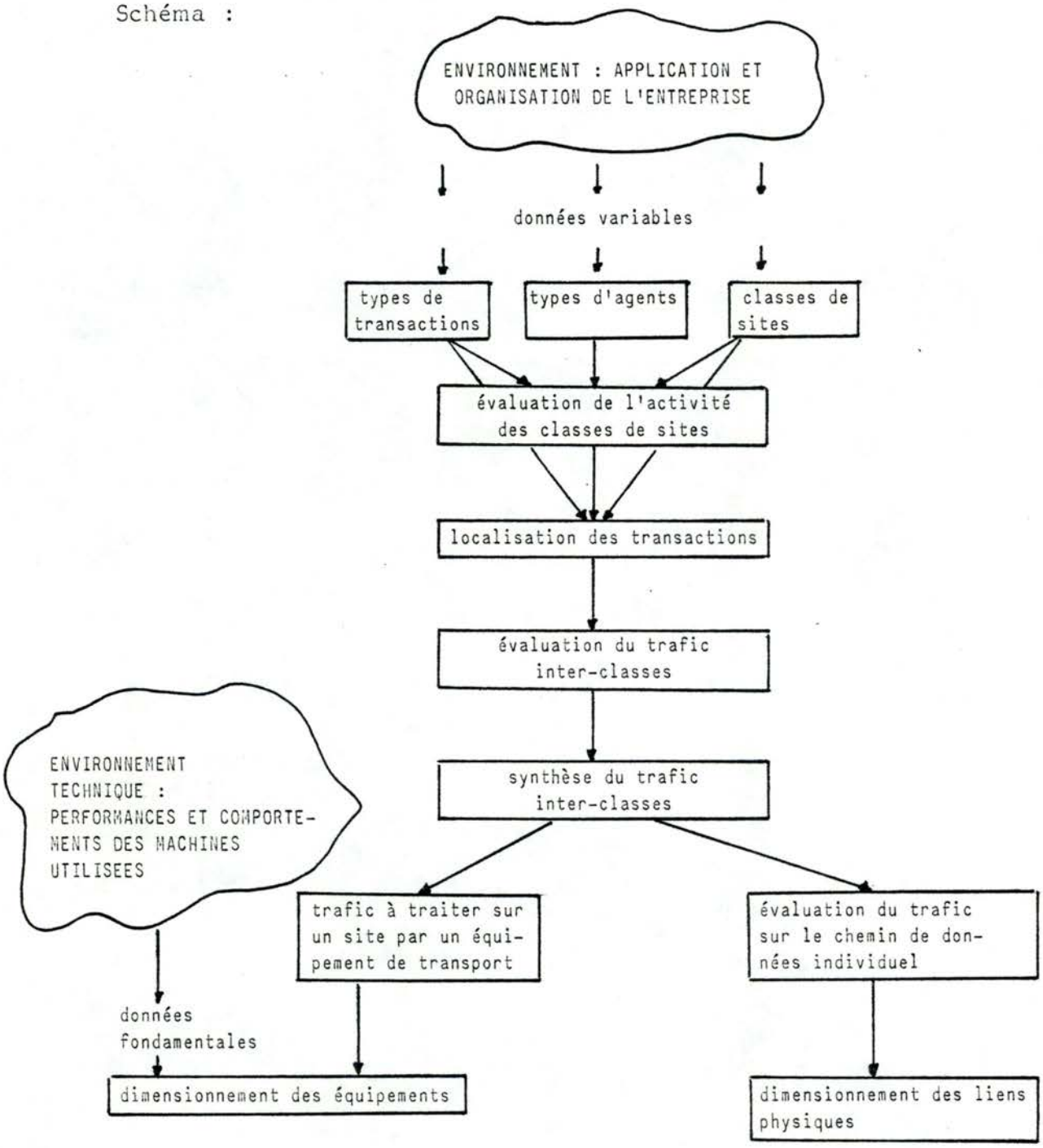
Cette charge CPU peut être détaillée par protocole supporté, par mode de fonctionnement, par réseau, etc... La décision sera du type : Si la charge CPU est inférieure ou égale à 90 % alors OK, sinon installer un équipement plus performant ou installer une unité supplémentaire du même type et répartir le travail sur ces deux machines.

Pour que la procédure du dimensionnement soit simpliste, le développement des modèles et formules exposé ci-dessus est loin d'être évident.

3.6. RESUME SCHEMATIQUE

Nous résumons schématiquement le principe de l'outil.

Schéma :



CHAPITRE 4 : AUTOMATISATION DE L'OUTIL

4.1. INTRODUCTION

Comme nous venons de le voir, l'outil présenté demande un volume considérable de manipulations et de traitements de données.

Une utilisation manuelle de l'outil est envisageable si l'utilisateur se donne des formulaires ou autres supports adéquats. Ainsi, toutes les données à collecter sont clairement spécifiées. Cependant, lors des manipulations, et principalement lors de la localisation des transactions, les risques d'erreurs sont très élevés.

Pour éviter ces problèmes, nous avons développé un outil automatisé, qui guide l'utilisateur d'une étape à l'autre, en assurant l'exploitation de toutes les combinaisons logiquement possibles. Ce type d'outil, que nous décrivons brièvement dans la suite, n'est pas uniquement un outil de calcul, mais il est aussi un instrument induisant des effets psychologiques auprès des utilisateurs, ainsi s'explique l'intérêt que les constructeurs ont pour de tels outils.

Il convient de remarquer que notre système automatisé peut être utilisé par n'importe quel constructeur d'équipements informatiques, après changement d'un certain nombre de détails. Ces détails se limitent aux procédures de calculs du temps CPU des équipements.

4.2. DESCRIPTION SUCCINCTE DE L'OUTIL

4.2.1. Fonctions de base

Les spécifications fonctionnelles du système automatisé ont été brièvement décrites au chapitre 3 de cette partie.

Les premières tâches de l'outil sont les collectes de données relatives aux notions de :

- types de transactions
- types d'agents
- classes de sites

La localisation des transactions, qui est l'étape la plus importante de tout l'outil, est entièrement automatisée, c'est-à-dire que tous les liens logiquement possibles entre classes de sites sont proposés à l'utilisateur, qui va retenir uniquement ceux requis par les applications.

Les phases de calculs, telles que l'évaluation de l'activité d'une classe, la synthèse du trafic inter-classes, ... etc se font essentiellement avec un minimum d'interaction avec l'utilisateur.

4.2.2. Fonctions "utilisateur"

L'outil automatisé doit fournir un certain nombre de services à l'utilisateur, sans lesquels l'utilisation du système est trop lourde et compliquée.

Tout au long des collectes de données, l'utilisateur est guidé dans son travail. L'outil automatisé est du type "self-explaining", c'est-à-dire qu'il affiche du commentaire et fournit des explications de façon automatique ou sur demande (par "?"). D'autres fonctions, comme par exemple le contrôle de validité des zones d'entrée sont assurées, et facilitent l'usage.

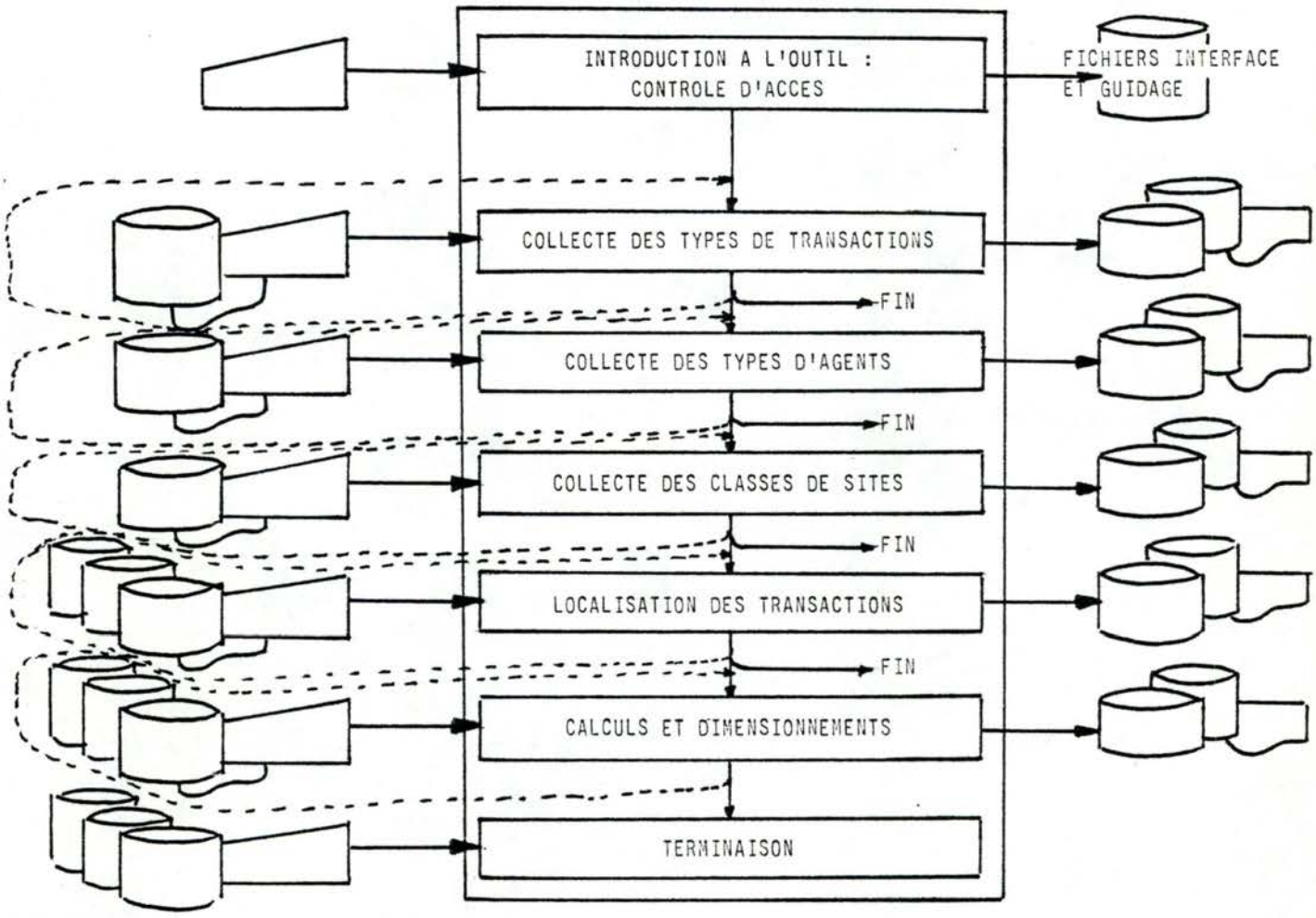
Les données collectées et les résultats générés constituent ce que nous avons appelé le "modèle" du système d'informatique distribuée. Les données qui sont stockées sur disques peuvent être modifiées, ainsi l'utilisateur peut simuler des hypothèses différentes sur son réseau.

Un dispositif de sécurité a été développé, qui protège, d'une part, le modèle de l'utilisateur de tout abus, et qui, d'autre part, assure la cohérence des données collectées, en refusant l'accès multiple au même modèle.

Le modèle dont nous présentons la structure au paragraphe suivant permet à l'utilisateur :

- d'interrompre son travail après chaque étape importante
- de refaire la même étape
- de continuer

4.2.3. Structure du système automatisé



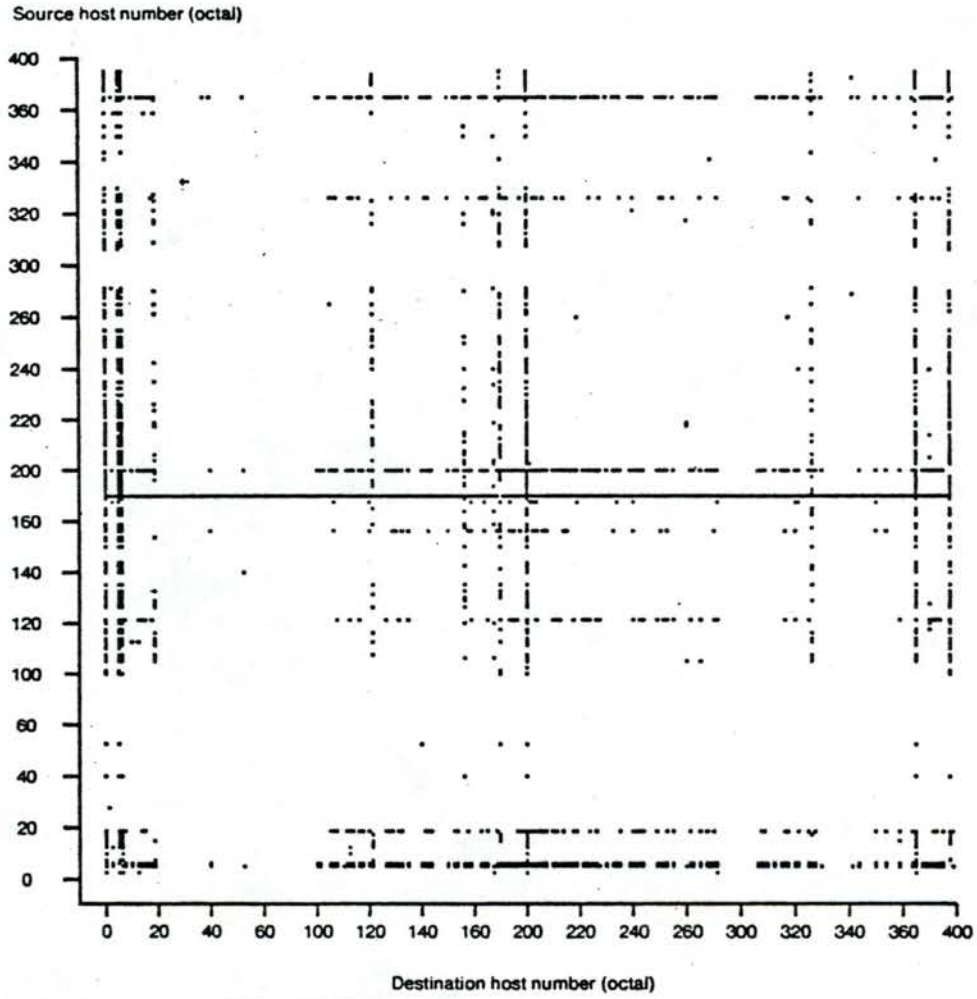
Pour donner un ordre de grandeur de la taille du système : nous avons développé un système de plus ou moins 7000 lignes d'instructions en pascal. L'outil s'exécute en interactif à partir d'un terminal à imprimante.

4.2.4. Problème des sorties

Après chaque étape, l'outil sort des rapports, sous forme de tableaux (pour les types de transactions, les types d'agents, les classes de sites, et la localisation des transactions), et sous forme de simple liste (dimensionnement des liens physiques et des équipements).

Malheureusement, le temps limité dont nous disposions ne nous a pas permis de développer des fonctions de présentation graphique.

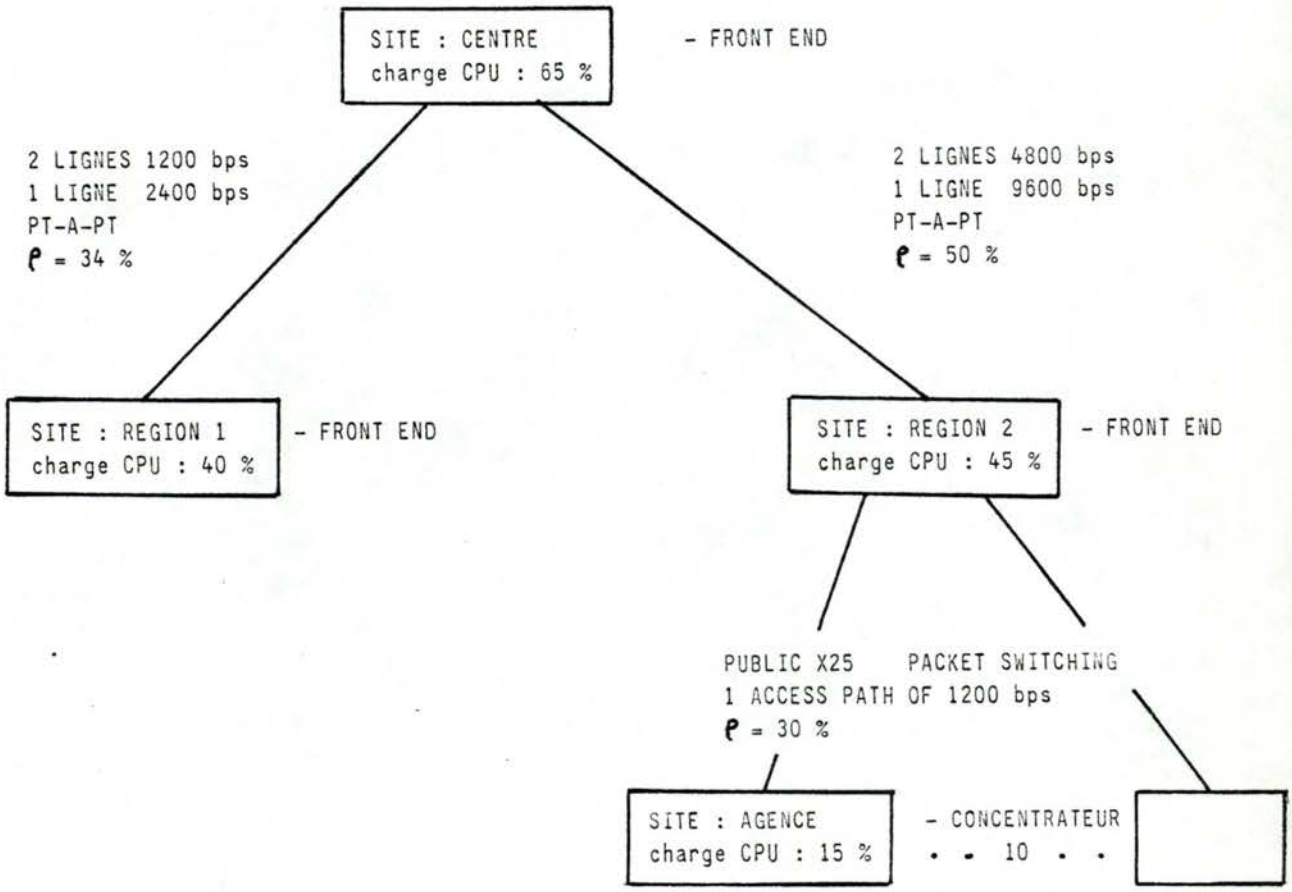
Une première présentation des résultats à laquelle nous avons pensé, c'est la présentation du trafic intersite sous forme matricielle analogue à la "source-destination traffic matrix" utilisée par les mesureurs de performances du réseau local "Ethernet" (S4).



Source-Destination Traffic Matrix.

Un deuxième graphique que l'outil pourrait générer est une présentation schématique du réseau avec les sites, les liens, et avec un minimum de résultats.

Exemple :



CHAPITRE 5 : PROCEDE ALTERNATIF DE MODELISATION DE RESEAUX

5.1. INTRODUCTION

Nous allons brièvement décrire une méthode de modélisation et de dimensionnement de réseaux que nous avons développée après avoir pris une certaine distance vis-à-vis de la première méthode.

La méthode présentée ci-dessous poursuit les mêmes buts, mais elle se base en partie sur des principes distincts, le principe fondamental étant une description de la topologie du réseau en fonction des applications qui tournent sur ce réseau. Cette description suivra les collectes des données relatives aux types de transactions, types d'agents et stations. Dans cette méthode, on ne parlera pas de sites ni de classes de sites, mais de stations.

Alors que la première méthode, et en particulier la localisation des transactions, se base sur l'aspect logique du réseau, cette deuxième méthode met l'accent sur la modélisation de la structure d'interconnexion (appelé par (L1) : shape) des diverses stations. Donc cette méthode met l'accent sur l'aspect physique du réseau.

Cette méthode devrait permettre une automatisation plus aisée, où le volume des calculs à effectuer est nettement moins élevé, et la nature des résultats plus réaliste que dans l'outil développé à partir de la première méthode. Ainsi allons-nous présenter cette méthode alternative, en faisant directement référence à une mise en oeuvre automatisée possible.

5.2. DESCRIPTION DE LA METHODE

5.2.1. Collecte des données

Comme la première méthode, la base du dimensionnement est constituée par les données "applications", c'est-à-dire les types de transactions et les types d'agents. Ensuite, les données relatives aux stations sont collectées.

Par station, nous entendons un équipement de traitement de données, de transport de données, et éventuellement un équipement de stockage des données.

Donc une station pourrait être par exemple un ordinateur hôte, un concentrateur de terminaux, un commutateur, un terminal, un processeur frontal, tandis qu'un site est un ensemble de machines, hommes et programmes (processus).

Avec cette méthode, on n'aura pas besoin de définir des classes de sites. Cependant, dans une optique de limitation des données à collecter, pour des stations identiques l'utilisateur ne spécifiera qu'une seule fois les caractéristiques de ces stations. Cette remarque est importante dans la mesure où on peut trouver beaucoup de stations du même type dans un réseau : exemple : si on considère les stations "processeur frontal" ou "concentrateur".

C'est seulement au niveau de la localisation des transactions que la collecte se fait par paires de stations. Cependant, comme le volume des données à fournir sera très faible, cette méthode reste applicable.

5.2.2. La topologie du système distribué, la matrice "émetteur-récepteur"

Dans la méthode précédente, la description de la topologie était purement logique, et elle était réalisée à l'aide de relations hiérarchiques, équidistribuées ou mixtes.

Ici, nous allons procéder d'une manière différente, à savoir que nous allons décrire la topologie physique, la structure d'interconnexion des stations. Cette description est basée sur l'utilisation d'une "matrice étendue du trafic émetteur-récepteur".

En effet, dans un réseau téléinformatique, à priori, les stations peuvent émettre des messages vers d'autres stations, ou recevoir des messages d'autres stations. Une station peut être uniquement émetteur ou uniquement récepteur. Il existe donc au maximum des échanges bidirectionnels de messages entre les stations. Ce phénomène peut être représenté par une matrice.

5.2.2.1. étude des éléments de la matrice

Nous allons décrire les différents éléments de la matrice, indépendamment de ce que les éléments contiennent, article ou pointeur vers des articles.

a) éléments diagonaux

Tous les éléments (i,i) de la matrice renferment les informations relatives aux stations, c'est-à-dire leurs caractéristiques, et éventuellement le volume et la nature des données à "traiter" (traitement au sens large). La définition des stations réalisée à l'étape précédente définit également la taille de la matrice.

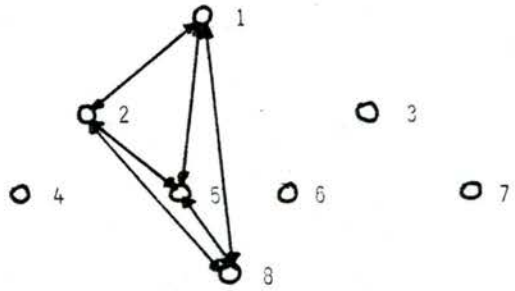
b) autres éléments

Par un procédé analogue à celui de la première méthode, on relève, par transaction, les stations à partir desquelles on peut l'initialiser. On demande ensuite à l'utilisateur de spécifier les destinations des transmissions résultant du traitement de cette transaction. Evidemment, les contrôles de cohérence évoqués dans la première méthode restent valables. Une autre possibilité, qui est d'ailleurs la meilleure, est de générer automatiquement les liens entre stations possibles. Dans ce cas, l'utilisateur choisit les interconnexions souhaitées, et il spécifie le pourcentage des données qui vont transiter par ces liens.

Le volume de données transmis de i vers j, en messages, en caractères, et en bits ("overheads" compris), le type de lien choisi, seront mémorisés à l'élément (i,j) de la matrice, tandis que les informations sur les données émises à la station i sont mémorisées à l'élément (i,i) et les données reçues à la station j à l'élément (j,j), tout en mémorisant le type de traitement (traitement au sens strict, transport, (commutation), stockage).

5.2.2.2. exemple

Un utilisateur a défini 8 stations distinctes, et il souhaite localiser un type de transaction.



Ce type de transaction ne peut être initialisé qu'à la station 8. Ce type de transaction peut être traité sur la station 1, 2 et 5. Les transmissions peuvent être commutées dans le cas de présence d'une fonction de commutation. Supposons la présence d'une telle fonction aux stations 2 et 5.

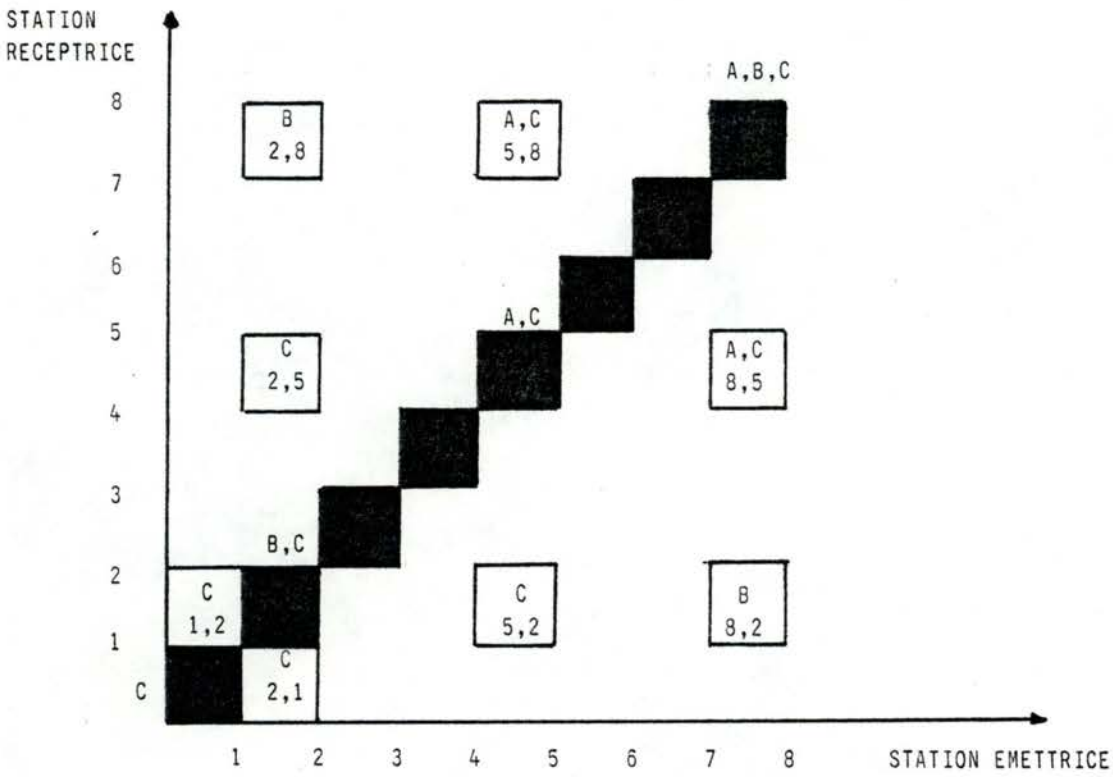
Les liens possibles sont :

(si on suppose pour ce type de transaction un échange de 1 message dans les deux sens)

station initialisant		station traitant	éléments de la matrice à remplir
(A)	8	5	(5,5) (8,8) (8,5) et (5,8)
(B)	8	2	(8,8) (2,2) (8,2) et (2,8) lien direct
	8	2	(8,8) (5,5) (2,2) (8,5) (5,2) et (2,5) (5,8) lien indirect
	8	1	(8,8) (1,1) (8,1) et (1,8) lien direct
	8	1	(8,8) (1,1) (5,5) (8,5) (5,1) et (1,5) (5,8) lien indirect
	8	1	(8,8) (1,1) (2,2) (8,2) (2,1) et (1,2),(2,8) lien indirect
(C)	8	1	(8,8) (5,5) (2,2) (1,1) (8,5) (5,2) (2,1) et (1,2) (2,5) (5,8) lien indirect

Chaque fois qu'un lien (i,j) est retenu, les informations sont enregistrées. Evidemment si le traitement de la transaction donne naissance à des messages de retour, les liens (j,i) sont automatiquement garnis.

Les éléments à garnir sont donc les suivants si on retient les liens A, B et C :



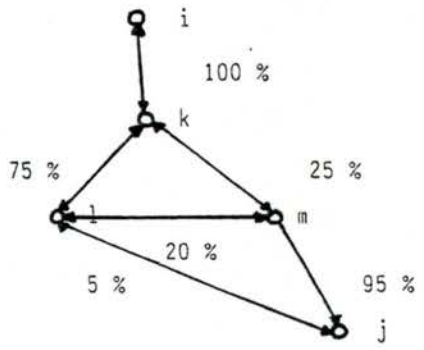
5.2.2.3. remarque

Cette localisation des transactions est beaucoup plus proche de la réalité, c'est-à-dire de la mise en oeuvre du réseau, car elle permet de définir rigoureusement la structure des interconnextions.

Chaque localisation est donc déterminée par une station émettrice (i), et par une station réceptrice (j). Le chemin inter-stations (i,j) peut être physiquement réalisé de plusieurs manières : (i,j) direct, point-à-point, X25 privé, X25 public, ou (i,j) ⟹ (i,k) (k,l) (l,m) (m,j) où on peut donner le pourcentage des transactions initialisées qui se font par ces chemins partiels.

Exemple : soit le chemin (i,j) qui est réalisé par la structure d'interconnexion suivante : (i,k) (k,l) (k,m) (l,m) (l,j) (m,j)

Graphiquement :



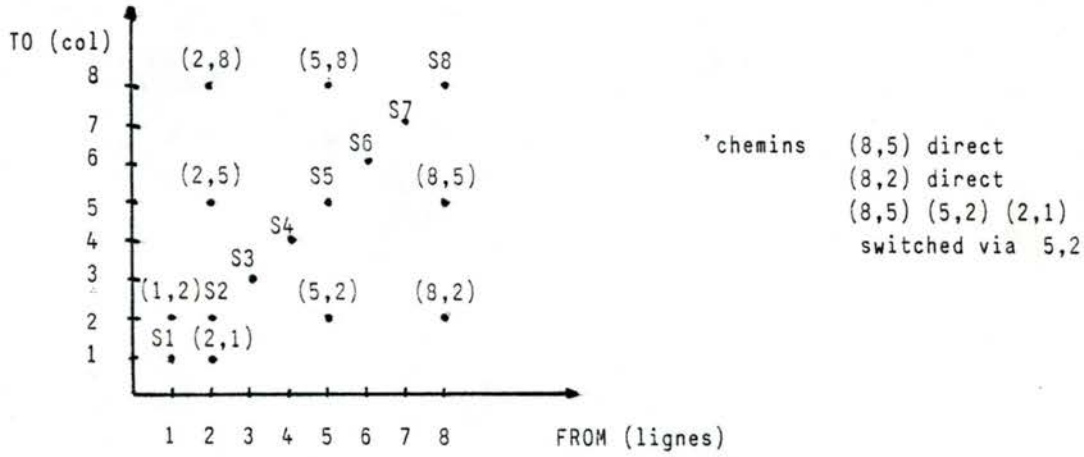
5.2.3. Dimensionnement

5.2.3.1. dimensionnement des liens

Vu que l'utilisateur a complètement spécifié la structure d'interconnexion de son réseau, le dimensionnement entre deux stations quelconques est immédiat : on connaît la nature du lien physique, on connaît les types de transactions et les volumes de données en caractères, bits, messages par unité de temps et par sens de transmission (i,j) et (j,i). Donc le dimensionnement ne pose pas de problèmes.

5.2.3.2. dimensionnement des stations de transport des données

Ce problème est aussi simplifié par la discrimination des informations relatives au trafic inter-stations. En effet, ayant dimensionné le lien entre i et tous les autres sites, on peut évaluer par exemple la charge CPU de cet équipement, grâce à la charge de travail connue.



Si les transactions se font par messages dans les deux sens, la matrice est symétrique par rapport à la diagonale.

5.3. APPRECIATION - CONCLUSION

La méthode que nous venons de décrire dans ce chapitre mérite d'être approfondie et d'être discutée. Elle permet, par ses possibilités, de modéliser le réseau, de générer des résultats correspondant à des réalités.

Cette méthode est facilement automatisable, et, vu sa base, elle permet l'utilisation de certains algorithmes de la théorie des graphes. Ainsi, on pourrait aboutir à l'élaboration d'un outil plus précis et plus extensible que celui issu de la première méthode. Finalement, cette deuxième méthode est capable de traiter le problème des systèmes d'informatique distribuée "general purpose", alors que la première méthode semble plus apte à traiter les problèmes des systèmes d'informatique distribuée présentant une architecture hiérarchique.

CONCLUSION

Dans ce mémoire, notre but était de contribuer à la conception de systèmes d'informatique distribuée. En fait, cette contribution a consisté en l'étude et l'automatisation d'un outil pragmatique d'aide à la conception développé chez CII-HB, outil destiné à être utilisé lors de la phase d'avant-vente de matériel informatique.

Parallèlement à notre travail pratique, qui consistait donc essentiellement en l'étude de l'outil pragmatique et la programmation d'un prototype de système automatisé, nous avons fait des études complémentaires dans un but de nous rendre compte de la complexité des problèmes à considérer lors des conceptions de systèmes d'informatique distribuée (SID).

Ainsi, nous nous sommes efforcés dans une première partie de retracer l'évolution des SID, d'en donner des exemples typiques, de proposer une classification, d'étudier brièvement les problèmes de la distribution des fonctions de traitement, de stockage et de mouvement des données dans un système distribué.

La deuxième partie nous a permis de passer en revue un certain nombre d'architectures de réseaux proposées par des constructeurs de matériel informatique. Ceci nous a permis d'abord de voir les efforts menés dans ce domaine encore jeune de l'informatique, et ensuite d'entrevoir l'enjeu commercial lié à ces développements.

Dans la troisième partie, nous avons résumé les études que nous avons faites sur les méthodes de conception de SID. Cette partie a été particulièrement difficile, dans la mesure où on peut trouver beaucoup de méthodes. Le point de vue à adopter dans cette matière est d'adapter une méthode de conception aux besoins.

La quatrième partie est consacrée à l'étude d'un outil de conception de SID que l'on pourrait utiliser dans l'une ou l'autre méthode présentée dans la partie 3. Nos efforts se sont portés sur l'étude de l'outil développé par Mademoiselle Claudie Chappuis de CII-Honeywell Bull. Cet outil, nous semble-t-il, est adapté aux besoins d'une conception de réseaux complexes, dans la mesure où, à l'aide d'un minimum de données et de calculs dont la complexité n'est pas trop importante, le concepteur dispose de résultats de dimensionnement globaux. L'application de cet outil en phase d'avant-vente justifie l'acceptation de ces résultats approchés.

Nous avons découvert un certain nombre d'inconvénients à la méthode, notamment le problème de la commutation, qui méritait une étude approfondie. Nous avons présenté une alternative à ce problème, dont il faudrait discuter l'utilité et l'intérêt.

Nous nous sommes également efforcés d'étudier les méthodes à l'aide desquelles on peut arriver à déterminer les performances des équipements face à une charge de travail. En effet, ces connaissances sont d'une importance majeure dans la mesure où elles permettent de dimensionner les noeuds.

Notre travail pratique a débouché sur la confection de programmes écrits en pascal. Cet outil répond aux spécifications de celui de Mademoiselle Claudie Chappuis que nous avons mis à jour. Lors de ce développement des programmes, nous nous sommes rendus compte des problèmes existant lors de l'élaboration de programmes destinés à être utilisés par des profanes. Finalement, nous estimons que l'outil pourrait être amélioré en utilisant des fonctions spécialisées de gestion d'écran.

Pour terminer, l'étude de ce mémoire nous a apporté les bénéfices suivants :

Sur un plan théorique :

- connaissances sur les SID
- compléments à des cours que nous avons eus
- connaissance de méthodes utilisées dans la vie pratique

Sur le plan pratique :

- confrontation avec la vie professionnelle
- travail de programmation

BIBLIOGRAPHIE

BIBLIOGRAPHIE

[A 1] : ANDERSON, G.A. ; JENSEN, E.D.

Computer interconnection, structures : taxonomy, characteristics and examples

ACM , NEW YORK (USA)
COMPUTING SURVEYS , 1975 , december , vol. 7 , no. 4
pp 197-213

[A 2] : AMERICAN NATIONAL STANDARDS INSTITUTE

Data processing - Open sytems interconnection - basic reference model : ISO/TC97/SC16 (doc.: 537 revised, circulating as DP 7498)

NORTH - HOLLAND PUBLISHING COMPANY , AMSTERDAM (NL)
COMPUTER NETWORKS , 1981 , no. 5 , pp 81-118

[B 1] : BOOTH, Grayce M. (HIS)

The distributed system environment - some practical approaches

McGRAW - HILL BOOK COMPANY , NEW YORK (USA)
1981 , ref.: ISBN 0-07-006507-1

[B 2] : BOCHMANN, Gregor von

Architecture of distributed computer systems

SPRINGER VERLAG , BERLIN (GFR)
LECTURE NOTES IN COMPUTER SCIENCE , 1979 , no. 77
ref.: ISBN 0-387-09723-6

[B 3] : BARDANE, Pierre

La régression un outil de prévision

01INFORMATIQUE , PARIS (FRANCE)
01INFORMATIQUE MENSUEL , 1982 , février , no.157
pp 55-57

BIBLIOGRAPHIE

[C 1] : CII - HONEYWELL BULL

Transmission des donnèes DATANET 7100
- introduction au système -

CII - HONEYWELL BULL , PARIS (FRANCE)
1980 , juin
rèf.: 15 F2 8026 REVO

[C 2] : CII - HONEYWELL BULL

DSA 200 - Performances measurement plan

CII - HONEYWELL BULL , PARIS (FRANCE)
internal use , 1981 , december

[C 3] : CII - HONEYWELL BULL

Architecture DSA

CII - HONEYWELL BULL , PARIS (FRANCE)
SYSTEMES D'INFORMATIQUE , 1981 , hiver , no. 34
rèf.: 01 F8 0084

[C 4] : CII - HONEYWELL BULL

Distributed systems architecture - summary description

CII - HONEYWELL BULL , PARIS (FRANCE)
ref.: 15 A8 8426

[C 5] : CII - HONEYWELL BULL

Les systèmes d'informatique distribuèe - DSE
- sèrie 60

CII - HONEYWELL BULL , PARIS (FRANCE)
rèf.: 00 F8 7350

[C 6] : COUPAYE, Jean (CII - HB)

Mesures CPU - DATANET release B V 2.5 (octobre 1981)

CII - HONEYWELL BULL , PARIS (FRANCE)
utilisation interne , 1981 , dècembre

[C 7] : CII - HONEYWELL BULL

DSE - Les systèmes d'informatique distribuée

CII - HONEYWELL BULL , PARIS (FRANCE)
réf.: 15.F8 - 8425

[C 8] : CII - HONEYWELL BULL

Mini 6 / DSS

CII - HONEYWELL BULL , PARIS (FRANCE)
réf.: 69.A8 - 8418

[C 9] : CII - HONEYWELL BULL

Datanet 7100 - summary description

CII - HONEYWELL BULL , PARIS (FRANCE)
ref.: 15.A8 - 8377

[C 10] : CII - HONEYWELL BULL

DSA - Distributed systems architecture

CII - HONEYWELL BULL , PARIS (FRANCE)
ref.: 15.A8 - 8001

[C 11] : CORNAFION (groupe d'auteurs)

Systèmes informatiques répartis - concepts et techniques

BORDAS , PARIS (FRANCE)
DUNOD - PHASE SPECIALITE INFORMATIQUE , 1981
réf.: ISBN 2-04-012135-8

[C 12] : CHAPPUIS, Claudie

Sizing a DSA network

CII - HONEYWELL BULL , PARIS (FRANCE)
SALES AID , internal use , 1980 , october , no. 80-034

[C 13] : CASTELLANI, Xavier

Méthode générale d'analyse d'une application informatique
- tome 1 : étapes et points fondamentaux de l'analyse de conception

MASSON , PARIS (FRANCE)
1978
réf.: ISBN 2-225-70576-3

[C 14] : CHAPPUIS, Claudie

DSA - network performances support
- market requirements -

CII - HONEYWELL BULL , PARIS (FRANCE)
internal use , 1981 , august

[C 15] : CII - HONEYWELL BULL

DSA : une architecture de réseau au service de l'usager

CII - HONEYWELL BULL , PARIS (FRANCE)
SYSTEMES D'INFORMATIQUE , 1979 , automne , no. 31

[D 1] : DALLE, François

Informatique et décentralisation des organisations
- actes du colloque international informatique et société -

INFORMATISATION ET SOCIETE - SERIE IMPACT , vol. 1
réf.: ISBN 2-11-000473-8

[D 2] : DATAPRO RESEARCH CORPORATION

DEC - Digital network architecture (DNA) and DECnet

NETWORKS AND ARCHITECTURE , 1981 , may
ref.: C 11-384-101

BIBLIOGRAPHIE

[E 1] : EDP - ANALYZER

Tools for building distributed systems

CANNING PUBLICATIONS , VISTA (USA)
EDP - ANALYZER , 1980 , october , vol. 18 , no. 10

[E 2] : EDP - ANALYZER

Application systems design aids

CANNING PUBLICATIONS , VISTA (USA)
EDP - ANALYZER , 1981 , october , vol. 19 , no. 10

[G 1] : GALLO, Arpad ; WILDER , Richard P. (HIS)

Performance measurement of data communications systems with emphasis on open system interconnections (OSI)

COMPUTER SOCIETY PRESS , NEW YORK (USA)
SIGARCH NEWSLETTER , 1981 , vol. 9 , no. 3 , pp 149-161

[L 1] : LORIN, Harold (IBM)

Aspects of distributed computer systems

JOHN WILEY & SONS , NEW YORK (USA)
WILEY - INTERSCIENCE PUBLICATION , 1980
ref.: ISBN 0-471-08114-0

[L 2] : LUSSATO, Bruno

Le dèfi informatique

FAYARD, PARIS (FRANCE)
1981
rèf.: ISBN 2-213-01005-6

[M 1] : MACCHI, Cèsar ; GUILBERT, Jean-François

Tèlèinformatique

BORDAS , PARIS (FRANCE)
DUNOD - PHASE SPECIALITE INFORMATIQUE , 1979
rèf.: ISBN 2-04-010361-9

[M 2] : McFAYDEN,J.H. (IBM)

SNA - Systems network architecture : an overview

IBM , NEW YORK (USA)
SYSTEMS JOURNAL 1976 , vol. 15 , no. 1 , pp 4-23

[M 3] : MARTIN, James

Systems analysis for data transmission

PRENTICE - HALL
SERIES IN AUTOMATIC COMPUTATION , 1972
ref.: ISBN 0-13-881300

[S 1] : SCHAFLITZL, Helmuth (SIEMENS)

Monitore zur Leistungsmessungen von DV-Systemen

SIEMENS , MUENCHEN (BRD)
DATA REPORT , 1977 , vol. 12 , no. 4 , pp 12-16
Ref .: ISSN 0374-289 X

[S 3] SPIEGEL, Murray R.

Thèorie et applications de la statistique

McGRAW - HILL BOOK COMPANY , NEW YORK (USA)
SERIE SCHAUM , 1980

[S 4] : SHOCH, John F. ; HUPP, Jon A. (XEROX)
Measured performance of an ETHERNET local network

ACM , NEW YORK (USA)
COMMUNICATIONS OF THE ACM , 1980 , december
vol. 23 , no. 12

[S 5] : SUP'AERO , Ecole nationale supèrieure de l'aèronautique et de l'espace

Mesure de systèmes informatiques à l'usage des ingènieurs système

CEPADUES EDITIONS , TOULOUSE (FRANCE)
1978
rèf.: ISBN 2-85428.036.9 ISSN 0337.6192

[S 6] : SINNOTT, M.R. ; BYTHEWAY, A.J.

What is going on inside the machine ?
(The effectiveness of hardware monitoring as a measurement tool)

ONLINE
COMPUTER PERFORMANCE EVALUATION , 1976
réf: ISBN 0-903796-14-7

[T 1] : TOBAGI, Fouad A. ; GERLA, Mario ; PEEBLES, Richard W. ; MANNING, Eric G.

Modeling and measurement techniques in packet communication networks

IEEE , NEW YORK (USA)
PROCEEDINGS OF THE IEEE , 1978 , november , vol. 66 , no. 11

[T 2] : TIMMONS, Michael L. (SPERRY UNIVAC)

Distributed communication architecture forms framework for network design

COMPUTER DESIGN PUBLISHING COMPANY , TULSA (USA)
COMPUTER DESIGN , 1981 , february , pp 121-125

[T 3] : TERPLAN, Kornel

Measuring and improving the performance of teleprocessing systems

ECOMA - 8 , LONDON (UK)
1980 , october 7-10 , pp 91-108

[V 1] : VAN BASTELEAR, Philippe (FNDP)

Bancs d'essai - choix d'un système

INSTITUT D'INFORMATIQUE , NAMUR (BELGIQUE)
TRANSPARENTS , support de cours

[V 2] : VANOIRBEEK, Christine

Implémentation d'une application de gestion sur réseau local

INSTITUT D'INFORMATIQUE , NAMUR (BELGIQUE)
MEMOIRE , 1981

[W 1] : WOODCOCK, A. W. (CII-HB)

Analyse de mesures de performances

CII - HONEYWELL BULL , PARIS (FRANCE)
utilisation interne , 1980 , juillet

[Z 1] : ZIMMERMAN, Hubert ; POUZIN, Louis

The standard network architecture developed by ISO

NETWORKS 80 , BOMBAY (INDIA)
DATA COMMUNICATION AND COMPUTER NETWORKS , symposium record , 1980 , february 4-6

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX - NAMUR (BELGIQUE),

INSTITUT D'INFORMATIQUE

CONTRIBUTION A LA CONCEPTION DE

SYSTEMES D'INFORMATIQUE DISTRIBUEE

- ANNEXES -

Promoteur : Ph. Van Bastelaer.

Eliane PERSOONS.
Marc HEMMERLING.

Mémoire présenté en vue
de l'obtention du grade de

LICENCIE ET MAITRE EN INFORMATIQUE

ANNEE ACADEMIQUE 1981 - 1982.

INTRODUCTION

L'annexe du mémoire comporte 4 grandes parties. Les 3 premières parties sont :

Partie 1 : "SIZING A DSA NETWORK", CII - HB sales aid 80-034 (révisé);

Partie 2 : DOSSIER DE PROGRAMMATION de l'outil que nous avons développé;

Partie 3 : MANUEL D'UTILISATION de l'outil que nous avons développé.

Ces parties sont présentées ci-après. La quatrième partie est relative au code Pascal constituant l'outil d'aide ala conception de réseau DSA. Cette partie est constitué de l'ensemble des listings présentés dans la farde accompagnant ce document.

D I M E N S I O N N E M E N T D' U N R E S E A U D S A

CII-HB SALES AID 80-034 REVISE

- Préparé par CLAUDIE CHAPPUIS
- Mis à jour par MARC HEMMERLING
  ELIANE PERSOONS
- Approuvé par GERARD YON
- Contrôlé par GILBERT LAQUAIS

TABLE DES MATIERES

PAGE

0. PREFACE I.1

1. INTRODUCTION I.2

2. MODELISATION DE RESEAUX DSA / GENERALITES I.3

3. NOTIONS FONDAMENTALES I.7

3.1. Introduction I.7
3.2. Notion de type I.7
3.3. Notion de classe I.9
3.4. Relations entre classes I.9
3.5. Relations intra-classe I.10
3.6. Exemples de relations I.11

4. LE DIMENSIONNEMENT I.13

4.1. Introduction I.13
4.2. Collecte des données I.14

4.2.1. les types de transactions I.14
4.2.2. les types d'agents I.20
4.2.3. les classes de sites I.22
4.2.4. la localisation des transactions I.26

4.3. Calculs et dimensionnement I.30

4.3.1. fréquence d'initialisation de transactions par sites I.30
4.3.2. flux des données générés entre les classes ... I.32
4.3.3. synthèse du trafic inter-sites I.35
4.3.4. flux de données sur les chemins de données individuels I.37
4.3.5. dimensionnement des chemins de données individuels I.40
4.3.6. dimensionnement des datanets DN 7100 ..... I.41

5. ANNEXE FORMULES I.44

6. ANNEXE FORMULAIRES I.49

0. PREFACE

Ce sales aid constitue un premier élément d'un ensemble d'outils d'évaluation de réseaux complexes DSA (distributed systems architecture).

Cette première version du sales aid a pour but de dimensionner :

- les liens du réseau primaire
- les noeuds constitués par des datanets DN7100

Les données nécessaires à la modélisation d'un réseau sont des données à la portée de chaque utilisateur.

Le sales aid est à la base d'un outil automatisé développé au DCT/DSR à Paris/Gambetta.

1. INTRODUCTION

Le sales aid est destiné à être utilisé pendant la phase d'avant-vente de matériel de réseau DSA.

L'utilisation de l'outil en avant-vente justifie les résultats approchés auxquels on aboutit. En fait, pour la plupart des cas, au moment de l'avant-vente il est complètement déplacé de procéder à des calculs de dimensionnement compliqués et longs ou à des simulations coûteuses.

Le lecteur voudra bien garder cette restriction d'utilisation à l'esprit; elle lui permettra de mieux comprendre la philosophie sous-jacente du sales-aid.

Les formulaires exposés sont supposés faire partie de DSA69, le standard pour modéliser un réseau DSA. Ces formulaires spécifient clairement ce que l'utilisateur doit fournir comme données pour arriver à dimensionner son réseau.

Tous ceux impliqués dans l'avant-vente des produits DSA sont priés

- d'utiliser l'outil mis à leur disposition

- de critiquer, de commenter le papier

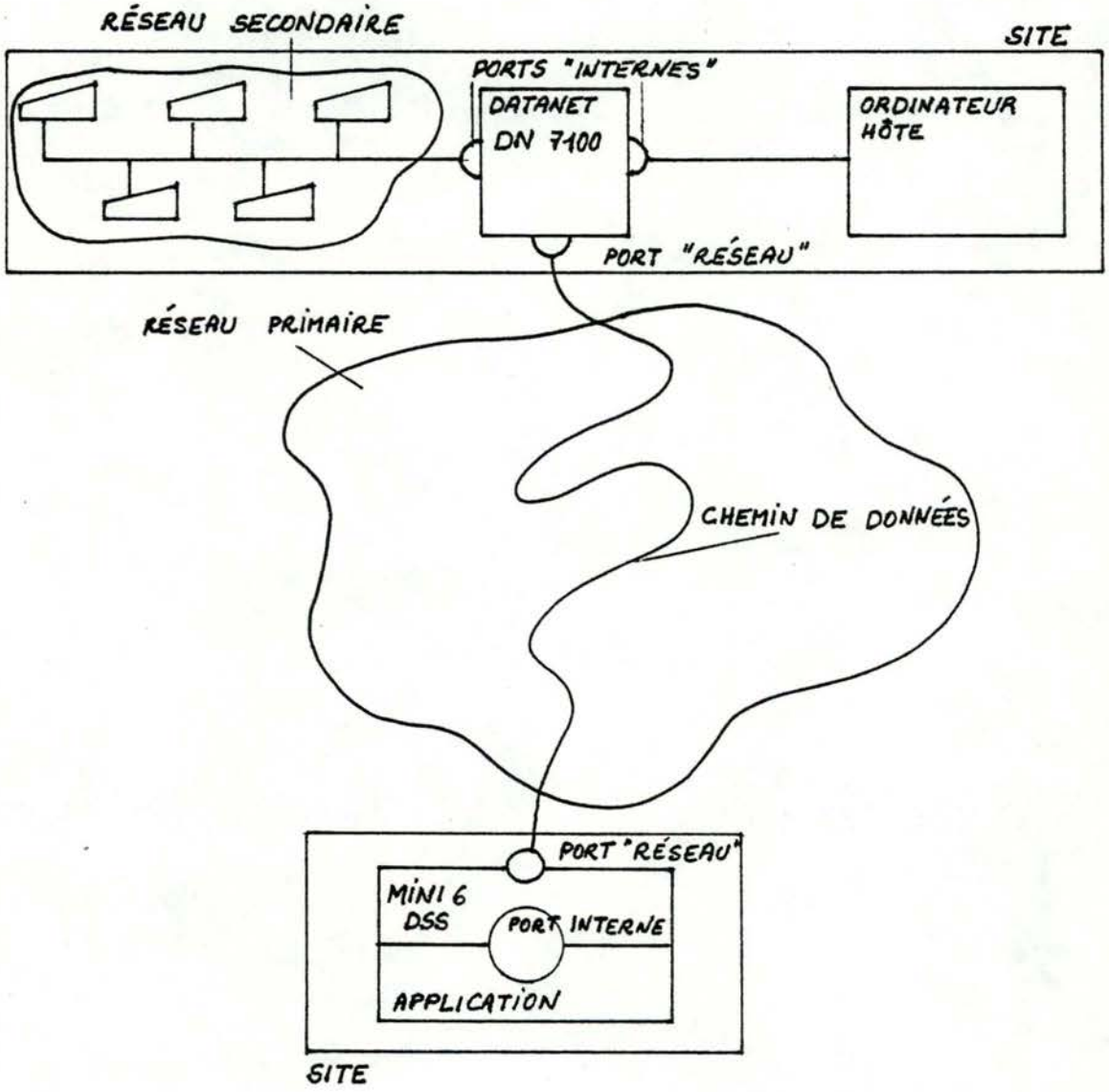
Cet outil doit être non seulement un outil de calcul, mais il doit avoir un effet psychologique positif sur l'utilisateur et doit aussi constituer une base aux discussions ultérieures clients/vendeurs.

2. MODELISATION DE RESEAUX DSA / GENERALITES

L'outil présenté dans ce papier (fin 1981) s'occupe uniquement de l'aspect "réseau primaire" et on ne tient compte que de la fonction "transport" de DSA (dans DSA il y a trois fonctions : traitement des données, transport des données et stockage des données).

La figure suivante sert à expliquer la terminologie employée et ne montre en aucune façon les possibilités DSA, ni les possibilités de modélisation de l'outil.

figure 2.1



Un réseau DSA est constitué d'un ensemble de sites interconnectés par des chemins de données.

Analysons les différents éléments du réseau

* les sites

Les sites sont des centres de traitement situés dans un lieu géographique donné.

Un site consiste en un composant DSA (pour le moment limité à un DN7100 ou à un MINI6/DSS) et en d'autres composants non typiquement DSA tels que :

- les processeurs "host", assurant un traitement de données et offrant des services aux utilisateurs (dans le réseau secondaire) et aussi aux autres systèmes dans le réseau primaire.

- les réseaux secondaires et terminaux, ces réseaux secondaires étant constitués d'un ensemble de terminaux et d'applications sous le contrôle d'un site.

- les applications utilisateur sur un MINI6/DSS

* les ports

Les différents composants sont interconnectés à travers des "ports". Un port est aussi le moyen d'accès entre un site et le chemin de données.

Exemples de ports :

- le "gateway" du niveau 66 sur le DN7100

- le TPI (transport programmatic interface) sur MINI6/DSS

- une station de transport en interface avec une connexion transpac

* les chemins de données

Les chemins de données représentent des connexions entre sites via le réseau primaire. Les chemins de données peuvent être de plusieurs types :

- circuit virtuel X25 sur réseau
  privé
  public (ex : transpac
  euronet
  datexp ...)

- point-à-point (HDLC) (peut se faire sur lien X21)

* les flux de données

Par flux de données nous entendons l'ensemble de caractères qui passent à travers le système entre deux points terminaux. Ces points, qui constituent la frontière entre les activités et le réseau, peuvent être :

- des terminaux

- des applications ... etc

Sont associés à la notion de flux :

- les ports et les chemins de données impliqués pour définir le chemin du flux

- une intensité, qui peut être mesurée en termes de paquets par seconde, caractères par seconde ou lettres par seconde selon les circonstances.

Calculs envisageables :

- si on totalise tout le flux des données traversant un certain port, on peut estimer l'activité CPU correspondante.

- si on totalise toutes les activités des ports du composant, on arrive à estimer la charge globale du CPU de ce composant, exprimée en ms/s.

- si on totalise tout le trafic transitant par un chemin de données, on peut évaluer la charge de la ligne, exprimée en bps. Ainsi ces calculs nous permettent de sélectionner la capacité adéquate de la ligne (c'est-à-dire la vitesse).

Ainsi nous arrivons à dimensionner grossièrement le réseau du point de vue liens entre sites et noeuds (DN7100).

3. NOTIONS FONDAMENTALES

3.1. Introduction

Pour dimensionner un réseau, il faut connaître un minimum d'informations telles que les applications mises en oeuvre, la topologie du réseau, ... etc.

Il est indispensable de discuter le deux caractéristiques essentielles des données : d'une part leur qualité ou nature et d'autre part leur nombre.

- la qualité : Afin d'utiliser ce sales aid comme support d'avant-vente, il est intéressant et souhaitable que les informations requises puissent être fournies par le client lui-même, qui vient de procéder à une analyse fonctionnelle profonde de ses applications.

- le nombre : Il est également souhaitable que le volume des données soit limité au strict nécessaire. Ceci est un but facile à atteindre si on profite des symétries et répétitions souvent présentes dans les réseaux.

3.2. Notion de type

La notion de type nous permet de décrire les paramètres de certains éléments génériques du modèle du réseau sous étude, sans devoir faire référence à leur localisation.

En particulier nous définissons :

* les types de processus

(on se place plutôt au niveau applications/programmes)
Les types de processus décrivent le travail à effectuer sur un "host" ou un "satellite".

exemple : type de processus : mise à jour d'un compte courant. Le type de processus représente une fonction précise à réaliser.

* les types de transactions

(on se place plutôt au niveau utilisateur/travail à réaliser)
Ces types décrivent les unités de travail vues par un agent du système (appelé utilisateur-terminal), en termes de processus utilisés et de messages échangés.

exemple : type de transaction : réservation de places d'avion. (tâche à accomplir par un agent/employé). Pour être traitée, cette transaction utilise (par exemple) le processus "consultation fichier" générant deux messages dans chaque sens, ayant chacun une longueur (exprimée en caractères), et le processus "réservation" générant un message dans chaque sens, ayant chacun une longueur bien précise.

* les types d'agents

Ces types décrivent l'agent vu par l'analyse fonctionnelle (et par l'organisation de l'entreprise). Ces types décrivent le travail d'un "utilisateur-terminal" (end-user) ; ce travail peut varier selon différentes heures en termes de transactions initialisées, lancées.

exemple : dans une agence de banque, l'employé travaillant au guichet peut initialiser les transactions "dépôt", "retrait" (d'argent). Pendant les heures normales il initialise 50 transactions par heure, dont 20 du type "dépôt" (40 %), et 30 du type "retrait" (60 %).

Si le suffixe "type" est absent, nous voulons exprimer que l'élément générique correspondant est associé à une localisation précise.

exemples :
- un processus est une occurence d'un type de processus implanté sur un "host" ou un "satellite" donné. (exemple : processus accès base de données dans une agence de banque à Paris)
- un agent est une occurence d'un type d'agent associé à un "lieu" à partir duquel une transaction peut être initialisée.
- une transaction est une occurence d'un type de transaction initialisée par un agent localisé. Remarquons que certains types de processus associés à une transaction peuvent être "non localisés". Il s'agit en particulier des types de processus initialisant les transactions à partir d'un terminal. Cette localisation est l'objet d'une procédure spéciale appelée "transaction location" (localisation des transactions).

3.3 Notion de classe

La notion de classe sera utilisée pour décrire un ensemble de sites qui sont identiques (excepté pour la localisation géographique).

Les critères de regroupement sont :

- les configurations hardware
- les types de processus implantés au site
- les types d'agents qui sont affectés à ce site
- le nombre des agents
  (ce nombre peut être approché, on regroupe toutes les classes où le nombre d'agents est égal (à un certain pourcentage près). Si cette marge (fixée par l'utilisateur lui-même) est dépassée, il est souhaitable de définir des sous-classes qui auront les mêmes propriétés qu'une classe pour la suite des calculs et qui ne diffèrent de la classe que par le nombre d'agents.

  par exemple :

  classe "région"
  nbre agents = 100 (à 10 % près)

  sous-classe "région2"
  nbre agents = 50

  sous-classe "région3"
  nbre agents = 150)

- les liens avec d'autres sites (situés dans la même classe ou dans d'autres classes).

Il est évident que si une classe contient plusieurs éléments, les calculs et spécifications sont suffisants pour un élément de la classe. (En fait, un site est équivalent à une classe contenant un élément).

3.4. Relations entre classes

Les processus qui traitent une transaction doivent communiquer entre eux. Cette communication donne naissance à un trafic dans le réseau.

Comme un type de processus est implanté sur tous les sites d'une classe, on se heurte à un problème : quel est le processus concerné par une activation donnée d'une transaction. En général ces ambiguïtés peuvent être éliminées si on définit de façon précise quel site sera impliqué dans le traitement de la transaction : par le mode.

Dans la majorité des cas réels, les sites sont arrangés dans une structure logique arborescente, qui peut être différente de la topologie physique du réseau. Typiquement, tous les sites se trouvant à un même niveau d'une structure arborescente forment une classe.

Analysons plus en détail le réseau organisé en forme d'arborescence :

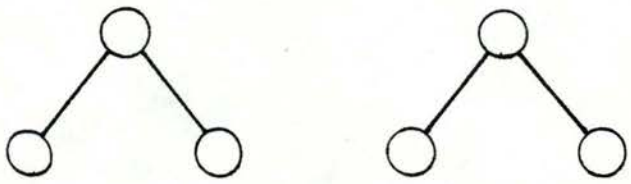
* le mode hiérarchique

La structure d'arbre est alors décrite comme étant une relation hiérarchique entre deux niveaux (classes). Dans ce cas, il est suffisant d'analyser les configurations, le trafic, ... etc pour deux sites communiquant, un par classe.

exemple :

classe 1

classe 2



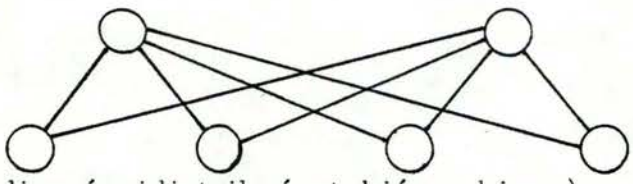
* le mode équidistribué

Ce mode définit une relation différente entre deux classes. Dans ce cas précis, n'importe quel site d'une classe peut communiquer avec n'importe quel site de l'autre classe (ce mode est le produit cartésien des deux classes). L'analyse sera limitée à un couple de sites.

exemple :

classe 1

classe 2



* le mode mixte (c'est-à-dire équidistribué et hiérarchique)

Ce mode est peut-être plus commun que le mode équidistribué (à 100 %). Ici la majorité des transmissions entre classes se fait en mode hiérarchique alors qu'un trafic équidistribué significatif subsiste.

3.5. Relations intra-classe

Dans une même classe on peut avoir des transmissions.

* le mode identique

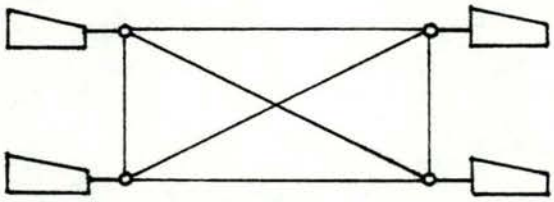
Ce mode définit une relation d'un site à lui-même, c'est-à-dire que le trafic provient du réseau secondaire attaché à ce site.

Outre les relations d'ordre hiérarchique, équidistribué ou mixte vues plus haut, il faut prendre en compte les relations entre les éléments d'une même classe.

* le mode mixte intra-classe (c'est-à-dire identique et équidistribué)

Ce mode est plus commun que le mode identique à 100 %.

exemple :



80 % du trafic se fait en mode (I)
20 % du trafic se fait en mode (E)

3.6. Exemples de relations

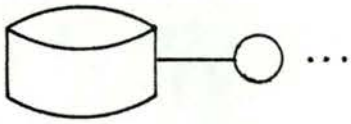
Soit une classe de sites "régionaux", supportant chacun une base de données "régionale", qui peuvent être interrogés par une classe de sites "agence" dispersés dans le pays. Ces agences sont regroupées par région.

La plupart du temps, le travail d'interrogation d'une base de données par une agence se fait au site "régional" auquel l'agence est rattachée. Toutefois il est concevable que cette agence fasse aussi des interrogations des autres bases de données.

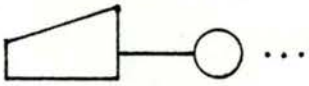
schématiquement :

- classe de sites "régionaux" : 37 sites

base de données régionale



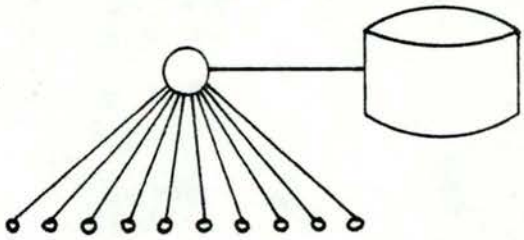
- classe de sites "agences" : 370 sites



Le problème consiste à spécifier quelle agence consulte quelle base de données.

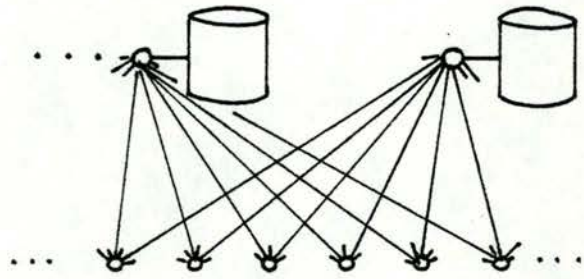
- solution 1 : mode hiérarchique (H)

chaque site "régional" est consulté par un groupe de 10 agences.



- solution 2 : mode équidistribué

  n'importe quel site "agence" peut consulter n'importe quelle base de données "régionale"



- solution 3 : mode mixte

  il est analogue à la solution 2 sauf que la majorité des consultations des bases de données "régionales" se fait selon le mode hiérarchique.

  exemple : 80 % mode H et 20 % mode E

4. LE DIMENSIONNEMENT

4.1. Introduction

Le dimensionnement d'un réseau peut être subdivisé en deux phases fondamentales :

- la première phase est consacrée à la collecte des données
- la deuxième phase s'occupe des calculs, c'est-à-dire du dimensionnement proprement dit.

La collecte des données est réalisée à l'aide de formulaires, définissant les besoins en données et informations pour résoudre le modèle du réseau. Les formulaires sont exposés en annexe-formulaires.

Les informations requises sont sans exception disponibles chez l'utilisateur (client) qui a procédé à une analyse fonctionnelle.

Les termes utilisés ne sont en aucune façon de nouveaux termes s'ajoutant à la terminologie DSA, mais des termes tirés du langage du client désirant informatiser une ou plusieurs applications.

La collecte des données concerne :

- la spécification des types de transactions (formulaire : transaction type (définition))
- la spécification des types d'agents (formulaire : agent type (définition))
- la spécification des classes de sites (formulaire : site-classes (définition))
- la localisation des transactions (formulaire : transaction location)

Les formulaires sont utilisés comme support des données.

Les calculs et le dimensionnement demandent l'utilisation des formulaires suivants :

- les fréquences d'initialisation de transactions par site pour calculer l'activité globale (formulaire : site transaction freqency)
- la synthèse du trafic intersite (formulaire : inter-site traffic synthesis)

Les calculs permettent de dimensionner les :

- liens physiques entre sites, en proposant la vitesse et le nombre de lignes à utiliser et en indiquant la charge de ces lignes.

- les datanets, en spécifiant la charge du processeur central du datanet.

4.2. Collecte des données

4.2.1. les types de transactions

a) méthodologie

Le travail d'un client (ou end-user) peut être subdivisé en différentes catégories :

- le transactionnel (1)

- les transferts de fichiers (2)

Expliquons ces catégories de travail :

(1) Une transaction est initialisée par un agent et la fréquence est typiquement liée à un événement externe, telle que l'arrivée d'un client dans un guichet de banque. Il est indésirable de permettre l'accumulation de travaux non réalisés, cependant il est aussi impossible de profiter des temps morts. Souvent certaines contraintes sont imposées en ce qui concerne la fréquence de travail d'un "utilisateur-terminal". Ces contraintes sont du style : "chaque agent du type A, affecté à la classe des sites S doit être capable de traiter T transactions par heure (avec un temps de réponse inférieur à tr secondes)". Il est évident que cette contrainte "T transactions par heure" tient déjà compte des files d'attente des clients devant le "guichet" des agents.

(2) Un transfert de fichier est caractérisé par la transmission de données à grande échelle. Les processus qui y sont impliqués (localisés dans les "hosts") sont des applications standard (c'est-à-dire des utilitaires), d'habitude présentes au niveau de chaque "host". Nous devons connaître la localisation des fichiers et le volume de données à transférer (et peut-être avec une contrainte temporelle).

b) collecte de données - le formulaire F1/annexe-formulaires

* type de transaction (colonne 1)

Il s'agit d'identifier les types de transactions, c'est-à-dire les unités de travail des agents (end-users).

exemples : paiement d'un chèque
dépôt d'argent

Il ne faut pas inclure un travail uniformément distribué dans le temps et qui forme une partie négligeable de la charge de travail.

exemple : dans un environnement bancaire : ouverture d'un compte bancaire.

Cependant il ne faut pas négliger le travail saisonnier, si à certains moments il représente une charge considérable.

exemple : à la fin de la semaine, dans une petite agence bancaire il y aura une activité plus importante lors du marché hebdomadaire.

* type de processus (colonne 2)

Pour chaque type de transaction identifié, déterminer tous les types de processus nécessaires au traitement de la transaction.

L'analyse fonctionnelle de l'application du client a fait apparaître toutes ces notions : l'application de l'utilisateur consiste à traiter un certain nombre et types de transactions ; l'analyse a également défini les types de processus qui vont traiter ces transactions. Chaque type de processus ainsi déterminé sera indistribuable, c'est-à-dire que le processus ne peut pas être implanté partiellement sur un site et partiellement sur un autre.

Cependant DSA permet la communication interprocessus et il est parfaitement possible de décrire des transactions qui nécessitent des traitements sur plusieurs sites. Typiquement, de telles méthodes sont nécessaires pour permettre aux transactions d'accéder à des bases de données situées sur des sites physiquement distincts.

exemple : Après l'analyse de l'implantation des fichiers du système, on réalise que certains d'entre-eux peuvent être accédés par un site à distance. On pourrait déclarer les types de processus "lire enregistrement", "écrire enregistrement", "mise à jour" comme étant implantés sur le même site, parce que chaque transaction les utilise en demandant accès à un fichier.

* transmissions (colonnes 3 à 6)

Pour chaque type de transaction, il faut déterminer le nombre de messages et leur longueur, transmis entre agent et type(s) de processus ; transmis entre type(s) de processus ; transmis entre type(s) de processus et agent.

remarque : il n'est pas permis à un processus A d'envoyer un message à B et du processus B de l'envoyer vers C et à C de répondre à A. De telles boucles ne peuvent être traitées si les processus sont localisés sur plusieurs sites d'une même classe.

donc $A \longrightarrow B \longrightarrow C$ est faux

si A, B et C appartiennent à la même classe.

c) exemple d'application

L'analyse de l'application implique 4 unités de travail à effectuer par les agents (end-users).

1) le dépôt d'argent "deposit"

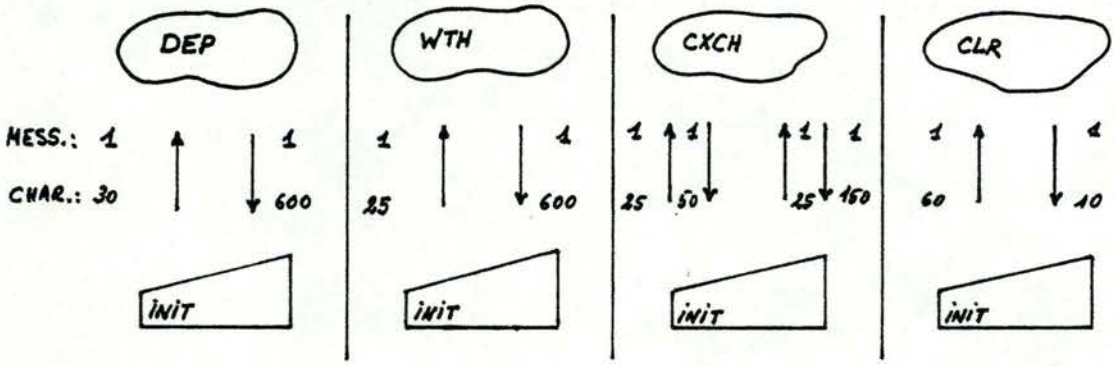
2) le retrait d'argent "withdrawal"

3) le change "currency-xch"

4) les chèques "cheque clear"

Ces 4 types de transactions sont traités chacun par un module, une "routine". Ce traitement donne naissance à des transmissions (dont on peut connaître le volume, ...)

Ces 4 modules sont :



Voir page suivante la collecte des données à l'aide du formulaire F1.

remarque : l'initialisation de la transaction (à partir d'un terminal) se fait par un processus, auquel on peut donner un nom quelconque. Nous avons pris la convention de le nommer "init".

TRANSACTION TYPES DEFINITION

NETWORK MODEL - NAME : EXEMPLE

TRANSACTION TYPES					
TRANSACTION TYPE ID	PROCESS TYPES EMPLOYED	TRANSMISSIONS			
		SENDING PROC. T.	RECEIVING PROC. T.	NBER OF MESS.	LENGTH (char)
DEPOSIT	DEP	INIT	DEP	1	30
		DEP	INIT	1	600
WITHDRAWAL	WTH	INIT	WTH	1	25
		WTH	INIT	1	600
CURRENCY-XCH	CXCH	INIT	CXCH	2	25
		CXCH	INIT	1	50
				1	150
CHEQUE-CLEAR	CLR	INIT	CLR	1	60
		CLR	INIT	1	10

d) remarque : une transaction spéciale : le transfert de fichiers

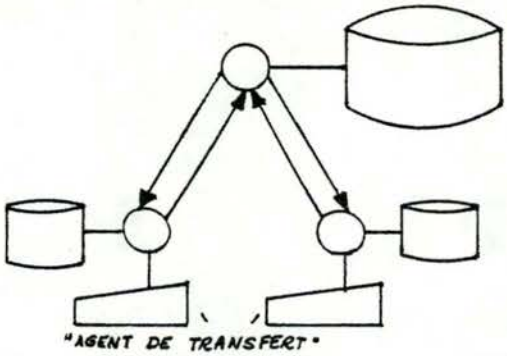
Le transfert de fichier peut être vu comme une transaction au sens défini plus haut. Elle est traitée par un processus. Le traitement donne naissance à deux messages : la demande de transfert et le transfert proprement dit.

Etant donné ces considérations et pour éviter de créer un nouveau formulaire, nous avons préféré assimiler le transfert à une transaction (un peu spéciale).

Pour prendre en compte la transaction dans les calculs qui suivront, il faut passer par un artifice. Le processus de transfert (utilitaire) est localisé sur des sites ; sur les sites recevant un fichier il faut qu'un agent spécial "transfert" soit présent.

exemple : site central

sites régionaux



TRANSACTION TYPES DEFINITION
NETWORK MODEL - NAME :

TRANSACTION TYPES					
TRANSACTION TYPE ID	PROCESS TYPES EMPLOYED	TRANSMISSIONS			
		SENDING PROC. T.	RECEIVING PROC. T.	NBER OF MESS.	LENGTH (char)
FILE-TRANS	FT	INIT	FT	1	15
	DEMANDE DE TRANSFERT	FT	INIT	1	3000000
	TRANSFERT				

Il est clair, vu la spécificité de ces transactions, qu'on peut simuler le comportement du réseau selon qu'il y ait beaucoup, peu ou pas de transfert.

4.2.2. les types d'agents

a) collecte des données - le formulaire F2/annexe-formulaires

* identificateur de type d'agent (colonne 1)

Il faut déterminer les différentes catégories d'agents (c'est-à-dire end-users). Cette procédure est quasi immédiate si l'analyse fonctionnele est bien faite.

exemple : - caissier

- agent de change

- etc ...

* type de la charge de travail (colonne 2)

Cette information n'est pas requise pour les calculs actuels, elle est demandée à titre informatif pour distinguer les agents "file-transfer" des autres.

deux possibilités : - T : transactionnel

- F : file-transfer

* profils d'utilisation / d'initialisation des transactions (colonnes 3 à 6)

Nous avons prévu trois profils possibles. Ces profils sont relatifs aux heures de fonctionnement "normal" (profil 1), aux heures d'utilisation de "pointe" (profil 2), et aux phénomènes saisonniers (profil 3). Le premier profil est obligatoire tandis que les deux autres sont optionnels.

Pour chaque type d'agent et pour chaque profil utilisé, il faut déterminer les types de transactions initialisés. En plus, il faut donner les proportions des différents types de transactions découverts plus haut.

Les profils 2 et 3 sont peut-être nécessaires pour compenser des variations importantes de l'activité.

b) exemple

AGENT TYPES DEFINITION

NETWORK MODEL - NAME : EXEMPLE

AGENT TYPES					
AGENT TYPE ID.	WORKLOAD TYPE	TRANSACTION USAGE PROFILES			
		TRANSACTIONS INITIATED	PROF. 1	PROF. 2	PROF. 3
TELLER	T	DEPOSIT	40		10
		WITHDRAWAL	50		70
		CURRENCY-XCH	10		20
OFFICE	T	CHEQUE-CLEAR	100		
MINI-TELLER	T	DEPOSIT	10		
		WITHDRAWAL	90		100

prof. 1 : heures normales
prof. 2 : heures de pointe
prof. 3 : saisonnalité
au niveau des banques en France, fermées samedi et dimanche, certaines agences sont quand-même ouvertes le samedi dans les régions paysannes. Ces agences sont confrontées à un accroissement considérable des dépôts d'argent à cause du marché hebdomadaire, d'où l'utilisation d'un profil particulier pour le samedi.

4.2.3. les classes de sites

a) remarque préliminaire

Dans le paragraphe 4.2.2., nous avons vu que l'utilisateur peut utiliser jusqu'à trois profils d'initialisation de transactions : un profil est relatif aux heures "normales", ce profil est obligatoire, le deuxième est relatif aux heures de pointe et le troisième tient compte des événements saisonniers.

Il se peut qu'au niveau des sites il y ait aussi des changements au niveau des agents qui y sont localisés. L'ensemble des types d'agents peut varier d'un profil à l'autre.

La collecte des données doit se faire pour chaque profil utilisé. A un profil correspond un problème et donc une solution globale.

b) collecte des données - le formulaire F3/annexe-formulaires

* identificateur d'une classe de sites (colonnes 1 et 2)

Tous les sites d'une même classe présentent :

- la même configuration (hardware)
- les mêmes types de processus au niveau du host
- les mêmes types d'agents

Ainsi tous les résultats seront identiques pour chaque élément appartenant à une classe.

Il ne faut pas oublier les sites existants, aussi lorsqu'ils utilisent des équipements de constructeurs étrangers ou des équipements HB non-DSA. Ces sites donnent également naissance à des transmissions.

Certains lecteurs vont se demander si l'utilisateur connaît toutes les données relatives à une classe de sites. Nous tenons à rappeler qu'un utilisateur doit avoir fait une analyse sérieuse de son application avant d'utiliser cet outil.

* la configuration (colonne 3)

Il faut indiquer les composants hardware du site. Pour le moment, les composants pris en compte par les procédures de calcul se limitent à :

- DPS 7
- DPS 8
- DN 7100
- MINI 6

Au moment de la rédaction de ce papier, les résultats de tests en cours n'étaient pas disponibles pour le MINI 6 .

* les processus implantés (colonne 4)

Il faut donner les noms des types de processus présents sur le site. Les processus peuvent être implantés uniquement sur un "host" (qui est un ordinateur offrant des services localement et à d'autres systèmes à travers le réseau primaire) ou sur un "satellite" (qui est un ordinateur rendant des services à un certain nombre d'utilisateurs locaux dans son réseau secondaire et communiquant avec d'autres hosts et satellites via le réseau primaire).

* les agents présents (colonnes 5 à 8)

Il faut spécifier les types d'agents localisés sur le site. Il est très probable que dans un réseau complexe il y ait des classes de sites, mais qu'à l'intérieur de ces classes on puisse distinguer plusieurs sous-classes se différenciant par l'activité, c'est-à-dire par le nombre des agents et par les intensités de travail (cfr notion de classe page I.9).

Les données à collecter pour les agents sont :

- nom des types d'agents implantés
- nombre d'agents par type
- procédure utilisée par un type d'agent dans le réseau secondaire
- intensité de travail d'un agent d'un type par unité de temps (qui sera l'heure)

Remarques :

- Pour un agent initialisant une transaction t_A :

  intensité * profil t_A = nombre de transactions t_A / heure

  exemple :

  l'agent A initialise deux transactions t_A et t_B , et son travail est organisé de la façon suivante :
  20 % pour t_A et 80 % pour t_B
  l'intensité de travail étant de 50 transactions par heure, l'agent initialise 10 transactions t_A par heure et 40 transactions t_B par heure.

- Si la somme des proportions des activités pour un profil d'utilisation d'un agent est égale à 1 (c'est-à-dire 100 %), alors l'intensité est simplement le nombre moyen de transactions par unité de temps (qui est l'heure).

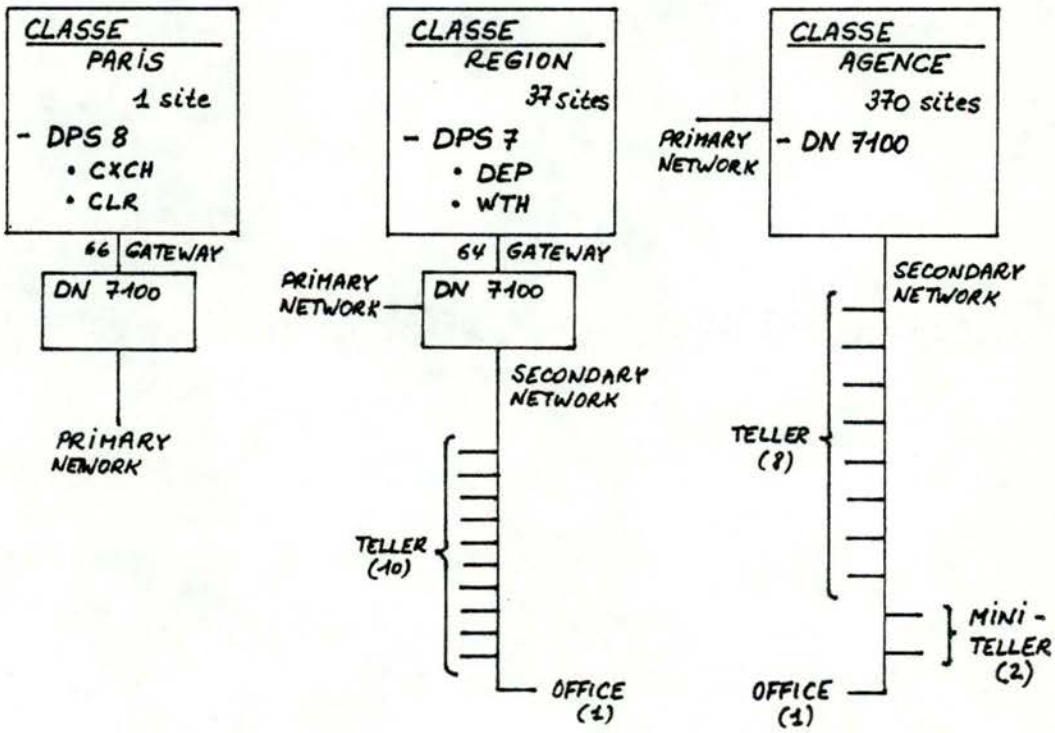
- Dans le cas d'une entrée de données, l'intensité doit être calculée à partir du nombre total de transactions à réaliser et du temps disponible pour cette réalisation.

  exemple : 700 transactions par jour ouvrable de 8 heures
  intensité = arrondi de (700/8) = 88 t/h

c) exemple

- si on utilise uniquement le profil 1 (concernant les heures normales)

- soit la configuration suivante : on distingue 3 classes :



SITE - CLASSES DEFINITION

NETWORK MODEL - NAME : EXEMPLE

SITE - CLASSES							
CLASS ID	NBER OF SITES	CONFIGURATION	PROCESSES ON SITE	INITIATING AGENTS AT SITE			
		COMPONENTS	TYPE - ID	TYPE-ID	NBER	INT.	PROC.
PARIS	1	DPS8 DN7100	CXCH CLR				
REGION	37	DPS7 DN7100	DEP WTH	TELLER OFFICE	10 1	30 120	VIP VIP
AGENCIES	370	DN7100		TELLER MINI-TELLER OFFICE	8 2 1	25 30 120	VIP VIP VIP

4.2.4. la localisation des transactions

a) remarques préliminaires

Le rôle de cette étape est de découvrir tous les liens logiquement possibles entre les classes de sites, de sélectionner et de définir les liens requis par l'analyse fonctionnelle de l'application.

Un même type de transaction initialisé à partir de différents sites sera localisé (c'est-à-dire pourra être traité) sur différents sites. Nous connaissons déjà les processus utilisés sur plusieurs sites regroupés dans une classe ; mais nous ignorons quel processus est concerné par une activation particulière. Ceci est réalisé par la localisation des transactions. Il y aura évidemment une localisation des transactions par profil utilisé.

b) procédure - le formulaire F4/annexe-formulaires

Il faut partir des définitions des types de transactions. Par type de transaction, copier le processus émetteur et le processus récepteur, les colonnes relatives aux transmissions, donc le nombre de messages et la longueur de ces messages.

A l'aide des définitions des classes de sites, entrer les classes de sites sur lesquelles les processus sont localisés, dans la rubrique site émetteur - site récepteur.

Remarques :

- Le site émetteur correspondant au processus "initialisation à partir d'un terminal" (notre convention : processus "init") doit évidemment supporter un agent dont le profil d'initialisation de transactions contient celle qu'on est en train de localiser.

- D'habitude, un site apparaîtra au moins deux fois dans la spécification de la localisation : une fois comme émetteur et une fois comme récepteur.

Afin de se retrouver dans les spécifications, il est conseillé de remplir également les colonnes "composants" ; pour les processus du type "init", il faut noter le type de protocole utilisé dans le réseau secondaire du site.

Il faut ensuite spécifier le mode de communication. Pour chaque paire de processus communiquant, il faut indiquer le mode de communication. Ceci est important si la classe contient plusieurs sites (cfr relations inter-classes et relations intra-classe pages I.9 à I.12)

Le mode peut être :

1) Pour la communication inter-classes :

- E ou équidistribué
- H ou hiérarchique
- E et H ou mixte

2) Pour la communication intra-classe :

- I ou identique
- E ou équidistribué
- I et E ou mixte

- pour chaque mode de communication, indiquer la proportion
- finalement, le nombre et la longueur des messages peuvent être copiés de la définition des transactions
- une fois le lien entre deux classes de sites ou le lien intra-classe sélectionné, on spécifiera la nature du lien :

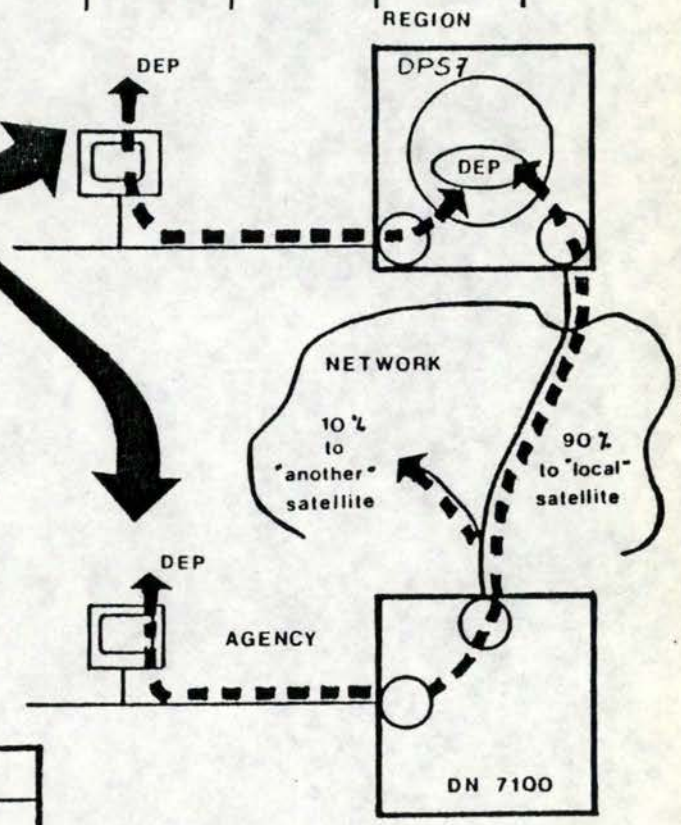
  - X 25 public : "U"
  - X 25 privé : "R"
  - point-à-point : "O"

c) exemples

- l'exemple de la page suivante visualise la procédure à utiliser
- l'exemple de la page I.29 reprend les données prises jusqu'ici comme exemple à chaque étape.

TRANSACTION LOCATION

TRANS-ACTION	SENDER			RECEIVER			MODE	PROPN.	MESSAGES PER TRANSACTION				CLASS ACTIVITY	INTER CLASS TRAFFIC		
	PROCESS	SITE	COMPONENT	PROCESS	SITE	COMPONENT			#	LENGTH	LINK	# OF TRANSP. UNIT		# OF LETTERS	CHARACTER VOLUME	# OF TRANSPORT UNITS
DEPOSIT	INIT	REGION	VIP	DEP	REGION	DPS7	I	100	1	30						
	DEP	"	DPS7	INIT	"	VIP	I	100	1	600						
DEPOSIT	INIT	AGENCY	VIP	DEP	REGION	DPS7	E	10	1	30						
							H	90	1	30						
	DEP	REGION	DPS7	INIT	AGENCY	VIP	E	10	1	600						
							H	90	1	600						



Message-flow for this transaction

(1)	(2)		(3)			
TRANSACTION TYPE ID	PROCESS - TYPES EMPLOYED		TRANSMISSIONS			
	ID	NB	SENDING PROCESS TYPE	RECEIVING PROCESS TYPE	NUMBER OF MESSAGES	LENGTH (CHAR)
DEPOSIT	DEP	1	INIT DEP	DEP INIT	1 1	30 600

TRANSACTION LOCATION																
TRANS-ACTION	SENDER			RECEIVER			MODE	PROPN.	MESSAGES PER TRANSACTION				CLASS ACTIVITY	INTER CLASS TRAFFIC		
	PROCESS	SITE	COMPONENT	PROCESS	SITE	COMPONENT			#	LENGTH	LINK	# OF TRANSP. UNIT		# OF LETTERS	CHARACTER VOLUME	# OF TRANSPORT UNITS
DEPOSIT	INIT	REGION	VIP	DEP	REGION	DPS7	I	100	1.	30						
	DEP	REGION	DPS7	INIT	REGION	VIP	I	100	1	600						
DEPOSIT	INIT	AGENCY	VIP	DEP	REGION	DPS7	E	10	1	30	U					
							H	90	1	30	U					
	DEP	REGION	DPS7	INIT	AGENCY	VIP	E	10	1	600	U					
							H	90	1	600	U					
WITHDRAWAL	INIT	REGION	VIP	WTH	REGION	DPS7	I	100	1	25						
	WTH	REGION	DPS7	INIT	REGION	VIP	I	100	1	600						
WITHDRAWAL	INIT	AGENCY	VIP	WTH	REGION	DPS7	E	20	1	25	U					
							H	80	1	25	U					
	WTH	REGION	DPS7	INIT	AGENCY	VIP	E	20	1	600	U					
							H	80	1	600	U					
CURRENCY-XCH	INIT	REGION	VIP	CXCH	PARIS	DPS8	H	100	2	25	0					
	CXCH	PARIS	DPS8	INIT	REGION	VIP	H	100	1	50	0					
							H	100	1	150	0					
CURRENCY-XCH	INIT	AGENCY	VIP	CXCH	PARIS	DPS8	H	100	2	25	0					
	CXCH	PARIS	DPS8	INIT	AGENCY	VIP	H	100	1	50	0					
							H	100	1	150	0					
CHEQUE-CLR	INIT	REGION	VIP	CLR	PARIS	DPS8	H	100	1	60	0					
	CLR	PARIS	DPS8	INIT	REGION	VIP	H	100	1	10	0					
CHEQUE-CLR	INIT	AGENCY	VIP	CLR	PARIS	DPS8	H	100	1	60	0					
	CLR	PARIS	DPS8	INIT	REGION	VIP	H	100	1	10	0					

4.3. Calculs et dimensionnement

4.3.1. fréquence d'initialisation de transactions par site

a) remarques préliminaires

Après avoir localisé les transactions, il est possible de calculer le trafic logique entre sites.

Ceci sera effectué pour chaque profil utilisé.

b) calcul de la fréquence des transactions par site

le formulaire F5/annexe-formulaires

Ce calcul est réalisé à l'aide des informations sur les classes de sites et les types d'agents (F2 et F3).

Pour chaque type d'agent localisé sur un site et qui initialise des transactions, faire le calcul suivant :

$$NA_{t,a} = p_{t,a} * i_a * n_a$$

$NA_{t,a}$ = nombre d'activations de la transaction du type t (par heure) par le type d'agent a sur le site nommé.

$p_{t,a}$ = proportion d'activations de la transaction du type t notée dans le profil du type d'agent a.

i_a = intensité de travail du type d'agent a sur le site (information qu'on a mémorisée dans les spécifications des sites).

n_a = nombre d'agents du type a au site (information qu'on a mémorisée dans les spécifications des sites).

(t : indice des transactions, a : indice des types d'agents)

Si tous les agents et toutes les transactions ont été traités, il faut sommer pour un type de transactions les $NA_{t,a}$ pour tous les agents.

Cette somme représente le taux total d'activation de transactions sur le site. Ells est appelée activité du site (pour la transaction concernée).

Si on multiplie cette somme par le nombre de sites contenus dans la classe, on obtient l'activité de la classe, résultat que l'on notera N(t).

c) EXEMPLE : calculons l'activité des sites de nos exemples (1 profil)

SITE TRANSACTION FREQUENCY

SITE CLASS-ID	# OF SITES IN CLASS	TRANS-ACTION ID	AGENTS LOCATED AT SITE								SITE ACTIVITY	CLASS ACTIVITY
			TELLER	OFFICE	MINI-TELLER							
REGION	37	DEPOSIT	120	-	-						120	4440
		WITHDR.	150	-	-						150	5550
		CURR.XC.	30	-	-						30	1110
		CHQUE.C.	-	120	-						120	4440
AGENCES	370	DEPOSIT	80	-	6						86	31820
		WITHDR.	100	-	54						154	56980
		CURR.XC.	20	-	-						20	7400
		CHQUE C.	-	120	-						120	44400

$$NA_{\text{chèque-clr,office}} = p_{\text{chèque-clr,office}} * i_{\text{office}} * n_{\text{office}}$$

$$= 1 * 120 * 1$$

$$= 120$$

$N_{\text{chèque-clr (site-activity)}}$: 120

$N_{\text{chèque-clr (class-activity)}}$: 120 * 370 = 44400

4.3.2. flux de données générés entre les classes

a) remarques préliminaires

Pour chaque type de transactions initialisé par la classe de sites considérée, nous calculons maintenant les flux de données générés entre les différentes classes. Ceci sera réalisé en appliquant les fréquences d'activations de transactions (qu'on vient de calculer au paragraphe précédent) aux informations concernées dans l'ensemble des informations relatives à la localisation des transactions.

b) méthode

- Chaque ligne du formulaire F4 (localisation des transactions) est associée à une transaction particulière, c'est-à-dire à un type de transaction initialisé à partir d'un site particulier. La même remarque est applicable à chaque ligne du formulaire F5 (fréquence d'initialisation des transactions). On peut donc reporter les informations du formulaire F5 au formulaire F4 (colonne 14).

- Ensuite il faut compléter les informations relatives aux messages (formulaire F4/colonne 13) : il faut calculer le nombre de paquets X25 ou le nombre de fragments point-à-point, selon le lien retenu lors de la localisation des transactions.

- Voici les formules à appliquer : (cfr annexe - formules/FL1)

  * nombre de fragments : F

  $$F = N * \text{arrondi} \left(\frac{L + 6}{1000}\right)$$

  N = nombre de messages de longueur L
  L = longueur des messages en caractères
  arrondi = arrondi à l'entier supérieur

  le 6 provient de deux caractères pour l' "en-tête du message" et quatre caractères pour l' "en-tête du segment de l'enregistrement".

  * nombre de paquets : X (privé ou public selon le choix)

  $$X = N * \text{arrondi} \left(\frac{L + 14}{128}\right)$$

  le 14 provient de six caractères expliqués ci-dessus, quatre caractères proviennent de l'ACK et quatre caractères de l' "en-tête du fragment".

- Pour les fragments, la valeur par défaut de la longueur est égale à 1000 ; cependant l'utilisateur peut spécifier une autre valeur.

- Pour les paquets, la valeur par défaut est égale à 128 caractères, mais l'utilisateur peut spécifier une autre longueur.

- Ensuite il faut compléter la section "trafic" du formulaire F4 (colonnes 15 à 17).

  Les calculs nécessaires se font selon les formules suivantes :

  * nombre de lettres : = P * A * N

  * volume de caractères : = P * A * N * L

  * nombre de fragments point-à-point : = P * A * F

  * nombre de paquets X25 : = P * A * X

  P = proportion du mode utilisé
  A = activité de la classe
  N = nombre de messages
  L = longueur du message en caractères
  F = nombre de fragments
  X = nombre de paquets X25

Remarques :

- On suppose que le nombre de lettres est égal au nombre de messages

- Si le mode est "I" (identique), les messages générés sur le site y sont traités, et donc il n'y aura pas de trafic sur le réseau primaire

- En ce qui concerne le volume de caractères, l'unité est l'octet (caractères), mais il pourrait aussi être le koctet (qui est égal à 1024 octets).

c) exemple

Voir page suivante, qui reprend les données collectées dans les exemples vus jusqu'ici, mais uniquement celles relatives au profil 1;

TRANSACTION LOCATION																
TRANS-ACTION	SENDER			RECEIVER			MODE	PROPN.	MESSAGES PER TRANSACTION				CLASS ACTIVITY	INTER CLASS TRAFFIC		
	PROCESS	SITE	COMPONENT	PROCESS	SITE	COMPONENT			#	LENGTH	LINK	# OF TRANSP. UNIT		# OF LETTERS	CHARACTER VOLUME	# OF TRANSPORT UNITS
DEPOSIT	INIT	REGION	VIP	DEP	REGION	DPS7	I	100	1	30			4440			
	DEP	REGION	DPS7	INIT	REGION	VIP	I	100	1	600			4440			
DEPOSIT	INIT	AGENCY	VIP	DEP	REGION	DPS7	E	10	1	30	U	1	31820	3182	95460	3182
							H	90	1	30	U	1	31820	28638	859140	28638
	DEP	REGION	DPS7	INIT	AGENCY	VIP	E	10	1	600	U	5	31820	3182	1909200	15910
							H	90	1	600	U	5	31820	28638	17182800	143190
WITHDRAWAL	INIT	REGION	VIP	WTH	REGION	DPS7	I	100	1	25			5550			
	WTH	REGION	DPS7	INIT	REGION	VIP	I	100	1	600			5550			
WITHDRAWAL	INIT	AGENCY	VIP	WTH	REGION	DPS7	E	20	1	25	U	1	56980	11396	284900	11396
							H	80	1	25	U	1	56980	45584	1139600	45584
	WTH	REGION	DPS7	INIT	AGENCY	VIP	E	20	1	600	U	5	56980	11396	6837600	56980
							H	80	1	600	U	5	56980	45584	27350400	227920
CURRENCY-XCH	INIT	REGION	VIP	CXCH	PARIS	DPS8	H	100	2	25	0	2	1110	2220	55500	2220
	CXCH	PARIS	DPS8	INIT	REGION	VIP	H	100	1	50	0	1	1110	1110	55500	1110
							H	100	1	150	0	1	1110	1110	166500	1110
CURRENCY-XCH	INIT	AGENCY	VIP	CXCH	PARIS	DPS8	H	100	2	25	0	2	7400	14800	370000	14800
	CXCH	PARIS	DPS8	INIT	AGENCY	VIP	H	100	1	50	0	1	7400	7400	370000	7400
							H	100	1	150	0	1	7400	7400	1110000	7400
CHEQUE-CLR	INIT	REGION	VIP	CLR	PARIS	DPS8	H	100	1	60	0	1	4440	4440	266400	4440
	CLR	PARIS	DPS8	INIT	REGION	VIP	H	100	1	10	0	1	4440	4440	44400	4440
CHEQUE-CLR	INIT	AGENCY	VIP	CLR	PARIS	DPS8	H	100	1	60	0	1	44400	44400	2664000	44400
	CLR	PARIS	DPS8	INIT	AGENCY	VIP	H	100	1	10	0	1	44400	44400	444000	44400

4.3.3. synthèse du trafic inter-sites

a) remarques préliminaires

Il faut produire autant de synthèses qu'il y a de profils utilisés.

Nous avons maintenant à notre disposition toutes les informations nécessaires pour dimensionner les chemins de données et les composants de transmission (DN 7100). Cependant l'information n'est pas présentée sous une forme convenable.

Le but de la synthèse inter-sites consiste en un regroupement de toutes les données relatives à un chemin de données entre deux sites.

b) méthode - le formulaire F6/annexe-formulaires

Sur le formulaire "synthèse du trafic inter-sites", il faut créer une entrée pour chaque couple distinct de site émetteur et site récepteur trouvé sur le formulaire F4.

Si la communication se fait sous le mode hiérarchique et équidistribué, il faut créer deux entrées distinctes, une pour chaque mode (car le trafic sur un chemin de données H est différent de celui sur un chemin de données E).

Nous ignorons toutes les lignes où le mode est I, car ce mode ne donne pas naissance à un trafic sur le réseau primaire.

Dans les colonnes appropriées, inscrire la somme des caractéristiques du trafic, c'est-à-dire le nombre de lettres et le volume de caractères de tout le trafic, transmis de la classe "sender" à la classe "receiver" pour le mode spécifié. Ces informations sont nécessaires pour dimensionner les chemins de données individuels entre sites (cfr étape suivante).

c) exemple

Voir page suivante, qui reprend la synthèse du trafic inter-sites pour le profil 1 des données collectées dans les exemples.

INTER - SITE TRAFFIC SYNTHESIS

SENDING SITE CLASS		RECEIVING SITE CLASS		MODE	## OF LETTERS MESSAGES	CHAR. VOLUME (LESS OVERHEAD)	## OF TRANSPORT UNITS
ID	##	ID	##				
AGENCY	370	REGION	37	E	3182	95460	3182
				E	11396	284900	11396
					14578	380360	14578
AGENCY	370	REGION	37	H	28638	859140	28638
				H	45584	1139600	45584
					74222	1998740	74222
REGION	37	AGENCY	370	E	3182	1909200	15910
				E	11396	6837600	56980
					14578	8746800	72890
REGION	37	AGENCY	370	H	28638	17182800	143190
				H	45584	27350400	227920
					74222	44533200	371110
REGION	37	PARIS	1	H	2220	55500	2220
				H	4440	266400	4440
					6660	321900	6660
PARIS	1	REGION	37	H	1110	55500	1110
				H	1110	166500	1110
				H	4440	44400	4440
					6660	266400	6660
AGENCY	370	PARIS	1	H	14800	370000	14800
				H	44400	2664000	44400
					59200	3034000	59200
PARIS	1	AGENCY	370	H	7400	370000	7400
				H	7400	1110000	7400
				H	44400	444000	44400
					59200	1924000	59200

4.3.4. flux de données sur les chemins de données individuels

a) remarques préliminaires

La synthèse du trafic inter-sites nous fait découvrir les flux de données à travers le réseau primaire. Il faut transformer les résultats de façon à obtenir un flux de données en bits par seconde sur un chemin de données individuel.

b) méthode

- Il faut utiliser le formulaire F6 rempli à l'étape précédente.
- Ensuite il faut sommer par mode les résultats obtenus entre "sender site" et "receiver site".

l'exemple nous donnera :

* entre : AGENCY et REGION pour le mode E,

  il y a : 29156 messages
  87468 paquets X25 public
  9127160 caractères

* entre : AGENCY et REGION pour le mode H,

  il y a : 148444 messages
  445332 paquets X25 public
  46531940 caractères

* entre : REGION et PARIS pour le mode H,

  il y a : 13320 messages
  13320 fragments point-à-point
  588300 caractères

* entre : AGENCY et PARIS pour le mode H,

  il y a : 118400 messages
  118400 fragments point-à-point
  4958000 caractères

Afin d'évaluer correctement le trafic entre les classes de sites, il faut ajouter l'overhead résultant des protocoles. (cfr formules/annexe-formules FL11/FL12). L'overhead change en fonction de la nature du chemin de données. Rappelons que nous prenons en compte la liaison par circuits virtuels X25 (public ou privé) et la liaison par lien HDLC point-à-point.

Le trafic comprenant l'overhead sera donné par les formules suivantes (en bits) :

lien point-à-point : 8 * (X + 6m + 13f)

lien X25 : 8 * (X + 14m + 8p)

où : X = volume de caractères
m = nombre total de messages (lettres)
f = nombre total de fragments
p = nombre total de paquets

où : 6 = overhead message (2 + 4)
13 = overhead fragment (8 + 5)
14 = overhead message (2 + 4 + 4 + 4)
8 = overhead·paquet (3 + 5)

Nous obtenons des bits par heure, transitant sur les chemins de données entre les classes de sites. Il faut convertir ces résultats en bits par seconde (bps).

pour l'exemple, nous allons avoir :

* entre : AGENCY et REGION pour le mode E,

  il y a 22745 bps

* entre : AGENCY et REGION pour le mode H,

  il y a 115940 bps

* entre : REGION et PARIS pour le mode H,

  il y a 1870 bps

* entre : AGENCY et PARIS pour le mode H,

  il y a 16017 bps

A présent, nous connaissons tout le trafic moyen en bps, entre deux classes de sites. A partir de cette information, nous devons calculer le trafic sur un chemin de données individuel entre deux sites appartenant chacun à une classe différente. Ceci nous permettra de déterminer la capacité requise de la ligne.

Comment répartir le trafic transitant entre deux classes de sites, afin de trouver le trafic sur un chemin de données individuel ?

Considérons deux classes de sites communiquant : A et B

exemple : classe B, $N_b = 2$ { O O

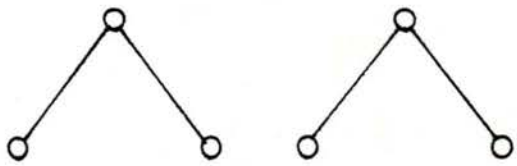
classe A, $N_a = 4$ { O O O O

Ou bien $N_a = N_b$, ou, sans perte de généralité, nous pouvons supposer que $N_a > N_b$ (comme ci-dessus).
En général, nous pouvons avoir $N_a * N_b$ chemins de données entre les sites de A et ceux de B (dans l'exemple : 2 * 4 = 8 chemins)

* dans le cas de trafic hiérarchique, nous avons besoin de N_a chemins de données individuels

classe B

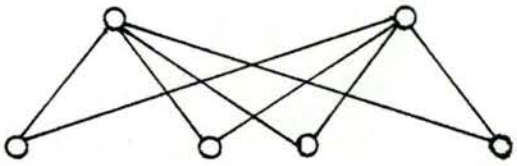
classe A



* dans le cas de trafic équidistribué, nous avons besoin de $N_a * N_b$ chemins de données

classe B

classe A



Pour passer au trafic en bps par chemin de données individuel, il faut utiliser les formules suivantes :

1) Si le mode entre les deux classes est uniquement H :

$$\text{trafic sur chemin individuel} = \frac{\text{trafic (de classe A à classe B)}}{\max(N_a, N_b)}$$

2) Si le mode est uniquement E :

- si classe A = classe B (trafic intra-classe) :

$$\text{trafic sur chemin individuel} = \frac{\text{trafic (intraclasse A)}}{\frac{N_a (N_a - 1)}{2}}$$

- sinon

$$\text{trafic sur chemin individuel} = \frac{\text{trafic (de classe A à classe B)}}{N_a * N_b}$$

3) Si les modes E et H existent entre deux classes :

il faut distinguer les chemins individuels où il y a du trafic uniquement en mode E : calcul selon 2), et les chemins qui supportent le trafic en mode H et E : calcul selon 1) et 2) puis additionner.

Notre exemple nous donnera les résultats suivants :

* chemin de données individuel entre : AGENCY et REGION pour mode E uniquement

  = 2 bps

* chemin de données individuel entre AGENCY et REGION pour mode E et H

  = 314 + 2 = 316 bps

* chemin de données individuel entre REGION et PARIS pour mode H uniquement

  = 51 bps

* chemin de données individuel entre AGENCY et PARIS pour mode H uniquement

  = 44 bps

4.3.5. dimensionnement des chemins de données individuels

Nous connaissons maintenant le flux de données sur les différents liens. Il faut à présent proposer des lignes physiques d'un point de vue nombre de lignes et capacité de ces lignes.

Le principe de cette proposition est qu'on ne tolère pas un taux d'utilisation supérieur à 60 %. Pour la combinaison nombre de lignes / vitesse de ces lignes, nous avons élaboré le "tableau des lignes" (cfr annexe-formules FL2).

Le choix se fait de la façon suivante :

- prendre le trafic du lien pour lequel il faut proposer une ligne
- parcourir le tableau ligne par ligne pour la première colonne
- dès que le trafic du lien est compris dans les bornes, proposer les lignes à utiliser et calculer le taux d'utilisation.

Pour notre exemple :

* entre AGENCY et REGION pour le mode E :

  1 ligne à 1200 bps ; taux d'utilisation : 1 %

* entre AGENCY et REGION pour le mode E et H :

  1 ligne à 1200 bps ; taux d'utilisation : 27 %

* entre REGION et PARIS pour le mode H :

  1 ligne à 1200 bps ; taux d'utilisation : 5 %

* entre AGENCY et PARIS pour le mode H :

  1 ligne à 1200 bps ; taux d'utilisation : 4 %

4.3.6. dimensionnement des datanets DN 7100

a) remarques préliminaires

Pour le moment (fin janvier 1982), nous nous limitons au dimensionnement des datanets DN 7100 sur les sites. Les composants de traitement de données doivent être analysés par des modèles spécialisés, ce qui dépasse le cadre de cet outil.

Un datanet est essentiellement constitué de ports subdivisés en ports de réseau et ports internes. La charge CPU du composant résultant de l'activité du port est déterminée par le trafic transitant par ce port. La charge CPU du composant DSA est la somme des charges CPU des différents ports. Ces hypothèses sont suffisamment précises pour un outil utilisé en avant-vente de produits DSA.

Pour mieux comprendre les calculs à effectuer, nous allons exposer les formules nécessaires et visualiser la méthode à l'aide d'un exemple.

Cet outil déterminera le modèle du datanet DN 7100 à choisir (du point de vue charge CPU) :

- DN 7102
- DN 7103
- DN 7102 + DCE 7104
  (mêmes performances que DN 7103 mais le nombre de lignes est limité à 48)

Les formules sont données par le "datanet 7100 message processing time in ms" (cfr annexe-formules/FL3)

b) méthodologie

Pour connaître le temps de traitement total (TT) :

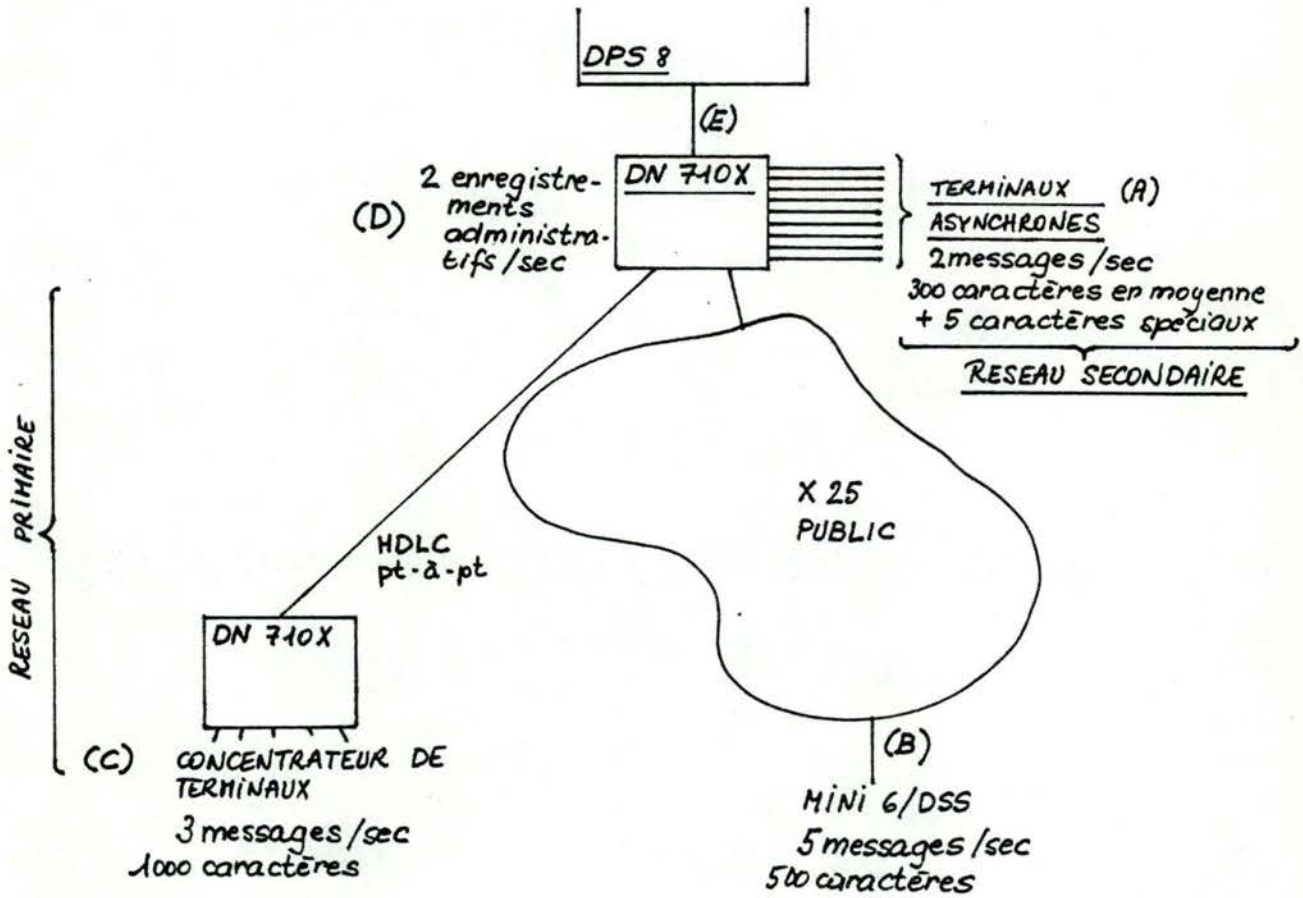
- pour chaque fonctionnalité utilisée, multiplier le temps (T) par le trafic correspondant

- additionner les différents temps

Si le trafic est exprimé en messages par seconde, le (TT) exprimé en ms ne doit pas dépasser la seconde, et des expériences ont montré qu'il vaut mieux ne pas dépasser les 0.8 secondes. Si les calculs sont faits pour le modèle DN 7102 sans DCE 7104 et que le (TT) est supérieur à 0.8 seconde, alors il faut refaire les calculs pour le modèle DN 7102 avec DCE 7104. Si le trafic est toujours supérieur à 0.8 seconde, alors il faut répartir le trafic sur plusieurs datanets.

Attention : il ne faut pas perdre de vue qu'un DN 7102 n'est pas transformable en un DN 7103.

c) exemple



Il s'agit d'évaluer l'activité du DN 7100 "frontal" d'un host du niveau 66 (DPS8).

Si le nombre total de lignes est inférieur à 48 lignes, nous pouvons essayer les calculs avec le modèle 7102.

1) c = 1

2) dimension du buffer = 100 caractères

3) réseau secondaire : terminaux asynchrones (A)

$$T_A = 1 * ((5.5 + 0.5 * \frac{300 - 1}{100} + 2 * 5) * 2) = 33$$

4) réseau primaire : X25 privé / DN full control (B)

$$T_B = 1 * ((18 + 8 * \frac{500 + 7}{128}) * 5) = 210$$

choix de contrôle :
- 1 : basic control
- 2 : flow control
- 3 : recovery
- 4 : full control
- M : multipathing

5) réseau primaire : point-à-point HDLC / full control (C)

$$T_C = 1 * ((11 + 10 * \frac{1000 - 1}{1000}) * 3) = 33$$

6) administration de réseau (host logfile) (D)

$$T_D = 1 * 16 * 2 = 32$$

7) "level-66 gateway" (E)

$$T_E = 1*((10+0.5*\frac{((300+500+1000)/3) - 1}{100})*(2+5*\frac{500}{128}+3+2)$$

$$= 12.5 * 26.5 = 331$$

8) TT = 639.25 ms

le résultat est inférieur à 0.8 sec
doncle modèle à utiliser est le DN 7102 ; charge CPU = 64 %

le modèle DN 7102/DCE 7104 a une charge CPU de
640 * 0.83 = 53 %

5. ANNEXE FORMULES

DSA PROTOCOLS OVERHEAD

The overhead, in characters, associated with each of the units previously described is :

- 6 characters per letter (2 characters "letter header" and 4 characters "record segment header")

- 4 characters per fragment

- A variable number of characters per transport unit. Transport unit size is determined so as to leave room for 4 characters if a full size fragment is to be transferred, extra characters (always a multiple of 4) may be included if the fragment is smaller.

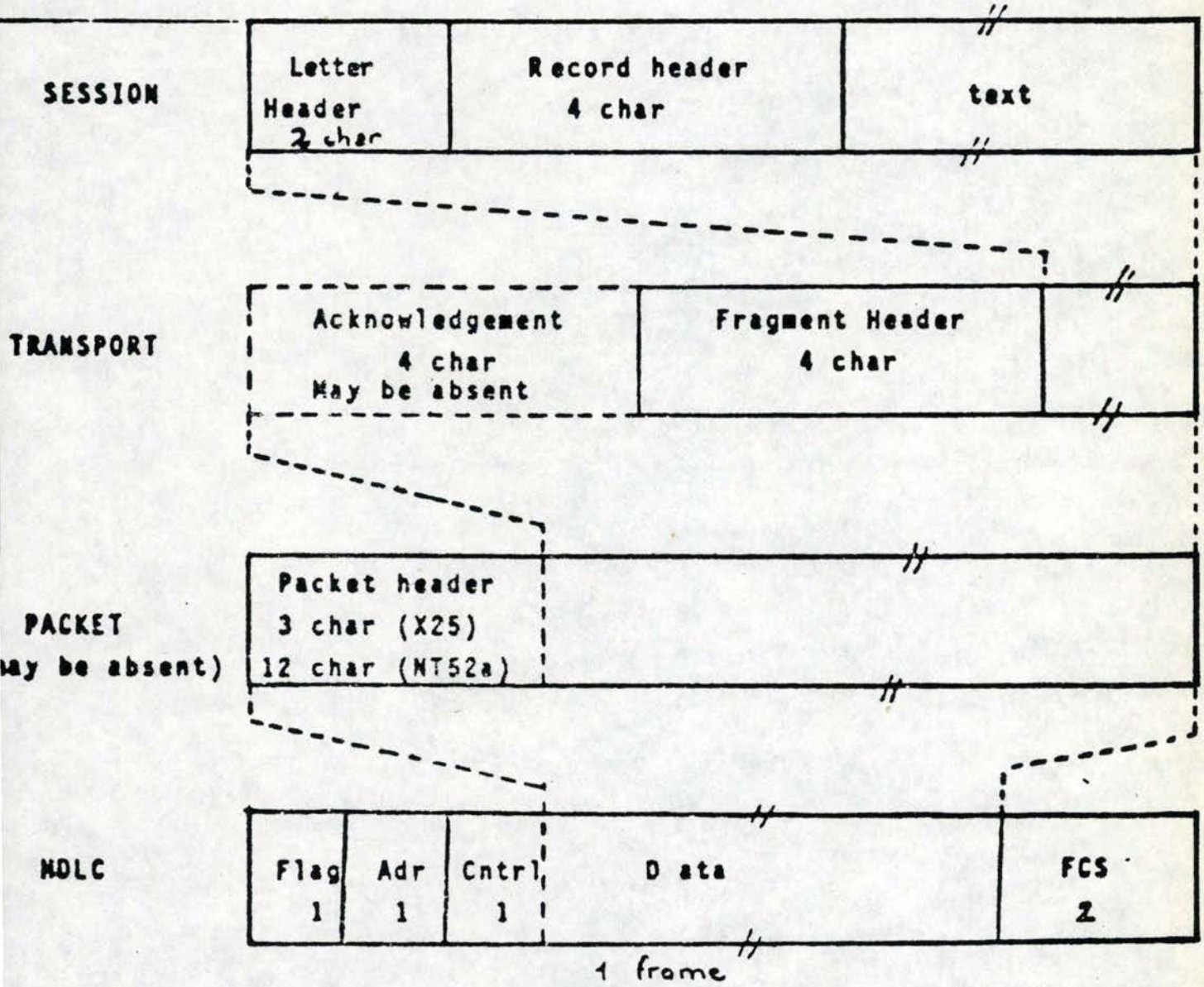
  The control-information usually consists of acknowledgments, and so its presence will be determined by the inverse traffic and the acknowledgment strategy of the Transport Station implementation. For most purposes, one may assume an overhead of 4 characters.

- 3 characters per X25 packet

- 12 characters per datagram packet

- 5 characters per frame

DSA PROTOCOLS / OVERHEAD



WORST CASES :

- NT52a 31 chars
- NT52b (X25) 22 chars
- X 21 and point-to - point 19 chars

FORM 4 DATANET 7100 MESSAGE PROCESSING TIME in Milliseconds

BASIC DNS STS 7112 or 7113

SECONDARY NETWORK

64/66

ASY STC 7105
T= (5,5 + 0,5 x B + 2 x nb of special characters)
Messages/sec:
LgIN:.... LgOUT:....

VIP STC 7106
T= (17 + 0,5 x B)
Messages/sec:....
LgIN:.... LgOUT:....

MINI6/PVE STC 7121
T= (17 + 0,5 x B)
Messages/sec:....
LgIN:.... LgOUT:....

PAD
T=(14 + 10 x P + 0,01 x L) x Mess./s.
XPAC / EURONET STC 7113
DATEX-P STC 7123
TELEPAC STC 7143

64/66

RCI STC 7119
T= (5,7 to 14,2) x
Messages/sec:.... } lines/s.
LgIN:.... LgOUT:.... } (132 ch.)

BSC 2780 STC 7109
T= 21 x
Messages/sec:.... } IN lines/s (80 ch.)
LgIN:.... LgOUT:.. } OUT lines/s (132 ch.)
T= 18 x

BSC 3270 STC 7108
T= x (25 + 0,015 x L)
Messages/sec:....
LgIN:.... LgOUT:....

66 only

DCU QUESTAR/T
T=(23 + 10 x P) x Mess./s.
XPAC / EURONET STC 7114
DATEX-P STC 7124
TELEPAC STC 7144

TMM-VU STC 7107
Messages/sec:....
LgIN:.... LgOUT:....

TMM-RB STC 7120
Messages/sec:....
LgIN:.... LgOUT:....

PARAMETER DEFINITIONS

-C Parameter: CPU Model power

C= 1 : DN7102 without DCE 7104
C= 0,83 : DN7102 with DCE 7104 or DN7103

EACH T MUST BE MULTIPLIED BY C

-L = (LgIN + LgOUT) / 2

-B= L - 1
Buffer size
System generation parameter which is a unique number.

-P= L + 7
Packet length
System generation parameter

-F= L - 1
Fragment size
System generation parameter

take the inferior integer

PUBLIC NETWORK

STANDARD	EXTENDED
XPAC / EURONET STC 7115	XPAC / EURONET STC 7116
DATEX-P STC 7125	DATEX-P STC 7126
TELEPAC STC 7145	TELEPAC STC 7146

T(1) = (14 + 10 x P) x Mess./s
T(2,3,4) = (20 + 10 x P) x Mess./s
T(M) = (21 + 10 x P) x Mess./s

DN1 STC 7135	DN1 STC 7136
NPDN STC 7117 x 1 (idem pt to pt)	NPDN STC 7118

PRIMARY NETWORK

T(1) : basic control , T(2,3,4) : flow control, recovery, full → default value
T(M) : Multipath

BASIC (pt to pt) STC 7101
T(1) = (9 + 8 x F) x Mess./s
T(2,3,4) = (11 + 10 x F) x Mess./s
T(M) = (12 + 11 x F) x Mess./s

T(1) = (12 + 8 x P) x Mess./s.
X25 PRIVATE STC 7103
T(2,3,4) = (18 + 8 x P) x Mess./s.

T= 14 x Packets/s
X25 PRIVATE SWITCHING STC 7104
T = 17 x Packets/s (Private - P.D.N.)

~~DATAGRAM STC 7102~~

HOST CONNECTION

GCOS 64 CONNECTION STM 7101
T= 5 x total traffic (mess. pack., AEP r./s.) records/s.

GCOS 3 CONNECTION STM 7103
T=(10 + 0,5 x B) x total traffic (mess., pack., AEP r./s) little links/s

NETWORK ADMINISTRATION

N.O.I STU 7111 || AUTO and or REMOTE LOAD STU 7112

NAD connected to a host LOGFILE of the Same SITE T= 16 x nb of AEP records/s.

NAD connected to a remote LOFFILE Point to Point Connection T= 26 x nb AEP/s.
X25 connection T= 46 x nb AEP r./s.

APPLICATIONS

FTF 626 : T = (11 + 10 x F + B + 1/x (21 + 0,5 B)) x nb of little links/s. 320 mots à 4 caract ASCII
FTF 66-66 — x every x little links ack.

RFA : T = 11 x nb of records/s. (records < 200 bytes)
FTF 64-64 T = (11 + 10 x F + B) x " " > "

LE TABLEAU DES LIGNES

	1200	2400	4800	9600	19200
0 - 720	1				
721 - 1440	2	1			
1441 - 2160	3	2	1		
2161 - 2880	4	2	1		
2881 - 4320		3	2	1	
4321 - 5760		4	2	1	
5761 - 8640			3	2	1
8641 - 11520			4	2	1

si trafic est supérieur à 11520 bps
il faut utiliser ligne à 48 Kbps ou
plusieurs lignes à 19200 bps ou
plusieurs lignes à 9600 bps.

6. ANNEXE FORMULAIRES

TRANSACTION TYPES DEFINITION

NETWORK MODEL - NAME :

TRANSACTION TYPES					
TRANSACTION TYPE ID	PROCESS TYPES EMPLOYED	TRANSMISSIONS			
		SENDING PROC. T.	RECEIVING PROC. T.	NBER OF MESS.	LENGTH (char)

AGENT TYPES DEFINITION

NETWORK MODEL - NAME :

AGENT TYPES					
AGENT TYPE ID.	WORKLOAD TYPE	TRANSACTION USAGE PROFILES			
		TRANSACTIONS INITIATED	PROF. 1	PROF. 2	PROF. 3

SITE - CLASSES DEFINITION

NETWORK MODEL - NAME :

SITE - CLASSES							
CLASS ID	NBER OF SITES	CONFIGURATION	PROCESSES ON SITE	INITIATING AGENTS AT SITE			
		COMPONENTS	TYPE - ID	TYPE-ID	NBER	INT.	PROC.

TRANSACTION LOCATION																	
TRANS-ACTION	SENDER			RECEIVER			MODE	PROPN.	MESSAGES PER TRANSACTION				CLASS ACT-IVITY	INTER CLASS TRAFFIC		
	PRO-CESS	SITE	COMP-ONENT	PRO-CESS	SITE	COMP-ONENT			#	LENGTH	LINK	# OF TRANSP. UNIT		# OF LETTERS	CHAR-ACTER VOLUME	# OF TRANSPORT UNITS

SITE TRANSACTION FREQUENCY											
SITE CLASS-ID	# OF SITES IN CLASS	TRANS-ACTION ID	AGENTS LOCATED AT SITE								SITE ACTIVITY	CLASS ACTIVITY

INTER - SITE TRAFFIC SYNTHESIS							
SENDING SITE CLASS		RECEIVING SITE CLASS		MODE	## OF LETTERS MESSAGES	CHAR. VOLUME (LESS OVERHEAD)	## OF TRANSPORT UNITS
ID	##	ID	##				

DIMENSIONNEMENT DE RESEAUX DSA

DOSSIER DE PROGRAMMATION

Version : 0.1 - avril 1982

Reférence : " SIZING A DSA NETWORK "
(sales aid 80 - 034 revisé)

CHAPITRE 1 : INTRODUCTION AU PROGRAMME PAGES

1. Introduction 1
2. Remarques - Le langage de programmation utilisé :
Pascal..... 2
3. Analyse de programmation - Généralités..... 3
3.1. Structure du programme - Introduction 3
3.1.1. Schéma 3
3.1.2. Description 3
3.2. Structure du programme - Modules 5
3.2.1. Introduction 5
3.2.2. Les modules 5
3.3. Mise en oeuvre sur un DPS8 / GCOS8 6
3.4. Structure du programme (1.) 7
3.4.1. Schéma 7
3.4.2. Description 8
3.4.3. Nécessité d'un programme "moniteur" 8

CHAPITRE 2 : MONITEUR ET DISPOSITIF DE SECURITE PAGES

1. Moniteur 9
1.1. Spécifications 9
1.2. Schéma 9
1.3. Description 10
1.4. Organigramme 11
1.5. Code Pascal correspondant 12
2. Protection des données de l'utilisateur 13
2.1. Introduction 13
2.2. Modèle de réseau 13
2.3. Schéma d'un exemple 13
2.4. Dispositif de sécurité 14
2.5. Accès concurrentiel 15
2.6. Remarques sur le fichier "catalogue" des modèles ... 16

CHAPITRE 3 : INTRODUCTION PAGES

1. Module 1 - Introduction - (1.1.) 17
2. Header - (1.1.1.) 18
3. Adjusting_form - (1.1.1.1.) 20
4. Printing_header - (1.1.1.2.) 21
5. Initializations - (1.1.2.) 22
6. Introduction_text - (1.1.3.) 24
7. System_guidance - (1.1.4.) 25
8. Access_control - (1.1.4.1.) 27
9. Initializations - (1.1.4.1.1.) 29
10. Control - (1.1.4.1.2.) 30

11. Seek - (1.1.4.1.2.1.) 33
12. Prepare_executions - (1.1.4.2.) 36
13. Sema - (1.1.4.2.1.) .....

CHAPITRE 4 : COLLECTE DES DONNEES - TRANSACTIONS PAGES

0. Remarques préliminaires 43
1. Module 2 - Data-collection - transactions - (1.2.) ... 44
2. Initializations - (1.2.1.) 45
3. Data_collection - (1.2.2.) 46
4. Transactions_types - (1.2.2.1.) 48
5. Print_transactions - (1.2.2.2.) 49
6. Modification - (1.2.2.3.) 50
7. Modify - (1.2.2.3.1.) 51
8. Insert - (1.2.2.3.2.) 53
9. Delete - (1.2.2.3.3.) 54
10. Closing - (1.2.2.3.4.) 55
11. Termination - (1.2.3.) 56

CHAPITRE 5 : COLLECTE DES DONNEES - AGENTS PAGES

1. Module 3 - Data-collection - agents - (1.3.) 58
2. Initializations - (1.3.1.) 59
3. Data_collection - (1.3.2.) 60
4. Agent_types - (1.3.2.1.) 62
5. Reenter_proportions - (1.3.2.1.1.) 63
6. Print_agents - (1.3.2.2.) 64
7. Modification - (1.3.2.3.) 65
8. Modify - (1.3.2.3.1.) 66
9. Insert - (1.3.2.3.2.) 68
10. Delete - (1.3.2.3.3.) 69
11. Closing - (1.3.2.3.4.) 70
12. Termination - (1.3.3.) 71

CHAPITRE 6 : COLLECTE DES DONNEES - SITES PAGES

1. Module 4 - Data-collection - sites - (1.4.) 73
2. Initializations - (1.4.1.) 74
3. Data_collection - (1.4.2.) 75
4. Sites - (1.4.2.1.) 77
5. Creation - (1.4.2.1.1.) 78
6. Print_sites - (1.4.2.2.) 79

7. Site_prof - (1.4.2.3.) 80
8. Modification - (1.4.2.4.) 81
9. Modify - (1.4.2.4.1.) 83
10. Insert - (1.4.2.4.2.) 84
11. Delete - (1.4.2.4.3.) 85
12. Closing - (1.4.2.4.4.) 86
13. Question_print - (1.4.2.5.) 87
14. Termination - (1.4.3.) 88

CHAPITRE 7 : COLLECTE DES DONNEES - LIENS PAGES

1. Module 5 - Link definition - (1.5.) 90
2. Initialization - (1.5.1) 91
3. Data_collection - (1.5.2.) 92
4. Trans_loc - (1.5.2.1.) 94
5. Initiating_sites - (1.5.2.1.1.) 95
6. Receiving_sites - (1.5.2.1.2.) 96
7. Proposition_link - (1.5.2.1.3.) 97
8. Print_trans_loc - (1.5.2.2.) 98
9. Handling - (1.5.2.3.) 99
10. Print - (1.5.2.4.) 100
11. Redefine - (1.5.2.5.) 101
12. Termination - (1.5.3.) 102

CHAPITRE 8 : CALCULS DE DIMMENSIONNEMENT PAGES

1. Module 6 - Computations - (1.6.) 104
2. Initializations - (1.6.1.) 105
3. Computing_schedule - (1.6.2.) 106
4. Class_activity - (1.6.2.1) 107
5. Compute - (1.6.2.1.1.) 108
6. Trans_loc - (1.6.2.2.) 110
7. Calcul_o - (1.6.2.2.1.) 111
8. Calcul_z - (1.6.2.2.1.1.) 112
9. Calcul_r - (1.6.2.2.2.) 113
10. Calcul_u - (1.6.2.2.3.) 114
11. Spacings - (1.6.2.2.4.) 115
12. Inter_site - (1.6.2.3.) 116
13. Traffic_list - (1.6.2.3.1.) 117
14. Compute_list - (1.6.2.3.2.) 119
15. Calcul_e - (1.6.2.3.2.1.) 120
16. Calcul_h - (1.6.2.3.2.2.) 121
17. Synthesis - (1.6.2.3.3.) 122
18. Compute_net - (1.6.2.4.) 123
19. Datanet_evaluation - (1.6.2.4.1.) 124
20. Create_datanet_file - (1.6.2.4.1.0.) 126
21. Search - (1.6.2.4.1.0.1.) 127

22. Search_dn - (1.6.2.4.1.0.1.1.) 128
23. Search_host - (1.6.2.4.1.1.) 129
24. Datanet_primary - (1.6.2.4.1.2.) 130
25. Compute_dn_size - (1.6.2.4.1.2.1.) 131
26. Datanet_secondary - (1.6.2.4.1.3.) 132
27. Calcul_nbr_mess - (1.6.2.4.1.3.1.) 133
28. Comp_asy - (1.6.2.4.1.3.1.2.) 134
29. Comp_vip - (1.6.2.4.1.3.1.3.) 135
30. Comp_bsc2 - (1.6.2.4.1.3.1.4.) 136
31. Comp_pad - (1.6.2.4.1.3.1.5.) 137
32. Comp_dcu - (1.6.2.4.1.3.1.6.) 138
33. Comp_rci - (1.6.2.4.1.3.1.7.) 138
34. Comp_bsc1 - (1.6.2.4.1.3.1.8.) 139
35. Comp_transf - (1.6.2.4.1.3.1.9.) 140
36. Datanet_host - (1.6.2.4.1.4.) 141
37. Print_results - (1.6.2.4.1.5.) 142
38. Primary_links_evaluation - (1.6.2.4.2.) 143
39. Primary_links_initializations - (1.6.2.4.2.1.) 144
40. Print_primary_links - (1.6.2.4.2.2.) 145
41. Compute_links_size - (1.6.2.4.2.2.1.) 146
42. Profile_handling - (1.6.2.5.) 147

CHAPITRE 9 : TERMINAISON PAGES

1. Module 7 - Termination - (1.7.) 149

CHAPITRE 10 : PROCEDURES DE BASE PAGES

0. Remarque 151
1. Newpage - (0.1.) 152
2. Newline - (0.2.) 153
3. Filestore - (0.3.) 154
4. Star - (0.4.) 155
5. Decoding_of_string - (0.5.) 156
6. Comment - (0.6.) 157
7. Verify - (0.7.) 158
8. Roundup - (0.8.) 159
9. Service - (0.9.) 160

CHAPITRE 11 : DESCRIPTION DES FICHIERS PAGES

0. Remarques 161

1. Exitfile 162
2. Fcata 162
3. Fica 163
4. F 163
5. Fg 164
6. Ft 165
7. Fit 165
8. Ft2 166
9. Fa 166
10. Fia 167
11. Fa2 167
12. Fs 167
13. Fis 168
14. Fs2 168
15. Fileaux 169
16. Fl 169
17. Fileact 170
18. Filedn 171
19. Filecal 171
20. Fileint2 172
21. Fileint 173
22. Tableau récapitulatif 174

CHAPITRE 12 : PROGRAMME DE SERVICE PAGES

1. Updating 175

CHAPITRE 1 : INTRODUCTION AU PROGRAMME

1. INTRODUCTION

Le programme DSANETWK est la version automatisée du sales aid " SIZING A DSA NETWORK " développé par Claudie Chappuis.

La version révisée de ce sales aid constitue la base théorique du programme que nous avons développé.

Avant d'entrer dans les détails de l'analyse de programmation, rappelons brièvement quelle est la tâche du programme : " dimensionner un réseau DSA ".. Le programme dimensionne les liens entre les différents sites du réseau, ensuite il évalue leur charge ainsi que la charge CPU des noeuds DSA. Actuellement les calculs se limitent au dimensionnement des ordinateurs de réseau DN 7100 (datanet), les résultats des expériences relatives aux MINI 6/DSS n'étant pas encore diffusés.

2. REMARQUES - LE LANGAGE DE PROGRAMMATION UTILISE : PASCAL

Antérieurement à nos efforts d'analyse nous avons dû choisir un langage de programmation. Ce choix a évidemment influencé en partie l'analyse par les contraintes imposées par le système d'exploitation GCOS 8 (General Comprehensive Operating System) tournant sur le DPS 8 (Data Processing System) du CTI à Paris / Gambetta sous TSS (Time Sharing System).

Nous avons choisi le langage PASCAL. Il y a plusieurs raisons qui nous ont amenés à ce choix. La première est que le PASCAL est un langage de haut niveau très proche de l'ALGOL 60, que nous connaissions; ceci nous a permis d'arriver plus tôt à des résultats acceptables.

La deuxième raison est que le PASCAL permet la programmation structurée, le langage possédant des instructions assez puissantes tout en restant très lisible ce qui présente un avantage non négligeable. Les raisons suivantes sont avant tout à chercher dans le matériel qui a été mis à notre disposition.

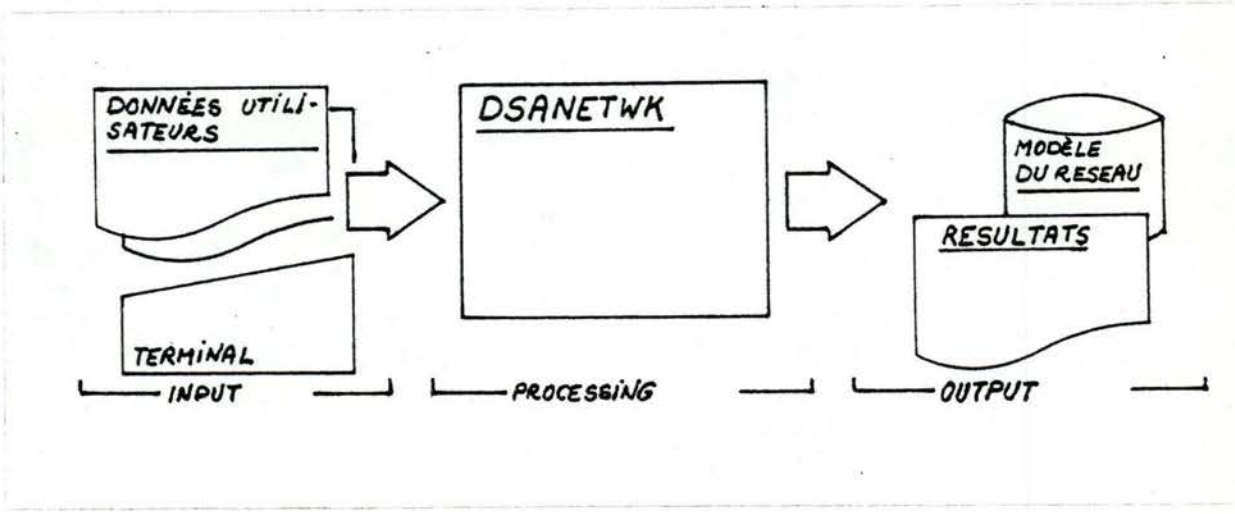
A l'intérieur de la compagnie, il y a des équipes qui utilisent le PASCAL pour la simulation, ce qui laisse supposer une vitesse d'exécution élevée.

Le compilateur PASCAL-66 respecte le PASCAL standard proposé par Kathleen Jensen et Niklaus Wirth; le compilateur a été développé à l'université de Waterloo (Canada). Le PASCAL-66 apporte des extensions considérables au standard. Par exemple : l'appel de procédures en langage B ou FORTRAN déclarées en externe; le lancement de code TSS à l'interieur d'un programme PASCAL. Il reste à préciser que les ordres d'entrées / sorties sont relativement simples à utiliser.

3. ANALYSE DE PROGRAMMATION - GENERALITES

3.1. STRUCTURE DU PROGRAMME - INTRODUCTION

3.1.1. SCHEMA



3.1.2. DESCRIPTION

* Entrées

Les entrées sont constituées par les informations concernant l'ensemble des applications de l'utilisateur mises en oeuvre sur le réseau sous étude.

* Sorties

Les sorties sont de deux natures : d'une part elles présentent les données du modèle du réseau à dimensionner et d'autre part elles fournissent les résultats du dimensionnement sous forme synthétique.

* Traitements

Les traitements sont spécifiés par le sales aid " SIZING A DSA NETWORK ". A cet ensemble de spécifications s'ajoutent un certain nombre de spécifications concernant des fonctionnalités qui facilitent la tâche de l'utilisateur du programme.

Le système assure d'abord une collecte des données utilisateur et procède ensuite à un traitement de ces informations pour déterminer la dimension des liens entre les différents sites et pour évaluer la charge CPU des ordinateurs de réseau " Datanet DN 7100 " installés sur les sites. Le programme assure un maximum d'assistance à l'utilisateur.

Les données entrées constituent le " modèle " du réseau de l'utilisateur. Ce modèle sera " stocké " en mémoire secondaire, ainsi il est modifiable et donc réutilisable. Les résultats calculés sont présentés au fur et à mesure de leur élaboration.

3.2. STRUCTURE DU PROGRAMME - MODULES

3.2.1. INTRODUCTION

Par module nous entendons un composant du système qui a des attributs biens définis et qui vérifie certaines propriétés. Par attributs on entend : les fonctions du module, c'est-à-dire ce qu'il doit faire; la structure interne, c'est-à-dire la manière avec laquelle il est couplé à d'autres modules de l'architecture; la dimension et les performances. Nous avons cherché à garder un couplage le plus faible entre les divers modules afin de limiter le nombre de connexions statiques entr les modules , mais tout en cherchant à réduire la complexité des interfaces au maximum. Le regroupement des fonctions à réaliser par module a été effectué dans une optique de recherche d'une cohésion interne forte, c'est-à-dire on voulait atteindre une forte interdépendance logique des actions dans un même module.

3.2.2. LES MODULES

Nous avons distingué 7 modules :

1) - le module " INTRODUCTION " ;
2) - le module " COLLECTE DES DONNEES RELATIVES AUX TRANSACTIONS " ;
3) - le module " COLLECTE DES DONNEES RELATIVES AUX AGENTS " ;
4) - le module " COLLECTE DES DONNEES RELATIVES AUX SITES " ;
5) - le module " LOCALISATION DES TRANSACTIONS " ;
6) - le module " CALCULS DE DIMENSIONNEMENT " ;
7) - le module " TERMINAISON " .

3.3. MISE EN OEUVRE SUR UN DPS8 / GCOS8

Le programme que nous avons développé à Paris, a été écrit pour tourner sur un DPS8 / GCOS8 travaillant en mode de temps partagé. Ce programme est un prototype qui est, rappelons-le, conforme aux exigences du sales aid " SIZING A DSA NETWORK ". Il faut cependant exposer un certain nombre de contraintes qui ont fortement influencé la structure du système mis en oeuvre.

Le matériel et logiciel du CTI que nous utilisions nous ont imposé certaines contraintes qu'il fallait respecter. Le système d'exploitation alloue à chaque utilisateur du système à temps partagé un espace mémoire recevant les fichiers temporaires utilisés lors de l'exécution de programmes. Un dépassement de la capacité de stockage de cette zone conduit à une fin prématurée du programme en exécution. Il s'agit d'une terminaison normale.

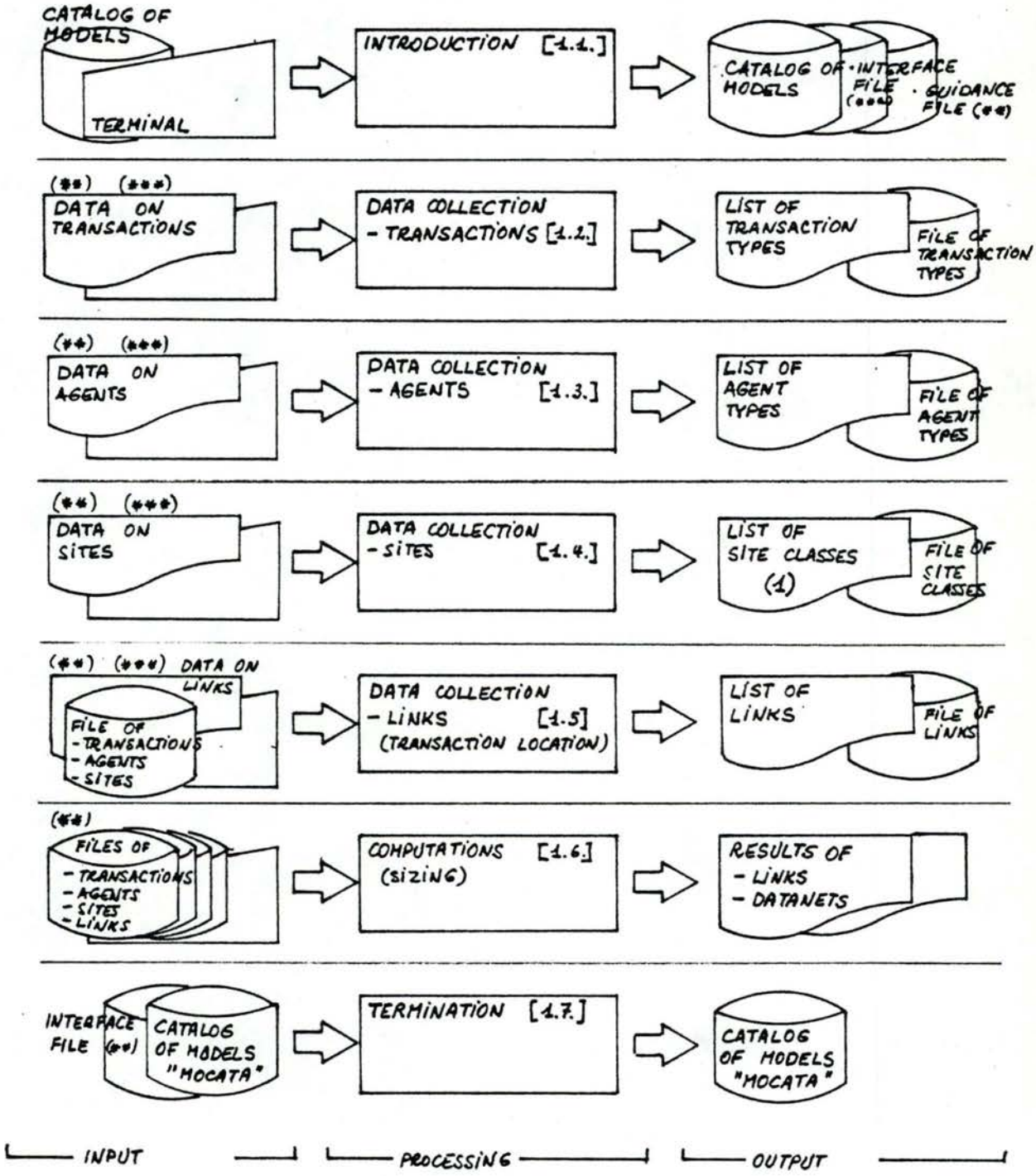
Le TSS (Time Sharing System) donne à l'utilisateur la possibilité de gérer la zone des fichiers temporaires. A cet effet l'utilisateur peut demander soit un vidage complèt de la zone, soit un effacement sélectif, c'est-à-dire de un ou plusieurs fichiers. A ces deux commandes s'ajoute la commande de transformation d'un fichier temporaire en un fichier permanent.

Ces commandes permettent d'éviter le problème de dépassement de la capacité de stockage. Cependant l'utilisateur doit veiller à bien appliquer ces commandes. Syntaxiquement le superviseur accepte ces ordres à n'importe quel moment d'une session TSS; l'expérience nous a quand- même montré qu'il ne faut pas lancer ces ordres à partir d'un programme PASCAL, bien que faisable.

La solution à prendre est de lancer ces commandes à l'extérieur des programmes travaillant sur les fichiers localisés dans la zone temporaire. Ceci nous a contraints de concevoir une architecture en modules où chacun des modules est matérialisé par un programme PASCAL en tant que tel.

3.4. STRUCTURE DU PPROGRAMME - NIVEAU 1

3.4.1. SCHEMA



3.4.2. DESCRIPTION

- le catalogue des modèles : fait partie du dispositif de sécurité du système; le catalogue contient les noms des modèles, les mots de passe associés et les états des modèles.

- le fichier guide : pour chaque modèle il faut connaitre l'avancement des travaux sur ce modèle.

- le fichier interface : contient lors de l'exécution du système le nom du modèle et son mot depasse associé.

(1) le sales aid prévoit la possibilité d'utiliser 3 profils d'activité des agents. Cette possibilité a été mise en oeuvre dans l'outil automatique, et dès qu'on a défini lors de la collecte des données " agents " l'utilisation de plusieurs profils, il faut exécuter toutes les opérations autant de fois qu'il y a des profils.

3.4.3. NECESSITE D'UN PROGRAMME " MONITEUR "

Avant l'exécution de chacun des modules cités ci-dessus, on va procéder à un vidage de l'espace des fichiers temporaires à l'exception du fichier interface. Il est évident que tous ces détails de mise en oeuvre du système doivent rester transparents pour un utilisateur du programme. Cette exigence entraîne la nécessité d'un programme " moniteur " qui à la fois gère l'espace des fichiers temporaires et lance les différents modules du système.

CHAPITRE 2 : MONITEUR ET DISPOSITIF DE SECURITE

1. MONITEUR

1.1. SPECIFICATIONS

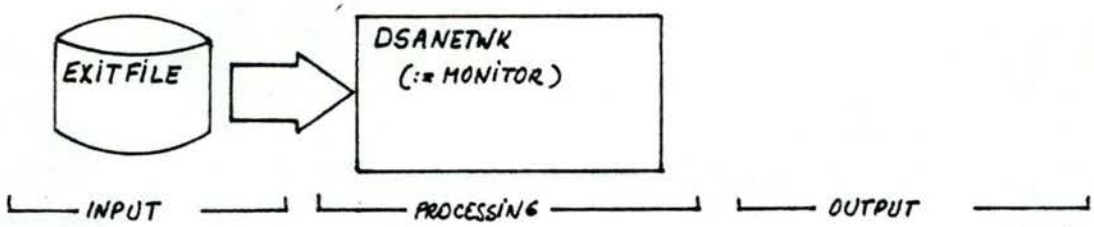
Le moniteur doit gérer d'une part l'espace des fichiers temporaires et d'autre part il doit assurer le lancement des sept modules.

Avant le lancement du premier module (introduction) et après l'exécution du septième (terminaison), le moniteur assure un vidage total de l'espace des fichiers temporaires. Entre chaque module appelé , un vidage partiel est assuré par le moniteur; c'est-à-dire tous les fichiers sont enlevés de la zone temporaire à l'exception du fichier interface sans lequel le programme ne peut pas être exécuté.

Si à la sortie du premier module l'utilisateur n'a pas eu de permission de travail, le moniteur doit immédiatement terminer l'exécution.

Entre les modules 1,2,3,4,5 et 6 le moniteur permet à l'utilisateur de (1) réexécuter le module qu'on vient de quitter, (2) de sortir du programme (arrêt du travail, reprise ultérieure) ou (3) d'aller au pas suivant (exécuter le module suivant).

1.2. SCHEMA



1.3. DESCRIPTION

1.3.1. Entrées

- fichier de travail "exitfile", recevant une valeur durant l'exécution du module 1. (cfr.: descriptions des fichiers en annexe de ce dossier).

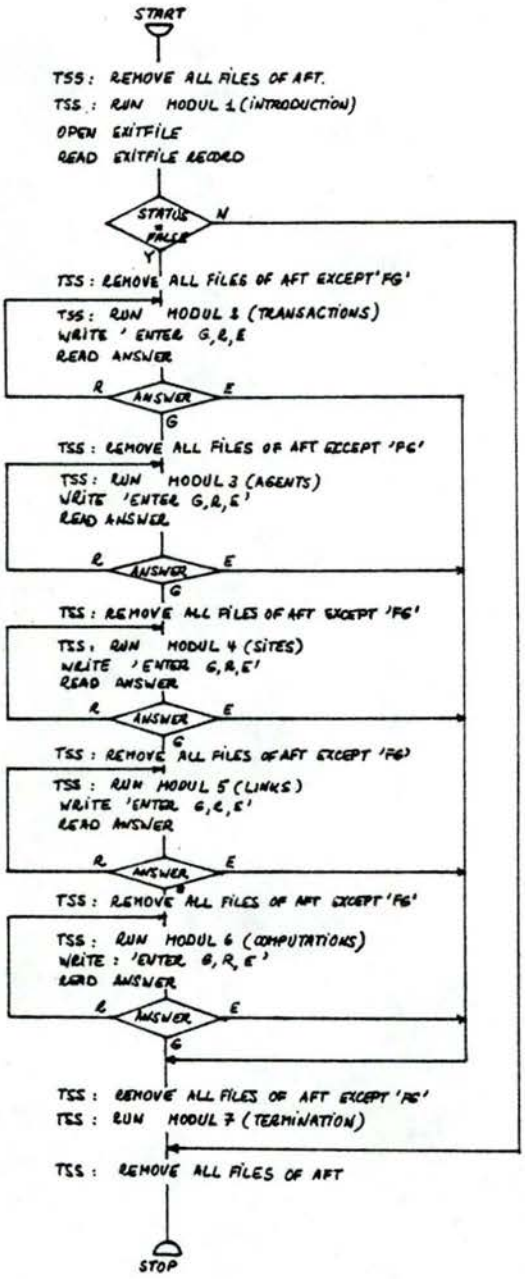
1.3.2. Sorties

- pas de sorties.

1.3.3. Traitements

- cfr.: schéma et organigramme.

1.3.4. Organigramme



1.3.5. Code PASCAL correspondant

cfr.: programme DSANETWK

2. PROTECTION DES DONNEES DE L'UTILISATEUR

2.1. INTRODUCTION

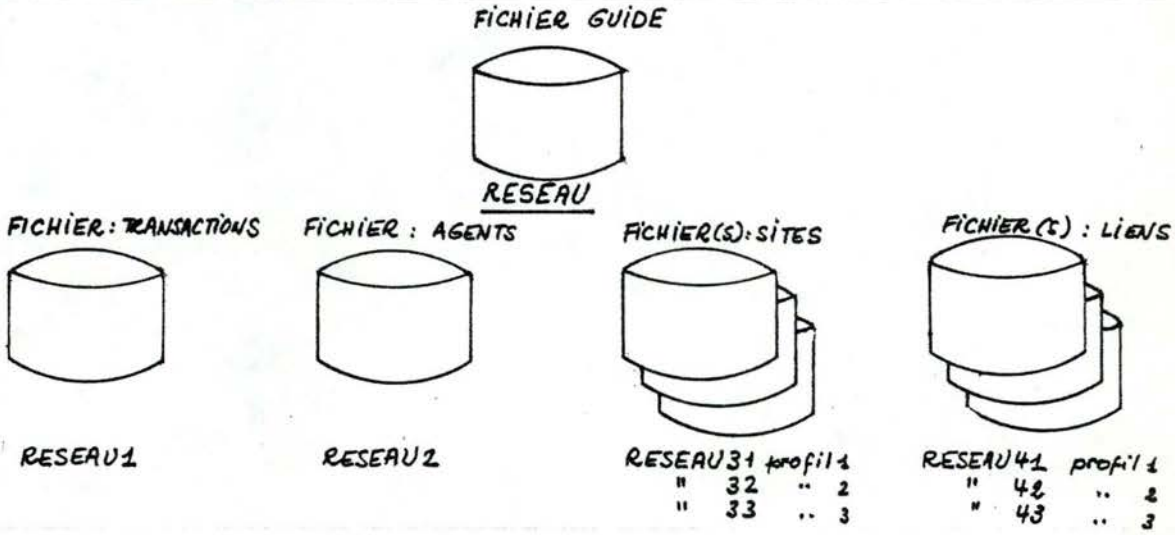
Les données de l'utilisateur constituant le "modèle" doivent être sauvées de façon permanente. Ceci permet une utilisation ultérieure de ces données.

2.2. MODELE DE RESEAU

Par modèle nous entendons un ensemble de données de types différents qui décrivent un système d'informatique distribuée. A chaque type de données correspond un fichier. Ainsi le modèle d'un réseau est constitué par les fichiers suivants : "transactions"; "agents";"sites" (1 par profil utilisé) et "liens" (1 par profil utilisé). A ces fichiers s'ajoute un fichier "guide" dont le rôle est de suivre l'état d'avancement des travaux de l'utilisateur sur son modèle.

2.3. SCHEMA D'UN EXEMPLE

Nous allons présenter de façon schématique les fichiers d'un exemple de modèle de réseau. Le nom de ce modèle est : "RESEAU" et il est constitué par l'ensemble des fichiers suivants:



Il reste à remarquer que l'utilisateur donne simplement le nom du modèle de réseau (dans l'exemple ci-dessus : RESEAU), c'est le programme qui se charge de la gestion des fichiers. En fait la notion de fichier est complètement transparente pour un utilisateur.

2.4. DISPOSITIF DE SECURITE

Pour éviter tout accès illégal et toute manipulation accidentielle, le modèle doit etre protégé. Pour réaliser cette protection nous associons à chaque nom de modèle un mot de passe. L'ensemble de tous les modèles est répertorié dans un fichier qui constitue le catalogue des modèles.

Chaque fois qu'un utilisateur demande un accès à un modèle il faut effectuer des contrôles : l'utilisateur spécifie s'il désire créer un nouveau modèle ou s'il souhaite accéder à un modèle existant. Ensuite le programme lui demande le nom et le mot de passe associé du modèle sur lequel il veut travailler. Un contrôle de validité sera assuré grâce à l'existance du catalogue des modèles.

En fonction du contrôle des données reçues, le programme doit réaliser certaines actions. La table de décision montre les détails de ce travail de contrôle.

création d'un modèle	Y	Y	Y	Y	N	N	N	N
nom existe	Y	Y	N	N	Y	Y	N	N
mot de passe ass. co.	Y	N	Y	N	Y	N	Y	N
actions	1	2	3	3	3	4	5	5

cette table de décisions peut etre réduite :

création d'un modèle	Y	Y	Y	N	N	N
nom existe	Y	Y	N	Y	Y	N
mot de passe associé correct	Y	N	-	Y	N	-
actions	1	2	3	3	4	5

Les actions

1. En principe on peut créer un modèle, mais un modèle sous ce mot et mot de passe existe déjà. Il faut prévenir l'utilisateur des conséquences de son action : il écrasera l'ancien modèle. A ce niveau il faut donner à l'utilisateur la possibilité de choisir un nouveau nom et un nouveau mot de passe.

2. On ne permet pas la création, car même si le couple (nom et mot de passe) n'existe pas, il y a une incompatibilité au niveau du nom du fichier. On demande à l'utilisateur de choisir un autre nom pour son modèle.

3. On permet de créer un nouveau modèle ou d'accéder à un modèle existant.

4. On ne permet pas l'accès au modèle car le mot de passe founi par l'utilisateur est incorrect.

5. Un modèle sous ce nom n'existe pas, donc pas de permission d'accès.

Le contrôle sera répétitif, toutefois le nombre d'essais est limité à 3. Si aucune permission n'a été donnée, le fichier 'exitfile' contient les informations nécessaires pour indiquer au programme moniteur de terminer immédiatement l'exécution du programme.

2.5. ACCES CONCURRENTIEL

Le dispositif décrit ci-dessus n'est pas encore suffisant pour résoudre les problèmes résultants de l'exécution en parallèle du programme par un ou plusieurs utilisateurs accédant au même modèle. Exemple : l'utilisateur 1 travaille sur MOD, lorsqu'il entre les informations relatives aux liens l'utlisateur

2 commence à travailler sur MOD. Ainsi il y aura des problèmes de cohérence des données.

Ce problème est résolu par le mécanisme des sémaphores. A chaque couple (nom et mot de passe d'un modèle) on associe un sémaphore qui peut prendre deux valeurs : T (true) = vrai et F (false) = faux. Par convention si le sémaphore est T, alors le modèle est inaccessible. Sinon le modèle est accessible. Ce contrôle d'accessibilité du modèle est effectué après avoir donné à l'utilisateur la permission de créer un nouveau modèle ou d'accéder à un ancien modèle. Si le modèle est inaccessible le fichier 'exitfile' contient les informations nécessaires pour indiquer au moniteur de terminer immédiatement l'exécution du programme.

2.6. REMARQUES SUR LE FICHIER "CATALOGUE DES MODELES"

Le fichier "catalogue" est la base du dispositif de sécurité. Ce fichier contient les informations suivantes par modèle :

- nom du modèle;
- mot de passe associé;
- valeur du sémaphore d'accès.

Chaque fois qu'un nouveau modèle est crée, on ajoute les nouvelles informations à la fin de ce fichier. Si le système est souvent utilisé, le fichier risque de grossir considérablement. De temps en temps il faut procéder à une mise à jour du fichier. Cette mise à jour consiste avant tout à retirer les modèles qui ne présentent plus aucun intérêt commercial. Cette fonction est assurée par un programme spécial de maintenance qui est détaillé au chapitre 12 de ce dossier.

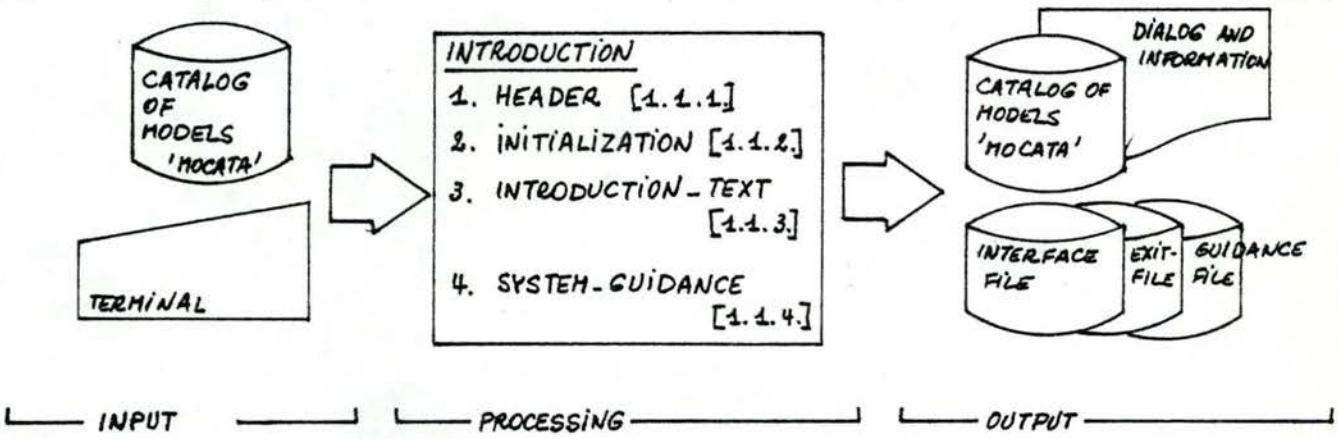
CHAPITRE 3 : INTRODUCTION

1. MODULE 1 - INTRODUCTION - NIVEAU D'ANALYSE 1.1.

1.1. SPECIFICATIONS

Le module "introduction" doit réaliser un certain nombre de fonctions qui ne sont pas contenues dans le sales aid, mais qui résultent avant tout de l'automatisation de ce dernier. D'une part l'utilisateur doit être informé brièvement sur le fonctionnement du système, d'autre part ce module doit effectuer les initialisations requises du point de vue sécurité et du point de vue collecte des données. Ces fonctions seront spécifiées avec plus de détails aux niveaux d'analyse suivants.

1.2. SCHEMA



1.3. DESCRIPTION

1.3.1. Entrées

- fichier catalogue des modèles, (fcata - mocata);
- entrées manuelles à partir du terminal.

1.3.2. Sorties

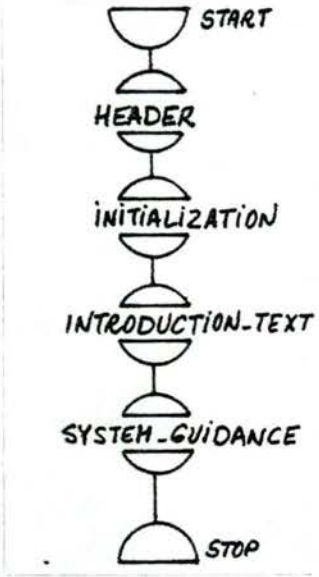
- fichier catalogue des modèles;
- fichier interface entre les modules, (fg -);
- fichier contenant les informations relatives à la

permission d'accès au modèle, (exitfile -);
- fichier d'avancement des travaux sur le modèle, (f - <nom_du_modèle>);
- sorties (dialogue) au terminal (télétype).

1.3.3. Traitements

- cfr.: schéma et organigramme.

1.4. ORGANIGRAMME



1.5. CODE PASCAL CORRESPONDANT

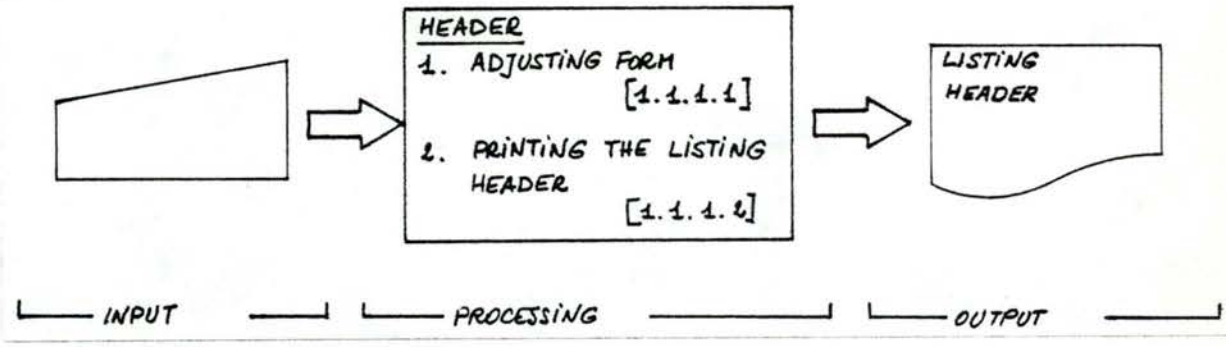
cfr.: programme INTRODUC - (1.1.)

2. HEADER - NIVEAU D'ANALYSE 1.1.1.

2.1. SPECIFICATIONS

Le système actuel est écrit pour qu'il puisse tourner sur des terminaux du type télétype. La première tâche est dès lors, de demander à l'utilisateur s'il veut avoir l'option de saut de page ou non. Si cette option n'est pas souhaitée, elle est automatiquement remplacée par un saut de ligne. Le choix de l'option reste valable pendant toute la durée de l'exécution du programme. Après avoir collecté cette information , le programme permet d'ajuster le papier sur l'imprimante. La deuxième tâche est de produire une page de garde pour le listing qui va recevoir le dialogue et les résultats.

2.2. SCHEMA



2.3. DESCRIPTION

2.3.1. Entrées

- réponses en provenance du terminal.

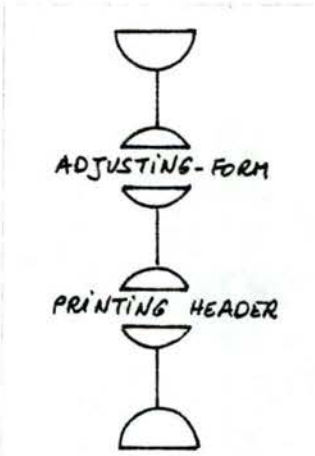
2.3.2. Sorties

- entête de listing.

2.3.3. Traitements

cfr.: schéma et organigramme.

2.4. ORGANIGRAMME



2.5. CODE PASCAL CORRESPONDANT

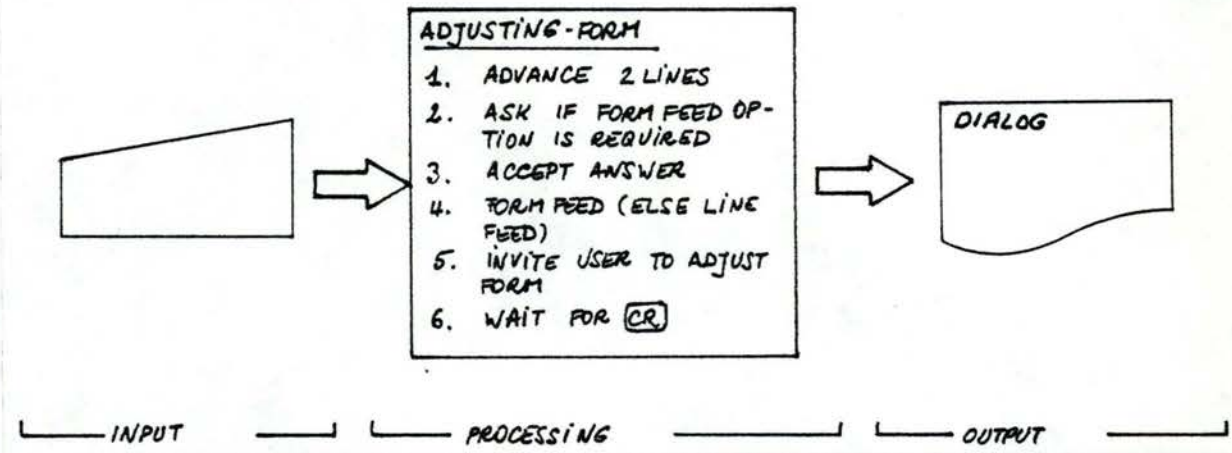
cfr.: programme INTRODUC - (1.1.1.)

3. ADJUSTING FORM - NIVEAU D'ANALYSE 1.1.1.1.

3.1. SPECIFICATIONS

La tâche de cette partie est de poser la question de l'option du saut de page et de permettre l'ajustement du papier sur l'imprimante.

3.2. SCHEMA



3.3. DESCRIPTION

3.3.1. Entrées

- réponses en provenance du terminal.

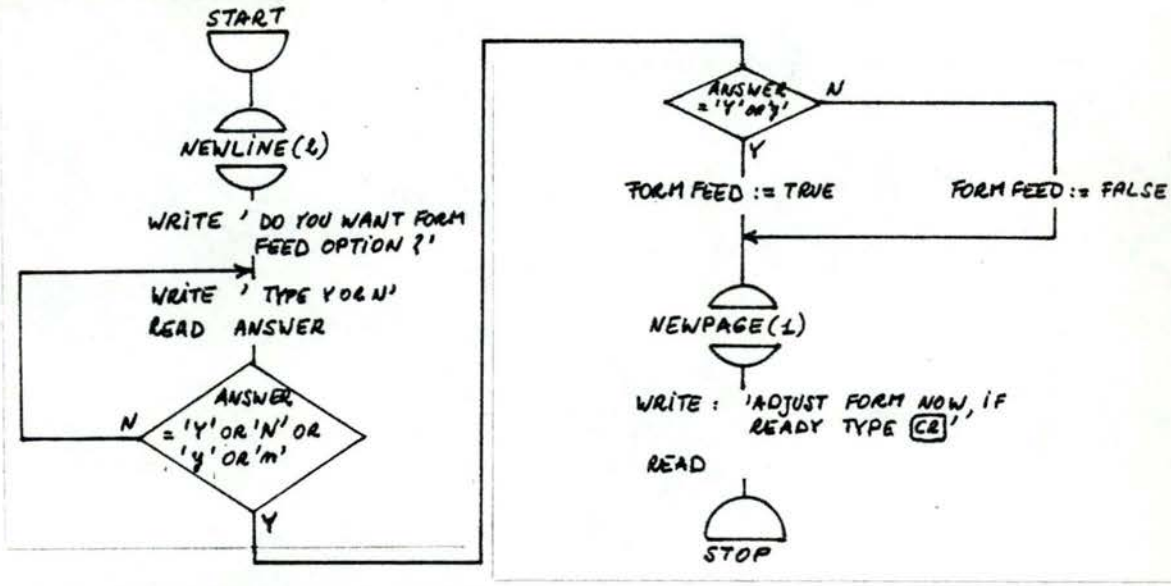
3.3.2. Sorties

- entête de listing.

3.3.3. Traitements

- cfr.: schéma et organigramme.

3.4. ORGANIGRAMME



3.5. CODE PASCAL CORRESPONDANT

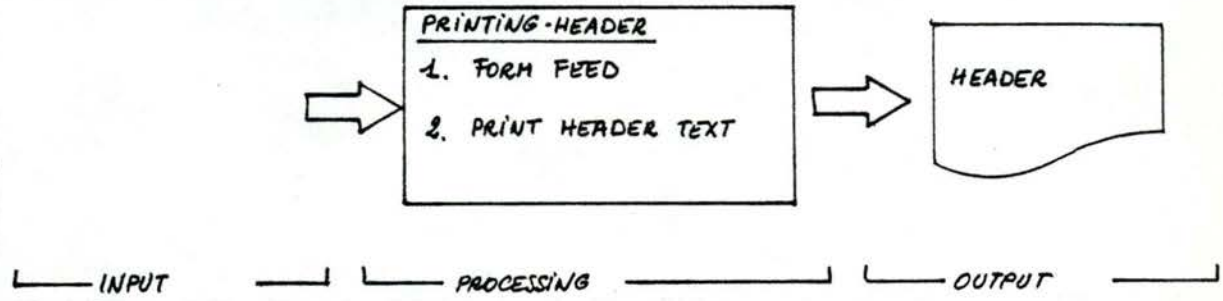
cfr.: programme INTRODUC - (1.1.1.1.)

4. PRINTING_HEADER - NIVEAU D'ANALYSE 1.1.1.2.

4.1. SPECIFICATIONS

Cette partie effectue un saut de page (ou saut de ligne selon le choix de l'utilisateur) et imprime l'entête du listing qui a été présentée dans le manuel d'utilisation de l'outil.

4.2. SCHEMA



4.3. DESCRIPTION

4.3.1. Entrées

- pas d'entrées.

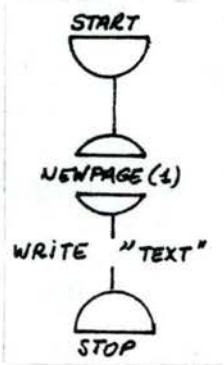
4.3.2. Sorties

- entête de listing.

4.3.3. Traitements

- cfr.: schèma et organigramme

4.4. ORGANIGRAMME



4.5. CODE PASCAL CORRESPONDANT

cfr.: programme INTRODUC - (1.1.1.2.)

5. INITIALIZATION - NIVEAU D'ANALYSE 1.1.1.

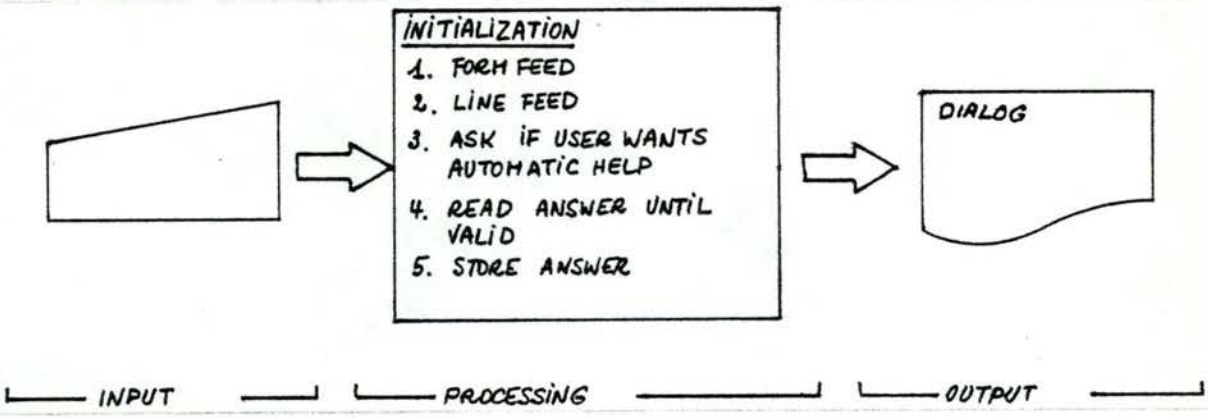
5.1. SPECIFICATIONS

Le système est confronté à deux types d'utilisateurs : ceux qui ne connaissent pas le sytème et ceux qui sont déjà initiés. Le programme a été développé dans une optique de "self

- explaining", c'est-à-dire il fournit à chaque question des commentaires. Pour satisfaire les deux groupes d'utilisateurs, nous avons mis en oeuvre deux mécanismes : à savoir le "point d'interrogation" et l'aide automatique. L'aide automatique est conçue pour les utilisateurs non initiés. A la première occurrence de chaque question, l'outil fournit automatiquement quelques lignes de commentaires. Pour les occurrences suivantes de chaque question, l'utilisateur non initié peut se servir du point d'interrogation. Evidemment l'utilisateur initié peut aussi se servir du point d'interrogation.

La tâche de cette partie est de demander l'utilisateur s'il veut ou ne veut pas avoir l'aide automatique.

5.2. SCHEMA



5.3. DESCRIPTION

5.3.1. Entrées

- réponse en provenance du terminal.

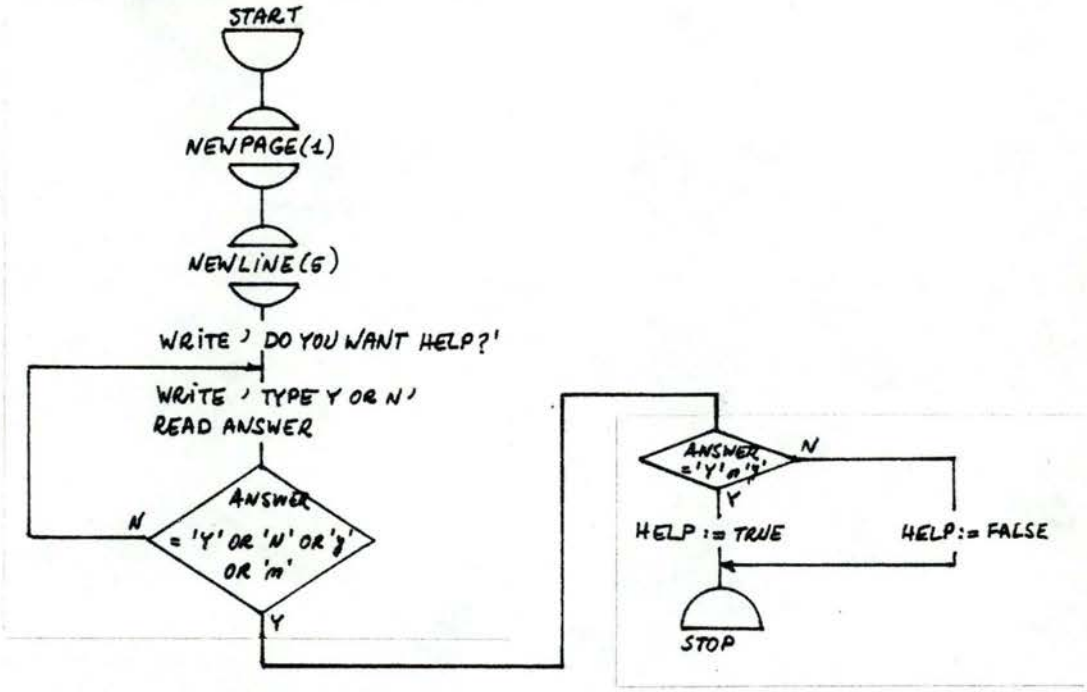
5.3.2. Sorties

- question affichée au terminal.

5.3.3. Traitements

- cfr.: schéma et organigramme.

5.4. ORGANIGRAMME



5.5. CODE PASCAL CORRESPONDANT

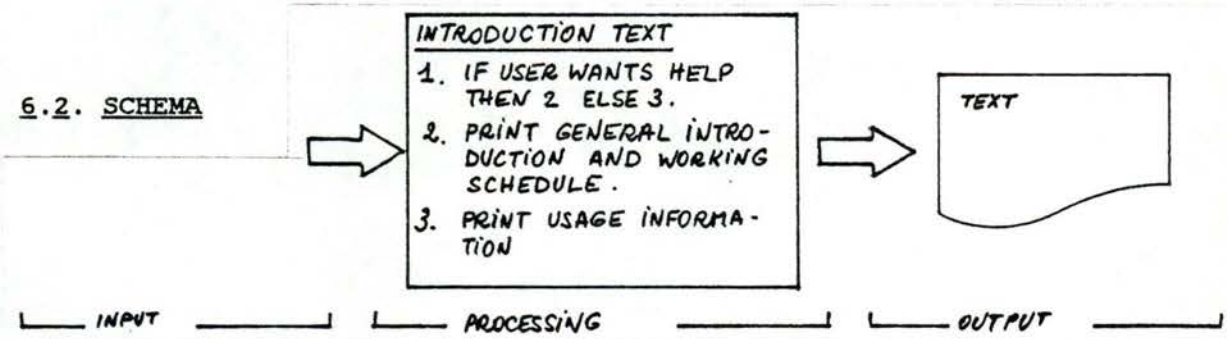
cfr.: programme INTRODUC - (1.1.1.)

6. INTRODUCTION_TEXT - NIVEAU D'ANALYSE 1.1.3.

6.1. SPECIFICATIONS

Cette partie réalise les tâches suivantes: Si l'utilisateur désire avoir l'aide automatique on lui fournit une introduction générale et des informations utiles à la compréhension des questions qui vont être posées. Elle fournit ensuite des informations d'ordre général à tout utilisateur.

6.2. SCHEMA



6.3. DESCRIPTION

6.3.1. Entrées

- pas d'entrées.

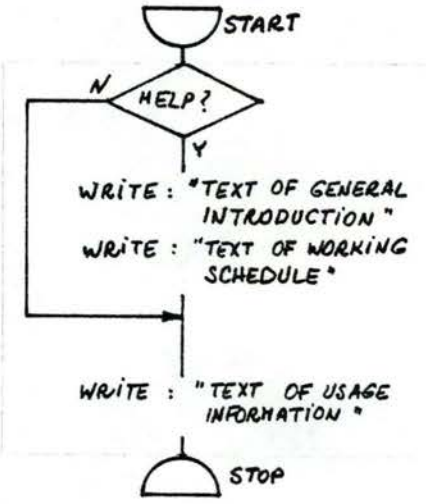
6.3.2. Sorties

- "general introduction" et "information for novice user" (présentés dans le manuel d'utilisation), (si l'utilisateur a demandé l'aide automatique);
- informations d'utilisation au terminal.

6.3.3. Traitements

- cfr.: schéma et organigramme.

6.4. ORGANIGRAMME



6.5. CODE PASCAL CORRESPONDANT

cfr.: programme INTRODUC - (1.1.3)

7. SYSTEM GUIDANCE - NIVEAU D'ANALYSE 1.1.4.

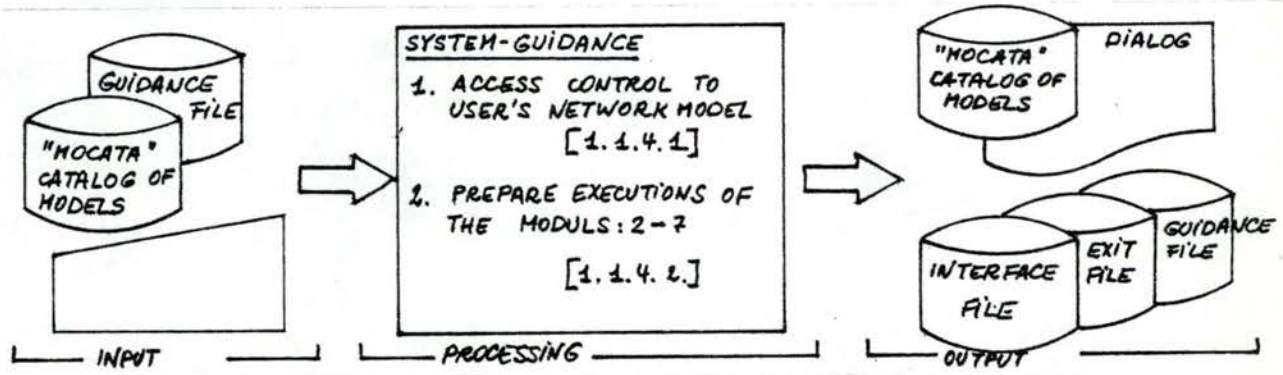
7.1. SPECIFICATIONS

Comme son nom l'indique, la partie "system_guidance" doit rassembler toutes les informations nécessaires pour initialiser et protéger le modèle de réseau de l'utilisateur. Les informations constituent la base des contrôles prévus dans le dispositif de sécurité que nous avons spécifié au deuxième chapitre. Les informations ainsi collectées servent à garnir le fichier d'interface 'fg' et le fichier de guidage 'f'.

Le dispositif de sécurité contrôle l'accès aux modèles existants et la création de nouveaux modèles. Il utilise le fichier catalogue des modèles 'mocata' comme référence de ces contrôles. Si la création ou l'accès est permis, le système doit mettre à jour , c'est-à-dire créer ou modifier le fichier de guidage et initialiser le fichier interface. Ce fichier interface contiendra le nom du modèle et le mot de passe associé. Le fichier de guidage contient toutes les informations concernant l'avancement des travaux sur le modèle, c'est-à- dire les étapes déjà franchies, les profils utilisés etc.). Ce fichier doit être initialisé lors de la création de modèle et lors d'un accès uniquement "l'aide automatique" sera mise à jour. Les refus d'accès ou de création sont toujours suivis d'un commentaire expliquant la cause.

Finalement le fichier 'exitfile' sera garni en fonction des décisions prises par le programme. L'information contenue dans ce fichier indiquera au moniteur si l'exécution du programme peut se poursuivre ou si elle doit être terminée.

7.2. SCHEMA



7.3. DESCRIPTION

7.3.1. Entrées

- fichier catalogue des modèles, (fcata - mocata);
- fichier de guidage, si accès à un modèle existant, (f - <nom_du_modèle>);
- réponses en provenance du terminal.

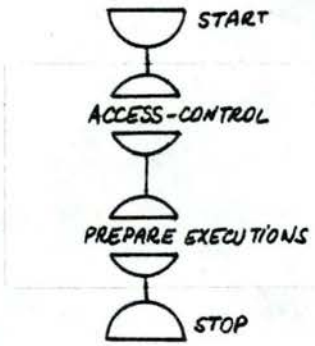
7.3.2. Sorties

- fichier contrôle de sortie, (exitfile -);
- fichier de guidage;
- fichier interface, (fg -);
- fichier catalogue des modèles;
- questions et messages affichés au terminal.

7.3.3. Traitements

- cfr.: schéma et organigramme.

7.4. ORGANIGRAMME



7.5. CODE PASCAL CORRESPONDANT

cfr.: programme INTRODUC - (1.1.4.)

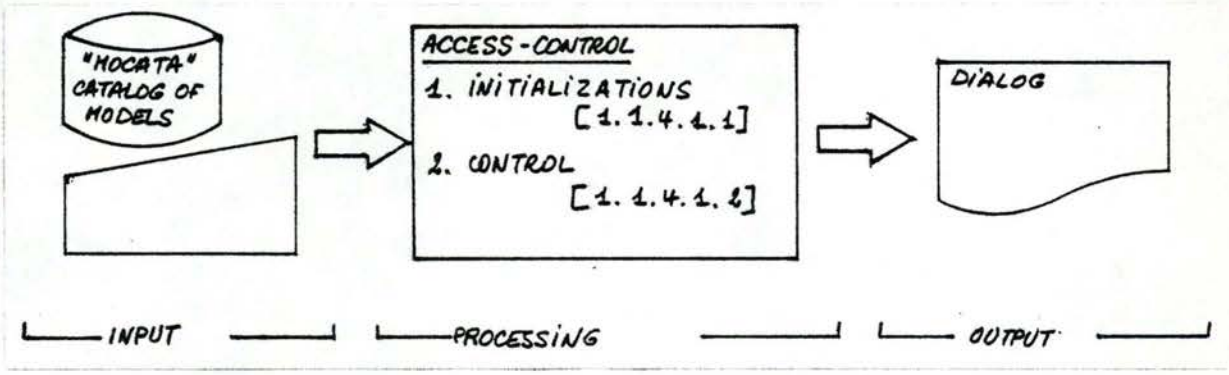
8. ACCESS_CONTROL - NIVEAU D'ANALYSE 1.1.4.1.

8.1. SPECIFICATIONS

Cette partie est chargée de contrôler l'accès au modèle

du réseau spécifié par l'utilisateur. Afin de réaliser cette tâche cette partie doit procéder aux initialisations nécessaires et ensuite au contrôle proprement dit.

8.2. SCHEMA



8.3. DESCRIPTION

8.3.1. Entres

- fichier catalogue des modèles, (fcata - mocata);
- entrées manuelles à partir du terminal.

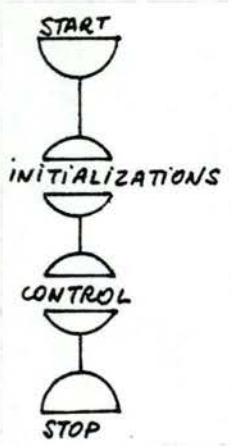
8.3.2. Sorties

- messages au terminal.

8.3.3. Traitements

- cfr.: schéma et organigramme.

8.4. ORGANIGRAMME



8.5. CODE PASCAL CORRESPONDANT

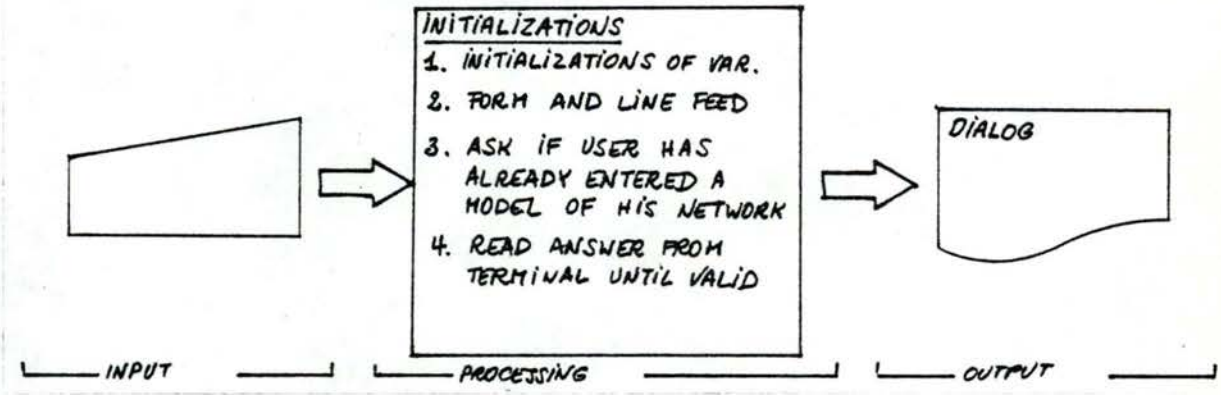
cfr.: programme INTRODUC - (1.1.4.1.)

9. INITIALIZATIONS - NIVEAU D'ANALYSE 1.1.4.1.1.

9.1. SPECIFICATIONS

Cette partie assure les initialisations nécessaires au contrôle d'accès au modèle. Elle détermine si l'utilisateur a déjà ou n'a pas encore introduit de modèle de réseau.

9.2. SCHEMA



9.3. DESCRIPTION

9.3.1. Entrées

- réponses en provenance du terminal.

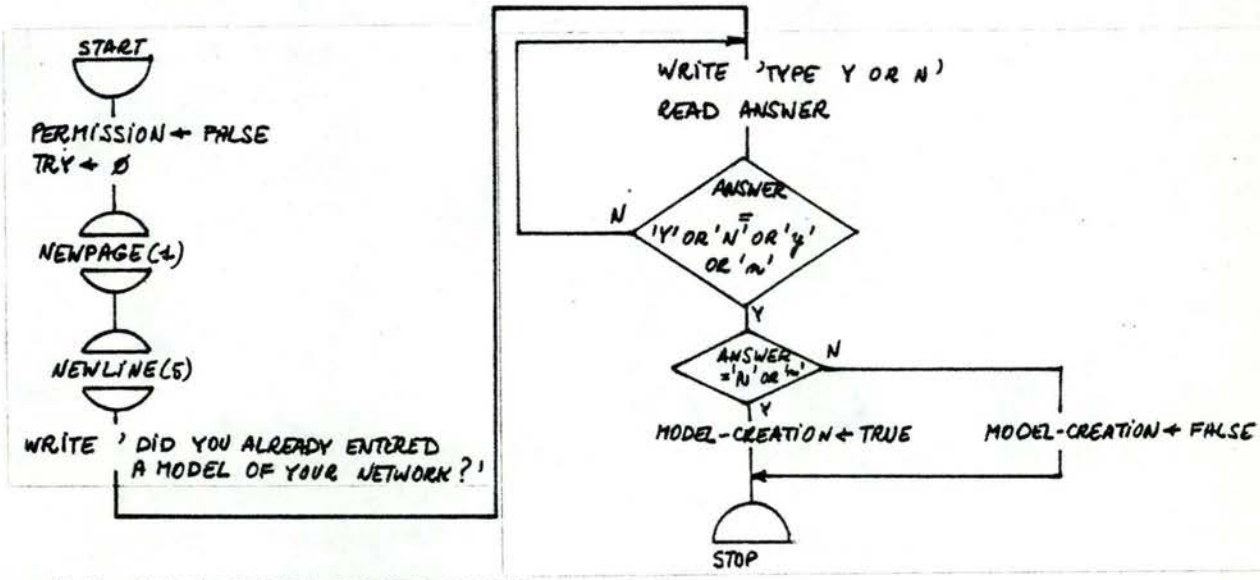
9.3.2. Sorties

- messages affichés au terminal.

9.3.3. Traitements

- cfr.: schèma et organigramme.

9.4. ORGANIGRAMME



9.5. CODE PASCAL CORRESPONDANT

cfr.: programme INTRODUC - (1.1.4.1.1.)

10. CONTROL - NIVEAU D'ANALYSE 1.1.4.1.2.

10.1. SPECIFICATIONS

Cette partie est responsable du contrôle de l'accès au modèle de rèseau. Elle demande le nom et le mot de passe associè du modèle auquel l'utilisateur dèsire accèder. Cette partie fait appel à une fonction de contrôle renvoyant un statut du modèle conformèment aux spècifications du dispositif de sècurite prèsentè au chapitre 2.

10.2. SCHEMA

(1)

10.3. DESCRIPTION

10.3.1. Entrèes

- rèponses en provenance du terminal;
- fichier catalogue des modèles, (fcata - mocata).

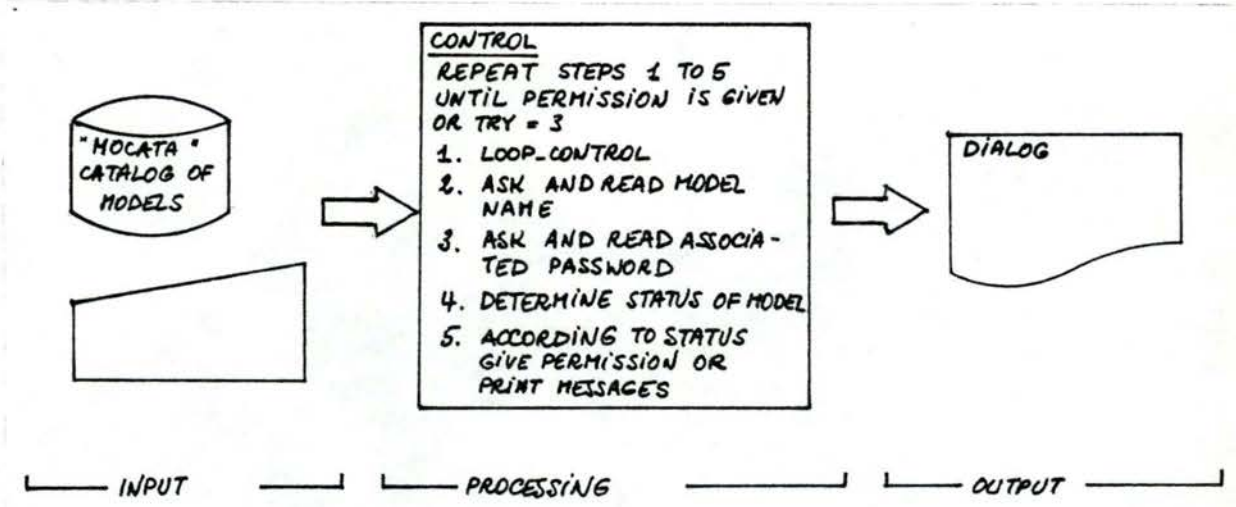
10.3.2. Sorties

- messages affichès au terminal.

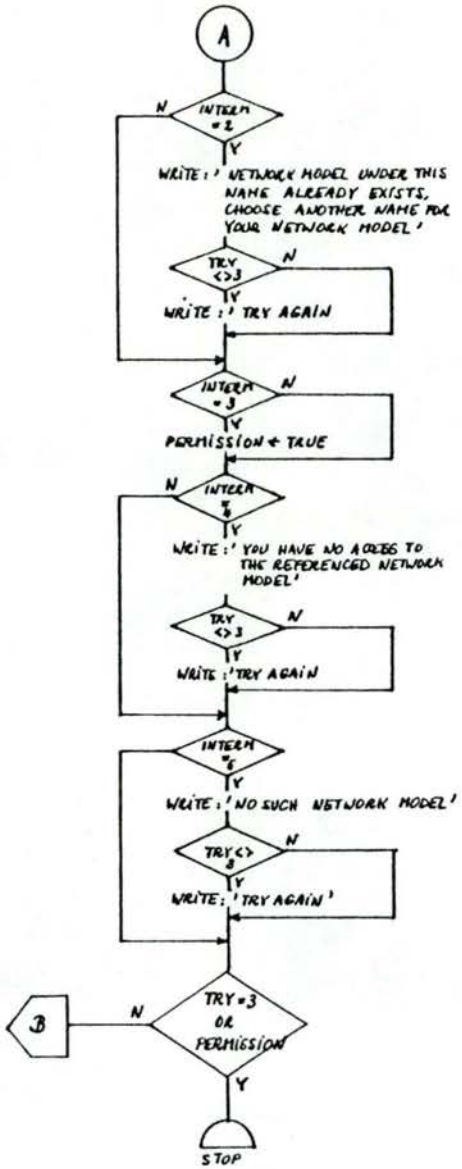
10.3.3. Traitements

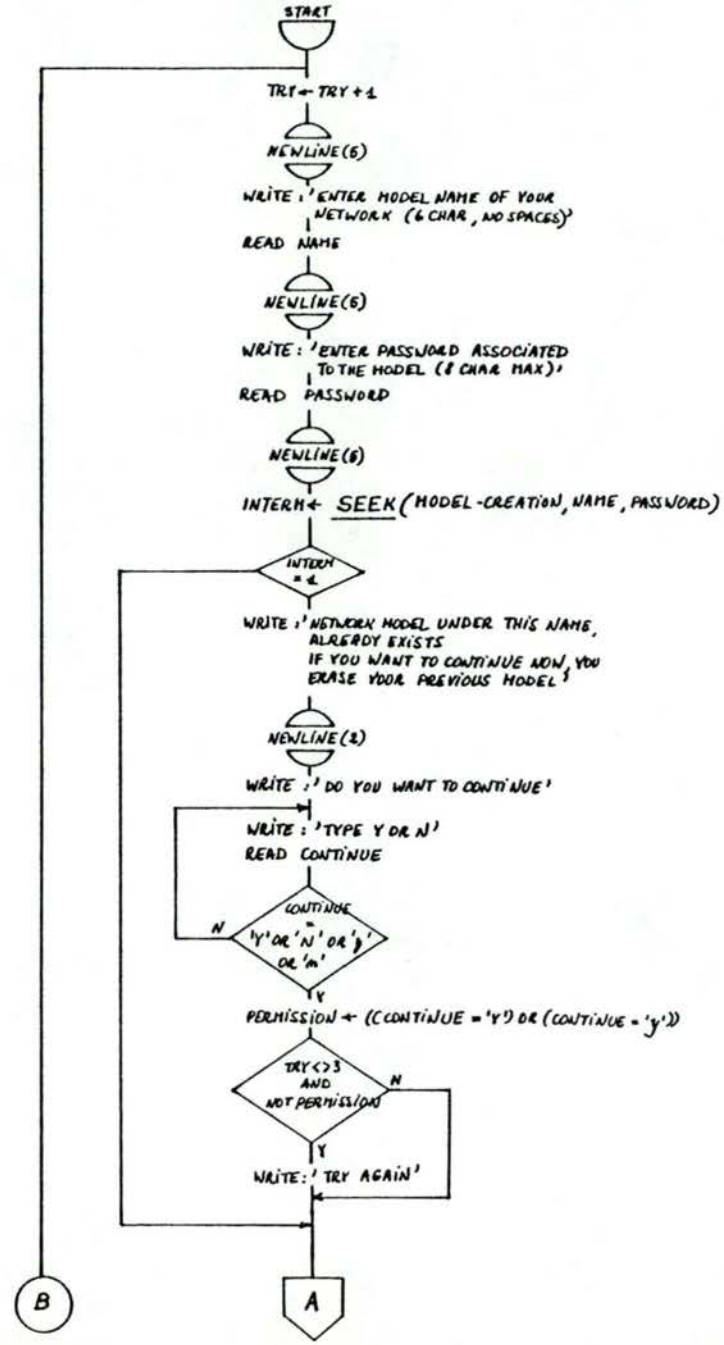
- cfr.: schèma et organigramme.

1.



10.4. ORGANIGRAMME





10.5. CODE PASCAL CORRESPONDANT

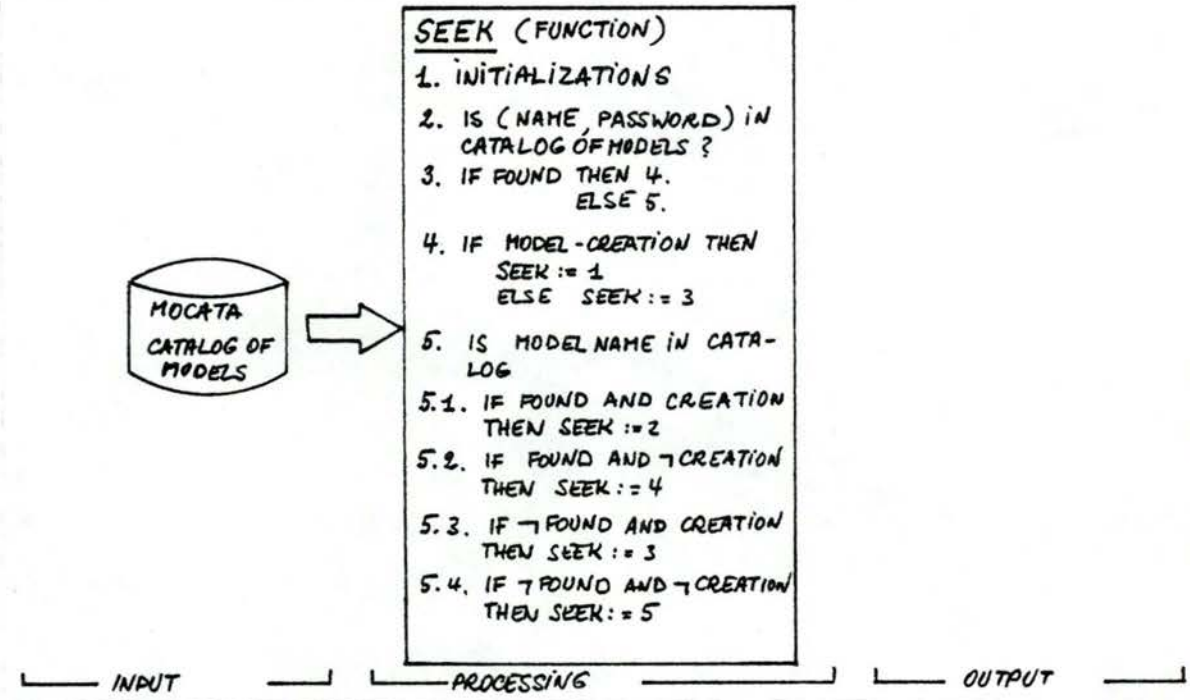
cfr.: programme INTRODUC - (1.1.4.1.2.)

11. SEEK - NIVEAU D'ANALYSE 1.1.4.1.2.1.

11.1. SPECIFICATIONS

Cette fonction de contrôle met en oeuvre les spécifications du dispositif de sécurité. La fonction détermine selon ces règles le statut (un nombre entre 1 et 5) du modèle auquel l'utilisateur veut accéder. La base de cette recherche sont le nom et le mot de passe associé.

11.2. SCHEMA



11.3. DESCRIPTION

11.3.1. Entrées

- fichier catalogue des modèles, (fcata - mocata).

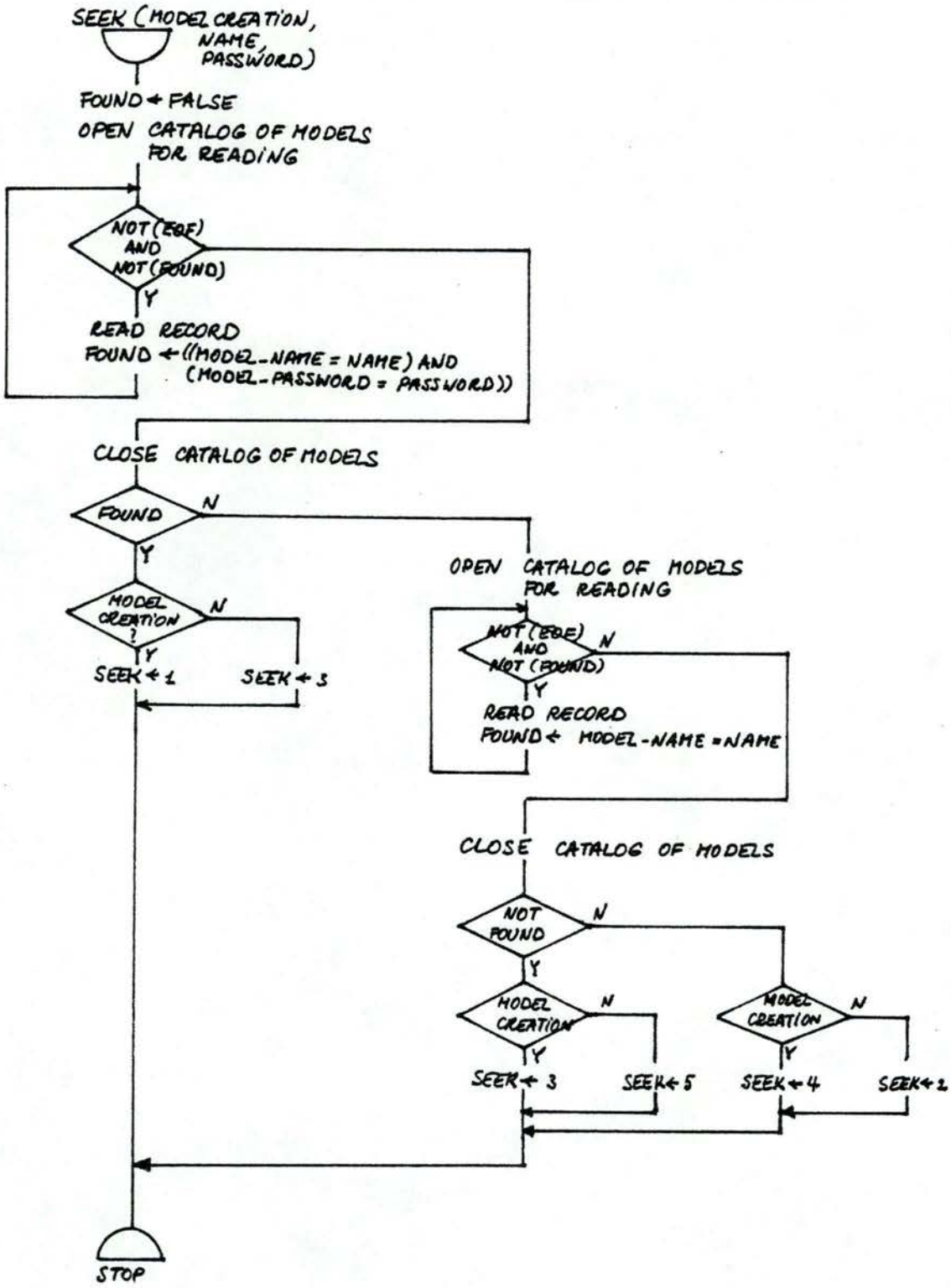
11.3.2. Sorties

- pas de sorties.

11.3.3. Traitements

- cfr.: schéma et organigramme.

11.4. ORGANIGRAMME



11.5. CODE PASCAL CORRESPONDANT

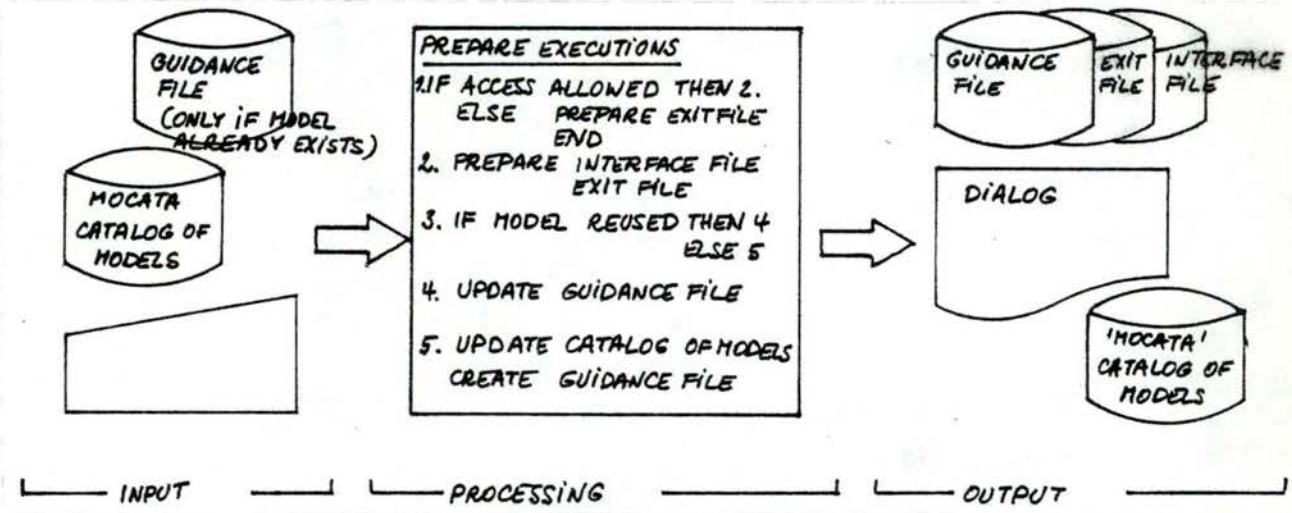
cfr.: programme INTRODUC - (1.1.4.1.2.1.)

12. PREPARE EXECUTIONS - NIVEAU D'ANALYSE 1.1.4.2.

12.1. SPECIFICATIONS

Cette partie réalise les initialisations nécessaires pour la suite des travaux. Si l'utilisateur a eu la permission d'accès, il faut encore voir si le modèle est libre. Selon la réponse, il faut garnir les fichiers d'interface 'fg', de sortie 'exitfile' et de guidage 'f'.

12.2. SCHEMA



12.3. DESCRIPTION

12.3.1. Entrées

- réponses en provenance du terminal;
- fichier de guidage si le modèle existe déjà, (f - <nom_du_modèle>);
- fichier catalogue des modèles, (fcata - mocata).

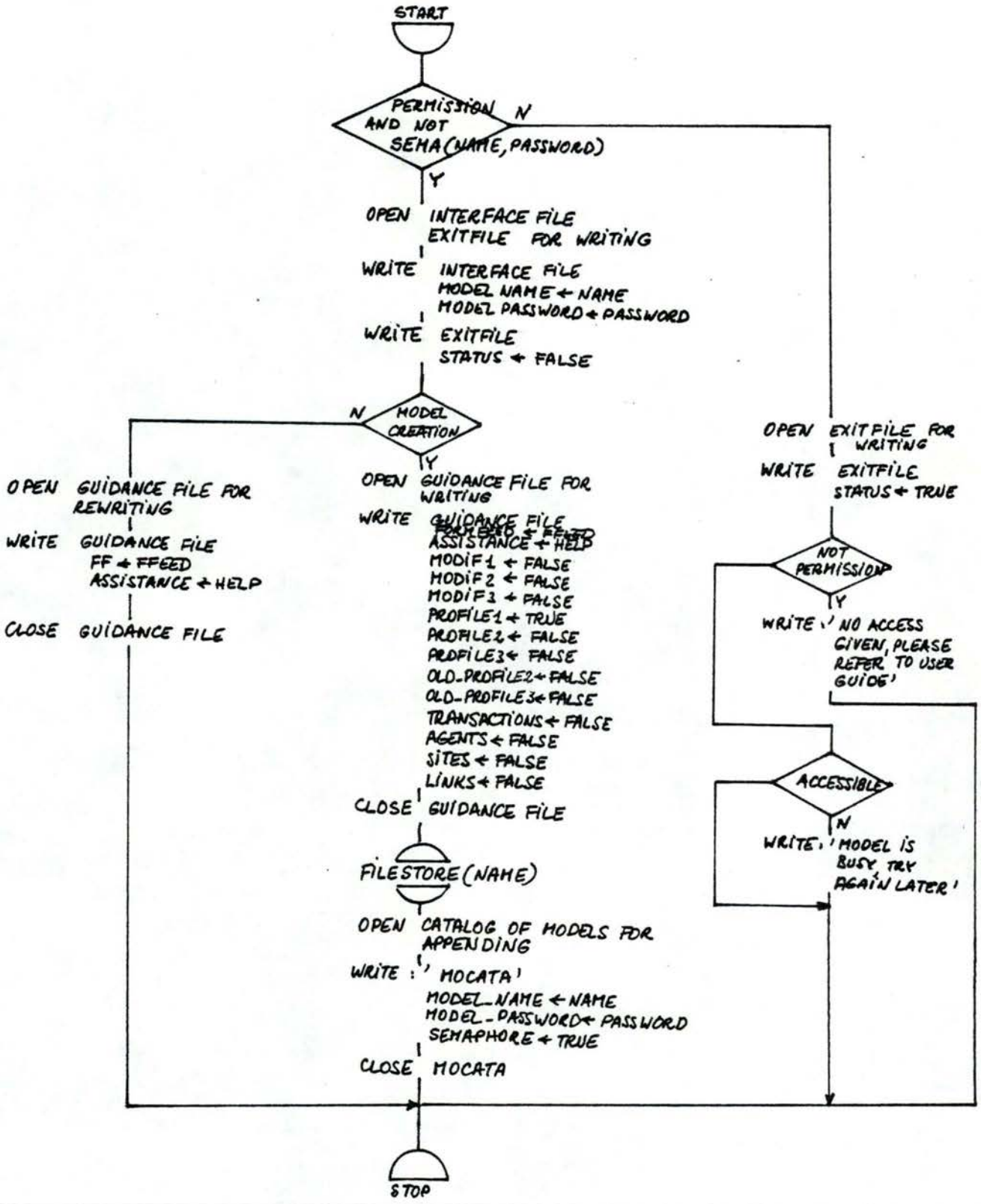
12.3.2. Sorties

- messages affichées au terminal;
- fichier catalogue des modèles;
- fichier de contrôle de sortie, (exitfile -);
- fichier de guidage;
- fichier interface, (fg -).

12.3.3. Traitements

- cfr.: schéma et organigramme.

12.4. ORGANIGRAMME



12.5. CODE PASCAL CORRESPONDANT

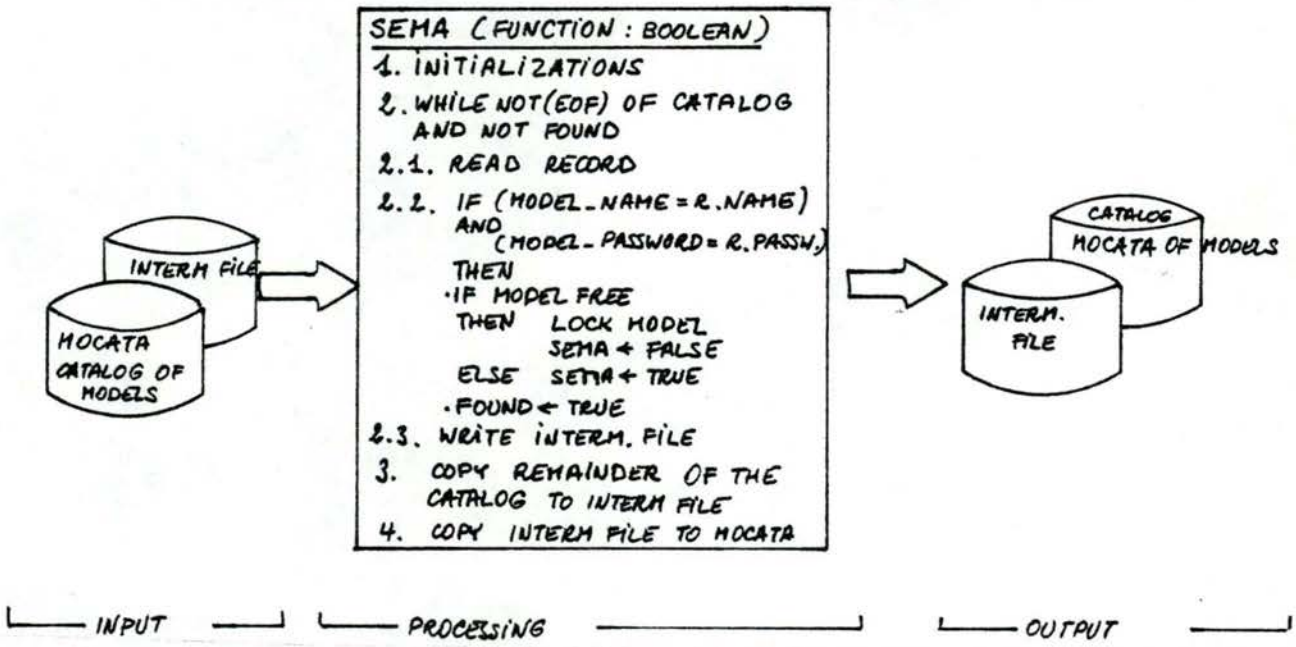
cfr.: programme INTRODUC - (1.1.4.2.)

13. SEMA - NIVEAU D'ANALYSE 1.1.4.2.1.

13.1. SPECIFICATIONS

En partant du nom de modèle et du mot de passe associé, cette fonction contrôle si le modèle est libre, c'est-à-dire s'il n'y a pas d'utilisateur qui travaille dessus. S'il y en a, le modèle est occupé et la fonction qui est du type booléen prend la valeur F (false). Sinon elle prend la valeur T (true) et elle bloque le modèle à tout autre utilisateur en faisant basculer le sémaphore associé au modèle.

13.2. SCHEMA



13.3. DESCRIPTION

13.3.1. Entrées

- fichier catalogue des modèles, (fcata - mocata);
- fichier intermédiaire du fichier catalogue, (fica -).

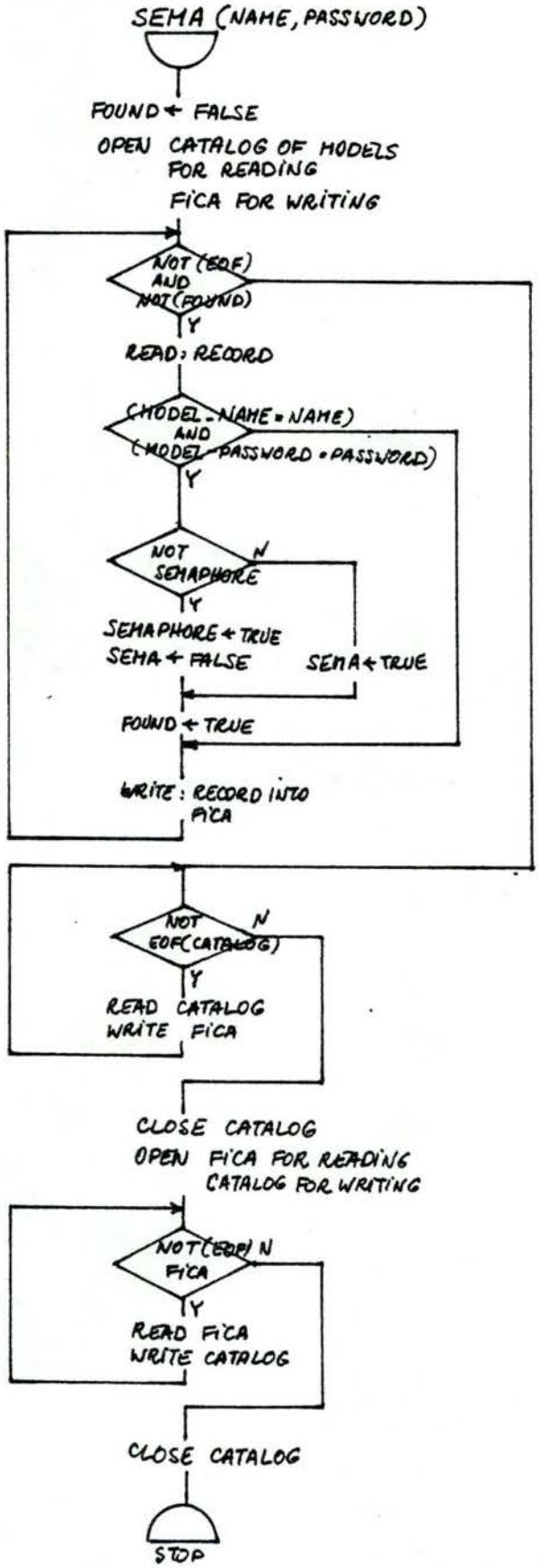
13.3.2. Sorties

- fichier intermédiaire du fichier catalogue;
- fichier catalogue des modèles.

13.3.3. Traitements

cfr.: schéma et organigramme.

13.4. ORGANIGRAMME



13.5. CODE PASCAL CORRESPONDANT

cfr.: programme INTRODUC - (1.1.4.2.1.)

CHAPITRE 4 : COLLECTE DES DONNEES - TRANSACTIONS

REMARQUE PRELIMINAIRE

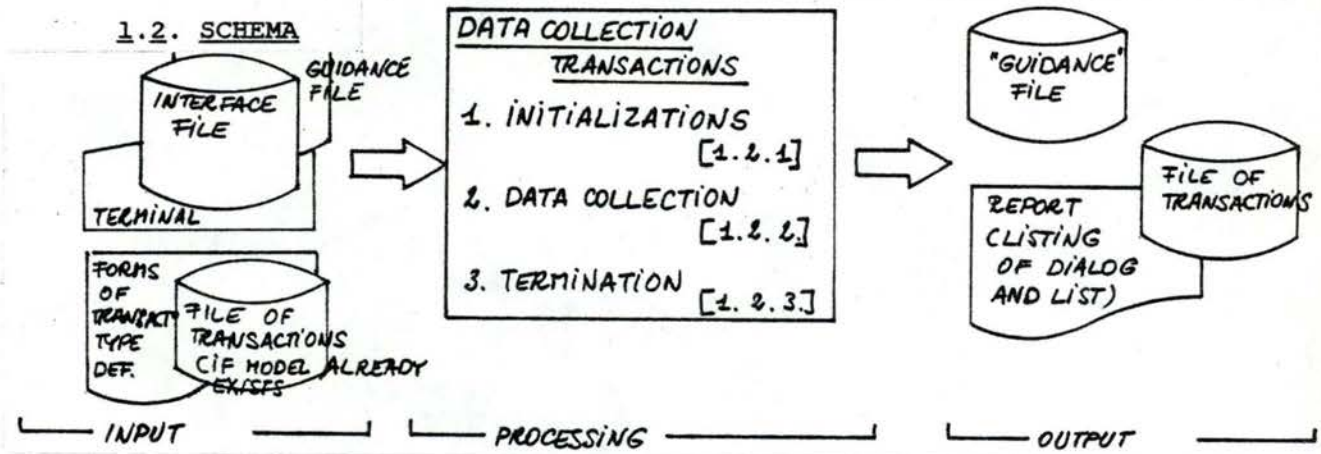
Vu la taille considérable de l'analyse il nous est impossible de présenter tous les organigrammes dans cette annexe du mémoire. Nous nous limitons à fournir les spécifications, le schéma et la description des entrées et sorties de chaque niveau de découpe de notre analyse. Cependant nous voulons signaler qu'un dossier complèt , reprenant tous les organigrammes a été remis à Mademoiselle Claudie Chappuis et à Monsieur Philippe Van Bastelaer afin de permettre des modifications ou des extensions du prototype que nous avons développé. Cette remarque s'applique aux chapitres 4 à 9.

1. MODULE 2 - DATA COLLECTION : TRANSACTIONS - NIVEAU D'ANALYSE 1.2.

1.1. SPECIFICATIONS

Le module "transactions" doit réaliser la collecte des données concernant les types de transactions générés par les applications de l'utilisateur. En partant des informations contenues dans les fichiers "interface" et "guide", le module crée un fichier des transactions et il le rend permanent ou il accède à un fichier de transactions existant. Le module doit également permettre des modifications sur ce fichier des transactions. Les fonctions de modification sont les suivantes : "insertion", "suppression" et "modification". Une fois la collecte et les modifications terminées, le module met à jour les informations du fichier de guidage.

1.2. SCHEMA



1.3. DESCRIPTION

1.3.1. Entrées

- fichier interface, (fg -);
- fichier de guidage associé au modèle, (f - <nom_du_modèle>);
- fichier des transactions, si modèle existe déjà, (ft - <nom_du_modèle>1);
- entrées manuelles au terminal.

1.3.2. Sorties

- rapports, messages et dialogue au terminal;
- fichier des transactions, (ft - <nom_du_modèle>1);
- fichier de guidage.

1.3.3. Traitements

- cfr.: schéma et code Pascal.

1.4. CODE PASCAL CORRESPONDANT

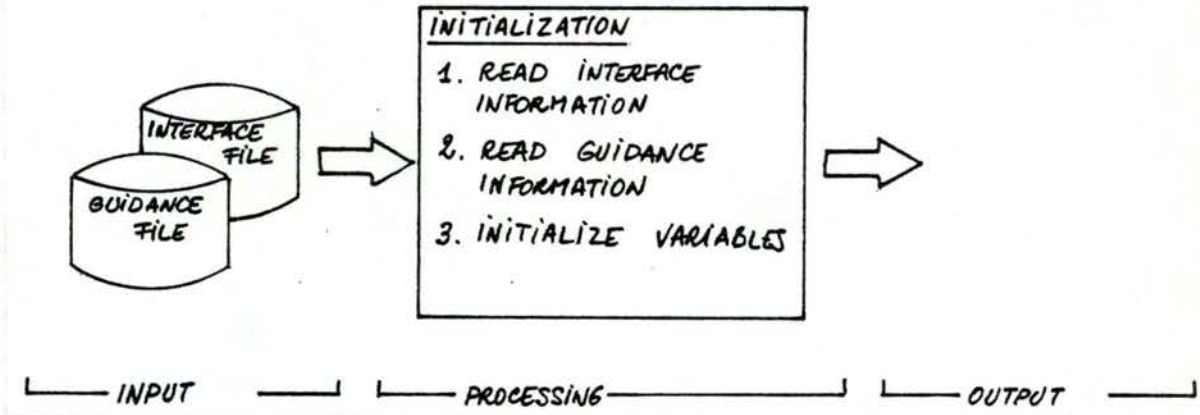
cfr.: programme TRANSACT - (1.2.)

2. INITIALIZATION - NIVEAU D'ANALYSE 1.2.1.

2.1. SPECIFICATIONS

Cette partie réalise les initialisations nécessaires du module. C'est-à-dire elle lit les informations dites d'interface. A partir de ces donnnées le système accède aux informations de guidage concernant le modèle de réseau sur lequel le programme s'exécute. Ces informations permettent de générer de façon automatique le nom du fichier externe stocké en mémoire secondaire.

2.2. SCHEMA



2.3. DESCRIPTION

2.3.1. Entrées

- fichier interface, (fg -);
- fichier de guidage, (f - <nom_du_modèle>).

2.3.2. Sorties

- pas de sorties.

2.3.3. Traitements

- cfr.: schéma et code Pascal.

2.4. CODE PASCAL CORRESPONDANT

cfr.: programme TRANSACT - (1.2.1.)

3. DATA COLLECTION - NIVEAU D'ANALYSE 1.2.2.

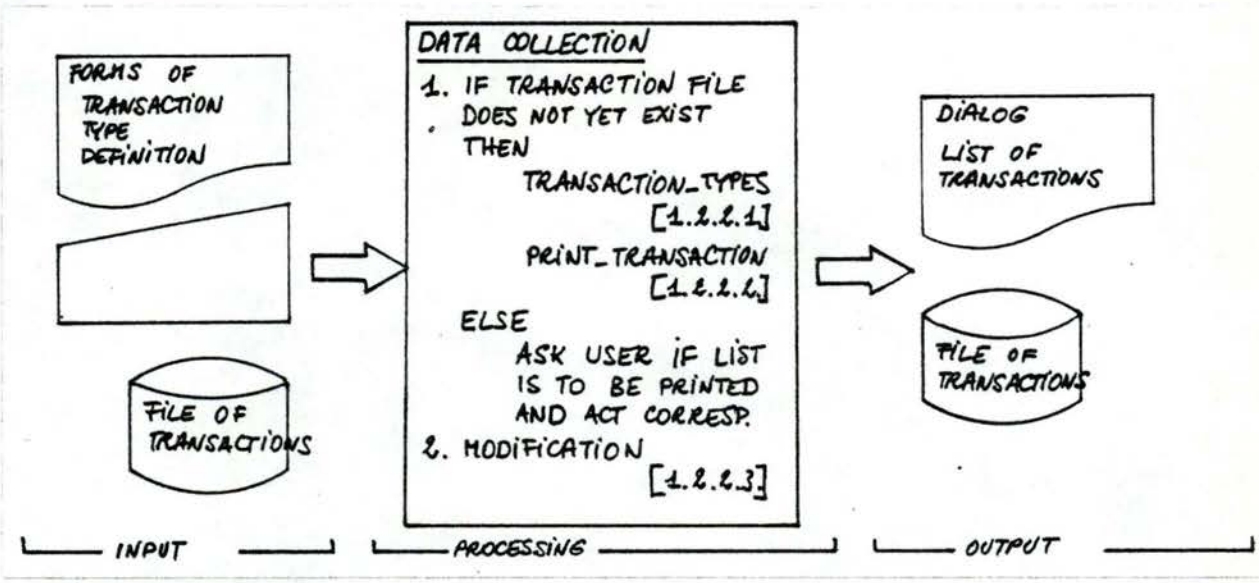
3.1. SPECIFICATIONS

A ce niveau le système connaît le nom du modèle sur lequel il doit travailler, ainsi que les informations concernant l'avancement des travaux sur ce modèle.

La première tâche du système est de voir dans les informations de guidage si l'utilisateur est déjà passé par la collecte des données relatives aux transactions. Si l'utilisateur n'a pas encore entré de données il faut passer par la collecte qui sera clôturée par une présentation des informations receuillies sous forme de tableau. Dans le manuel d'utilisation se trouve un exemple qui visualise ce tableau. Si l'utilisateur a déjà entré des informations relatives aux transactions , on demande à l'utilisateur s'il veut ou ne veut pas avoir une impression du contenu du fichier.

L'opération suivante est la modification possible du contenu du fichier des transctions.

3.2. SCHEMA



3.3. DESCRIPTION

3.3.1. Entrèes

- fichier des transactions, si dèjà existant, (ft - ‹nom_du_ modèle›1);
- entrèes des donnèes au terminal.

3.3.2. Sorties

- questions et messages affichés au terminal;
- fichier des transactions;
- tableau prèsentant le contenu du fichier des transactions.

3.3.3. Traitements

- cfr.: schèma et code Pascal.

3.4. CODE PASCAL CORRESPONDANT

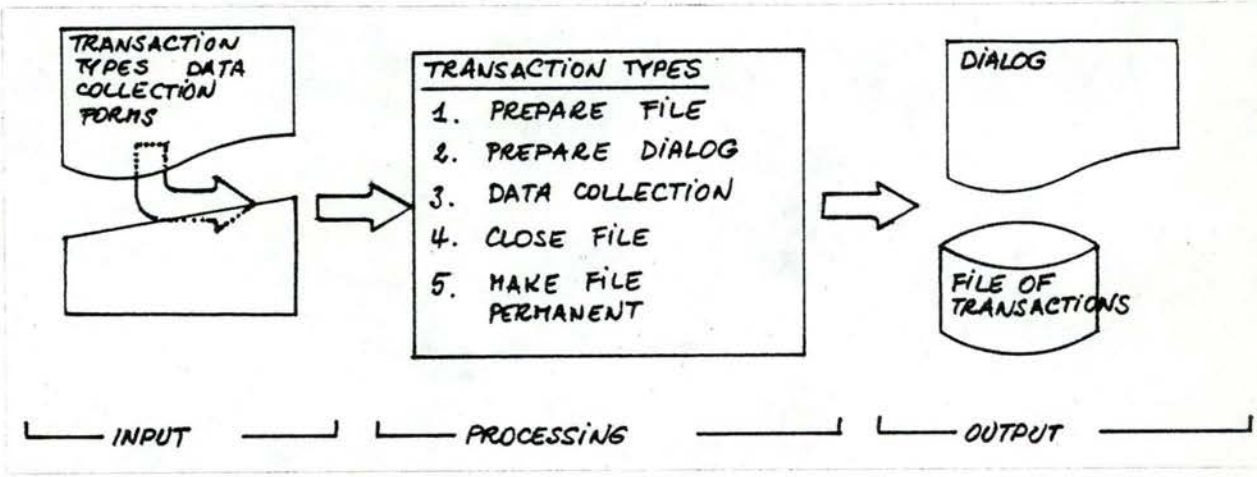
cfr.: programme TRANSACT - (1.2.2.)

4. TRANSACTION_TYPES - NIVEAU D'ANALYSE 1.2.2.1.

4.1. SPECIFICATIONS

Cette partie est responsable de la collecte des donnèes et de la confection du fichier des transactions. Le travail de l'utilisateur est guidè au maximum par des commentaires mis à sa disposition. Le système assure des contrôles de validitè des rèponses.

4.2. SCHEMA



4.3. DESCRIPTION

4.3.1. Entrèes

- entrèes manuelles au terminal.

4.3.2. Sorties

- questions et messages affichès au terminal;
- fichier des transactions, (ft - ‹nom_du_modèle›1);

4.3.3. Traitements

- cfr.: schèma et code Pascal.

4.4. CODE PASCAL CORRESPONDANT

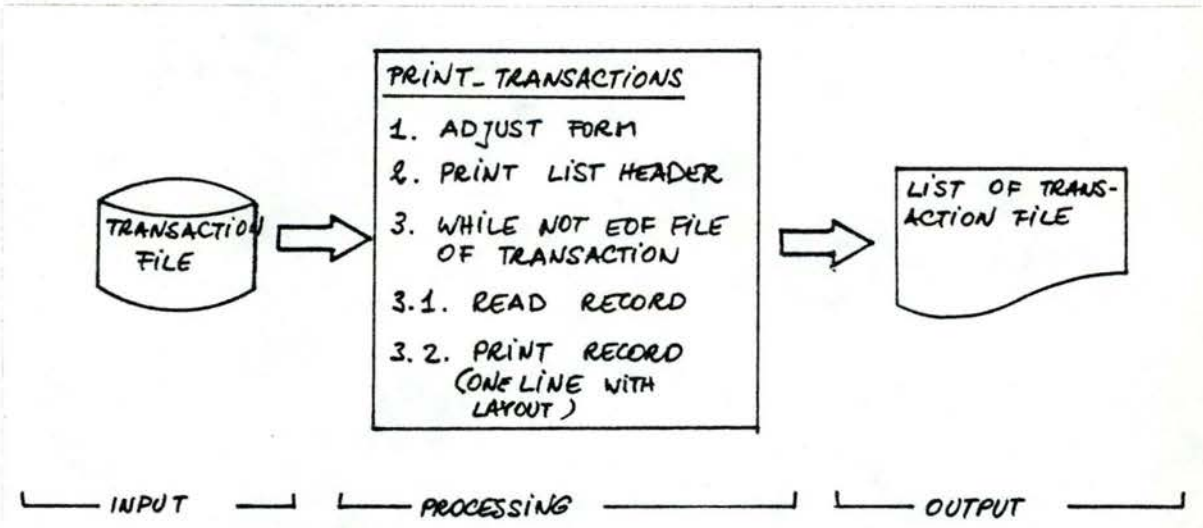
cfr.: programme TRANSACT - (1.2.2.1.)

5. PRINT_TRANSACTIONS - NIVEAU D'ANALYSE 1.2.2.2.

5.1. SPECIFICATIONS

En partant du fichier des transactions, cette procédure imprime un tableau des transactions dont la présentation a été donnée au chapitre 7 du manuel d'utilisation.

5.2. SCHEMA



5.3. DESCRIPTION

5.3.1. Entrées

- fichier des transactions, (ft - <nom_du_modèle>1).

5.3.2. Sorties

- tableau des transactions imprimé au terminal.

5.3.3. Traitements

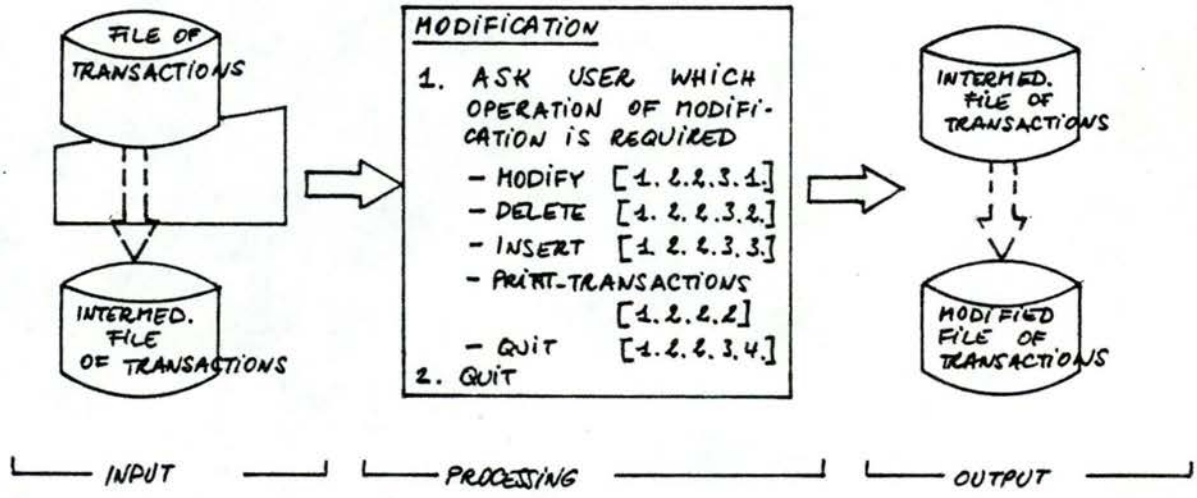
- cfr.: schéma et code Pascal.

6. MODIFICATION - NIVEAU D'ANALYSE 1.2.2.3.

6.1. SPECIFICATIONS

La procédure de modification permet de modifier le contenu du fichier des transactions. Ainsi l'utilisateur peut changer son modèle sans devoir tout recommencer. Par modification nous entendons : modifications des attributs d'un enregistrement, insertion d'un enregistrement , effacement d'un enregistrement et l'impression du contenu du fichier des transactions, pour faciliter les modifications.

6.2. SCHEMA



6.3. DESCRIPTION

6.3.1. Entrées

- fichier des transactions, (ft - ‹nom_du_modèle›1);
- fichiers intermédiaires crées dans cette procédure;
- entrées manuelles au terminal.

6.3.2. Sorties

- fichiers intermédiaires;
- questions et messages affichés au terminal;
- fichier des transactions modifié;
- tableau des transactions affiché au terminal (sur demande).

6.3.3. Traitements

- cfr.: schéma et code Pascal.

6.4. CODE PASCAL CORRESPONDANT

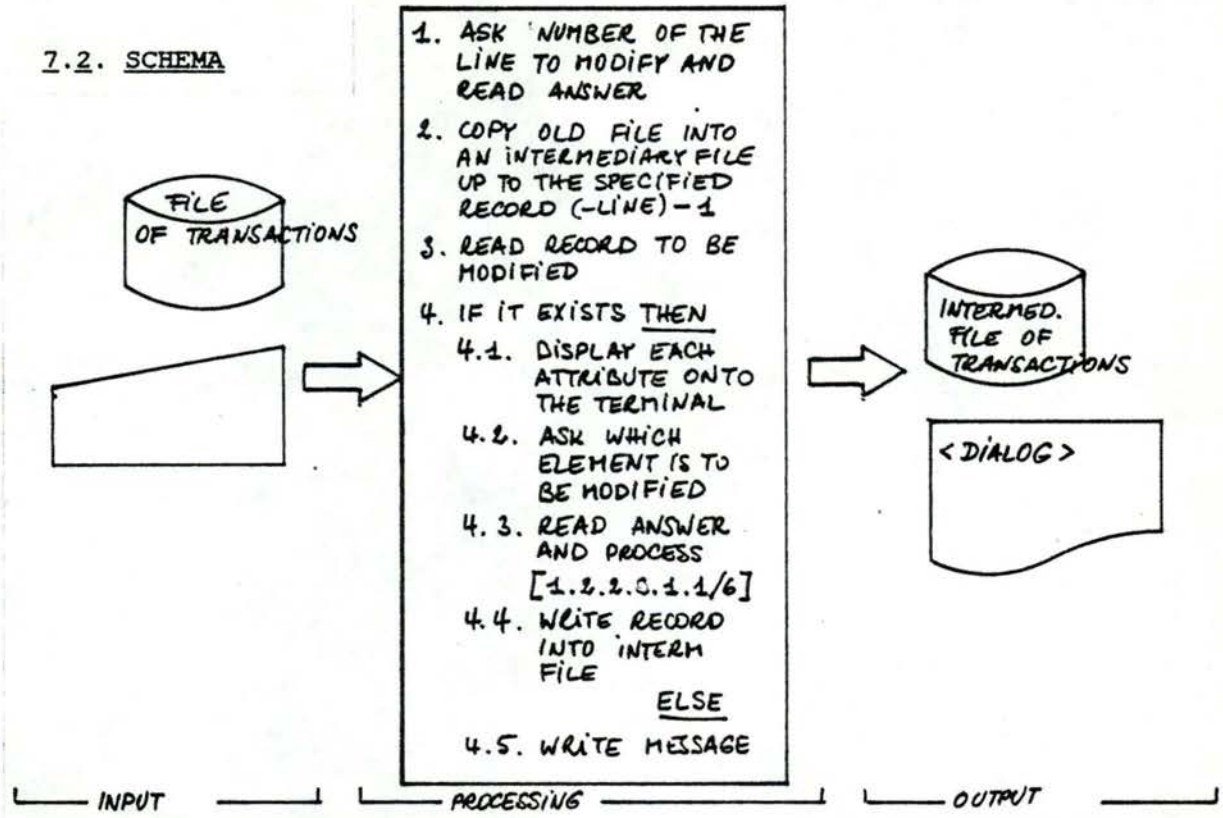
cfr.: programme TRANSACT - (1.2.2.3.)

7. MODIFY - NIVEAU D'ANALYSE 1.2.2.3.1.

7.1. SPECIFICATIONS

La procédure "modify" permet de modifier les attributs d'un enregistrement qui est repéré par un numéro identique à celui affiché dans le tableau. Cette procédure affiche les différents attributs de l'enregistrement et procède aux modifications souhaitées. La procédure assure le contrôle de validité du numéro de ligne spécifié par l'utilisateur.

7.2. SCHEMA



7.3. DESCRIPTION

7.3.1. Entrèes

- fichier des transactions, (fit - <nom_du_modèle>1);
- entrèes manuelles au terminal.

7.3.2. Sorties

- fichier intermèdiaire des transactions, (ft2 -);
- questions et messages affichès au terminal.

7.3.3. Traitements

- cfr.: schèma et code Pascal.

7.4. CODE PASCAL CORRESPONDANT

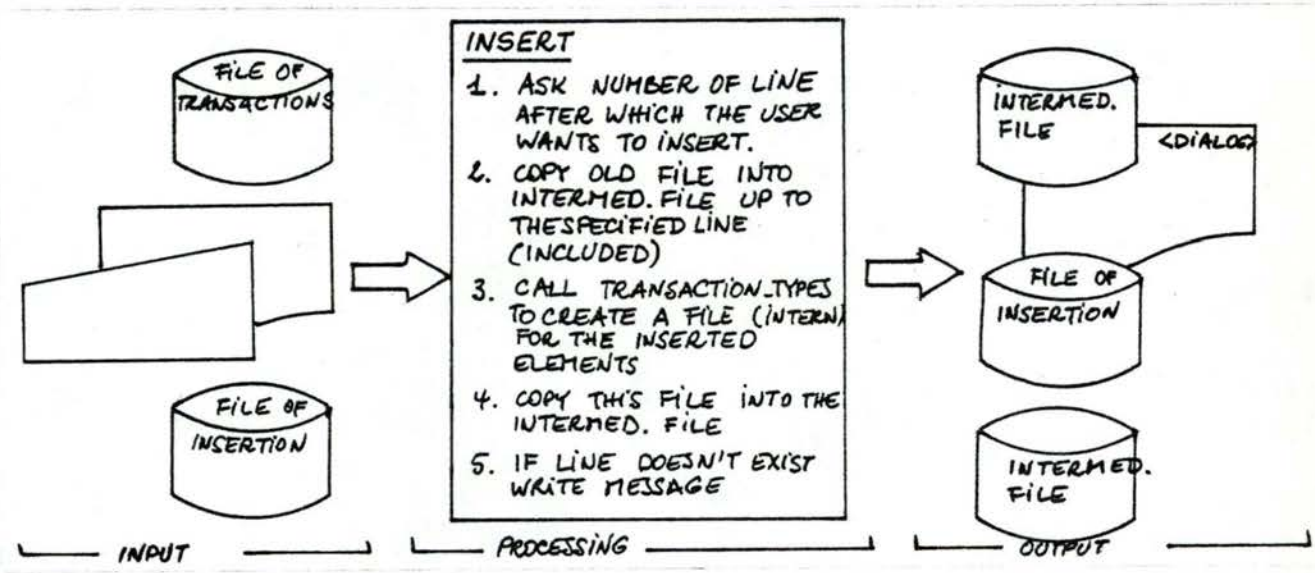
cfr.: programme TRANSACT - (1.2.2.3.1.)

8. INSERT - NIVEAU D'ANALYSE 1.2.2.3.2.

8.1. SPECIFICATIONS

La procédure "insert" permet d'inserrer une ou plusieurs lignes derrière la ligne du tableau spécifiée. La procédure doit copier le fichier des transactions jusqu'à cette ligne (y comprise) dans un fichier intermédiaire. Ensuite cette procédure fait elle-même appel à la procédure "transaction-types" qui crée un autre fichier intermédiaire qui sera copié à la fin des insertions dans le premier fichier intermédiaire et ensuite détruit. La procédure assure le contrôle de validité du numéro de ligne spécifié.

8.2. SCHEMA



8.3. DESCRIPTION

8.3.1. Entrées

- fichier des transactions, (fit - <nom_du_modèle>1);
- fichier des insertions, (ft - ft3);
- entrées manuelles au terminal.

8.3.2. Sorties

- questions et messages au terminal;
- fichier intermédiaire des transactions, (ft2 -);
- fichier des insertions, (ft - ft3).

8.3.3. Traitements

- cfr.: schéma et code Pascal.

8.4. CODE PASCAL CORRESPONDANT

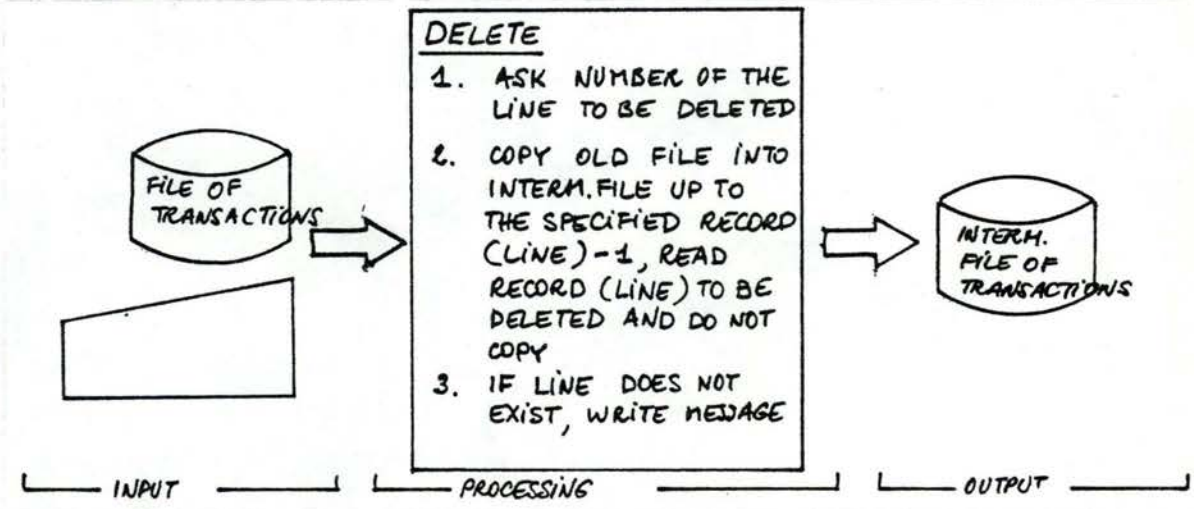
cfr.: programme TRANSACT - (1.2.2.3.2.)

9. DELETE - NIVEAU D'ANALYSE 1.2.2.3.3.

9.1. SPECIFICATIONS

La procédure "delete" détruit la ligne spécifiée par l'utitilisateur. La recherche se fait séquentiellement et un contrôle de validité du numéro de ligne est assuré.

9.2. SCHEMA



9.3. DESCRIPTION

9.3.1. Entrées

- fichier des transactions, (fit - <nom_du_modèle>1);
- numéro de ligne au terminal.

9.3.2. Sorties

- questions et messages affichés au terminal;
- fichier intermédiaire des transactions, (ft2 -).

9.3.3. Traitements

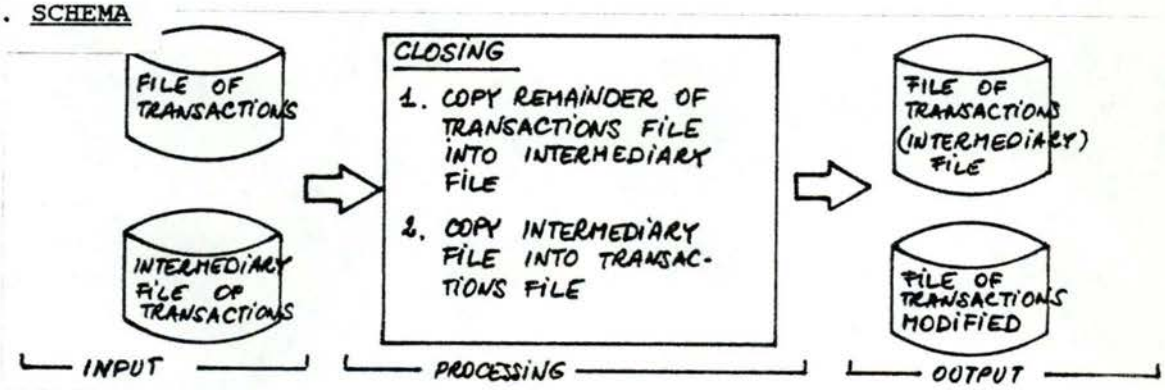
- cfr.: schéma et code Pascal.

10. CLOSING - NIVEAU D'ANALYSE 1.2.2.3.4.

10.1. SPECIFICATIONS

Cette procédure doit fermer les fichiers encore ouverts et bien remettre les transactions définies et modifiées dans le fichier des transactions '<nom_du_modèle>1'. Cette procédure se base sur le fichier intermédiaire constitué dans les procédures "modify", "insert" et "delete".

10.2. SCHEMA



10.3. DESCRIPTION

10.3.1. Entrées

- fichier des transactions, (fit - <nom_du_modèle>1);
- fichier intermédiaire des transactions, (ft2 -).

10.3.2. Sorties

- fichier intermédiaire des transactions;
- fichier des transactions mis à jour.

10.3.3. Traitements

- cfr.: schéma et code Pascal.

10.4. CODE PASACL CORRESPONDANT

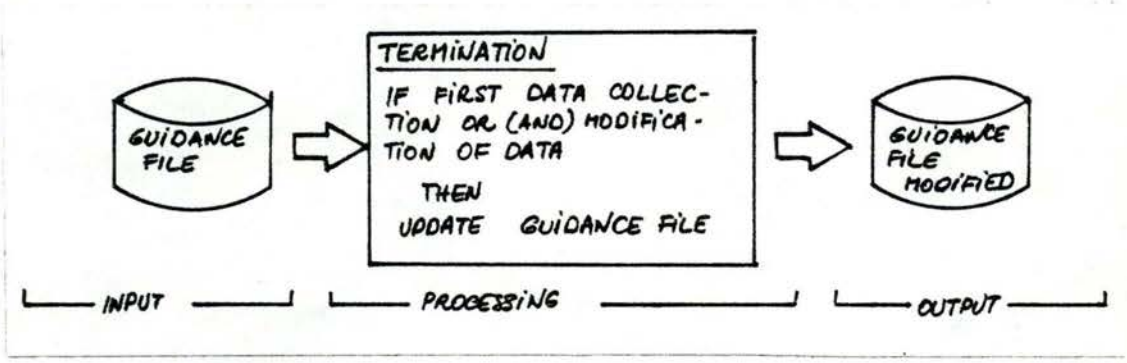
cfr.: programme TRANSACT - (1.2.2.3.4.)

11. TERMINATION - NIVEAU D'ANALYSE 1.2.3.

11.1. SPECIFICATIONS

Le rôle de cette partie est de mémoriser le fait qu'on est passé par la collecte des données relatives aux transactions. Cependant cette opération n'est exécutée que dans le cas d'un premier enregistrement des informations. Une mise à jour des transactions peut nécessiter une redéfinition des liens (chapitre 7) , c'est pourquoi la modification des données est mémorisée.

11.2. SCHEMA



11.3. DESCRIPTION

11.3.1. Entrèes

- fichier de guidage, (f - ‹nom_du_modèle›).

11.3.2. Sorties

- fichier de guidage.

11.3.3. Traitements

- cfr.: schèma et code Pascal.

11.4. CODE PASCAL CORRESPONDANT

cfr.: programme TRANSACT -(1.2.3.)

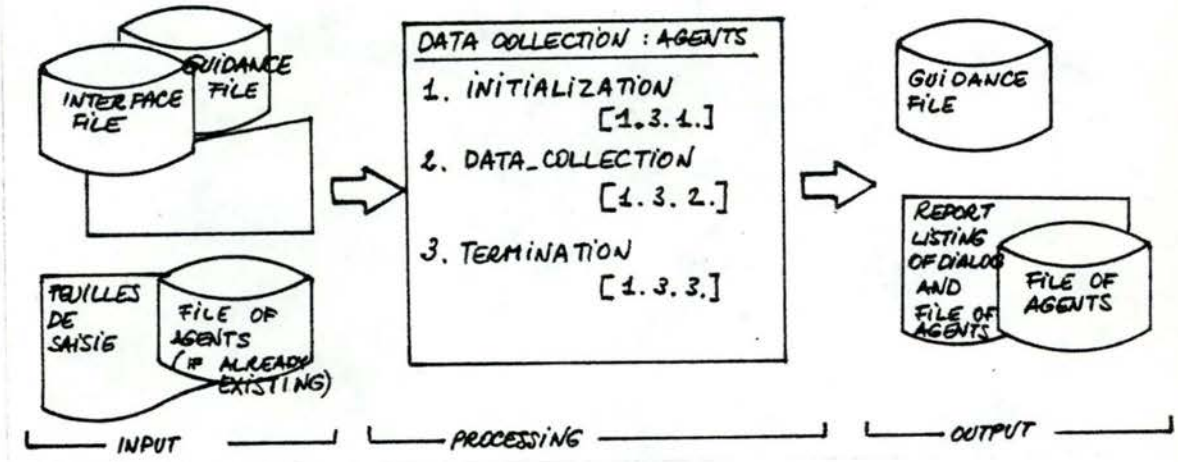
CHAPITRE 5 : COLLECTE DES DONNEES - AGENTS

1. MODULE 3 - DATA COLLECTION : AGENTS - NIVEAU D'ANALYSE 1.3.

1.1. SPECIFICATIONS

Le module "agents" doit réaliser la collecte des données concernant les types d'agents initialisant les transactions de l'application. En partant des informations contenues dans les fichiers "interface" et "guide" , le module crée un fichier des agents et le rend permanent ou le module accède à un fichier existant. Le module permet également de faire les modifications suivantes : "modify","insert" et "delete". Le module met à jour les informations de guidage.

1.2. SCHEMA



1.3. DESCRIPTION

1.3.1. Entrées

- fichier interface, (fg -);
- fichier de guidage associé au modèle, (f - <nom_du_modèle>);
- entrées manuelles au terminal;
- fichier des agents si modèle existe déjà, (fa - <nom_du_modèle>2).

1.3.2. Sorties

- fichier des agents , (fa - ‹nom_du_modèle›2);
- questions, messages et tableau des agents affichés au terminal;
- fichier de guidage mis à jour, (f - ‹nom_du_modèle›).

1.3.3. Traitements

- cfr.: schèma et code Pascal.

1.4. CODE PASCAL CORRESPONDANT

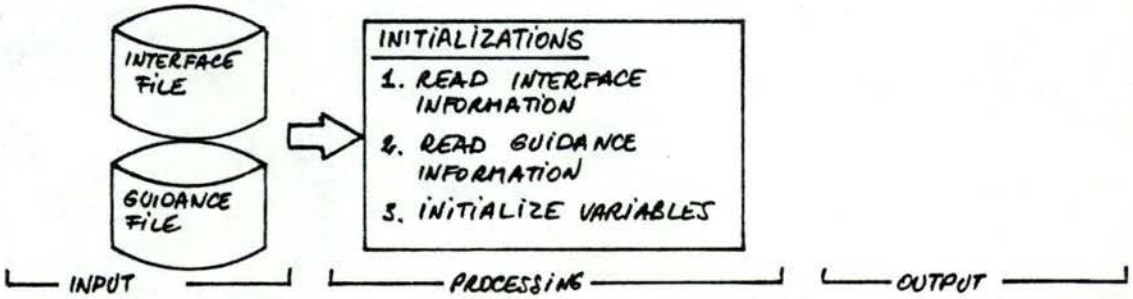
cfr.: programme agents - (1.3.)

2. INITIALIZATION - NIVEAU D'ANALYSE 1.3.1.

2.1. SPECIFICATIONS

Cette partie est responsable d'initialiser le module, c'est-à-dire de lire les informations dites d'interface. A partir de ces donnèes le système accède aux informations contenues dans le fichier de guidage associé au modèle de rèseau sur lequel s'exècute le programme. Ces informations permettent de gènèrer automatiquement le nom du fichier externe dans lequel seront ou sont stockèes les informations relatives aux agents.

2.2. SCHEMA



2.3. DESCRIPTION

2.3.1. Entrées

- fichier interface, (fg -);
- fichier de guidage, (f - ‹nom_du_modèle›).

2.3.2. Sorties

- pas de sorties.

2.3.3. Traitements

- cfr.: schéma et code Pascal.

2.4. CODE PASCAL CORRESPONDANT

- cfr.: programme AGENTS - (1.3.1.)

3. DATA_COLLECTION - NIVEAU D'ANALYSE 1.3.2.

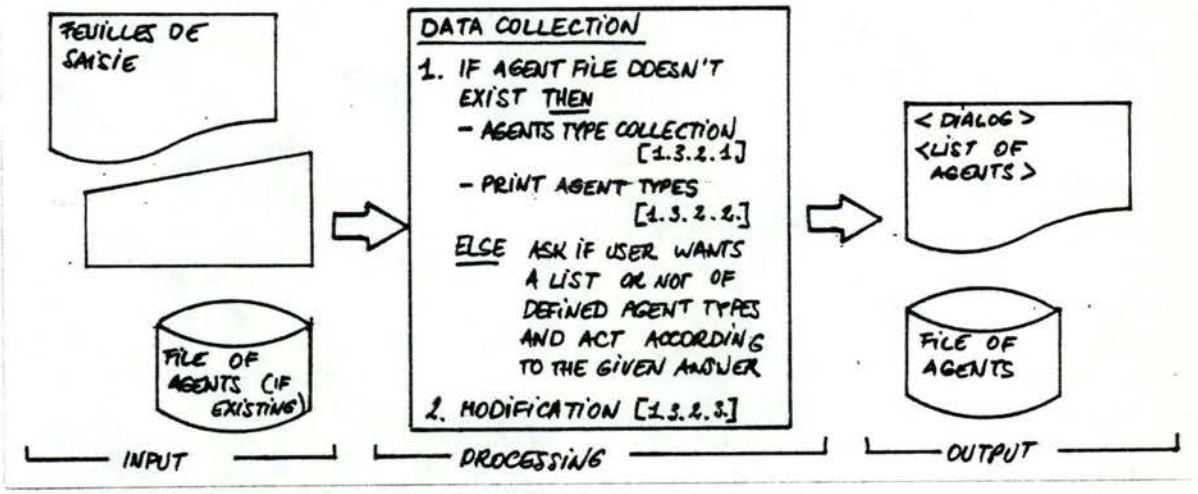
3.1. SPECIFICATIONS

A ce niveau le système connaît le nom du modèle sur lequel il doit travailler, ainsi que les informations concernant l'avancement des travaux sur ce modèle.

La première tâche de cette partie est de voir si l'utilisateur est déjà passé par la collecte des données relatives aux agents. S'il n'a pas encore entré de données , il faut passer par la collecte. Cette collecte se termine par la présentation des données receuillies sous forme de tableau. Si l'utilisateur a déjà entré des informations sur les agents, on lui demande s'il veut ou ne veut pas avoir une impression du contenu du fichier des agents.

La deuxième tâche consiste à permettre la modification des données contenues dans le fichier des agents.

3.2. SCHEMA



3.3. DESCRIPTION

3.3.1. Entrées

- entrées manuelles au terminal;
- fichier des agents si déjà existant, (fa - ‹nom_du_modèle›2).

3.3.2. Sorties

- questions, messages et tableau affichés au terminal;
- fichier des agents, (fa - ‹nom_du_modèle›2).

3.3.3. Traitements

- cfr .: schéma et code Pascal.

3.4. CODE PASCAL CORRESPONDANT

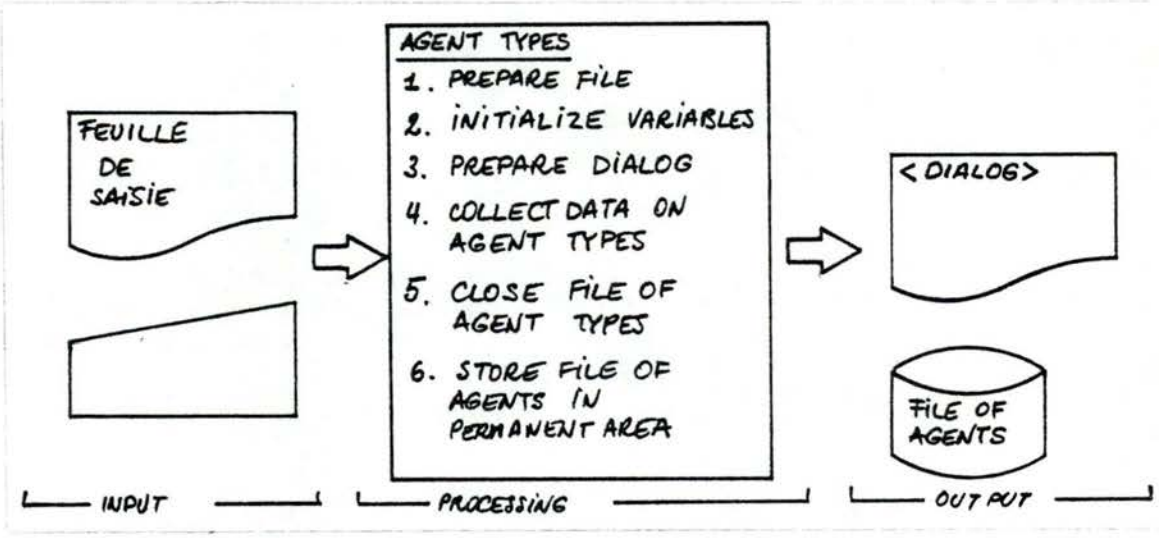
cfr.: programme AGENTS - (1.3.2.)

4. AGENT_TYPES - NIVEAU D'ANALYSE 1.3.2.1.

4.1. SPECIFICATIONS

Cette procédure doit assurer la collecte des données et la confection d'un fichier permanent des agents. Le travail de l'utilisateur est assisté au maximum par le programme, c'est-à-dire le programme assure des contrôles et il fournit des commentaires et explications à l'utilisateur.

4.2. SCHEMA



4.3. DESCRIPTION

4.3.1. Entrées

- réponses en provenance du terminal;

4.3.2. Sorties

- questions et messages affichés au terminal;
- fichier des agents, (fa - ‹nom_du_modèle›2).

Traitements

- cfr.: schéma et code Pascal.

4.4. CODE PASCAL CORRESPONDSANT

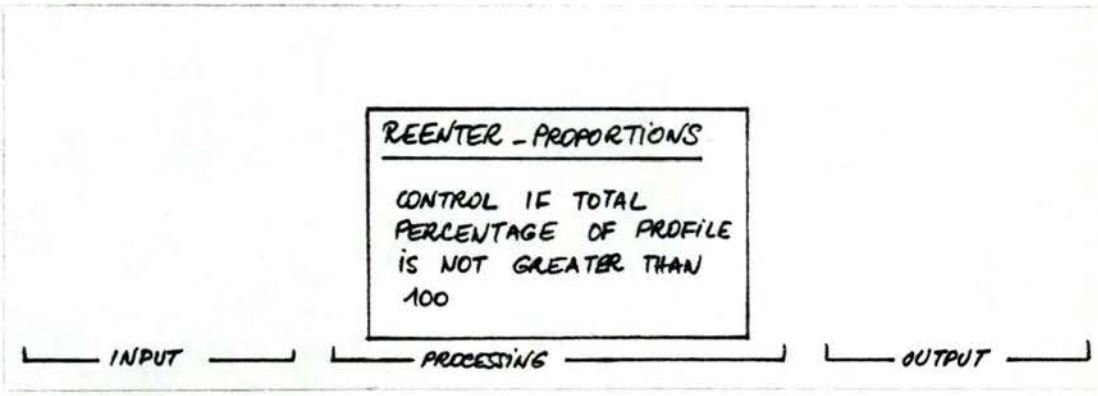
cfr.: programme AGENTS - (1.3.2.1.)

5. REENTER_PROPORTIONS - NIVEAU D'ANALYSE 1.3.2.1.1.

5.1. SPECIFICATIONS

Cette procédure vérifie que l'utilisateur ne déclare pas de types d'agents effectuant un travail > 100 %.

5.2. SCHEMA



5.3. DESCRIPTION

5.3.1. Entrées

- pas d'entrées.

5.3.2. Sorties

- pas de sorties.

5.3.3. Traitements

- cfr.: schéma et code Pascal.

5.4. CODE PASCAL CORRESPONDANT

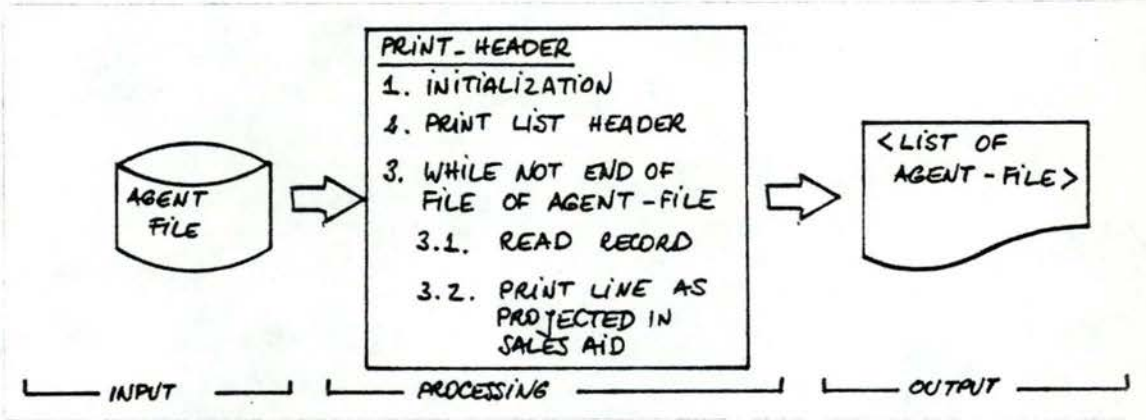
cfr.: programme AGENTS - (1.3.2.1.1.)

6. PRINT_AGENTS - NIVEAU D'ANALYSE 1.3.2.2.

6.1. SPECIFICATIONS

En partant du fichier des agents, cette procédure imprime une liste des agents. Le lecteur intéressé pourra consulter un exemple de sortie au chapitre 7 du manuel d'utilisation.

6.2. SCHEMA



6.3. DESCRIPTION

6.3.1. Entrées

- fichier des agents, (fa - ‹nom_du_modèle›2).

6.3.2. Sorties

- liste (tableau) des agents imprimée au terminal.

6.3.3. Traitements

- cfr.: schéma et code Pascal.

6.4. CODE PASCAL CORRESPONDANT

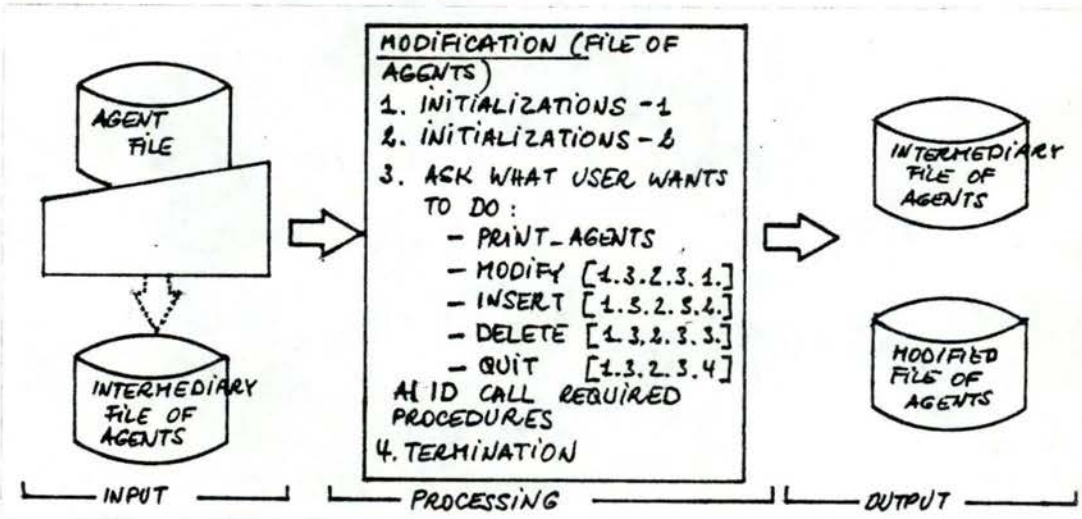
cfr.: programme AGENTS - (1.3.2.2.)

7. MODIFICATION - NIVEAU D'ANALYSE 1.3.2.3.

7.1. SPECIFICATIONS

La procédure de modification permet de modifier le contenu du fichier des agents. Ainsi l'utilisateur peut changer son modèle sans devoir tout recommencer. Par modification nous entendons : modification des attributs d'un enregistrement, insertion d'un enregistrement, effacement d'un enregistrement et l'impression du contenu du fichier des agents, pour faciliter les modifications.

7.2. SCHEMA



7.3. DESCRIPTION

7.3.1. Entrées

- fichier des agents, (fia - ‹nom_du_modèle›2);
- fichiers intermédiaires des agents créés dans cette procédure;
- réponses en provenance du terminal.

7.3.2. Sorties

- fichiers intermédiaires des agents;
- questions et messages affichés au terminal;
- fichier des agents mis à jour, (fia - ‹nom_du_modèle›2).

7.3.3. Traitements

- cfr.: schéma et code Pascal.

7.4. CODE PASCAL CORRESPONDANT

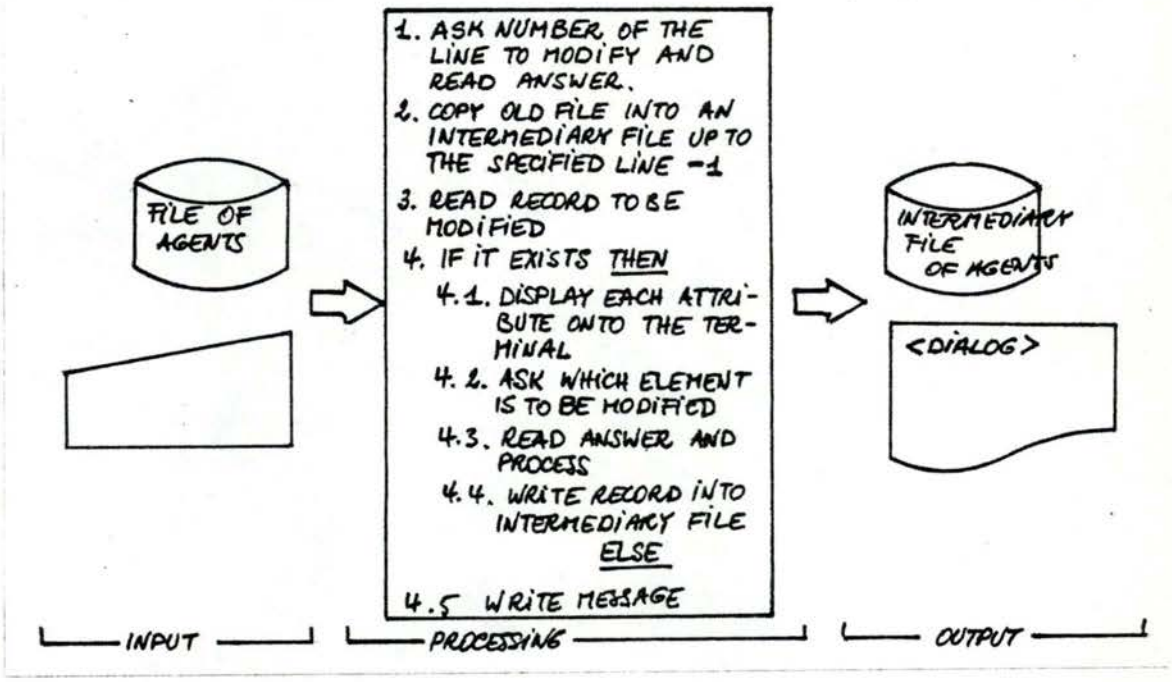
cfr.: programme AGENTS - (1.3.2.3.)

8. MODIFY - NIVEAU D'ANALYSE 1.3.2.3.1.

8.1. SPECIFICATIONS

La procédure "modify" permet de modifier les attributs d'un enregistrement qui est repéré par un numéro identique à celui affiché dans le tableau. Cette procédure affiche les différents attributs de l'enregistrement et procède aux modifications souhaitées. La procédure assure le contrôle de validité du numéro de ligne spécifié par l'utilisateur.

8.2. SCHEMA



8.3. DESCRIPTION

8.3.1. Entrées

- fichier des agents, (fia - ‹nom_du_modèle›2);
- réponses (modifications) en provenance du terminal.

8.3.2. Sorties

- fichier intermédiaire des agents, (fa2 -);
- questions et messages affichés au terminal.

8.3.3. Traitements

- cfr.: schéma et code Pascal.

8.4. CODE PASCAL CORRESPONDANT

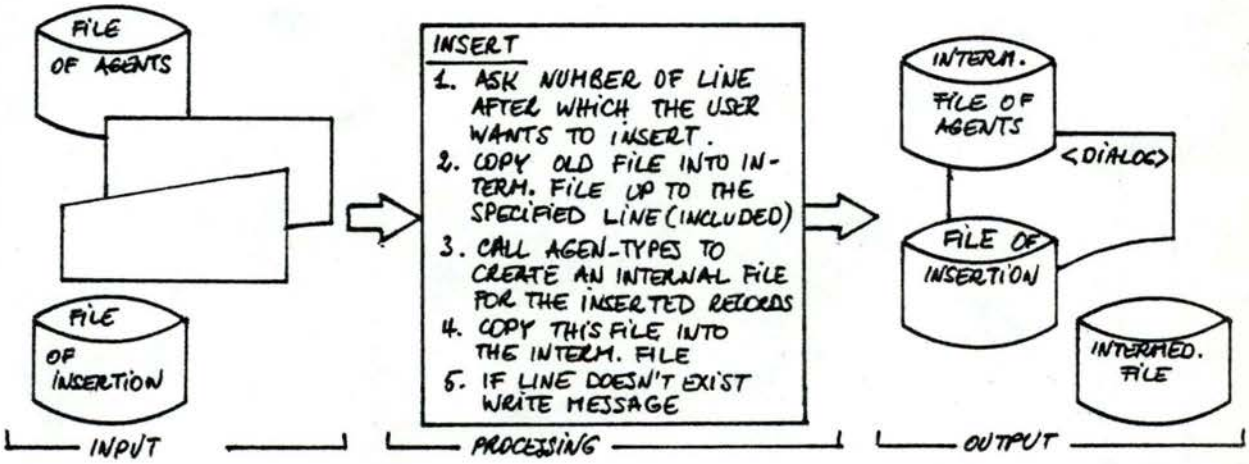
cfr.: programme AGENTS - (1.3.2.3.1.)

9. INSERT - NIVEAU D'ANALYSE 1.3.2.3.2.

9.1. SPECIFICATIONS

La procedure "insert" permet d'inserrer une ou plusieurs lignes derrière la ligne spécifiée du tableau. La procédure doit copier le fichier des agents jusqu'à cette ligne (y comprise) dans un fichier intermédiaire (fa2). Ensuite cette procédure fait elle-mme appel à la procédure "agent_types" qui crée un autre fichier intermédiaire (fa - fa3) qui sera copié à la fin des insertions dans le premier fichier intermédiaire et ensuite détruit. La procédure assure le contrôle de validité du numéro de ligne spécifié.

9.2. SCHEMA



9.3. DESCRIPTION

9.3.1. Entrées

- fichier des agents, (fia - <nom_du_modèle>2);
- réponses en provenance du terminal;
- fichier d'insertion crée lors de l'appel à "agent_types" , (fa - fa3).

9.3.2. Sorties

- questions et messages affichés au terminal;
- fichier intermédiaire des agents, (fa2 -);
- fichier des insertions, (fa -fa3).

9.3.3. Traitements

- cfr.: schéma et code Pascal.

9.4. CODE PASCAL CORRESPONDANT

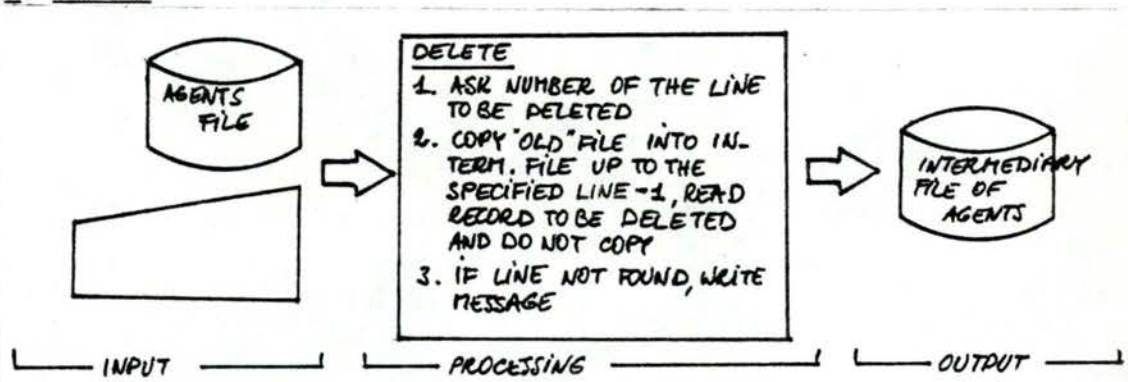
cfr.: programme AGENTS - (1.3.2.3.2.)

10. DELETE - NIVEAU D'ANALYSE - 1.3.2.3.3.

10.1. SPECIFICATIONS

La procédure "delete" détruit la ligne spécifiée par l'utilisateur. La recherche se fait séquentiellement et un contrôle de validité du numéro de ligne est assuré.

10.2. SCHEMA



10.3. DESCRIPTION

10.3.1. Entrées

- fichier des agents, (fa - <nom_du_modèle>2);
- réponses en provenance du terminal.

10.3.2. Sorties

- questions et messages affichés au terminal;
- fichier intermédiaire des agents, (fa2 -).

10.3.3. Traitements

- cfr.: schéma et code Pascal.

10.4. CODE PASCAL CORRESPONDANT

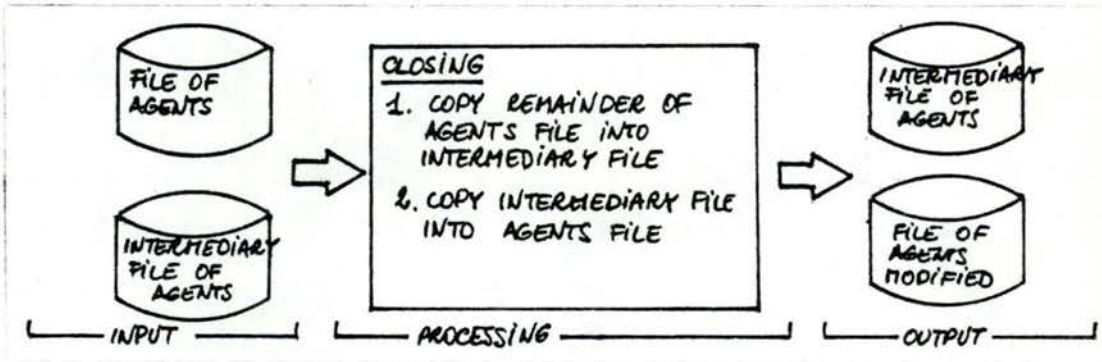
cfr.: programme AGENTS - (1.3.2.3.3.)

11. CLOSING - NIVEAU D'ANALYSE 1.3.2.3.4.

11.1. SPECIFICATIONS

Cette procédure doit fermer les fichiers encore ouverts et bien remettre les agents définis et modifiés dans le fichier des agents intermédiaire constitué dans les procédures "modify", "insert" et "delete" (fa2).

11.2. SCHEMA



11.3. DESCRIPTION

11.3.1. Entrées

- fichier des agents, (fia - <nom_du_modèle>2);
- fichier intermédiaire des agents, (fa2 -).

11.3.2. Sorties

- fichier intermédiaire des agents;
- fichier des agents mis à jour.

11.3.3. Traitements

- cfr.: schéma et code Pascal.

11.4. CODE PASCAL CORRESPONDANT

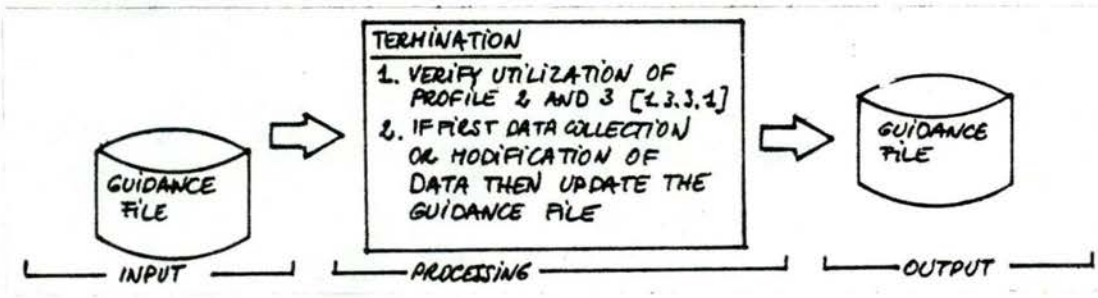
cfr.: programme AGENTS - (1.3.2.3.4.)

12. TERMINATION - NIVEAU D'ANALYSE 1.3.3.

12.1. SPECIFICATIONS

Le rôle de cette partie est de mémoriser le fait qu'on est passé par la collecte des données relatives aux agents. Cependant cette opération n'est exécutée que dans le cas d'un premier enregistrement des informations. Une mise à jour des informations relatives aux agents peut nécessiter une redéfinition des liens (chapitre 7), c'est pourquoi la modification des données est mémorisée.

12.2. SCHEMA



12.3. DESCRIPTION

12.3.1. Entrées

- fichier de guidage, (f - ‹nom_du_modèle›).

12.3.2. Sorties

- fichier de guidage mis à jour.

12.3.3. Traitements

- cfr.: schéma et code Pascal.

12.4. CODE PASCAL CORRESPONDANT

cfr.: programme AGENTS - (1.3.3.)

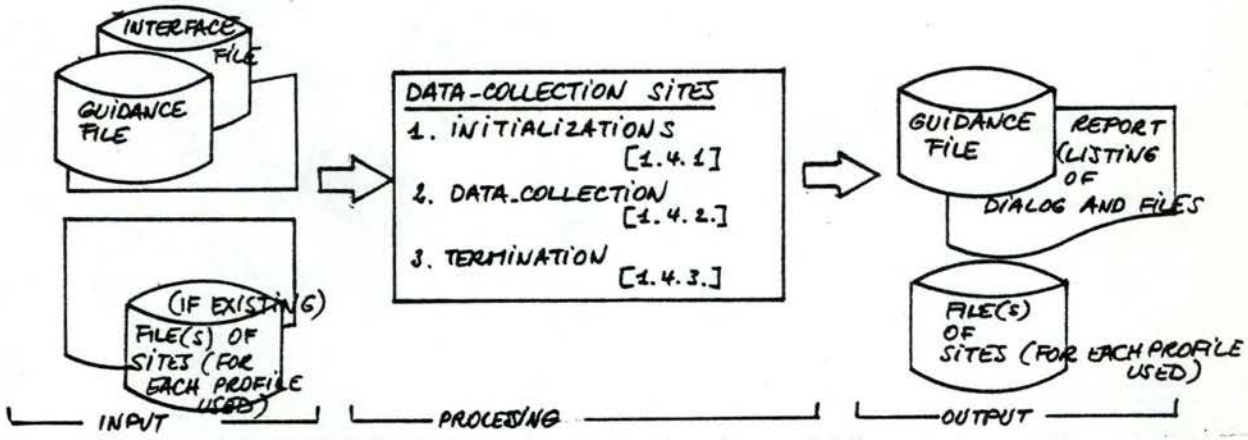
CHAPITRE 6 : COLLECTE DES DONNEES - SITES

1. MODULE 4 - DATA COLLECTION : SITES - NIVEAU D'ANALYSE 1.4.

1.1. SPECIFICATIONS

Le module "sites" doit réaliser la collecte de données concernant les sites. En partant des informations interface et de guidage, le module crée un fichier des sites et le rend permanent ou accède au fichier existant. Le module doit également permettre des modifications sur le fichier des sites. Les fonctions de modification sont les suivantes : "insertion", "suppression" et "modification". Une fois la collecte et les modifications terminées, le module met à jour les informations du fichier de guidage.

1.2. SCHEMA



1.3. DESCRIPTION

1.3.1. Entrées

- fichier interface, (fg -);
- fichier de guidage, (f - ‹nom_du_modèle›);
- fichier(s) des sites (un par profil) s'ils existent déja, (fs - ‹nom_du_modèle›31/32/33);
- réponses en provenance du terminal.

1.3.2. Sorties

- fichiers des sites, (fs - <nom_du_modèle>31/32/33);
- fichier de guidage;
- questions, messages et tableaux imprimés au terminal.

1.3.3. Traitements

- cfr.: schéma et code Pascal.

1.4. CODE PASCAL CORRESPONDANT

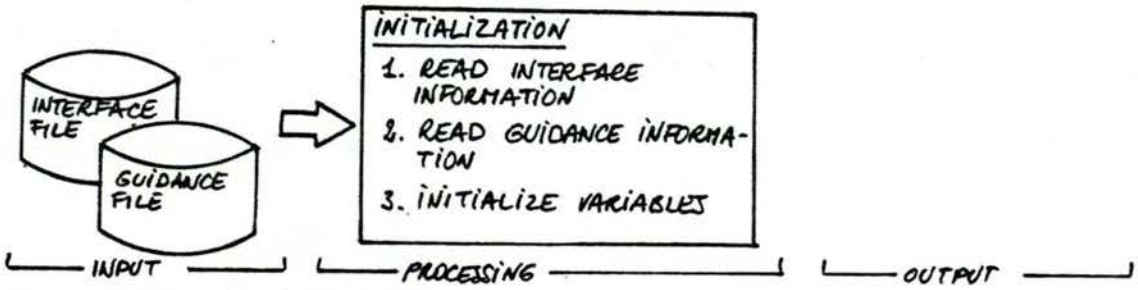
cfr.: programme SITES -(1.4.)

2. INITIALIZATION - NIVEAU D'ANALYSE 1.4.1.

2.1. SPECIFICATIONS

La première tâche de cette partie est d'initialiser le module, c'est-à-dire de lire les informations dites d'interface. A partir de ces données, le système accède aux informations de guidage concernant le modèle de réseau sur lequel le programme s'exécute. Ces informations permettent de générer de façon automatique des fichiers externes dans lesquels seront stocké les informations sur les sites.

2.2. SCHEMA



2.3. DESCRIPTION

2.3.1. Entrées

- fichier interface, (fg -);
- fichier de guidage, (f - <nom_du_modèle>).

2.3.2. Sorties

- pas de sorties.

2.3.3. Traitements

- cfr.: schèma et code Pascal.

2.4. CODE PASCAL CORRESPONDANT

cfr.: programme SITES - (1.4.1.)

3. DATA_COLLECTION - NIVEAU D'ANALYSE 1.4.2.

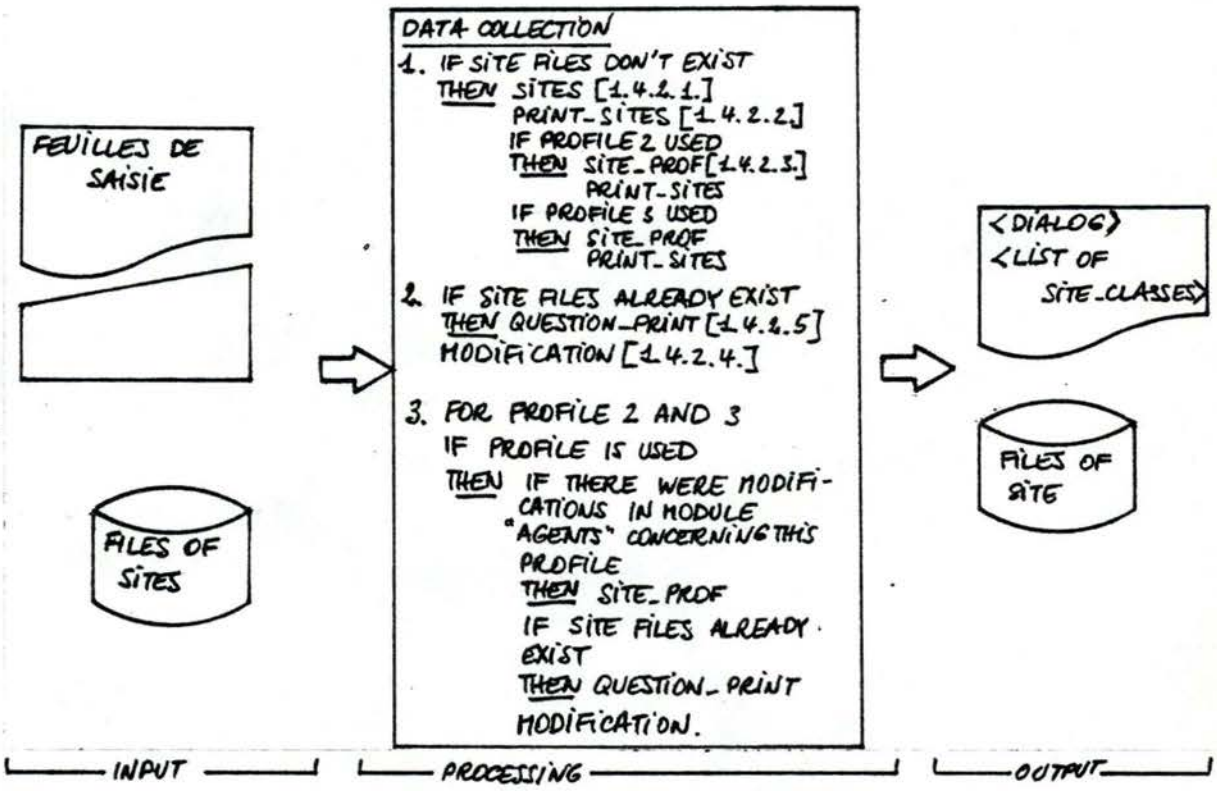
3.1. SPECIFICATIONS

A ce niveau le système connaît le nom du modèle sur lequel il doit travailler, ainsi que les informations concernant l'avancement des travaux sur le modèle considèrè.

La première tâche du système est de voir si l'utilisateur est dèjà passè par la collecte des donnèes relatives aux sites. S'il n'a pas encore entrè de donnèes il faut passer par la collecte, ensuite le système prèsente ces donnèes sous forme d'un tableau. Le lecteur intèressè trouvera un exemple de ce tableau au chapitre 7 du manuel d'utilisation. Si l'utilisateur a dèjà entrè des informations sur les sites, on demande à l'utilisateur s'il veut ou ne veut pas avoir une impression du contenu des fichiers (un par profil utilisè).

La deuxième tâche du système est de permettre des modificationss des fichiers relatifs aux donnèes "sites".

3.2. SCHEMA



3.3. DESCRIPTION

3.3.1. Entrées

- réponses en provenance du terminal;
- fichier(s) des sites, s'ils existent déja, (fs - <nom_du_modèle>31/32/33).

3.3.2. Sorties

- questions,messages et tableaux imprimés au terminal;
- fichier(s) des sites.

3.3.3. Traitements

- cfr.: schéma et code Pascal.

3.4. CODE PASCAL CORRESPONDANT

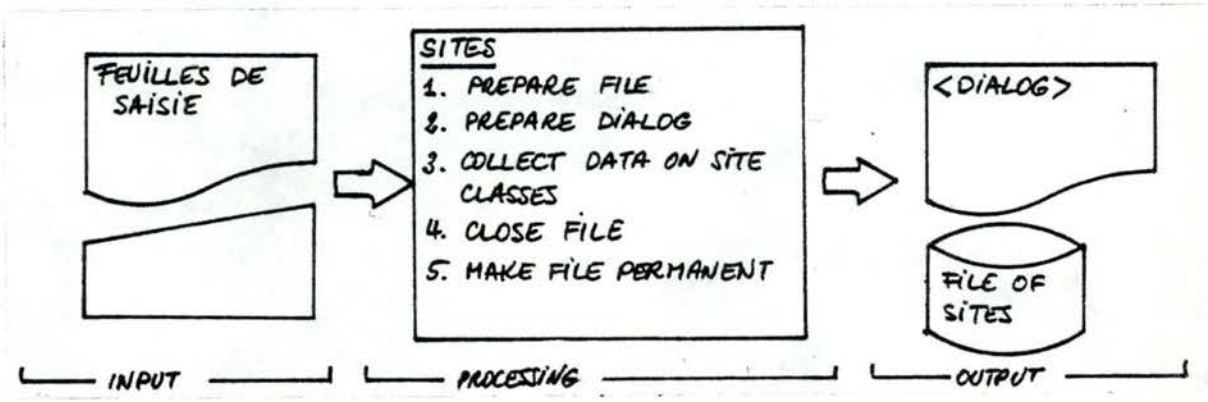
cfr.: programme SITES - (1.4.2.)

4. SITES - NIVEAU D'ANALYSE 1.4.2.1

4.1. SPECIFICATIONS

Cette procédure doit assurer la collecte des données et la confection d'un fichier pour le premier profil. Le travail de l'utilisateur est assisté au maximum par le système, c'est-à-dire les système assure des contrôles et fournit des commentaires et explications à l'utilisateur.

4.2. SCHEMA



4.3. DESCRIPTION

4.3.1. Entrées

- rèponses en provenance du terminal.

4.3.2. Sorties

- fichier des sites, (fs - ‹nom_du_modèle›31);
- questions et messages affichès au terminal.

4.3.3. Traitements

- cfr.: schèma et code Pascal.

4.4. CODE PASCAL CORRESPONDANT

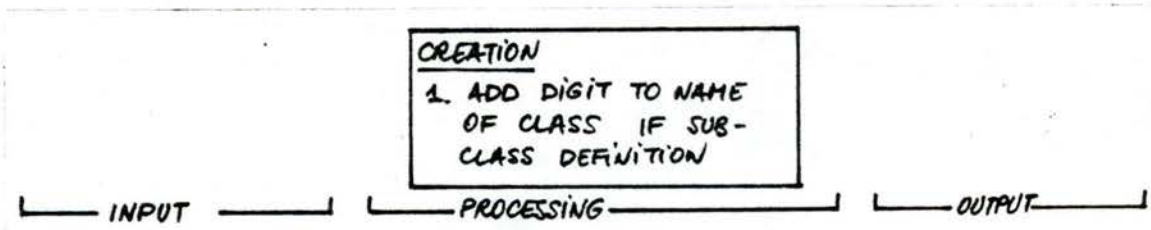
cfr.: programme SITES - (1.4.2.1.)

5. CREATION - NIVEAU D'ANALYSE 1.4.2.1.1.

5.1. SPECIFICATIONS

Des sous-classes de sites pouvant être dèfinies, et l'utilisateur ne fournissant qu'une fois le nom de la classe, il est nècessaire de crèer dynamiquement le nom des sous-classes afin qu'elles portent chacune un nom distinct.

5.2. SCHEMA



5.3. DESCRIPTION

5.3.1. Entrées

- pas d'entrées.

5.3.2. Sorties

- pas de sorties.

5.3.3. Traitements

- cfr.: schéma et code Pascal.

5.4. CODE PASCAL CORRESPONDANT

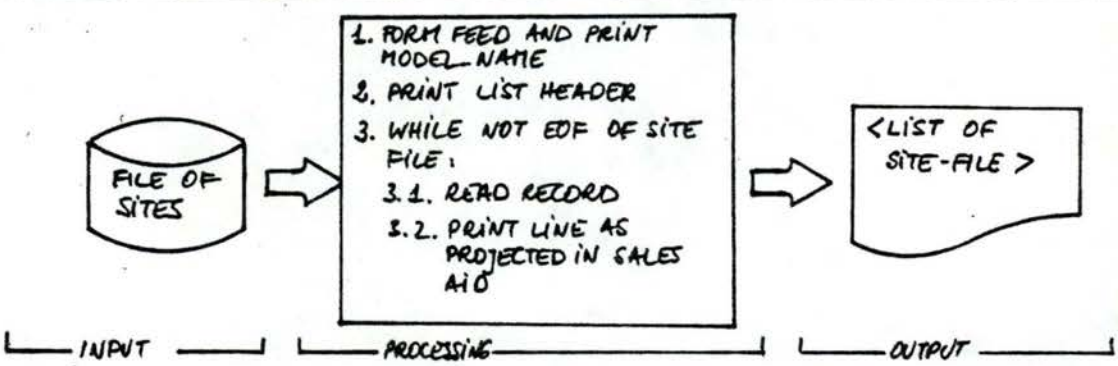
cfr.: programme SITES - (1.4.2.1.1.)

6. PRINT_SITES - NIVEAU D'ANALYSE 1.4.2.2.

6.1. SPECIFICATIONS

En partant du fichier des sites, la procédure imprime une liste des sites (voir chapitre 7 du manuel d'utilisation).

6.2. SCHEMA



6.3. DESCRIPTION

6.3.1. Entrées

- fichier des sites, (fs - <nom_du_modèle>31 ou 32 ou 33).

6.3.2. Sorties

- liste (tableau) imprimée au terminal.

6.3.3. Traitements

- cfr.: schéma et code Pascal.

6.4. CODE PASCAL CORRESPONDANT

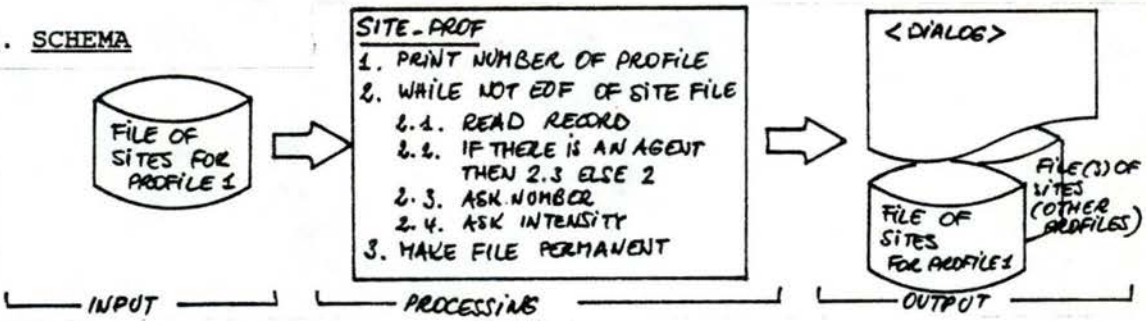
cfr.: programme SITES - (1.4.2.2.)

7. SITE_PROF - NIVEAU D'ANALYSE 1.4.2.3.

7.1. SPECIFICATIONS

Dans le cas où le(s) profil(s) 2 et/ou 3 est(sont) utilisé(s), seules les informations caractérisant les agents implantés sur le site peuvent changer. En reprenant les informations enregistrées pour le premier profil, les questions seront reposées uniquement pour les agents.

7.2. SCHEMA



7.3. DESCRIPTION

7.3.1. Entrées

- fichier des sites relatif au profil 1, (fs - <nom_du_modèle>31).
- réponses en provenance du terminal.

7.3.2. Sorties

- questions et messages affichés au terminal;
- fichier(s) des sites relatif(s) au (x) profil(s) 2 et 3, (fs - <nom_du_modèle>32 et 33).

7.3.3. Traitements

- cfr.: schéma et code Pascal.

7.4. CODE PASCAL CORRESPONDANT

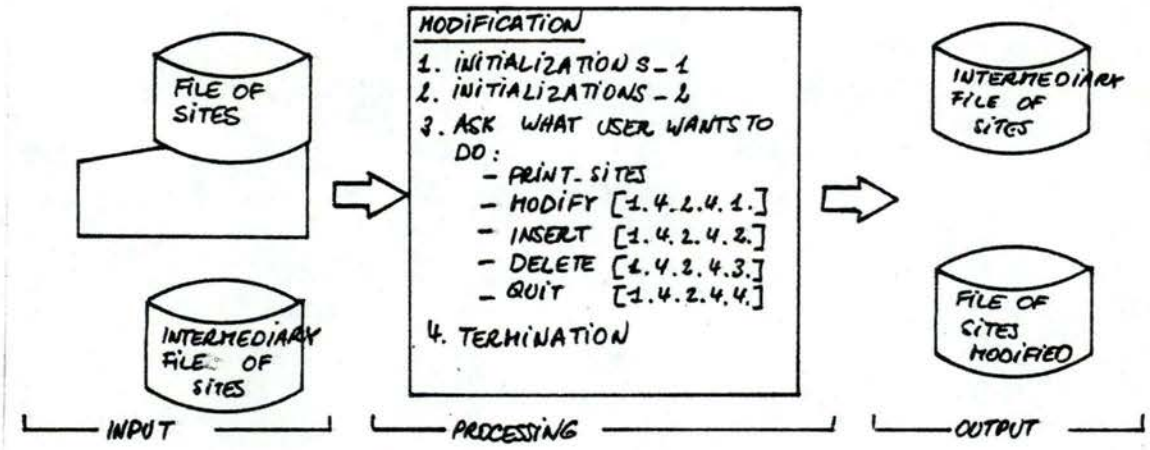
cfr.: programme SITES - (1.4.2.3.)

8. MODIFICATION - NIVEAU D'ANALYSE 1.4.2.4.

8.1. SPECIFICATIONS

La procédure de modification permet de modifier le contenu du fichier des sites. Ainsi l'utilisateur peut changer son modèle de réseau sans devoir tout recommencer. Par modification nous entendons : modification des attributs d'un enregistrement, insertion de un ou plusieurs enregistrements, la suppression d'un enregistrement et l'impression du contenu du fichier pour faciliter les modifications.

8.2. SCHEMA



8.3. DESCRIPTION

8.3.1. Entrèes

- fichier des sites relatif à un profil, (fis - <nom_du_modèle>3?);
- rèponses en provenance du terminal;
- fichiers intermèdiaires des sites.

8.3.2. Sorties

- questions et messages affichèes au terminal;
- fichier intermèdiaires des sites
- fichier des sites mis à jour.

8.3.3. Traitements

- cfr.: schèma et code Pascal.

8.4. CODE PASCAL CORRESPONDANT

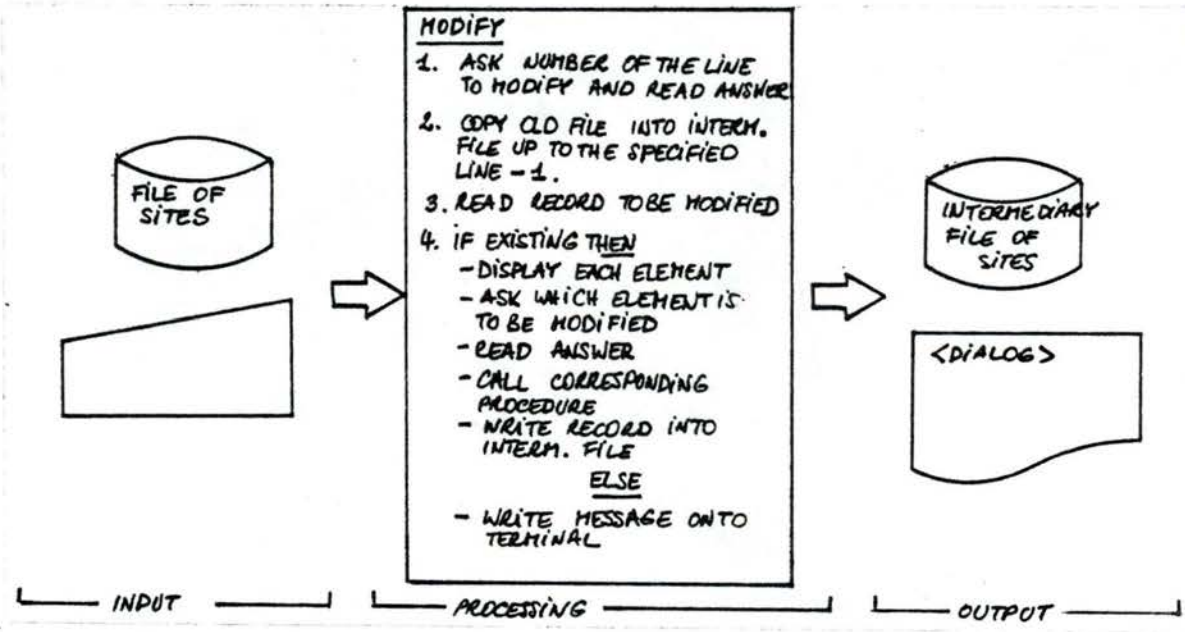
cfr.: programme SITES - (1.4.2.4.)

9. MODIFY - NIVEAU D'ANALYSE 1.4.2.4.1.

9.1. SPECIFICATIONS

La procédure "modify" permet de modifier les attributs d'un enregistrement spécifié par le numéro de ligne de cet enregistrement dans le tableau des sites. La procédure affiche les différents attributs de l'enregistrement et procède aux modifications souhaitées. Bien entendu, la procédure assure le contrôle de validité du numéro de ligne à modifier.

9.2. SCHEMA



9.3. DESCRIPTION

9.3.1. Entrées

- fichier des sites relatif à un profil, (fis - ‹nom_du_modèle›3?);
- réponses en provenance du terminal.

9.3.2. Sorties

- fichier intermédiaire des sites, (fs2 -).
- questions et messages affichés au terminal.

9.3.3. Traitements

- cfr.: schéma et code Pascal.

9.4. CODE PASCAL CORRESPONDANT

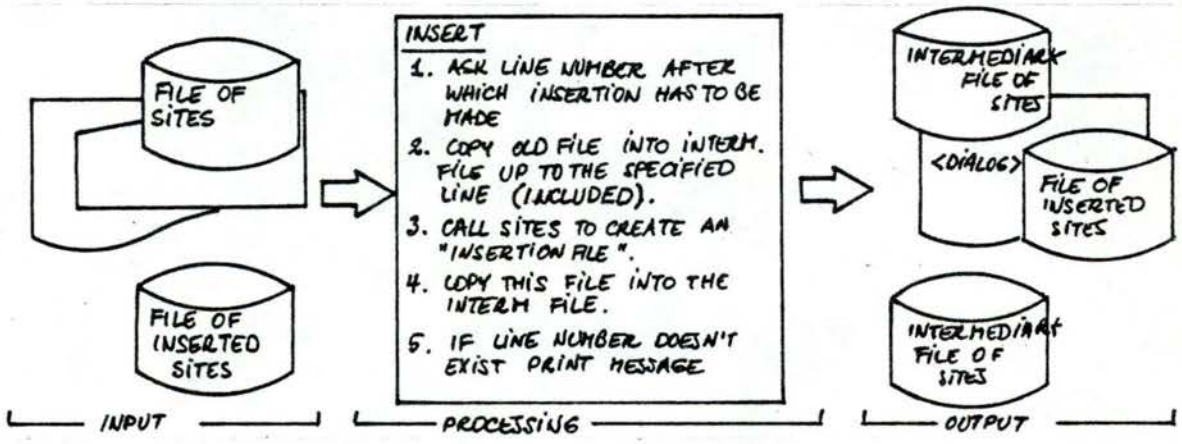
cfr.: programme SITES - (1.4.2.4.1.)

10. INSERT - NIVEAU D'ANALYSE 1.4.2.4.2.

10.1. SPECIFICATIONS

La procédure permet d'inserrer une ou plusieurs lignes derrière la ligne spécifiée. La procédure copie le fichier jusqu'à la ligne (y comprise) dans un fichier intermédiaire (fs2). Elle fait appel à la procédure "sites" pour l'insertion . Il y a création d'un autre fichier intermédiaire (fs - fs3), qui à son tour sera copié et ensuite détruit. La procédure assure le contrôle de validité sur le numéro de ligne.

10.2. SCHEMA



10.3. DESCRIPTION

10.3.1. Entrèes

- fichier des sites relatif à un profil, (fis - ‹nom_du_modèle›3?);
- fichier des insertions, (fs - fs3);
- rèponses en provenance du terminal.

10.3.2. Sorties

- questions et messages affichèes au terminal;
- fichier intermèdiaire des sites, (fs2);
- fichier des insertions.

10.3.3. Traitements

- cfr.: schèma et code Pascal.

10.4. CODE PASCAL CORRESPONDANT

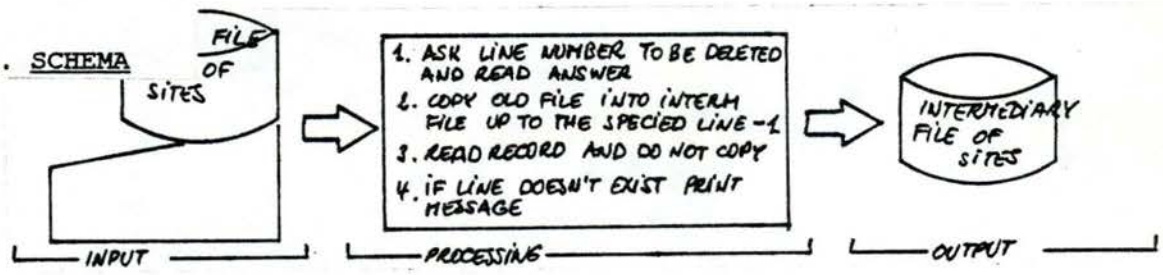
cfr.: programme SITES - (1.4.2.4.2.)

11. DELETE - NIVEAU D'ANALYSE 1.4.2.4.3.

11.1. SPECIFICATIONS

La procèdure "delete" dètruit la ligne spècifièe par l'utilisateur. La recherche se faisant sèquentiellement, il faut faire un contrôle sur le numèro de ligne à dètruire.

11.2. SCHEMA



11.3. DESCRIPTION

11.3.1. Entrées

- fichier des sites relatif à un profil, (fis - ‹nom_du_modèle›3?);
- réponses en provenance du terminal.

11.3.2. Sorties

- fichier intermédiaire des sites, (fs2);
- questions et messages affichés au terminal.

11.3.3. Traitements

- cfr.: schéma et code Pascal.

11.4. CODE PASCAL CORRESPONDANT

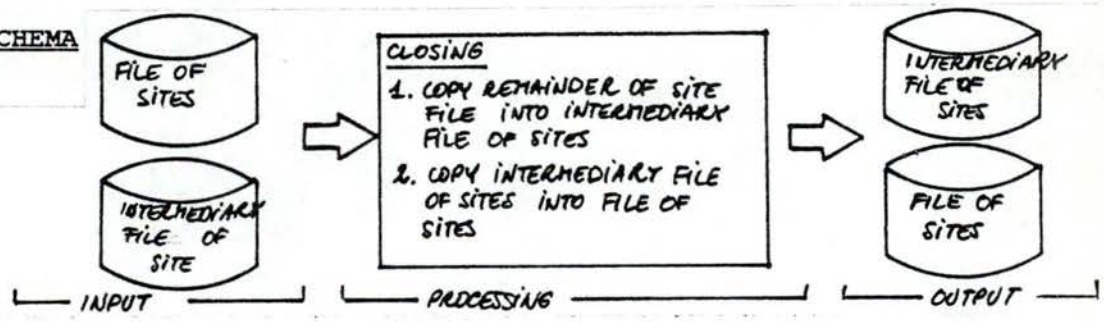
cfr.: programme SITES - (1.4.2.4.3.)

12. CLOSING - NIVEAU D'ANALYSE 1.4.2.4.4.

12.1. SPECIFICATIONS

Cette procédure est destinée à fermer les fichiers encore ouverts et de bien remettre les sites définis et modifiés dans le fichier des sites connu sous le nom externe '‹nom_du_modèle›3?'.

12.2. SCHEMA



12.3. DESCRIPTION

12.3.1. Entrées

- fichier des sites relatif à un profil, (fis - <nom_du_modèle>3?);
- fichier intermédiaire des sites, (fs2).

12.3.2. Sorties

- fichier intermédiaire des sites;
- fichier des sites relatif à un profil mis à jour.

12.3.3. Traitements

- cfr.: schéma et code Pascal.

12.4. CODE PASCAL CORRESPONDANT

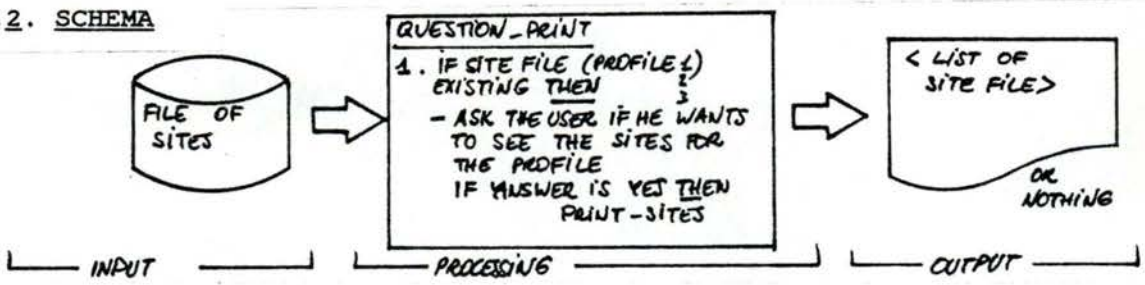
cfr.: programme SITES - (1.4.2.4.4.)

13. QUESTION_PRINT - NIVEAU D'ANALYSE 1.4.2.5.

13.1. SPECIFICATIONS

Dans le cas où le fichier des sites a déjà été introduit pour le profil concerné, la procédure permet une impression du contenu de ce fichier.

13.2. SCHEMA



13.3. DESCRIPTION

13.3.1. Entrées

- fichier des sites, (fs - <nom_du_modèle>3?);
- réponses en provenance du terminal.

13.3.2. Sorties

- question affichée au terminal;
- liste des sites imprimée au terminal.

13.3.3. Traitements

- cfr.: schéma et code Pascal.

13.4. CODE PASCAL CORRESPONDANT

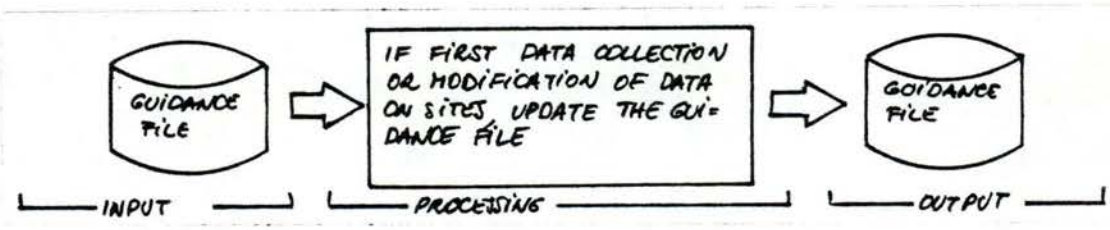
cfr.: programme SITES - (1.4.2.5.)

14. TERMINATION - NIVEAU D'ANALYSE 1.4.3.

14.1. SPECIFICATIONS

Le rôle de cette partie est de mémoriser le fait qu'on est passé par la collecte des données "sites". Ceci est uniquement fait dans le cas d'un premier enregistrement des données ou d'une mise à jour des données.

14.2. SCHEMA



14.3. DESCRIPTION

14.3.1. Entrées

- fichier de guidage, (f - ‹nom_du_modèle›).

14.3.2. Sorties

- fichier de guidage mis à jour.

14.3.3. Traitements

- cfr.: schéma et code Pascal.

14.4. CODE PASCAL CORRESPONDANT

cfr.: programme SITES - (1.4.3.)

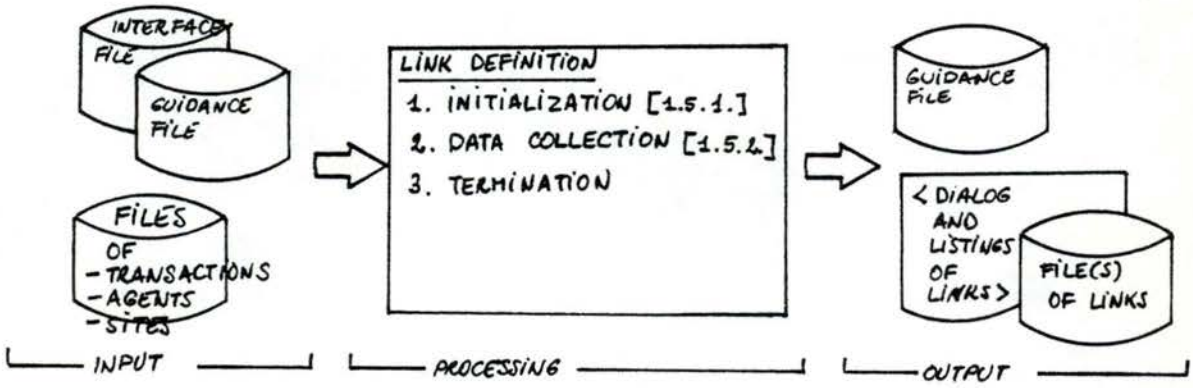
CHAPITRE 7 : COLLECTES DES DONNEES - LIENS

1. MODULE 5 : LINK DEFINITION - NIVEAU D'ANALYSE 1.5.

1.1. SPECIFICATIONS

Le module "link definition" propose tous les liens logiquement possibles entre classes de sites. L'utilisateur sélectionnera et caractérisera les liens qu'il désire garder. Le module crée un fichier des liens (un fichier par profil utilisé) et il le rend permanent, ou accède aux fichiers existant. Le module permet de redéfinir complètement tous les liens. Cette définition repose sur le procédé de localisation des transactions exposé dans le sales aid.

1.2. SCHEMA



1.3. DESCRIPTION

1.3.1. Entrées

- fichier interface, (fg -);
- fichier de guidage, (f - ‹nom_du_modèle›);
- fichier des transactions, (ft - ‹nom_du_modèle›1);
- fichier des agents, (fa - ‹nom_du_modèle›2);
- fichier(s) des sites, (fs - ‹nom_du_modèle›31 ou/et 32 ou/et 33);
- réponses en provenance du terminal.

1.3.2. Sorties

- fichier de guidage;
- fichiers des liens, (fl - <nom_du_modèle>41 ou/et 42 ou/et 43);
- questions, messages et tableaux affichés au terminal.

1.3.3. Traitements

- cfr.: schèma et code Pascal.

1.4. CODE PASCAL CORRESPONDANT

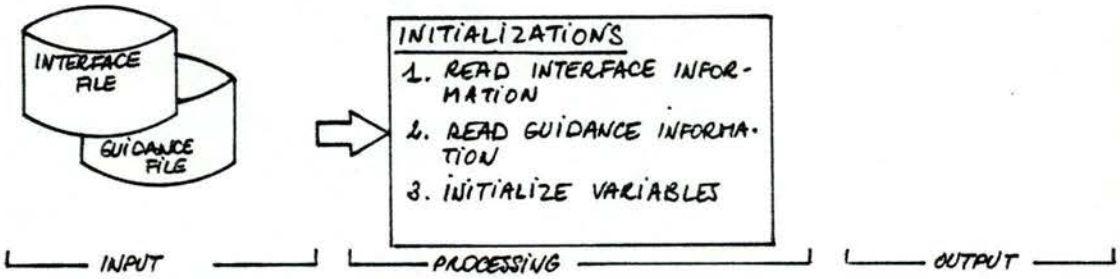
cfr.: programme TRANSLOC - (1.5.)

2. INITIALIZATION

2.1. SPECIFICATIONS

La première tâche est d'initialiser le module, c'est-à-dire de lire les informations dites d'interface. A partir de ces données, le système accède aux informations de guidage concernant le modèle de réseau sur lequel le programme s'exécute. Ces informations permettent de générer de façon automatique le nom des fichiers externes dans lesquels seront stocké les informations relatives aux liens.

2.2. SCHEMA



2.3. DESCRIPTION

2.3.1. Entrées

- fichier interface, (fg -);
- fichier de guidage, (f - ‹nom_du_modèle›).

2.3.2. Sorties

- pas de sorties.

2.3.3. Traitements

- cfr.: schéma et code Pascal.

2.4. CODE PASCAL CORRESPONDANT

cfr.: programme TRANSLOC - (1.5.1.)

3. DATA_COLLECTION - NIVEAU D'ANALYSE 1.5.2.

3.1. SPECIFICATIONS

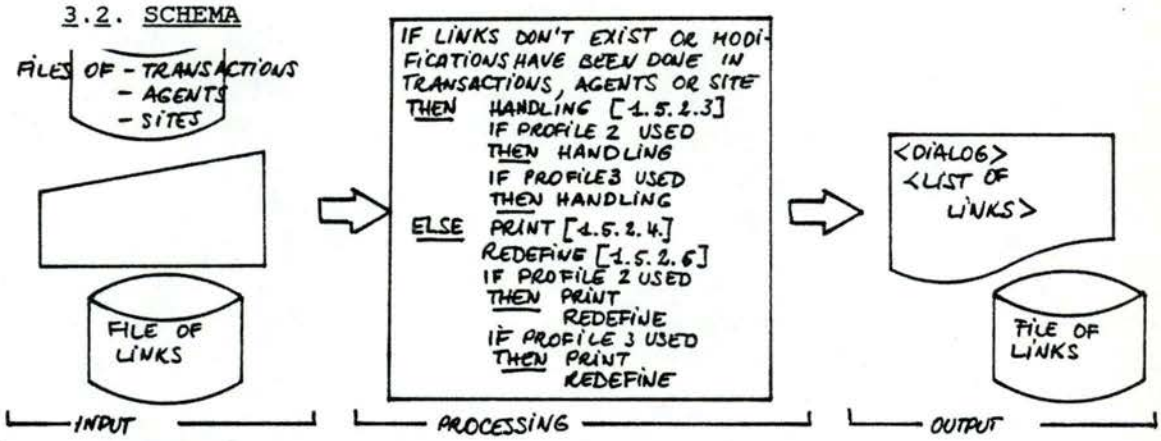
A ce niveau le système connaît le nom du modèle sur lequel il doit travailler, ainsi que les informations concernant l'avancement des travaux sur le modèle considéré.

La première tâche du système est de voir si l'utilisateur est déjà passé par la définition des liens, ou s'il a fait une modification dans les fichiers transactions, agents et / ou sites. Dans ce cas, il faut passer par la redéfinition des liens, ensuite le système affiche les résultats sous forme d'un tableau présenté au chapitre 7 du manuel d'utilisation. Il y a une définition des liens par profil utilisé.

Si l'utilisateur a déjà entré des informations sur les liens et s'il n'a fait aucune modification dans le fichier des transactions, des agents et des sites, on demande à l'utilisateur s'il veut ou ne veut pas avoir une impression du contenu des

fichiers (un par profil utilisé). Ensuite l'utilisateur peut procéder à une redéfinition complète des liens.

3.2. SCHEMA



3.3. DESCRIPTION

3.3.1. Entrées

- fichier(s) des liens, (fl - ‹nom_du_modèle›41 ou/et 42 ou/et 43);
- fichier des transactions, (ft - ‹nom_du_modèle›1);
- fichier des agents, (fa - ‹nom_du_modèle›2);
- fichier(s) des sites, (fs - ‹nom_du_modèle›31 ou/et 32 ou/et 33);
- réponses en provenance du terminal.

3.3.2. Sorties

- questions, messages et tableaux affichés au terminal;
- fichiers des liens.

3.3.3. Traitements

- cfr.: schéma et code Pascal.

3.4. CODE PASCAL CORRESPONDANT

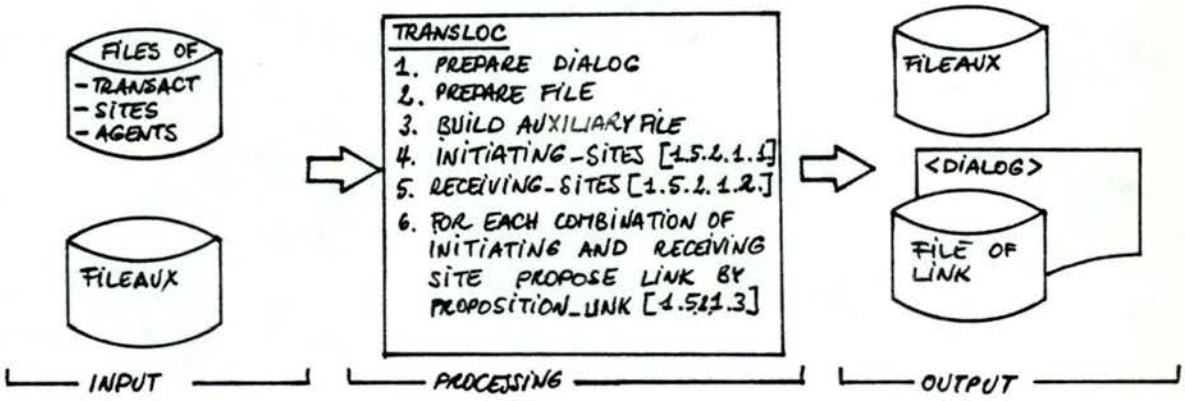
cfr.: programme TRANSLOC - (1.5.2.)

4. TRANS_LOC - NIVEAU D'ANALYSE 1.5.2.1.

4.1. SPECIFICATIONS

Cette procédure fournira tous les liens logiquement possibles entre classes de sites. L'utilisateur sélectionne les liens qu'il désire conserver et donne leurs caratéristiques. Son travail est assisté au maximum par le système, c'est-à-dire le système assure des contrôles et fournit des commentaires et explications à l'utilisateur.

4.2. SCHEMA



4.3. DESCRIPTION

4.3.1. Entrées

- fichier des transactions, (ft - ‹nom_du_modèle›1);
- fichier des agents, (fa - ‹nom_du_modèle›2);
- fichier(s) des sites, (fs - ‹nom_du_modèle›31 ou/et 32 ou/et 33);
- réponses en provenance du terminal;
- fileaux.

4.3.2. Sorties

- questions et messages affichés au terminal;
- fichier(s) des liens, (fl - ‹nom_du_modèle›41 ou/et 42 ou/et 43);
- fileaux.

4.3.3. Traitements

- cfr.: schéma et code Pascal.

4.4. CODE PASCAL CORRESPONDANT

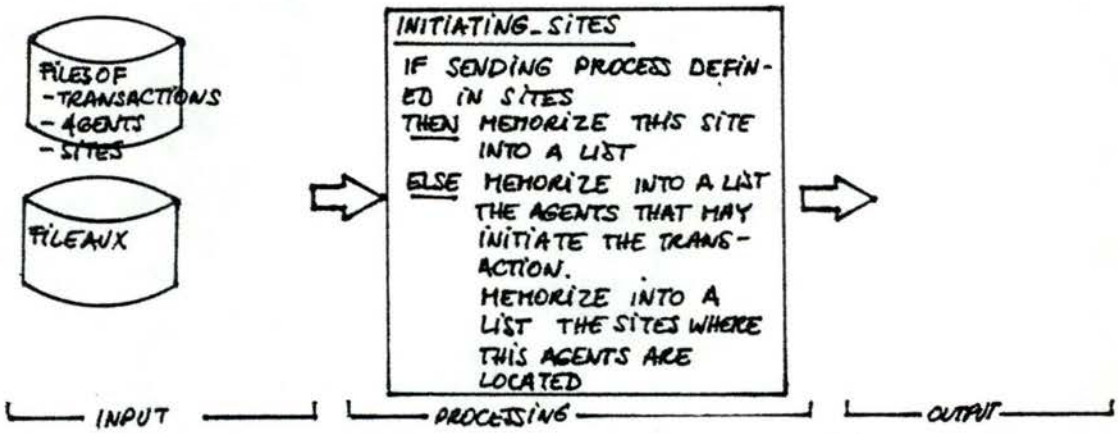
cfr.: programme TRANSLOC - (1.5.2.1.)

5. INITIATING_SITES - NIVEAU D'ANALYSE 1.5.2.1.1.

5.1. SPECIFICATIONS

Pour une ligne du tableau des transactions, cette procédure va rechercher toutes les classes de sites à partir desquelles un agent peut initialiser le processus qui traite la transaction et mémoriser ces sites dans une liste.

5.2. SCHEMA



5.3. DESCRIPTION

5.3.1. Entrées

- fichier des transactions, (ft - <nom_du_modèle>1);
- fichier des agents, (fa - <nom_du_modèle>2);
- fichier(s) des sites, (fs - <nom_du_modèle>31 ou/et 32 ou/et 33);
- fichier auxiliaire des transactions, (fileaux -).

5.3.2. Sorties

- pas de sorties.

5.3.3. Traitements

- cfr.: schèma et code Pascal.

5.4. CODE PASCAL CORRESPONDANT

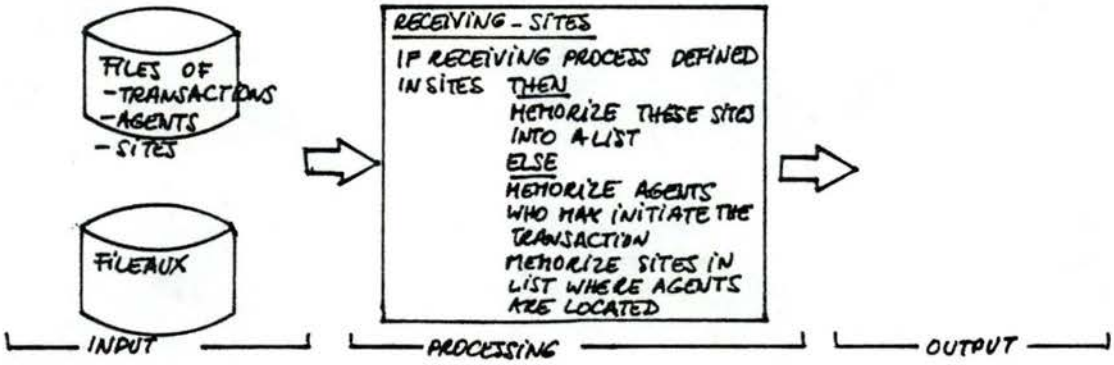
cfr.: programme TRANSLOC - (1.5.2.1.1.)

6. RECEIVING_SITES - NIVEAU D'ANALYSE 1.5.2.1.2.

6.1. SPECIFICATIONS

Pour une ligne du tableau des transactions, cette procedure va rechercher tous les sites à partir desquels un agent peut initialiser le processus qui reçoit le message et mèmoriser ces sites dans une liste.

6.1. SCHEMA



6.3. DESCRIPTION

6.3.1. Entrèes

- fichier des transactions, (ft - <nom_du_modèle>1);
- fichier des agents, (fa - <nom_du_modèle>2);
- fichier(s) des sites, (fs - <nom_du_modèle>31 ou/et 32 ou/et 33);
- fichier auxiliaire des transactions, (fileaux -).

6.3.2. Sorties

- pas de sorties.

6.3.3. Traitements

- cfr.: schèma et code Pascal.

6.4. CODE PASCAL CORRESPONDANT

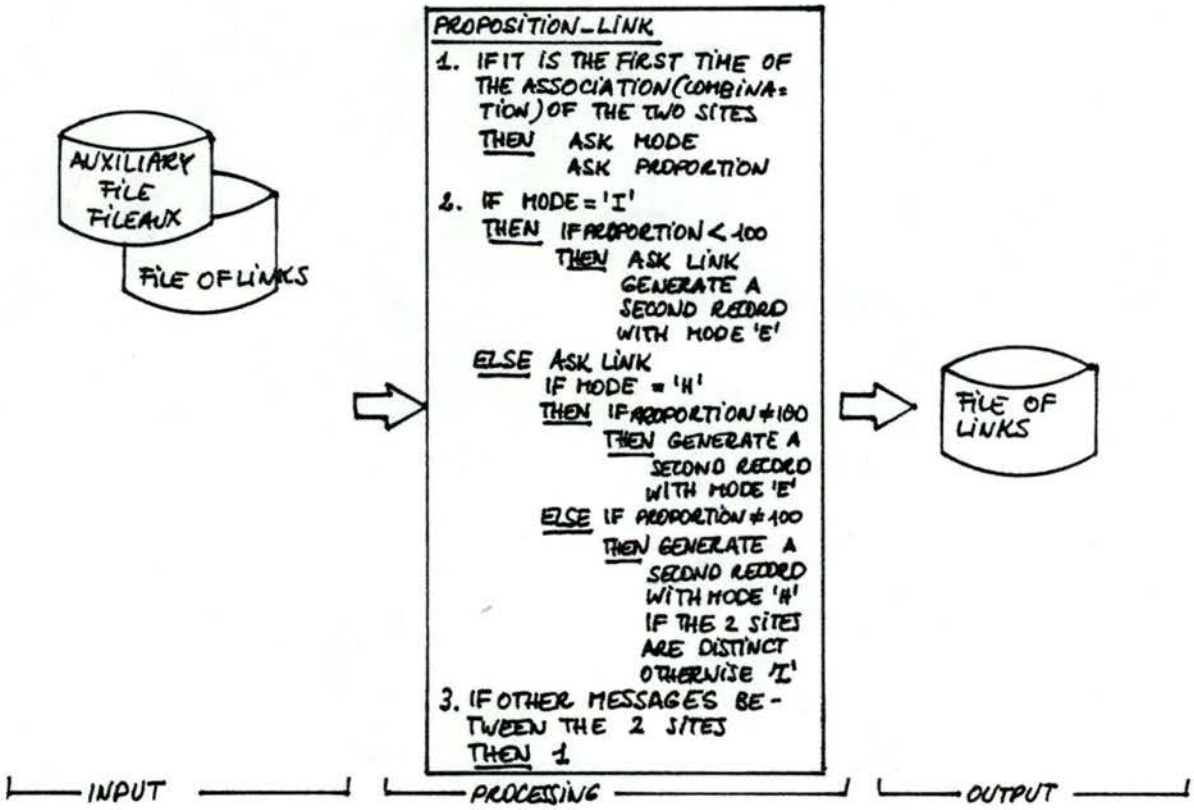
cfr.: programme TRANSLOC - (1.5.2.1.2.)

7. PROPOSITION_LINK - NIVEAU D'ANALYSE 1.5.2.1.3.

7.1. SPECIFICATIONS

Cette procèdure va proposer le lien entre deux classes de sites, demander le mode de communication, la proportion et le type de lien si nècessaire et enregistrer ces informations dans le fichier des liens.

7.2. SCHEMA


AUXILIARY FILE FILEAUX
FILE OF LINKS
PROPOSITION_LINK
1. IF IT IS THE FIRST TIME OF THE ASSOCIATION (COMBINATION) OF THE TWO SITES
THEN ASK MODE
ASK PROPORTION
2. IF MODE = 'I'
THEN IF PROPORTION < 100
THEN ASK LINK
GENERATE A SECOND RECORD WITH MODE 'E'
ELSE ASK LINK
IF MODE = 'H'
THEN IF PROPORTION ≠ 100
THEN GENERATE A SECOND RECORD WITH MODE 'E'
ELSE IF PROPORTION ≠ 100
THEN GENERATE A SECOND RECORD WITH MODE 'H' IF THE 2 SITES ARE DISTINCT OTHERWISE 'I'
3. IF OTHER MESSAGES BETWEEN THE 2 SITES
THEN 1
FILE OF LINKS
INPUT
PROCESSING
OUTPUT

7.3. DESCRIPTION

7.3.1. Entrèes

- fichier auxiliaire, (fileaux -);
- fichier(s) des liens, (fl - ‹nom_du_modèle›41 ou/et 42 ou/et43);
- rèponses en provenance du terminal.

7.3.2. Sorties

- fichier des liens;
- questions et messages affichès au terminal.

7.3.3. Traitements

- cfr.: schèma et code Pascal.

7.4. CODE PASCAL CORRESPONDANT

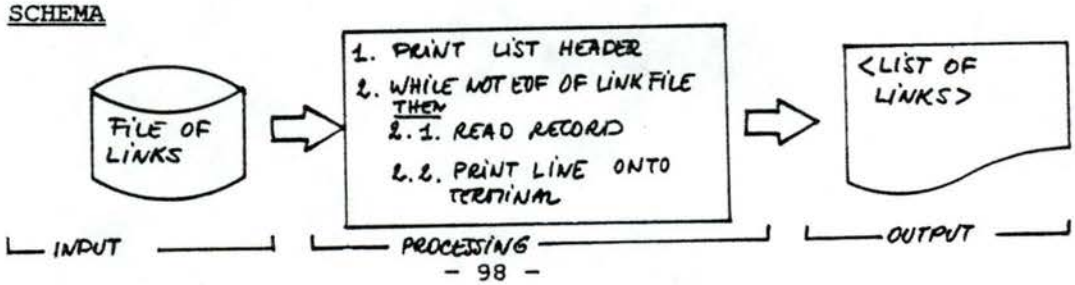
cfr.: programme TRANSLOC - (1.5.2.1.3.)

8. PRINT_TRANS_LOC - NIVEAU D'ANALYSE 1.5.2.2.

8.1. SPECIFICATIONS

En partant du fichier des liens, la procèdure imprime la liste de ces liens. Le lecteur intèressè trouvera un exemple de cette liste au chapitre 7 du manuel d'utilisation.

8.2. SCHEMA



8.3. DESCRIPTION

8.3.1. Entrèes

- fichier des liens, (fl - ‹nom_du_modèle›4?).

8.3.2. Sorties

- tableau affiché au terminal.

8.3.3. Traitements

- cfr.: schèma et code Pascal.

8.4. CODE PASCAL CORRESPONDANT

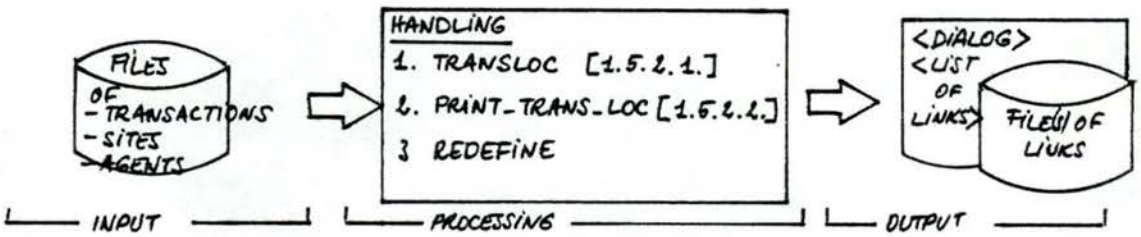
cfr.: programme TRANSLOC - (1.5.2.2.)

9. HANDLING - NIVEAU D'ANALYSE 1.5.2.3.

9.1. SPECIFICATIONS

Cette procèdure assure la collecte des liens sèlectionnès par l'utilisateur, les imprime et lui donne la possibilitè de redèfinir tous les liens qui viennent d'être dèfinis.

9.2. SCHEMA



9.3. DESCRIPTION

9.3.1. Entrèes

- fichier des transactions, (ft - ‹nom_du_modèle›1);
- fichier des agents, (fa - ‹nom_du_modèle›2);
- fichier(s) des sites, (fs - ‹nom_du_modèle›31 ou/et 32 ou/et 33);
- rèponses en provenance du terminal.

9.3.2. Sorties

- fichiers des liens;
- questions et messages affichès au terminal.

9.3.3. Traitements

- cfr.: schèma et code Pascal.

9.4. CODE PASCAL CORRESPONDANT

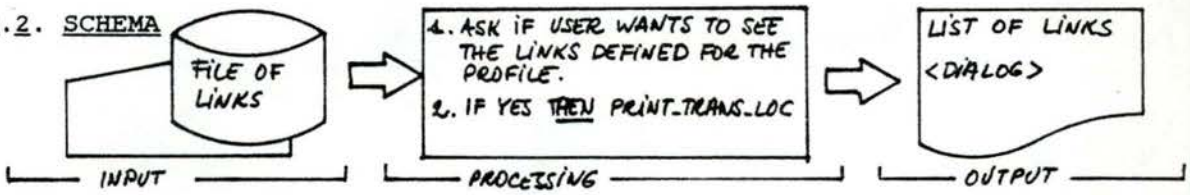
cfr.: programme TRANSLOC - (1.5.2.3.)

10. PRINT - NIVEAU D'ANALYSE 1.5.2.4.

10.1. SPECIFICATIONS

La procèdure demande à l'utilisateur s'il veut ou ne veut pas avoir une impression des liens qu'il a dèfinis, pour le profil concernè, dans le cas où les liens ont dèjà ètè dèfinis ou s'ils ont ètè redèfinis.

10.2. SCHEMA



10.3. DESCRIPTION

10.3.1. Entrèes

- fichier(s) des liens, (fl - <nom_du_modèle>41 ou/et 42 ou/et 43);
- rèponses en provenance du terminal.

10.3.2. Sorties

- questions, messages et listes des liens affichès au terminal.

10.3.3. Traitements

- cfr.: schèma et code Pascal.

10.4. CODE PASCAL CORRESPONDANT

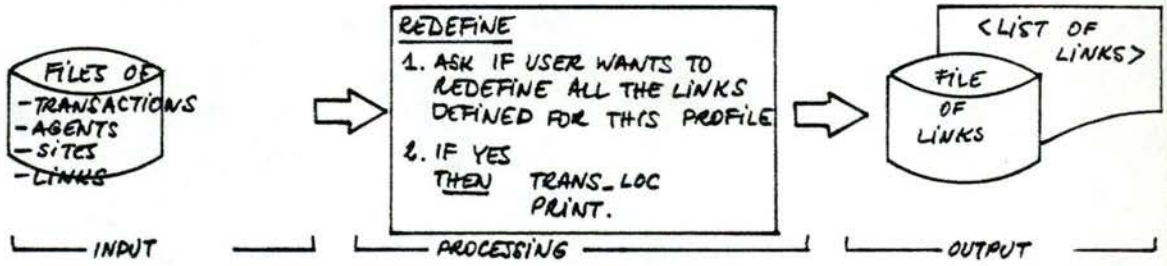
cfr.: programme TRANSLOC - (1.5.2.4.)

11. REDEFINE - NIVEAU D'ANALYSE 1.5.2.5.

11.1. SPECIFICATIONS

Cette procèdure permet à l'utilisateur de redèfinir les liens qui ont ète dèfinis pour le profil considèrè.

11.2. SCHEMA



11.3. DESCRIPTION

11.3.1. Entrées

- fichier des transactions, (ft - ‹nom_du_modèle›1);
- fichier des agents, (fa - ‹nom_du_modèle›2);
- fichier(s) des sites, (fs - ‹nom_du_modèle›31 ou/et 32 ou/et 33);
- fichier(s) des liens, (fl - ‹nom_du_modèle›41 ou/et 42 ou/et 43);
- réponses en provenance du terminal.

11.3.2. Sorties

- fichier des liens;
- questions , liste et messages affichés au terminal.

11.3.3. Traitements

- cfr.: schéma et code Pascal.

11.4. CODE PASCAL CORRESPONDANT

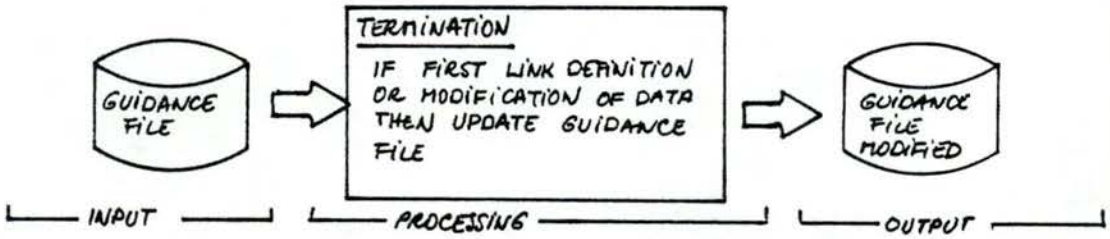
cfr.: programme TRANSLOC - (1.5.2.5.)

12. TERMINATION -NIVEAU D'ANALYSE 1.5.3.

12.1. SPECIFICATIONS

Le rôle de cette procédure est de mémoriser le fait qu'on est passé par la définition des liens. Ceci est uniquement fait dans le cas d'un premier enregistrement des liens, ou d'une mise à jour des données.

12.2. SCHEMA



12.3. DESCRIPTION

12.3.1. Entrèes

- fichier de guidage, (f - ‹nom_du_modèle›).

12.3.2. Sorties

- fichier de guidage.

12.3.3. Traitements

- cfr.: schèma et code Pascal.

12.4. CODE PASCAL CORRESPONDANT

cfr.: programme TRANSLOC - (1.5.3.)

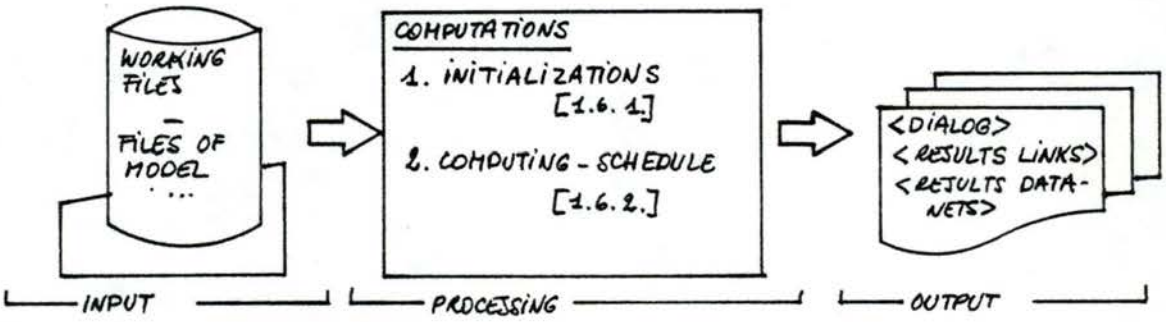
CHAPITRE 8 : CALCULS DE DIMENSIONNMENT

1. MODULE 6 : COMPUTATIONS - NIVEAU D'ANALYSE 1.6.

1.1. SPECIFICATIONS

Le module "computations" doit effectuer les calculs prèvus par le sales aid pour arriver à dimensionner les liens du rèseau primaire et à èvaluer la charge CPU des datanets implantès. Le module se base sur les donnèes utilisateurs collectèes pendant l'exècution des modules prècèdants. Un certain nombre d'informations complèmentaires sont saisies. Ce module, après avoir terminè les calculs, imprime les rèsultats au terminal.

1.2. SCHEMA



1.3. DESCRIPTION

1.3.1. Entrèes

- fichier interface, (fg -);
- fichier de guidage, (f - ‹nom_du_modèle›);
- fichier des transactions, (ft - ‹nom_du_modèle›1);
- fichier des agents, (fa - ‹nom_du_modèle›2);
- fichier(s) des sites (un par profil), (fs - ‹nom_du_modèle›3?);
- fichier(s) des liens (id), (fl - ‹nom_du_modèle›4?);
- fichier des datanets, (filedn -);
- fichier auxiliaire des transactions, (fileaux -);

- fichier des activités des classes, (fileact -);
- fichier de calcul, (filecal -);
- fichier(s) du trafic intersite, (fileint;fileint2 -);
- réponses en provenance du terminal.

1.3.2. Sorties

- questions, messages et résultats imprimés au terminal.

1.3.3. Traitements

- cfr.: schéma et code Pascal.

1.4. CODE PASCAL CORRESPONDANT

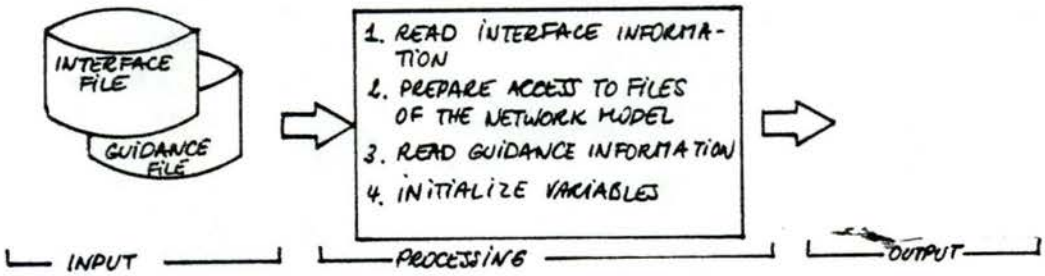
cfr.: programme COMPUTAT - (1.6.)

2. INITIALIZATIONS - NIVEAU D'ANALYSE 1.6.1.

2.1. SPECIFICATIONS

Cette partie accède aux informations contenues dans le fichier interface. A partir de ces informations elle accède aux informations relatives à l'avancement des travaux sur le modèle en question. Cette partie prépare l'accès aux fichiers qui constituent ce modèle et elle initialise des variables de travail.

2.2. SCHEMA



2.3. DESCRIPTION

2.3.1. Entrées

- fichier de guidage, (f - ‹nom_du_modèle›);
- fichier interface, (fg -).

2.3.2. Sorties

- pas de sorties.

2.3.3. Traitements

- cfr.: schéma et code Pascal.

2.4. CODE PASCAL CORRESPONDANT

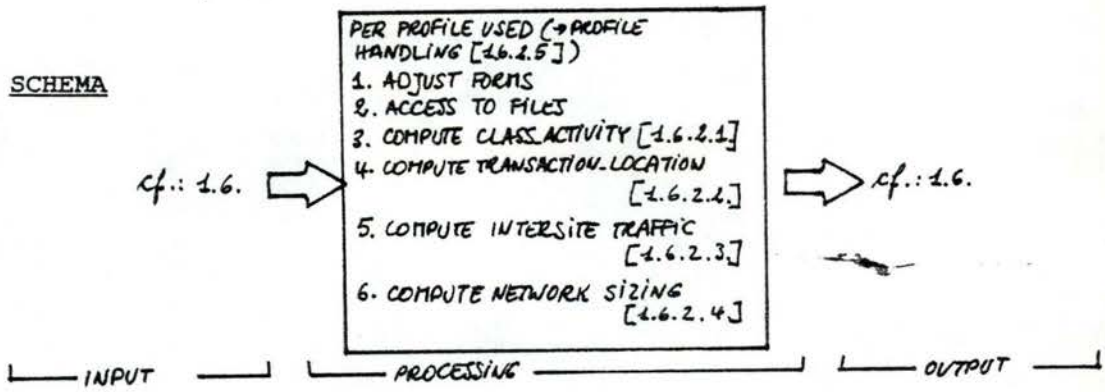
cfr.: programme COMPUTAT - (1.6.1.)

3. COMPUTING_SCHEDULE - NIVEAU D'ANALYSE 1.6.2.

3.1. SPECIFICATIONS

Comme son nom l'indique, cette partie assure l'ordonnancement des calculs à faire. Les calculs à effectuer par profil utilisé sont : évaluer l'activité des différentes classes de sites, compléter la localisation des transactions, évaluer le trafic intersite et effectuer les calculs de dimmensionnement proprement dits.

3.2. SCHEMA



3.3. DESCRIPTION

3.3.1. Entrées

- cfr.: niveau d'analyse 1.6.1. sauf fichiers interface et guide.

3.3.2. Sorties

- cfr.: niveau d'analyse 1.6.1.

3.3.3. Traitements

- cfr.: schéma et code Pascal.

3.4. CODE PASCAL CORRESPONDANT

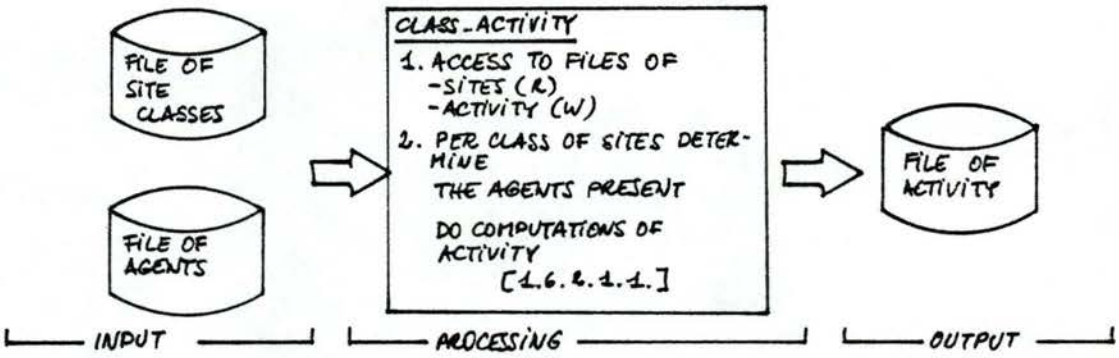
cfr.: programme COMPUTAT - (1.6.2.)

4. CLASS_ACTIVITY - NIVEAU D'ANALYSE 1.6.2.1.

4.1. SPECIFICATIONS

Le but de cette partie est d'évaluer l'activité de chaque classe de sites en nombre de transactions initialisées par heure. Les résultats sont donnés par type de transactions et sont stockés dans un fichier. Par site nous allons déterminer les agents présents. Cette information nous permet d'évaluer par agent et transaction le nombre moyen de transactions initialisées, si on fait la somme pour tous les agents pour un type de transactions on obtient l'activité de la classe pour ce type de transactions.

4.2. SCHEMA



4.3. DESCRIPTION

4.3.1. Entrèes

- fichier des sites, (fs - <nom_du_modèle>3?);
- fichier des agents, (fa - <nom_du_modèle>2).

4.3.2. Sorties

- fichier des activitès, (fileact -).

4.3.3. Traitements

- cfr.: schèma et code Pascal.

4.4. CODE PASCAL CORRESPONDANT

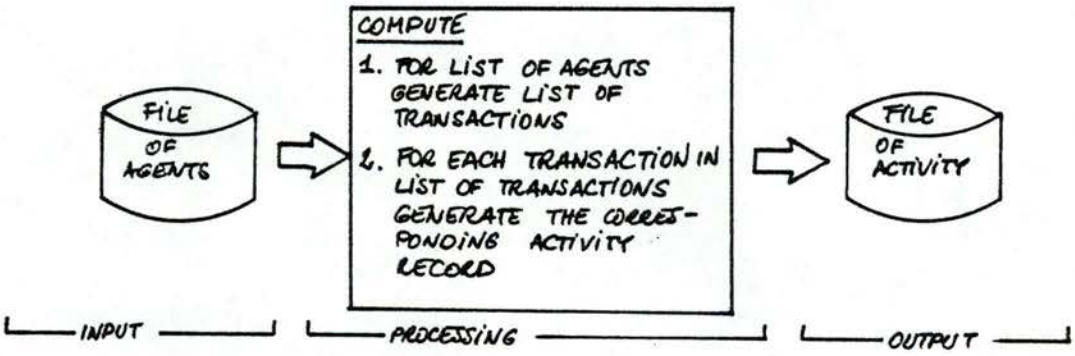
cfr.: programme COMPUTAT - (1.6.2.1.)

5. COMPUTE - NIVEAU D'ANALYSE 1.6.2.1.1.

5.1. SPECIFICATIONS

Pour chaque classe de site, déterminer la liste des agents présents et déterminer ainsi les transactions initialisées. Pour chaque type de transactions déterminer l'activité de la classe et mémoriser ce résultat dans un fichier (des activités). (Cfr.: sales aid).

5.2. SCHEMA



5.3. DESCRIPTION

5.3.1. Entrées

- fichier des agents, (fa - ‹nom_du_modèle›2).

5.3.2. Sorties

- fichier des activités, (fileact -).

5.3.3. Traitements

- cfr.: schéma et code Pascal.

5.4. CODE PASCAL CORRESPONDANT

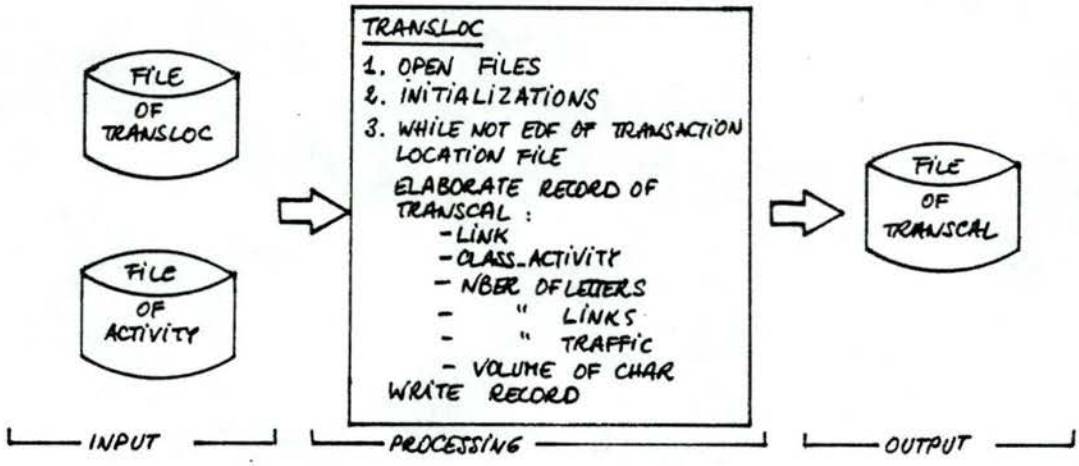
cfr.: programme COMPUTAT - (1.6.2.1.1.)

6. TRANS_LOC - NIVEAU D'ANALYSE 1.6.2.2.

6.1. SPECIFICATIONS

Le but de cette partie est de compléter les informations collectées lors de la localisation des transactions (cfr.: chapitre 7). Le fichier des localisations des transactions est lu séquentiellement et à l'aide du fichier des activités des classes cette partie confectionne un fichier parallèle à celui des localisations et contient les informations relatives au nombre de lettres, au nombre de fragments ou paquets (X25 privé ou public) par message, au nombre total de fragments ou paquets et au volume total de caractères.

6.2. SCHEMA



6.3. DESCRIPTION

6.3.1. Entrèes

- fichier des localisations des transactions, (fl - ‹nom_du_modèle›4?);
- fichier des activits, (fileact -).

6.3.2. Sorties

- fichier des calculs des transactions, (filecal -).

6.3.3. Traitements

- cfr.: schèma et code Pascal.

6.4. CODE PASCAL CORRESPONDANT

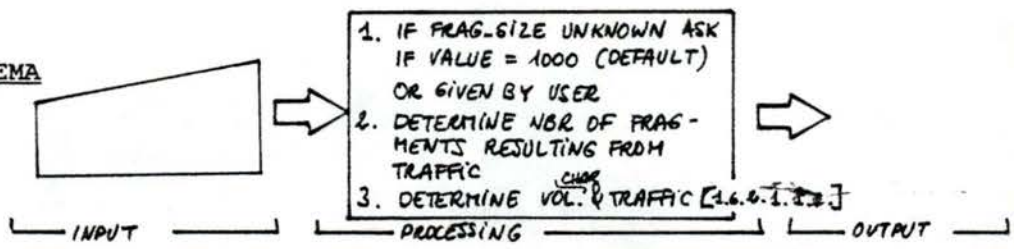
cfr.: programme COMPUTAT - (1.6.2.2.)

7. CALCUL O - NIVEAU D'ANALYSE 1.6.2.2.1.

7.1. SPECIFICATIONS

Cette procèdure a pour but d'èvaluer le nombre de fragments point-à-point nècessaires pour transmettre les messages entre les "senders" et "receivers". L'utilisateur choisit la longueur des fragments en caractères. En connaissant le nombre de lettres transitant de l'èmetteur vers le rècepteur on calcule le nombre de fragments ainsi que le volume de caractères.

7.2. SCHEMA



7.3. DESCRIPTION

7.3.1. Entrées

- entrées manuelles au terminal.

7.3.2. Sorties

- pas de sorties.

7.3.3. Traitements

- cfr.: schéma et code Pascal.

7.4. CODE PASCAL CORRESPONDANT

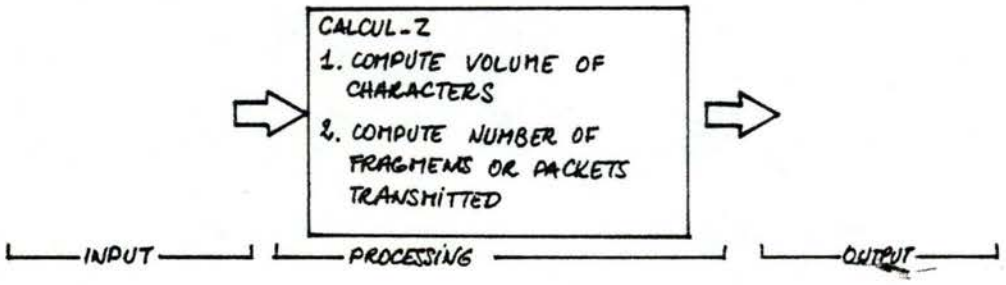
cfr.: programme COMPUTAT - (1.6.2.2.1.)

8. CALCUL_Z - NIVEAU D'ANALYSE 1.6.2.2.1.1.

8.1. SPECIFICATIONS

Cette partie a pour but d'évaluer le volume de caractères transmis ainsi que le nombre de paquets ou de fragments.

8.2. SCHEMA



8.3. DESCRIPTION

8.3.1. Entrèes

- pas d'entrèes.

8.3.2. Sorties

- pas de sorties.

8.3.3. Traitements

- cfr.: schèma et code Pascal.

8.4. CODE PASCAL CORRESPONDANT

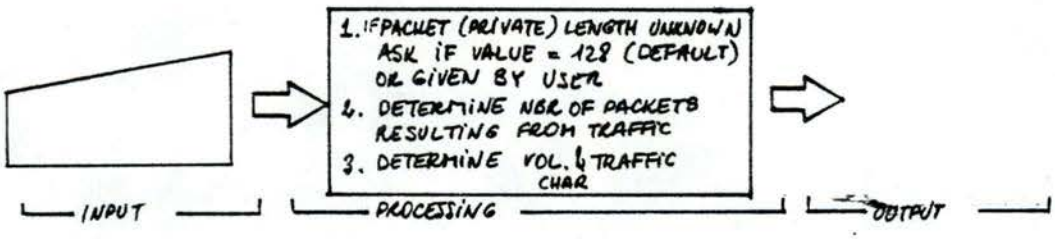
cfr.: programme COMPUTAT - (1.6.2.2.1.1.)

9. CALCUL_R - NIVEAU D'ANALYSE 1.6.2.2.2.

9.1. SPECIFICATIONS

Cette procèdure a pour but d'èvaluer le nombre de paquets X25 privè nècessaires pour transmettre les messages entre èmetteur et rècepteur. L'utilisateur choisit la longueur des paquets en caractères. En connaissant le nombre de lettres, on calcule le nombre de paquets ainsi que le volume de caractères.

9.2. SCHEMA



9.3. DESCRIPTION

9.3.1. Entrées

- entrées manuelles au terminal.

9.3.2. Sorties

- pas de sorties.

9.3.3. Traitements

- cfr.: schéma et code Pascal.

9.4. CODE PASCAL CORRESPONDANT

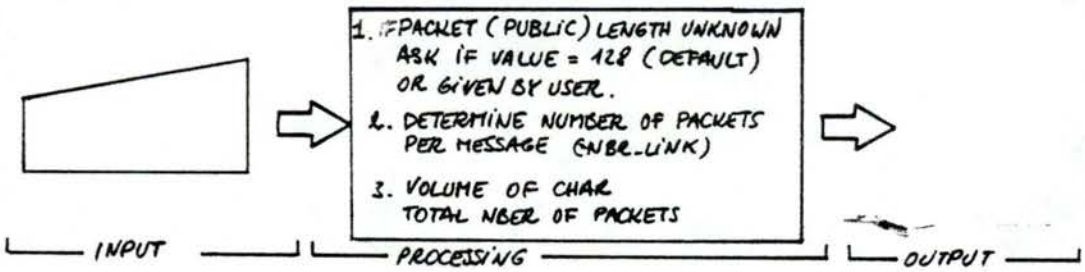
cfr.: programme COMPUTAT - (1.6.2.2.2.)

10. CALCUL_U - NIVEAU D'ANALYSE 1.6.2.2.3.

10.1. SPECIFICATIONS

Cette procédure a pour but d'évaluer le nombre de paquets X25 public nécessaires aux transmissions des messages entre émetteur et récepteur. L'utilisateur choisit la longueur des paquets. La partie évalue aussi le volume de caractères transmis.

10.2. SCHEMA



10.3. DESCRIPTION

10.3.1. Entrées

- entrées manuelles au terminal.

10.3.2. Sorties

- pas de sorties.

10.3.3. Traitements

- cfr.: schêma et code Pascal.

10.4. CODE PASCAL CORRESPONDANT

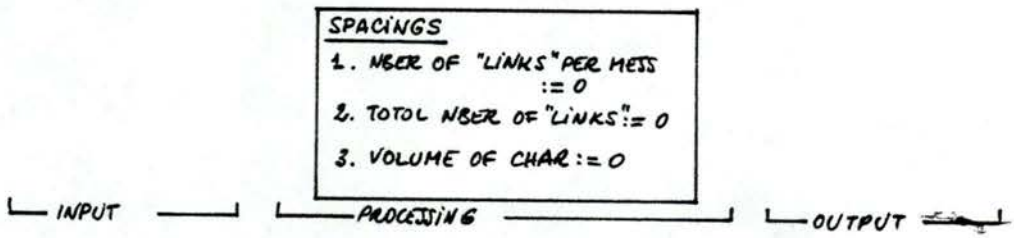
cfr.: programme COMPUTAT - (1.6.2.2.3.)

11. SPACINGS - NIVEAU D'ANALYSE 1.6.2.2.4.

11.1. SPECIFICATIONS

Cette partie met à zêro les valeurs suivantes :
- nombre de paquets ou de fragments par message
- nombre total de paquets ou de fragments
- volume de caractères transmis.

11.2. SCHEMA



11.3. DESCRIPTION

11.3.1. Entrèes

- pas d'entrèes.

11.3.2. Sorties

- pas de sorties.

11.3.3. Traitements

- cfr.: schèma et code Pascal.

11.4. CODE PASCAL CORRESPONDANT

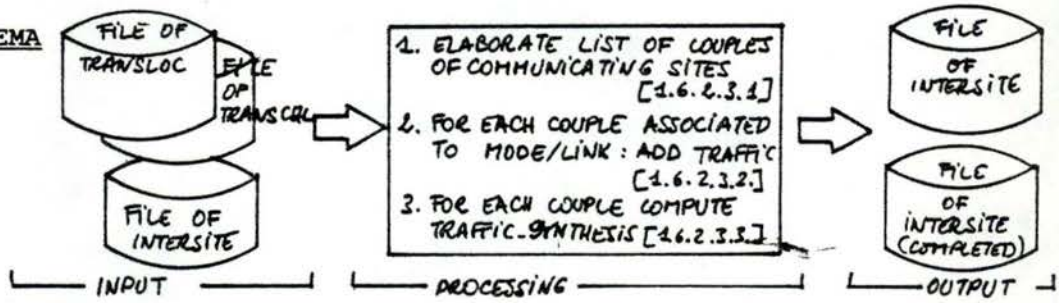
cfr.: programme COMPUTAT - (1.6.2.2.4.)

12. INTER_SITE - NIVEAU D'ANALYSE 1.6.2.3.

12.1. SPECIFICATIONS

Cette partie a pour but d'èvaluer le trafic entre classes de sites. Les rèsultats sont prèsentès par mode de communication et par nature du lien choisi entre deux classes de sites communiquant. Cette partie doit donc dèterminer toutes les paires de classes de sites communiquant pour un mode et un type de liens et dèterminer le trafic total pour chaque paire trouvèe.

12.2. SCHEMA



12.3. DESCRIPTION

12.3.1. Entrées

- fichier des localisations des transactions, (fl - <nom_du_modèle>4?);
- fichier des calculs relatifs au fichier des localisations, (filecal -);
- fichier intermédiaire concernant le trafic intersite confectionné dans cette partie, (fileint -).

12.3.2. Sorties

- fichier intermédiaire du trafic intersite, (fileint -);
- fichier complèt du trafic intersite, (fileint2 -).

12.3.3. Traitements

- cfr.: schèma et code Pascal.

12.4. CODE PASCAL CORRESPONDANT

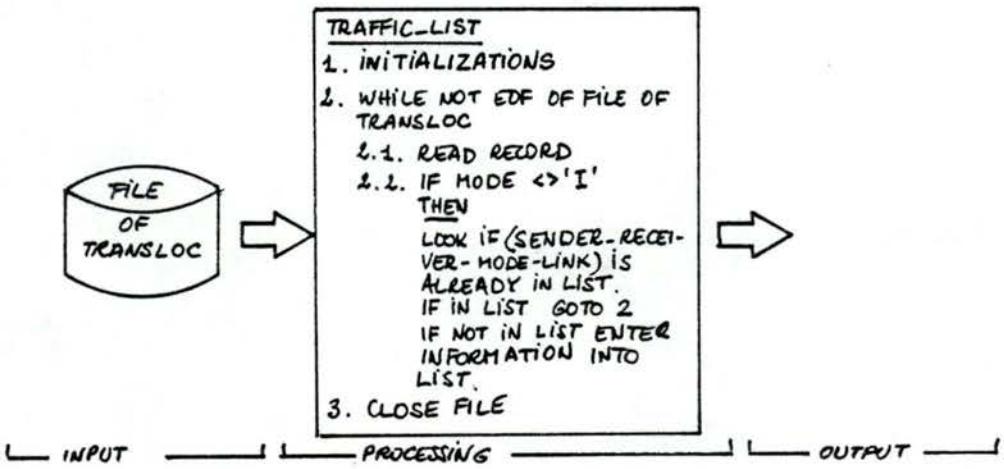
cfr.: programme COMPUTAT - (1.6.2.3.)

13. TRAFFIC_LIST - NIVEAU D'ANALYSE 1.6.2.3.1.

13.1. SPECIFICATIONS

Cette procédure lit le fichier de la localisation des transactions et détermine tous les couples de classes de sites (émetteur,récepteur) par nature du lien et par mode de communication. Ces informations sont mises dans une liste qui servira plus loin comme plan de recherche d'autres informations.

13.2. SCHEMA



13.3. DESCRIPTION

13.3.1. Entrèes

- fichier des localisations des transactions, (fl - <nom_du_modèle>4?) .

13.3.2. Sorties

- pas de sorties.

13.3.3. Traitements

- cfr.: schèma et code Pascal.

13.4. CODE PASCAL CORRESPONDANT

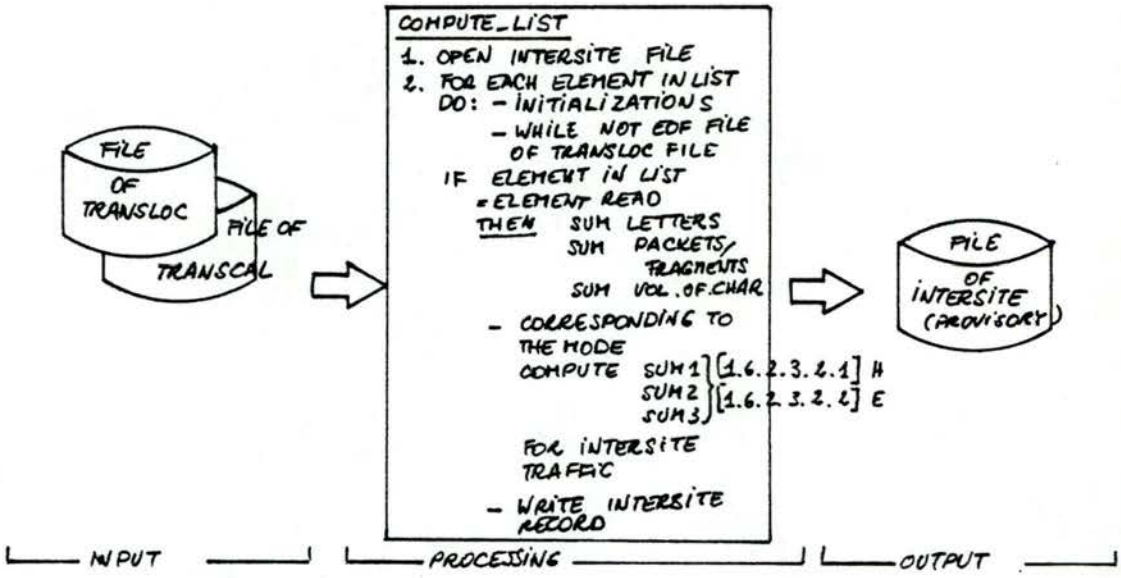
cfr.: programme COMPUTAT - (1.6.2.3.1.)

14. COMPUTE_LIST - NIVEAU D'ANALYSE 1.6.2.3.2.

14.1. SPECIFICATIONS

Cette procédure prend comme base des traitements à effectuer la liste confectionnée par la procédure "traffic_list". Pour chaque couple (émetteur,récepteur) de la liste on va faire la somme des messages, la somme du volume de caractères et la somme des paquets ou fragments transitant de l'émetteur vers le récepteur. Les informations ainsi élaborées vont être mises dans un fichier intermédiaire d'intersite.

14.2. SCHEMA



14.3. DESCRIPTION

14.3.1. Entreses

- fichier des localisations des transactions, (fl - <nom_du_modèle>4?);
- fichier parallèle des calculs, (filecal -).

14.3.2. Sorties

- fichier intersite, (fileint -).

14.3.3. Traitements

- cfr.: schèma et code Pascal.

14.4. CODE PASCAL CORRESPONDANT

cfr.: programme COMPUTAT - (1.6.2.3.2.)

15. CALCUL E - NIVEAU D'ANALYSE 1.6.2.3.2.1.

15.1. SPECIFICATIONS

cfr.: base thèorique : calcul du trafic intersite en partant du trafic interclasse.

15.2. DESCRIPTION

15.2.1. Entrèes

- pas d'entrèes.

15.2.2. Sorties

- pas de sorties.

15.2.3. Traitements

- cfr.: code Pascal.

15.3. CODE PASCAL CORRESPONDANT

cfr.: programme COMPUTAT - (1.6.2.3.2.1.)

16. CALCUL_H - NIVEAU D'ANALYSE 1.6.2.3.2.2.

16.1. SPECIFICATIONS

cfr.: base théorique : calcul du trafic intersite en partant du trafic interclasses.

16.2. DESCRIPTION

16.2.1. Entrées

- pas d'entrées.

16.2.2. Sorties

- pas de sorties.

16.2.3. Traitements

- cfr.: code Pascal.

16.3. CODE PASCAL CORRESPONDANT

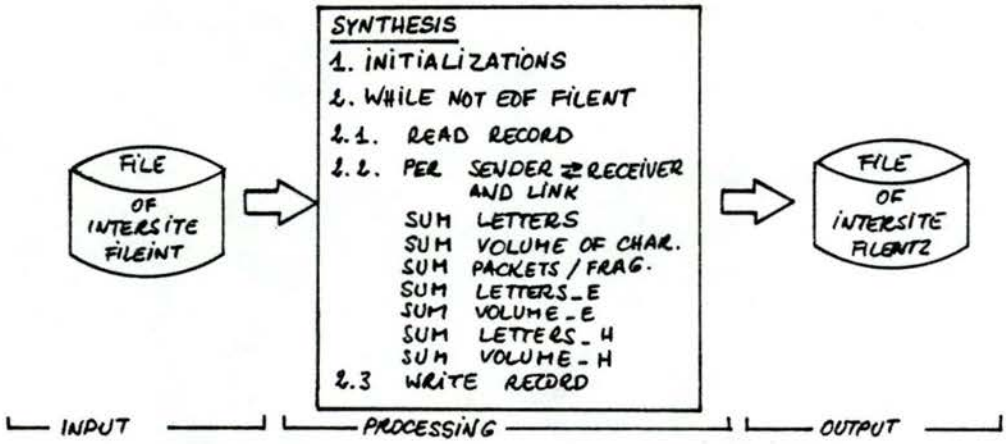
cfr.: programme COMPUTAT - (1.6.2.3.2.2.)

17. SYNTHESIS - NIVEAU D'ANALYSE 1.6.2.3.3.

17.1. SPECIFICATIONS

Cette procédure part des informations du type "intersites" pour faire la synthèse du trafic intersite . Le but principal est de sommer tous les messages, nombre de paquets ou fragments et caractères transitant entre deux sites sans distinguer le sens de ce transfert, ni le mode. Les informations ainsi élaborées sont mémorisées dans un fichier du type "intersite".

17.2. SCHEMA



17.3. DESCRIPTION

17.3.1. Entrées

- fichier intermédiaire intersite, (fileint -).

17.3.2. Sorties

- fichier final intersite, (fileint2 -).

17.3.3. Traitements

- cfr.: schèma et code Pascal.

17.4. CODE PASCAL CORRESPONDANT

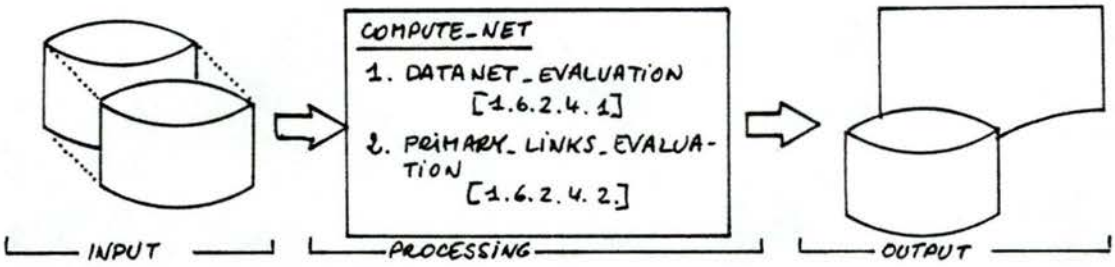
cfr.: programme COMPUTAT - (1.6.2.3.3.)

18. COMPUTE_NET - NIVEAU D'ANALYSE 1.6.2.4.

18.1. SPECIFICATIONS

Cette partie assure deux types de calculs différents. Les calculs sont relatifs au dimensionnement de rèseau. Le premier type de calculs concerne l'évaluation des datanets, le deuxième type concerne le dimensionnment des liens du rèseau primaire.

18.2. SCHEMA



18.3. DESCRIPTION

18.3.1. Entrées

- fichier des sites, (fs - <nom_du_modèle>3?);
- fichier intersite, (fileint2 -);
- fichier des datanets, (filedn -);
- réponses en provenance du terminal.

18.3.2. Sorties

- fichier des datanets;
- résultats affichés au terminal.

18.3.3. Traitements

- cfr.: schéma et code Pascal.

18.4. CODE PASCAL CORRESPONDANT

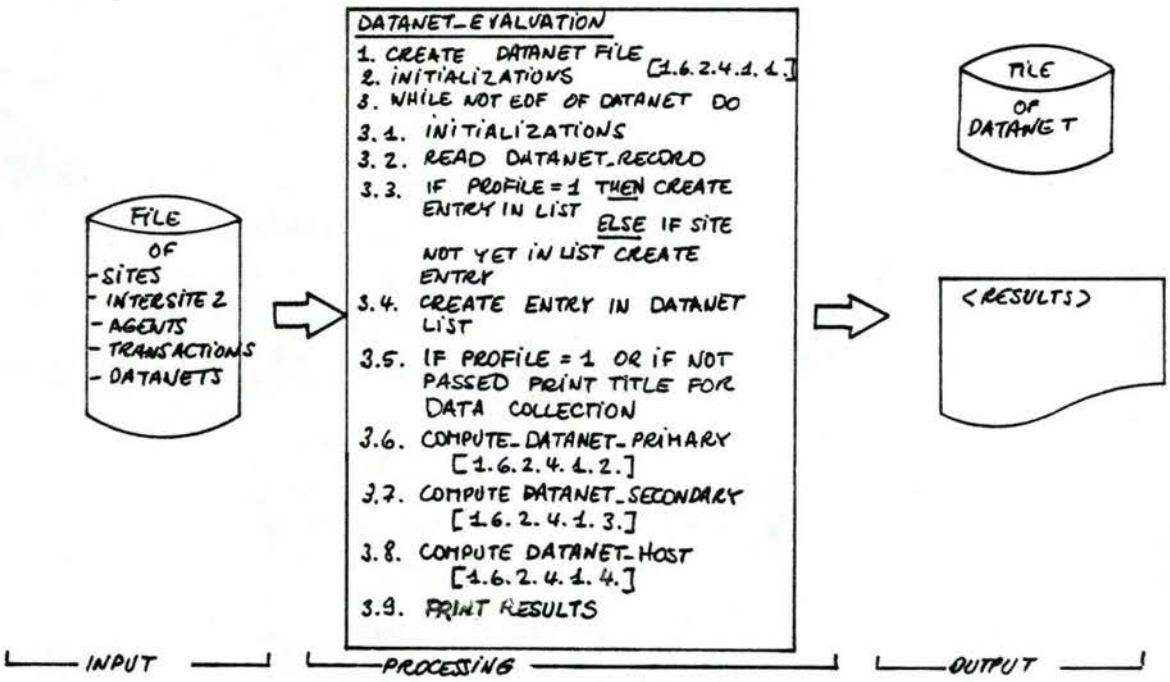
cfr.: programme COMPUTAT - (1.6.2.4.)

19. DATANET_EVALUATION - NIVEAU D'ANALYSE 1.6.2.4.1.

19.1. SPECIFICATIONS

Cette partie a pour but d'évaluer la charge CPU des datanets du réseau. La première tâche de la procédure est de créer un fichier reprenant tous les sites où il y a un ou plusieurs datanets. En partant de ce fichier la procédure crée une liste de ces sites où chaque élément de la liste mémorisera les informations fournies par l'utilisateur pour qu'il ne soit plus obligé de les entrer si plusieurs profils sont utilisés. Donc par site lu dans le fichier, on le met dans la liste, on évalue la charge du datanet situé à ce site pour le réseau primaire, pour le réseau secondaire et pour la connexion du datanet à l'ordinateur hôte . Après le traitement, la procédure déclenche l'impression des résultats.

19.2. SCHEMA



19.3. DESCRIPTION

19.3.1. Entrèes

- fichier des transactions, (ft - ‹nom_du_modèle›1);
- fichier des agents, (fa - ‹nom_du_modèle›2);
- fichier des sites, (fs - ‹nom_du_modèle›3?);
- fichier des datanets (crèes dans cette partie), (filedn -);
- fichier intersite, (fileint2 -).

19.3.2. Sorties

- fichier des datanets;
- questions, messages et rèsultats affichès au terminal.

19.3.3. Traitements

- cfr.: schèma et code Pascal.

19.4. CODE PASCAL CORRESPONDANT

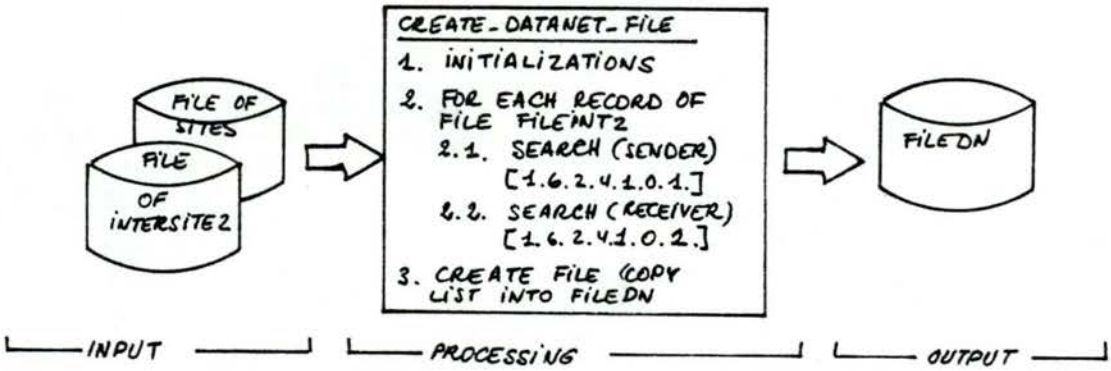
cfr.: programme COMPUTAT - (1.6.2.4.1.)

20. CREATE_DATANET-FILE - NIVEAU D'ANALYSE 1.6.2.4.1.0.

20.1. SPECIFICATIONS

Le rôle de cette procédure est de créer le fichier reprenant toutes les classes de sites où un datanet est localisé. L'information de base est contenue dans le fichier intersite intermédiaire et un fichier des datanets sera créé.

20.2. SCHEMA



20.3. DESCRIPTION

20.3.1. Entrées

- fichier intersite intermédiaire, (fileint -);
- fichier des sites, (fs - ‹nom_du_modèle›3?).

20.3.2. Sorties

- fichier des datanets, (filedn -).

20.3.3. Traitements

- cfr.: schéma et code Pascal.

20.4. CODE PASCAL CORRESPONDANT

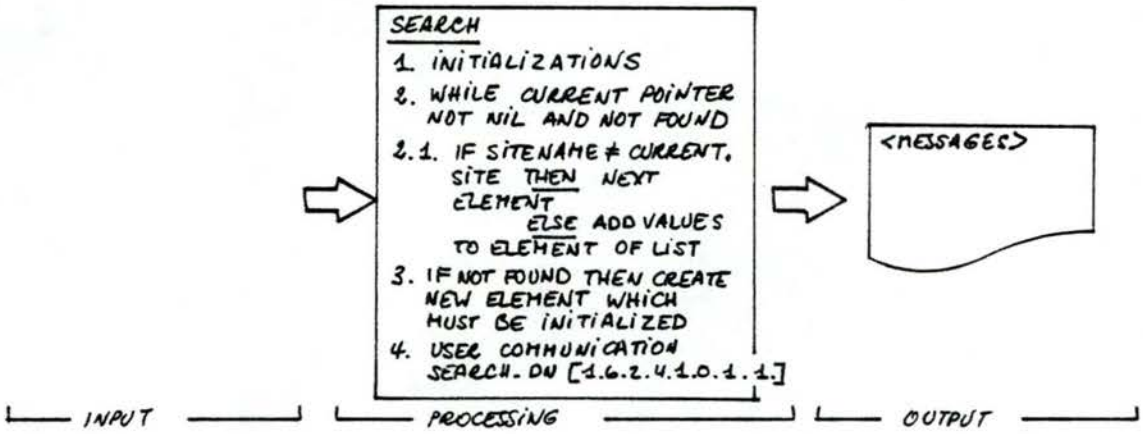
cfr.: programme COMPUTAT - (1.6.2.4.1.0.)

21. SEARCH - NIVEAU D'ANALYSE 1.6.2.4.1.0.1.

21.1. SPECIFICATIONS

A partir du nom d'une classe de sites on va voir si un datanet se trouve au niveau des sites de cette classe. Si oui, la classe de sites est retenue , sinon le datanet est supposé être présent. Il y a création d'une liste dont chaque élément reprend les nombres de paquets X25 privé et public et le nombre de trames.

21.2. SCHEMA



21.3. DESCRIPTION

21.3.1. Entrées

- pas d'entrées.

21.3.2. Sorties

- messages èventuels affichès au terminal.

21.3.3. Traitements

- cfr.: schèma et code Pascal.

21.4. CODE PASCAL CORRESPONDANT

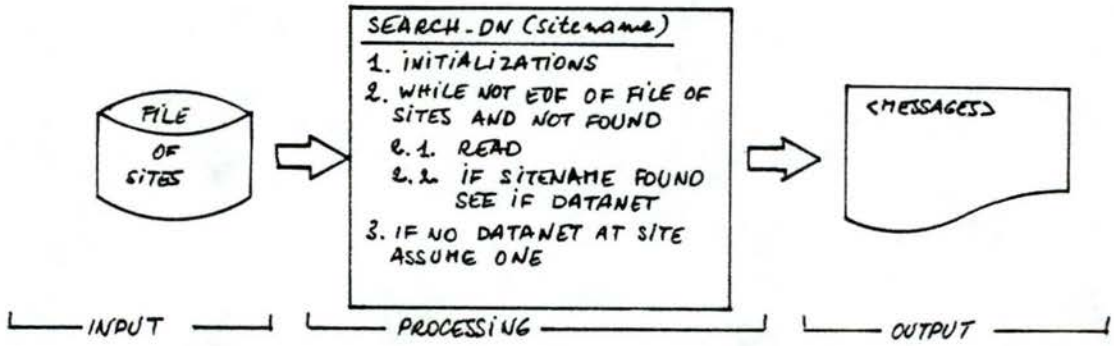
cfr.: programme COMPUTAT - (1.6.2.4.1.0.1.)

22. SEARCH_DN - NIVEAU D'ANALYSE 1.6.2.4.1.0.1.1.

22.1. SPECIFICATIONS

Comme on ne peut effectuer des calculs que pour les datanets, le programme assure à chaque site la prèsence d'un datanet et le signale à l'utilisateur.

22.2. SCHEMA



22.3. DESCRIPTION

22.3.1. Entrèes

- fichier des sites, (fs - <nom_du_modèle>3?).

22.3.2. Sorties

- messages affichès au terminal.

22.3.3. Traitements

- cfr.: schèma et code Pascal.

22.4. CODE PASCAL CORRESPONDANT

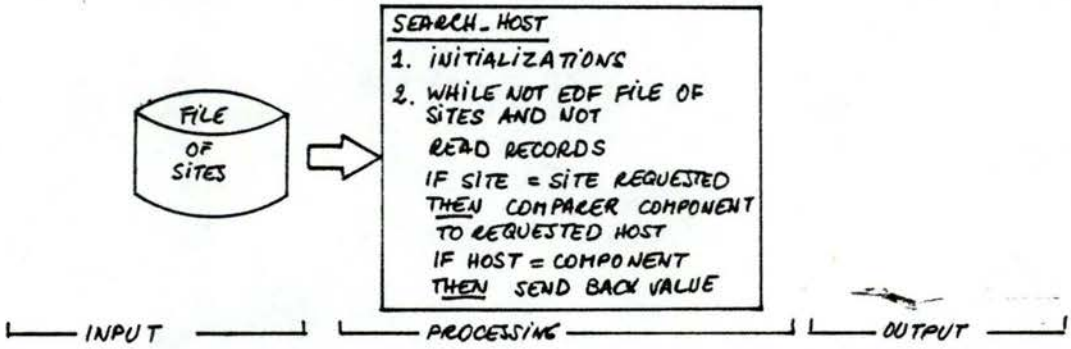
cfr.: programme COMPUTAT - (1.6.2.4.1.0.1.1.)

23. SEARCH_HOST - NIVEAU D'ANALYSE 1.6.2.4.1.1.

23.1. SPECIFICATIONS

Cette procèdure vèrifie si un composant du type "hôte" est prèsent à un certain site.

23.2. SCHEMA



23.3. DESCRIPTION

23.3.1. Entrées

- fichier des sites, (fs - ‹nom_du_modèle›3?).

23.3.2. Sorties

- pas de sorties.

23.3.3. Traitements

- cfr.: schèma et code Pascal.

23.4. CODE PASCAL CORRESPONDANT

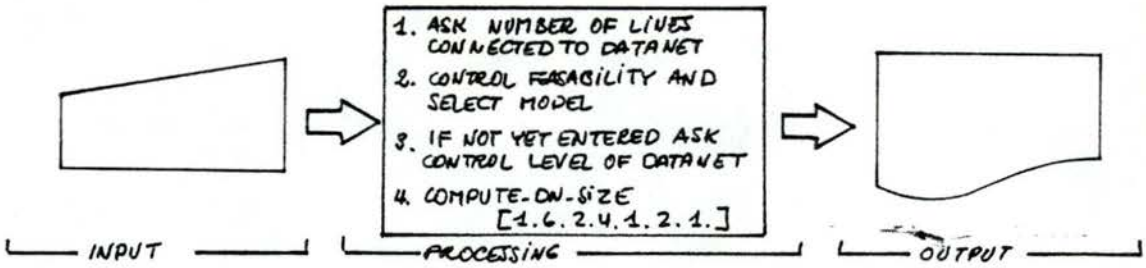
cfr.: programme COMPUTAT - (1.6.2.4.1.1.)

24. DATANET_PRIMARY - NIVEAU D'ANALYSE 1.6.2.4.1.2.

24.1. SPECIFICATIONS

Cette procèdure èvalue la charge CPU du datanet rèsultant des transmissions se faisant à travers le rèseau primaire. Cette procèdure collecte l'information relative aux services fournis par le datanet.

24.2. SCHEMA



24.3. DESCRIPTION

24.3.1. Entrées

- réponses en provenance du terminal.

24.3.2. Sorties

- questions affichées au terminal.

24.3.3. Traitements

- cfr.: schéma et code Pascal.

24.4. CODE PASCAL CORRESPONDANT

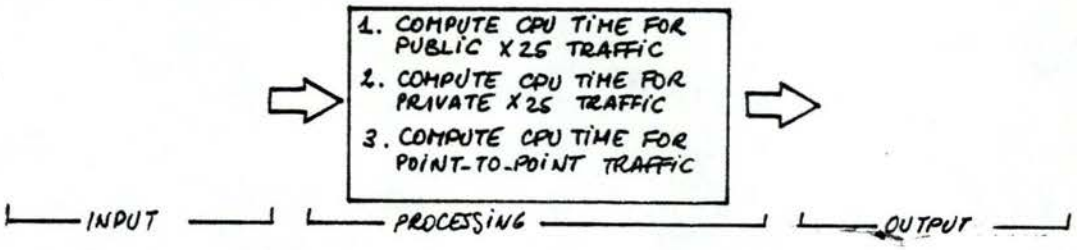
cfr.: programme COMPUTAT - (1.6.2.4.1.2.)

25. COMPUTE_DN_SIZE - NIVEAU D'ANALYSE 1.6.2.4.1.2.1.

25.1. SPECIFICATIONS

Cette procédure évalue la charge CPU du datanet résultant des transmissions se faisant à travers le réseau primaire, c'est-à-dire la charge CPU résultant du traitement des paquets X25 public ou privé et / ou des fragments.

25.2. SCHEMA



25.3. DESCRIPTION

25.3.1. Entrèes

- pas d'entrèes.

25.3.2. Sorties

- pas de sorties.

25.3.3. Traitements

- cfr.: schèma et code Pascal.

25.4. CODE PASCAL CORRESPONDANT

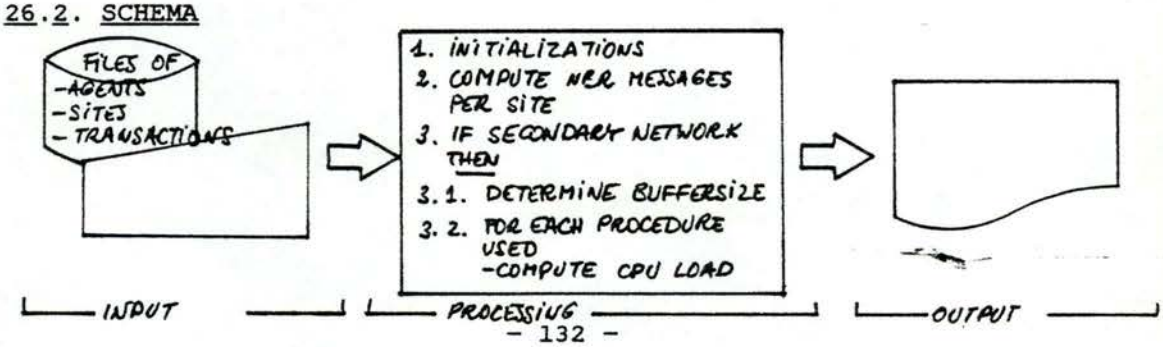
cfr.: programme COMPUTAT - (1.6.2.4.1.2.1.)

26. DATANET SECONDARY - NIVEAU D'ANALYSE 1.6.2.4.1.3.

26.1. SPECIFICATIONS

Cette procèdure èvalue la charge CPU du datanet rèsultant du traitement des transactions initialisèes dans le rèseau secondaire. Selon le type des protocoles utilisès elle èvalue cette charge. Il y a un contrôle de dèpassement de la capacitè du CPU.

26.2. SCHEMA



26.3. DESCRIPTION

26.3.1. Entrées

- fichier des transactions, (ft - ‹nom_du_modèle›1);
- fichier des agents, (fa - ‹nom_du_modèle›2);
- fichier des sites, (fs - ‹nom_du_modèle›3?);
- réponses en provenance du terminal.

26.3.2. Sorties

- messages affichés au terminal.

26.3.3. Traitements

- cfr.: schéma et code Pascal.

26.4. CODE PASCAL CORRESPONDANT

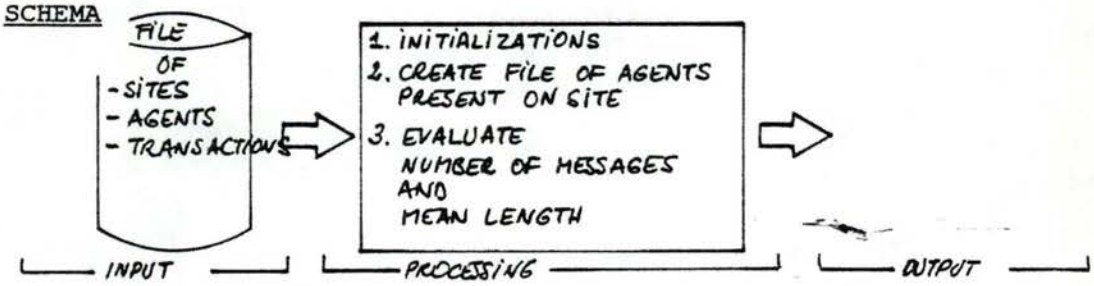
cfr.: prgramme COMPUTAT - (1.6.2.4.1.3.)

27. CALCUL_NBR_MESS - NIVEAU D'ANALYSE 1.6.2.4.1.3.1.

27.1. SPECIFICATIONS

Cette procédure évalue pour un site donné le nombre de messages initialisés dans son réseau secondaire et la longueur moyenne de ces messages.

27.2. SCHEMA



27.3. DESCRIPTION

27.3.1. Entrées

- fichier des transactions, (ft - <nom_du_modèle>1);
- fichier des agents, (fa - <nom_du_modèle>2);
- fichier des sites, (fs - <nom_du_modèle>3?).

27.3.2. Sorties

- pas de sorties.

27.3.3. Traitements

- cfr.: schèma et code Pascal.

27.4. CODE PASCAL CORRESPONDANT

cfr.: programme COMPUTAT - (1.6.2.4.1.3.1.)

28. COMP_ASY - NIVEAU D'ANALYSE 1.6.2.4.1.3.2.

28.1. SPECIFICATIONS

Cette procédure évalue la charge CPU du datanet résultant des transmissions utilisant un protocole asynchrone. (Cfr.: sales aid).

28.2. DESCRIPTION

28.2.1. Entrées

- réponse en provenance du terminal.

28.2.2. Sorties

- question affichée au terminal.

28.2.3. Traitements

- cfr.: code Pascal.

28.3. CODE PASCAL CORRESPONDANT

cfr.: programme COMPUTAT - (1.6.2.4.1.3.2.)

29. COMP_VIP - NIVEAU D'ANALYSE 1.6.2.4.1.3.3;

29.1. SPECIFICATIONS

Cette procédure évalue la charge CPU du datanet résultant des transmissions utilisant le protocole VIP. (Cfr.: sales aid).

29.2. DESCRIPTION

29.2.1. Entrées

- pas d'entrées.

29.2.2. Sorties

- pas de sorties.

29.2.3. Traitements

- cfr.: code Pascal.

29.3. CODE PASCAL CORRESPONDANT

cfr.: programme COMPUTAT - (1.6.2.4.1.3.3.)

30. COMP_BSC2 - NIVEAU D'ANALYSE 1.6.2.4.1.3.4.

30.1. SPECIFICATIONS

Cette procédure évalue la charge CPU du datanet résultant des transmissions utilisant le protocole BSC 3270. (Cfr.: sales aid).

30.2. DESCRIPTION

30.2.1. Entrées

- pas d'entrées.

30.2.2. Sorties

- pas de sorties.

30.2.3. Traitements

- cfr.: code Pascal.

30.3. CODE PASCAL CORRESPONDANT

cfr.: programme COMPUTAT - (1.6.2.4.1.3.4.)

31. COMP_PAD - NIVEAU D'ANALYSE 1.6.2.4.1.3.5.

31.1. SPECIFICATIONS

Cette procédure évalue la charge CPU du datanet résultant des transmissions utilisant le protocole PAD (packet assembly/deassembly). (Cfr.: sales aid).

31.2. DESCRIPTION

31.2.1. Entrées

- pas d'entrées.

31.2.2. Sorties

- pas de sorties

31.2.3. Traitements

- cfr.: code Pascal.

31.3. CODE PASCAL CORRESPONDANT

cfr.: programme COMPUTAT - (1.6.2.4.1.3.5.)

32. COMP_DCU - NIVEAU D'ANALYSE 1.6.2.4.1.3.6.

32.1. SPECIFICATIONS

Cette procédure évalue la charge CPU du datanet résultant des transmssions utilisant le protocole DCU. (Cfr.: sales aid).

32.2. DESCRIPTION

32.2.1. Entrées

- pas d'entrées.

32.2.2. Sorties

- pas de sorties.

32.2.3. Traitements

- cfr.: code Pascal.

32.3. CODE PASCAL CORRESPONDANT

cfr.: programme COMPUTAT - (1.6.2.4.1.3.6.)

33. COMP_RCI - NIVEAU D'ANALYSE 1.6.2.4.1.3.7.

33.1. SPECIFICATIONS

Cette prcédure évalue la charge CPU du datanet résultant des transmissions utilisant le protocole RCI. (Cfr.: sales aid).

33.2. DESCRIPTION

33.2.1. Entrées

- rèponses en provenance du terminal.

33.2.2. Sorties

- question affichèe au terminal.

33.2.3. Traitements

- cfr.: code Pascal.

33.3. CODE PASCAL CORRESPONDANT

cfr.: programme COMPUTAT - (1.6.2.4.1.3.7.)

34. COMP_BSC1 - NIVEAU D'ANALYSE 1.6.2.4.1.3.8.

34.1. SPECIFICATIONS

Cette procèdure èvalue la charge CPU du datanet rèsultant des transmissions utilisant le protocole BSC 2780 . (Cfr.: sales aid).

34.2. DESCRIPTION

34.2.1. Entrèes

- rèponses en provenance du terminal.

34.2.2. Sorties

- question affichée au terminal.

34.2.3. Traitements

- cfr.: code Pascal.

34.3. CODE PASCAL CORRESPONDANT

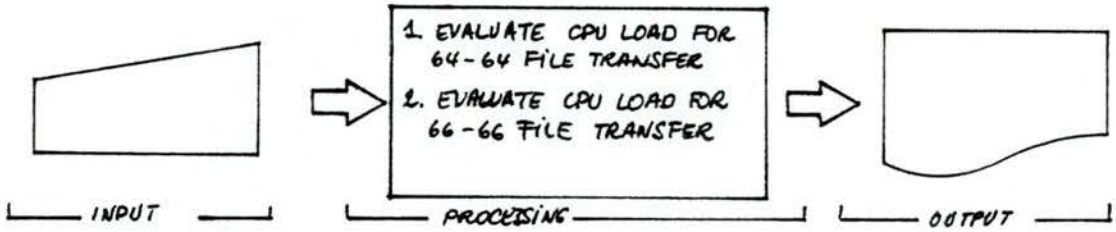
cfr.: programme COMPUTAT - (1.6.2.4.1.3.8.)

35. COMP_TRANSF - NIVEAU D'ANALYSE 1.6.2.4.1.3.9.

35.1. SPECIFICATIONS

Cette procédure assure l'évaluation de la charge CPU du datanet résultant de transferts de fichiers. (Cfr.: sales aid).

35.2. SCHEMA



35.3. DESCRIPTION

35.3.1. Entrées

- réponses en provenace du terminal.

35.3.2. Sorties

- questions affichées au terminal.

35.3.3. Traitements

- cfr.: schéma et code Pascal.

35.4. CODE PASCAL CORRESPONDANT

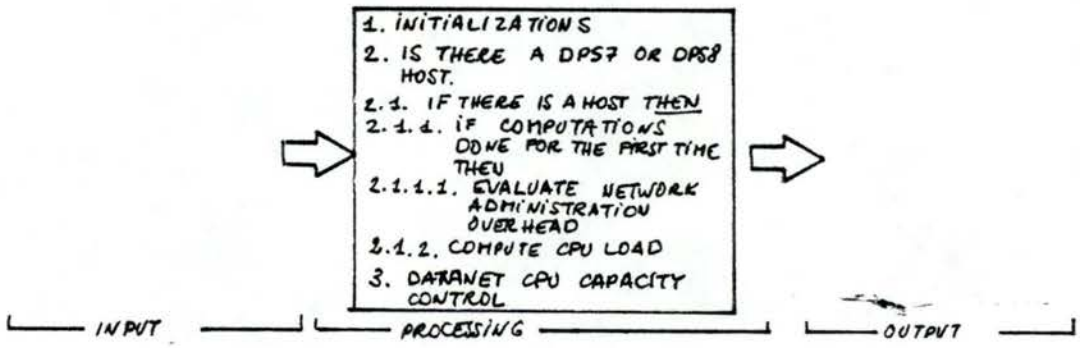
cfr.: programme COMPUTAT - (1.6.2.4.1.3.9.)

36. DATANET_HOST - NIVEAU D'ANALYSE 1.6.2.4.1.4.

36.1. SPECIFICATIONS

Cette procédure assure l'évaluation de la charge CPU du datanet résultant de la connexion de l'ordinateur hôte au datanet. Après avoir fait les initialisations, la procédure vérifie s'il y a un datanet au site. S'il y en a, elle règle d'abord la question des enregistrements administratifs et ensuite elle évalue la charge CPU pour le trafic global. Ceci se fait avec un contrôle de dépassement de la capacité du datanet.

36.2. SCHEMA



36.3. DESCRIPTION

36.3.1. Entrées

- pas d'entrées.

36.3.2. Sorties

- pas de sorties.

36.3.3. Traitements

- cfr.: schéma et code Pascal.

36.4. CODE PASCAL CORRESPONDANT

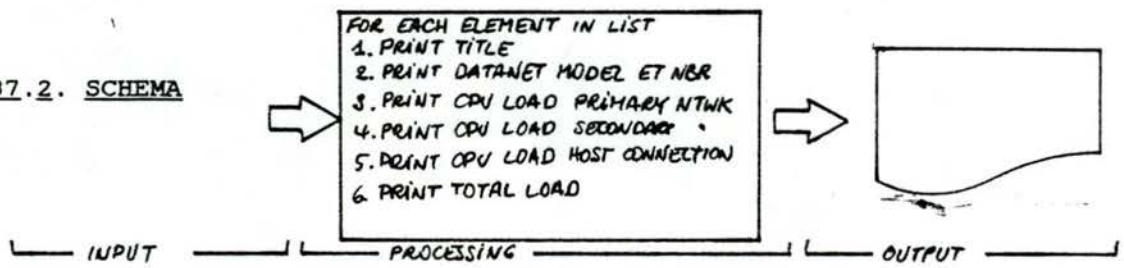
cfr.: programme COMPUTAT - (1.6.2.4.1.4.)

37. PRINT_RESULTS - NIVEAU D'ANALYSE 1.6.2.4.1.5.

37.1. SPECIFICATIONS

Cette procédure imprime pour chaque site la charge CPU du datanet. Elle donne le type (modèle) du datanet à utiliser ainsi que le nombre de datanets de ce type. Si le taux d'utilisation est > 80 %, elle procède à une mise à jour du nombre de datanets et des résultats obtenus. Après ce contrôle la procédure imprime le modèle du datanet, le nombre de datanets, la charge CPU "réseau primaire", la charge CPU "réseau secondaire", la charge CPU "connexion datanet-hôte" et finalement la charge totale du CPU.

37.2. SCHEMA



37.3. DESCRIPTION

37.3.1. Entrées

- pas d'entrées.

37.3.2. Sorties

- sorties au terminal.

37.3.3. Traitements

- cfr.: schéma et code Pascal.

37.4. CODE PASCAL CORRESPONDANT

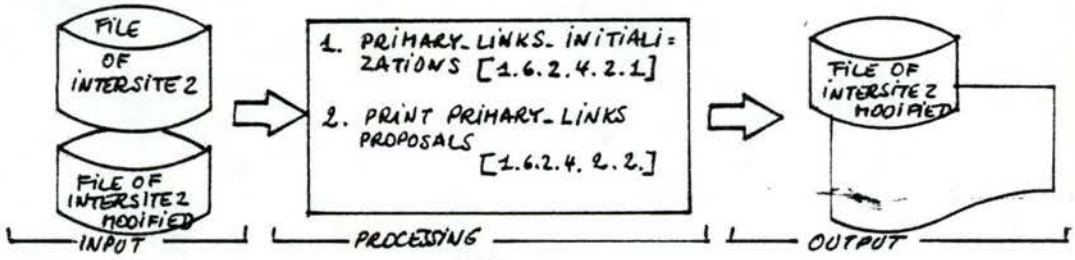
cfr.: programme COMPUTAT - (1.6.2.4.1.5.)

38. PRIMARY_LINKS_EVALUATION - NIVEAU D'ANALYSE 1.6.2.4.2.

38.1. SPECIFICATIONS

Cette partie est responsable du dimensionnement des liens du réseau primaire. La procédure crée d'abord le tableau des lignes présenté dans le sales aid et modifie ensuite le fichier intersite. En se basant sur ces informations, la procédure propose les lignes pour les liens retenuus. Les résultats sont ramenés au niveau de sites et non de classes de sites.

38.2. SCHEMA



38.3. DESCRIPTION

38.3.1. Entrèes

- fichier intersite, (fileint2 -);
- fichier intersite (fileint2 -) modifiè.

38.3.2. Sorties

- fichier intersite, (fileint2 -);
- rèsultats affichès au terminal.

38.3.3. Traitements

- cfr.: schèma et code Pascal.

38.4. CODE PASCAL CORRESPONDANT

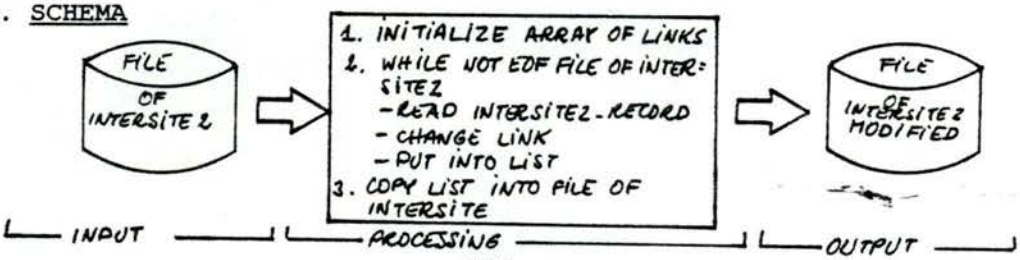
cfr.: programme COMPUTAT - (1.6.2.4.2.)

39. PRIMARY LINKS INITIALIZATIONS - NIVEAU D'ANALYSE 1.6.2.4.2.1.

39.1. SPECIFICATIONS

Le but de cette procèdure est de charger le tableau des lignes en mèmoire. La procèdure lit ensuite le fichier intersite et change la nature du lien, c'est-à-dire on ne diffèrencie plus que deux types de liens : lien X25 public et lien X25 privè ou lien point-à-point.

39.2. SCHEMA



39.3. DESCRIPTION

39.3.1. Entrèes

- fichier intersite, (fileint2 -).

39.3.2. Sorties

- fichier intersite (fileint2 -) modifiè.

39.3.3. Traitements

- cfr.: schèma et code Pascal.

39.4. CODE PASCAL CORRESPONDANT

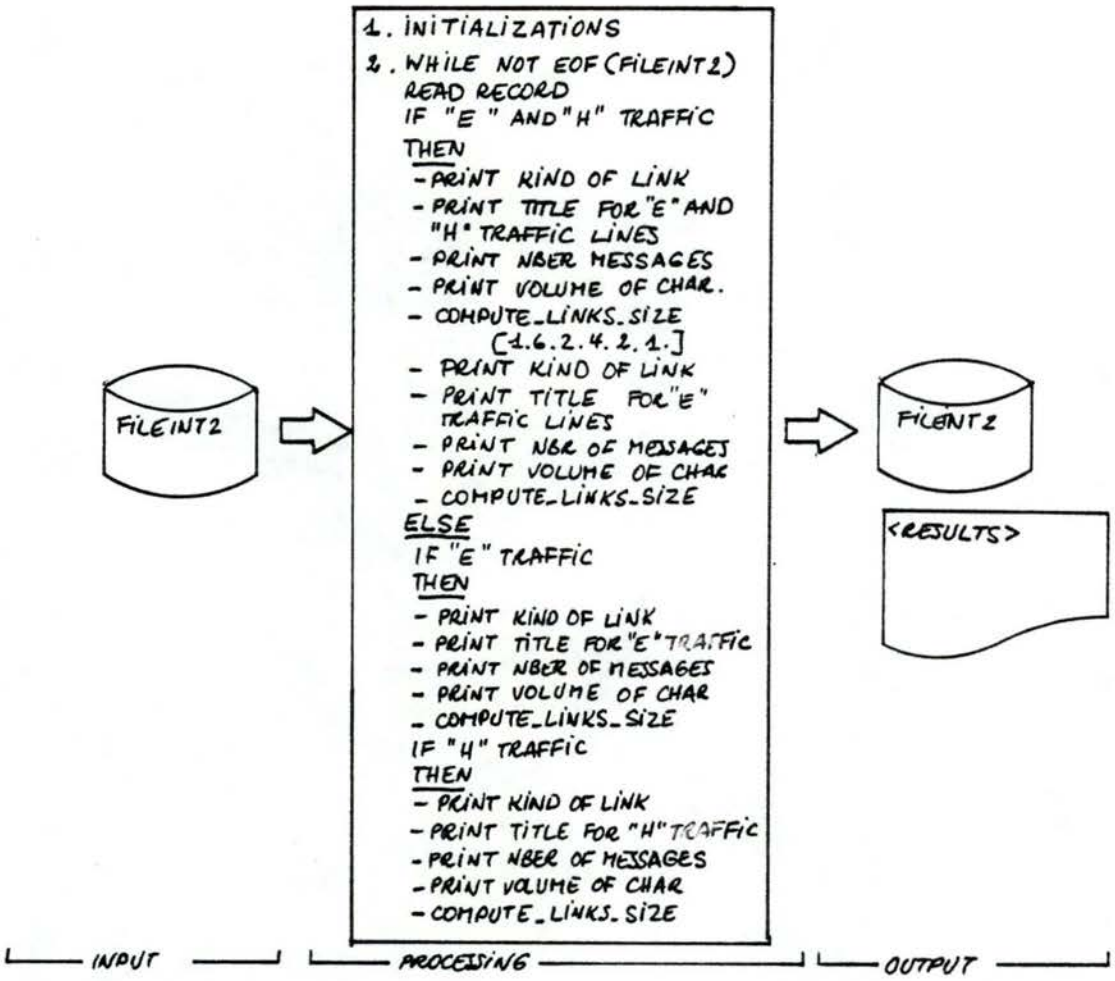
cfr.: programme COMPUTAT - (1.6.2.4.2.1.)

40. PRINT_PRIMARY_LINKS - NIVEAU D'ANALYSE 1.6.2.4.2.2.

40.1. SPECIFICATIONS

Cette procèdure a pour but d'imprimer les propositions de lignes pour les liens retenus. Les lignes sont prèsentèes pour le lien X25 public et pour les liens X25 privè ou point-à-point. Pour chaque lien on propose plusieurs vitesses diffèrentes (si possibles).

40.2. SCHEMA


1. INITIALIZATIONS
2. WHILE NOT EOF (FILEINT2)
READ RECORD
IF "E" AND "H" TRAFFIC
THEN
- PRINT KIND OF LINK
- PRINT TITLE FOR "E" AND "H" TRAFFIC LINES
- PRINT NBER MESSAGES
- PRINT VOLUME OF CHAR.
- COMPUTE_LINKS_SIZE [1.6.2.4.2.1.]
- PRINT KIND OF LINK
- PRINT TITLE FOR "E" TRAFFIC LINES
- PRINT NBR OF MESSAGES
- PRINT VOLUME OF CHAR
- COMPUTE_LINKS_SIZE
ELSE
IF "E" TRAFFIC
THEN
- PRINT KIND OF LINK
- PRINT TITLE FOR "E" TRAFFIC
- PRINT NBER OF MESSAGES
- PRINT VOLUME OF CHAR
- COMPUTE_LINKS_SIZE
IF "H" TRAFFIC
THEN
- PRINT KIND OF LINK
- PRINT TITLE FOR "H" TRAFFIC
- PRINT NBER OF MESSAGES
- PRINT VOLUME OF CHAR
- COMPUTE_LINKS_SIZE
FILEINT2
FILEINT2
<RESULTS>
INPUT
PROCESSING
OUTPUT

40.3. DESCRIPTION

40.3.1. Entrèes

- fichier intersite, (fileint2 -).

40.3.2. Sorties

- fichier intersite (fileint2 -) modifiè.

40.3.3. Traitements

- cfr.: schèma et code Pascal.

40.4. CODE PASCAL CORRESPONDANT

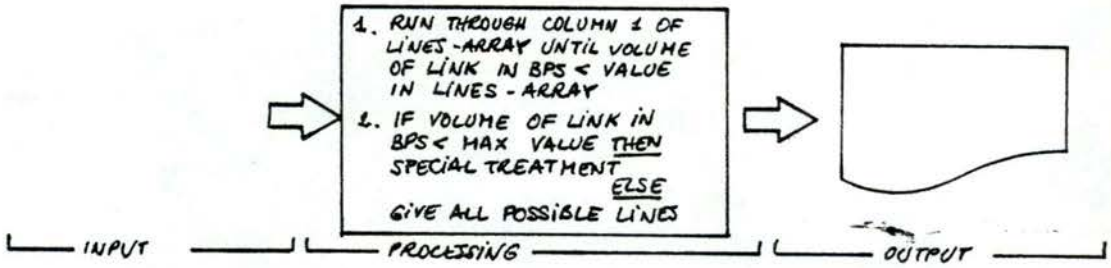
cfr.: programme COMPUTAT - (1.6.2.4.2.2.)

41. COMPUTE_LINKS_SIZE - NIVEAU D'ANALYSE 1.6.2.4.2.2.1.

41.1. SPECIFICATIONS

En connaissant le volume de caractères à transporter par heure, cette procèdure dimensionne le lien physique requis. C'est-à-dire elle propose une ou plusieurs capacitès de ligne avec le taux d'utilisation.

41.2. SCHEMA



41.3. DESCRIPTION

41.3.1. Entrées

- pas d'entrées.

41.3.2. Sorties

- résultats affichés au terminal.

41.3.3. Traitements

- cfr.: schéma et code Pascal.

41.4. CODE PASCAL CORRESPONDANT

cfr.: programme COMPUTAT - (1.6.2.4.2.2.1.)

42. PROFILE_HANDLING - NIVEAU D'ANALYSE 1.6.2.5.

42.1. SPECIFICATIONS

Cette procédure permet d'éviter les 4 calculs suivants : activité des classes, compléments de la localisation des transactions, trafic intersite et calculs de dimensionnement si les profils 2 et / ou 3 sont utilisés.

42.2. DESCRIPTION

- cfr.: niveau d'analyse 1.6.2.

42.3. CODE PASCAL CORRESPONDANT

cfr.: programme COMPUTAT - (1.6.2.4.5.)

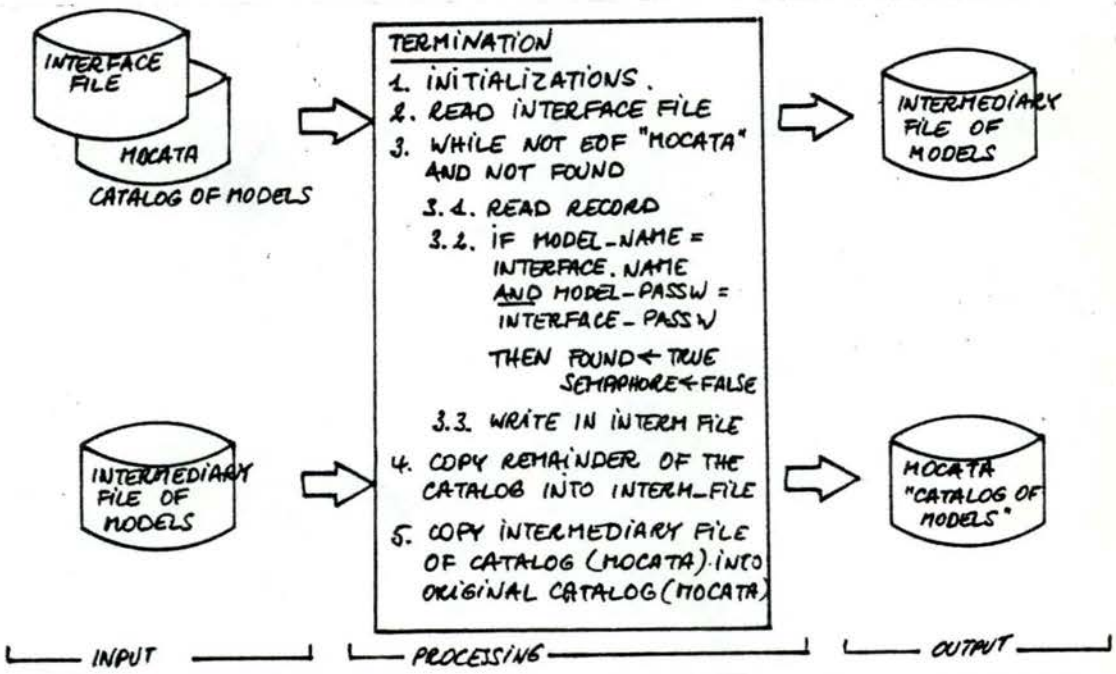
CHAPITRE 9 : TERMINAISON

1. MODULE 7 - TERMINATION - NIVEAU D'ANALYSE 1.7.

1.1. SPECIFICATIONS

Les calculs sur le modèle sont terminés, il faut maintenant libérer l'accès au modèle. C'est-à-dire il faut mettre à jour le dispositif de sécurité et en particulier changer l'état du sémaphore.

1.2. SCHEMA



1.3. DESCRIPTION

1.3.1. Entrées

- fichier interface, (fg -);
- fichier catalogue des modèles, (fcata - mocata);
- fichier catalogue intermédiaire, (fica -).

1.3.2. Sorties

- fichier catalogue des modèles intermédiaires, (fica -);

- fichier catalogue des modèles, (fcata - mocata).

1.3.3. Traitements

- cfr.: schèma et organigramme.

1.4. CODE PASCAL CORRESPONDANT

cfr.: programme TERMINAT - (1.7.)

CHAPITRE 10 : PROCEDURES DE BASE

REMARQUES

Dans ce chapitre nous allons présenter un certain nombre de procédures utilisées dans la plupart des modules analysés ci-dessus. Nous avons désigné le niveau d'analyse de ces procédures par des numéros de référence commençant par "0.". Ces procédures on été élaborées suite à une analyse "bottom - up".

1. NEWPAGE - NIVEAU D'ANALYSE 0.1.

1.1. SPECIFICATIONS

Cette procédure a pour objectif d'effectuer un certain nombre de sauts de page spécifiés par l'utilisateur. Ces sauts de page sont remplacés par des sauts de lignes si l'utilisateur n'a pas opté pour le saut de page.

1.2. DESCRIPTION

1.2.1. Entrées

- pas d'entrées.

1.2.2. Sorties

- saut(s) de page ou de ligne.

1.2.3. Traitements

- cfr.: code Pascal.

1.3. CODE PASCAL CORRESPONDANT

```
procedure newpage(n : integer);
var     i : integer;
begin
        for i:=1 to n do if guidance_record.ff
                        then write(^\^^) else write(^\^^)
end;
```

2. NEWLINE - NIVEAU D'ANALYSE 0.2.

2.1. SPECIFICATIONS

Cette procédure a pour objectif d'effectuer un certain nombre de sauts de ligne spécifiés par l'utilisateur.

2.2. DESCRIPTION

2.2.1. Entrées

- pas d'entrées.

2.2.2. Sorties

- saut(s) de ligne.

2.2.3. Traitements

- cfr.: code Pascal.

2.3. CODE PASCAL CORRESPONDANT

```
procedure newline(n : integer);
var     i : integer;
begin
        for i:=1 to n do write(#13#10)
end;
```

3. FILESTORE - NIVEAU D'ANALYSE 0.3.

3.1. SPECIFICATIONS

Cette procédure a pour objectif de transformer un fichier temporaire se trouvant en "aft" en un fichier permanent qui se trouvera dans la directory à laquelle l'exécution du programme est attachée.

3.2. DESCRIPTION

3.2.1. Entrées

- fichier qui se trouve dans la zone des fichiers temporaires.

3.2.2. Sorties

- fichier rendu permanent.

Traitements

- cfr.: code Pascal.

3.3. CODE PASCAL CORRESPONDANT

```
procedure filestore(var filename : chain8);
var     i,j : integer;
        cmd : packed array[1..22] of char;
begin
        for i := 1 to 22 do cmd[i] := ' ';
        j := 0;
        for i := 1 to 5 do
        begin
             case i of
                        1 : cmd[i] := 'p';
                        2 : cmd[i] := 'e';
                        3 : cmd[i] := 'r';
                        4 : cmd[i] := 'm';
                        5 : cmd[i] := ' ';
                        end;
             j := j+1;
        end;
        i := 0;
        repeat    i := i + 1;
                  if filename[i] <> ' ' then
                  begin
                       cmd[5 + i] := filename[i];
                       j := j + 1;
                  end;
```

```
1)      until (filename[i] = ' ') or (i = 8);
        j := j + 1;
        cmd[j] := ';';
        i := 0;
        repeat    i := i + 1;
                  if filename[i] <> ' ' then
                  begin
                       cmd[j + i] := filename[i];
                  end;
        until ( filename[i] = ' ') or (i = 8);
        run(cmd);
end;
```

4. STAR - NIVEAU D'ANALYSE 0.4.

4.1. SPECIFICATIONS

Le but de cette procédure est d'imprimer sur une ligne un certain nombre de "*".

4.2. DESCRIPTION

4.2.1. Entrées

- pas d'entrées.

4.2.2. Sorties

- ligne d'étoiles imprimée.

4.2.3. Traitements

- cfr.: code Pascal.

4.3. CODE PASCAL CORRESPONDANT

```
procedure star(nbr_of_characters : integer );
var        i : integer;
begin
        for i := 1 to nbr_of_characters do write('*');
        writeln
end;
```

5. DECODING_OF_STRING - NIVEAU D'ANALYSE 0.5.

5.1. SPECIFICATIONS

Cette procédure a pour objectif de lire une chaîne de caractères. La procédure contrôle si cette chaîne correspond à un entier ou pas. Si le résultat de ce contrôle est négatif, c'est-à-dire la chaîne contient des caractères non numériques, elle met une variable d'erreur à 'true' (vrai). Sinon cette variable est à 'false' (faux).

5.2. DESCRIPTION

5.2.1. Entrées

- pas d'entrées.

5.2.2. Sorties

- pas de sorties.

5.2.3. Traitements

- cfr.: code Pascal.

5.3. CODE PASCAL CORRESPONDANT

cfr.: page - 156 bis -

```
procedure decimal_or_string(string : string_of_char ;
                            var error : boolean ;
                            var nber : integer );
label    1;
const    max_value = 2147483647;
         max_dimension = 10;
var      index : integer;
         digit : 0..9;
begin
         error := true;
         nber := 0;
         index := 1;
         while (index <= max_dimension) and (string[index] = ' ') do
              index := index + 1;
         if index > max_dimension then
         begin
              writeln('  invalid data : non signed integer required');
              goto 1
         end;
         while (index <= max_dimension) and (string[index] <> ' ') do
         begin
              if not (string[index] in ['0'..'9']) then
              begin
                   writeln('  invalid data : non signed integer required');
                   goto 1
              end;
              if index = max_dimension then
                   if max_value - string; then
                   begin
                   writeln('  invalid data : non signed integer required');
                   goto 1
                   end;
              digit := ord(string[index])-ord('0');
              nber := nber*10 + digit;
              index := index + 1;
         end;
         error := (index <= max_dimension) and (string[index] in ['0'..'9']);
1:end;
```

6. COMMENT - NIVEAU D'ANALYSE 0.6.

6.1. SPECIFICATIONS

Cette procédure a pour objectif d'imprimer du commentaire. Ce commentaire est référencé par un numéro attribué lors de la phase de l'analyse du problème.

6.2. DESCRIPTION

6.2.1. Entrées

- pas d'entrées.

6.2.2. Sorties

- commentaire imprimé.

6.2.3. Traitements

- cfr.: code Pascal.

6.3. CODE PASCAL CORRESPONDANT

```
procedure comment(nbr : integer);
begin
        if nbr = 1 then
        begin
            writeln('  enter an integer');
        end;
        if nbr = 2 then
        begin
            writeln('  1 for begin control');
            writeln('  2 for full control (default value)');
            writeln('  3 for multipath action');
            newline(1);
            writeln('  enter an integer between 1 and 3, corresponding');
            writeln('  to the control chosen');
        end;
end;
```

7. VERIFY - NIVEAU D'ANALYSE 0.7.

7.1. SPECIFICATIONS

Le but de cette procédure est de contrôler l'entrée de données numériques. Aussi longtemps que l'utilisateur n'a pas entré une donnée correcte, cette procédure continue à boucler sur la demande de donnée. Cette procédure utilise la procédure "decoding_of_ string" (cfr.: 0.5.) et la procédure "comment" (cfr.: 0.6.).

7.2. DESCRIPTION

7.2.1. Entrées

- entrée en provenance du terminal.

7.2.2. Sorties

- pas de sorties.

7.2.3. Traitements

- cfr.: code Pascal.

7.3. CODE PASCAL CORRESPONDANT

```
procedure verify(n :integer ; var int : integer);
var     error : boolean ;
begin
        error := true;
        while error do
        begin
            repeat    if [illegible]
                      readln(buffer);
                      if not [illegible]
            until     [illegible]
            decoding_of_string( [illegible] )
        end
end
```

8. ROUNDUP - NIVEAU D'ANALYSE 0.8.

8.1. SPECIFICATIONS

Cette procédure a pour but d'arrondir des nombres réels à l'entier immédiatement supérieur.

8.2. DESCRIPTION

8.2.1. Entrées

- pas d'entrées.

8.2.2. Sorties

- pas de sorties.

8.2.3. Traitements

- cfr.: code Pascal.

8.3. CODE PASCAL CORRESPONDANT

```
function roundup(n : real):integer;

begin
        if n <> trunc(n) then roundup := trunc(n) + 1
                         else roundup := trunc(n);
end;
```

9. SERVICE - NIVEAU D'ANALYSE 0.9.

9.1. SPECIFICATIONS

Cette procédure sert à mettre en oeuvre le "point d'interrogation". Elle utilise la procédure "comment".

9.2. DESCRIPTION

9.2.1. Entrées

- entrée en provenance du terminal.

9.2.2. Sorties

- pas de sorties.

9.2.3. Traitements

- cfr.: code Pascal.

9.3. CODE PASCAL CORRESPONDANT

```
procedure service(n : integer ; var texte : chain);
begin
        repeat    if help then comment(n);
                  readln(texte);
                  if not help and (texte[1] = qm) then comment(n);
        until     texte[1] <> qm
end;
```

CHAPITRE 11 : DESCRIPTION DES FICHIERS

REMARQUES

Dans ce chapitre nous donnons une description des fichiers utilisés par les divers modules. Cette description est complètée par un tableau récapitulatif dit "cross-reference". Les fichiers sont désignés par leur nom interne.

1. EXITFILE

Nom interne : EXITFILE
Nom externe : -

Organisation : séquentielle
Nature : binaire
Durée de vie : temporaire

Type : file of EXITSTATUS

Structure des enregistrements : EXITFILE_RECORD : EXITSTATUS

status ;

Description

Le fichier EXITFILE est un fichier spécial ne contenant qu'un seul enregistrement. Cet enregistrement contient l'information nécessaire pour continuer ou arrêter l'exécution du programme.

2. FCATA

Nom interne : FCATA
Nom externe : MOCATA

Organisation : séquentielle
Nature : binaire
Durée de vie : permanente

Type : file of MODEL

Structure des enregistrements : MODEL_RECORD : MODEL

name;
password;
semaphore;

Description

Le fichier contient le nom, le mot de passe et le sèmaphore aasociès de tous les modèles crèès par le programme. Ce fichier constitue la base du dispositif de protection des modèles des utilisateurs.

3. FICA

Nom interne : FICA
Nom externe : -

cfr.: FCATA

Description

Ce fichier est un fichier intermèdiaire du fichier FCATA.

4. F

Nom interne : F
Nom externe : ‹nom_du_modèle›

Organisation : sèquentielle
Nature : binaire
Durèe de vie : permanente

Type : file of GUIDANCE

Structure des enregistrements : GUIDANCE_RECORD : GUIDANCE

```
assistance;
ff;
modif1;
modif2;
```

```
modif3;
profile1;
profile2;
profile3;
old_profile1;
old_profile2;
transaction;
agent;
site;
link;
```

Description

Le fichier F est un fichier spécial ne contenant qu'un seul enregistrement. Ce enregistrement est unique par modèle et l'information qu'il contient sert à guider le programme dans l'exécution des tâches.

5. FG

Nom interne	: FG
Nom externe	: -
Organisation	: séquentielle
Nature	: binaire
Durée de vie	: temporaire
Type	: file of INTERFACE
Structure des enregistrements	: INTERFACE_RECORD : INTERFACE

```
name;
password;
```

Description

Le fichier FG est un fichier spécial ne contenant qu'un seul enregistrement. L'information contenue dans cet enregistrement sert d'information interface entre les diffèerents modules qui constituent le programme. Cette information assure l'enchaînement correct des modules et l'exécution correcte du programme.

6. FT

Nom interne : FT
Nom externe : <nom_du_modèle>1 et FT3

Organisation : séquentielle
Nature : binaire
Durée de vie : permanente pour <nom_du_modèle>1;
temporaire pour FT3

Type : file of TRANSACT

Structure des enregistrements : TRANSACT_RECORD : TRANSACT

```
typeid;
process;
sender;
receiver;
messnbr;
length;
```

Description

Le fichier contient les enregistrements relatifs à un type de transactions.

7. FIT

Nom interne : FIT
Nome externe : <nom_du_modèle>1

cfr.: FT

Description

Fichier interne FIT dont on se sert pour effectuer les modifications.

8. FT2

Nom interne : FT2
Nom externe : -

cfr.: FT

Description

Ce fichier est un fichier intermédiaire et temporaire.

9. FA

Nom interne : FA
Nom externe : <nom_du_modèle>2 et FA3

Organisation : séquentielle
Nature : binaire
Durée de vie : permanente pour <nom_du_modèle>2
temporaire pour FA3

Type : file of AGENT

Structure des enregistrements : AGENT_RECORD : AGENT

```
agetyp;
workload;
transinit;
profile1;
profile2;
profile3;
```

Description

Le fichier contient les enregistrements relatifs aux types d'agents, un enregistrement se rapportant à un agent et à un type de transactions initialisées.

10. FIA

Nom interne : FIA
Nom externe : <nom_du_modèle>2

cfr.: FA

Description

Ce fichier est utilisé lors des modifications sur <nom_du_modèle>2.

11. FA2

Nom interne : FA2
Nom externe : -

cfr.: FA

Description

Le fichier FA2 est un fichier intermédiaire et temporaire.

12. FS

Nom interne : FS
Nom externe : <nom_du_modèle>31(32)(33)
un par profil
et FS3

Organisation : séquentielle
Nature : binaire

Durée de vie : permanente pour ‹nom_du_modèle›3?
temporaire pour FS3

Type : file of SITE

Structure des enregistrements : SITE_RECORD : SITE

```
classid;
numsite;
composant;
typproc;
typage;
numage;
intensity;
proced;
```

Description

Le fichier contient les enregistrements relatifs aux classes de sites.

13. FIS

Nom interne : FIS
Nom externe : ‹nom_du_modèle›31 (32) (33)
un par profil

cfr.: FS

Description

Ce fichier est utilisé lors des modifications de ‹nom_du_modèle›3?.

14. FS2

Nom interne : FS2

Nom externe : -

cfr.: FS

Description

Ce fichier est un fichier intermédiaire et temporaire.

15. FILEAUX

Nom interne : FILEAUX
Nom externe : -

cfr.: FT

Description

Ce fichier est un fichier intermédiaire et temporaire.

16. FL

Nom interne : FL
Nom externe : <nom_du_modèle>41 (42) (43)
un par profil

Organisation : séquentielle
Nature : binaire
Durée de vie : permanente

Type : file of TRANSLOC

Structure des enregistrements : TRANSLOC_RECORD : TRANSLOC
trans;

```
send_proc;
send_site;
s_site_nber;
receiv_proc;
receiv_site;
r_site_nber;
number;
length;
mode;
proportion;
link;
```

Description

Ce fichier contient les informations relatives aux localisations des transactions. Il y a autant de fichiers de ce type que de profils utilisés.

17. FILEACT

Nom interne	: FILEACT
Nom externe	: _
Organisation	: séquentielle
Nature	: binaire
Durée de vie	: temporaire
Type	: file of ACTIVITY
Structure des enregistrements	: ACTIVITY_RECORD : ACTIVITY

```
classid;
transinit;
sum;
```

Description

Le fichier contient les enregistrements relatifs aux différentes classes de sites (par rapport à un type de transactions).

18. FILEDN

Nom interne : FILEDN
Nom externe : -

Organisation : séquentielle
Nature : binaire
Durée de vie : temporaire

Type : file of DNSITE

Structure des enregistrements : DNSITE_RECORD : DNSITE

```
site;
sum_r;
sum_m_r;
sum_v_r;
sum_u;
sum_m_u;
sum_v_u;
sum_o;
sum_m_o;
sum_v_o;
```

Description

Le fichier contient les informations relatives aux transmissions qui se font au niveau des datanets.

19. FILECAL

Nom interne : FILECAL
Nom externe : -

Organisation : séquentielle
Nature : binaire
Durée de vie : temporaire

Type : file of TRANSCAL

Structure des enregistrements : TRANSCAL_RECORD : TRANSCAL

```
                                    link;
                                    class_act;
                                    nbr_letters;
                                    nbr_link;
                                    nbr_traffic;
                                    volume;
```

Description

Ce fichier est un fichier parallèle au fichier FL. L'information qu'il contient est la suite de celle qui est contenue dans le fichier FL.

20. FILEINT2

Nom interne	: FILEINT2
Nom externe	: -
Organisation	: séquentielle
Nature	: binaire
Durée de vie	: temporaire
Type	: file of INTERSITE
Structure des enregistrements	: INTERSITE_RECORD : INTERSITE

```
                                    send_site;
                                    receiv_site;
                                    mode;
                                    nbr_letters;
                                    volume;
                                    nbr_traffic;
                                    nbr_letters_e;
                                    volume_e;
                                    nbr_letters_h;
                                    volume_h;
                                    link;
```

Description

Ce fichier contient les informations de synthèse du trafic intersite.

21. FILEINT

Nom interne : FILEINT
Nom externe : -

cfr.: FILEINT2

Description

Ce fichier est un fichier intermédiaire et temporaire.

22. TABLEAU RECAPITULATIF

nom interne	nom externe	moni-teur	modul 1	modul 2	modul 3	modul 4	modul 5	modul 6	modul 7	upda-te
1.EXITFILE	-	R	C	-	-	-	-	-	-	-
2.FCATA	mocata	-	A,R	-	-	-	-	-	R,W	R,W
3.FICA	-	-	C,R	-	-	-	-	-	C,R	C,R
4.F	n.d.m.	-	C,W	R,W	R,W	R,W	R,W	R	-	-
5.FG	-	-	C	R	R	R	R	R	R	-
6.FT	n.d.m.1	-	-	C,R	-	-	R	R	-	-
7.FIT	id	-	-	R,W	-	-	-	-	-	-
8.FT2	-	-	-	C,R	-	-	-	-	-	-
9.FA	n.d.m.2	-	-	-	C,R	-	R	R	-	-
10.FIA	id	-	-	-	R,W	-	-	-	-	-
11.FA2	-	-	-	-	C,R	-	-	-	-	-
12.FS	n.d.m3.	-	-	-	-	C,R	R	R	-	-
13.FIS	id	-	-	-	-	R,W	-	-	-	-
14.FS2	-	-	-	-	-	C,R	-	-	-	-
15.FILEAUX	-	-	-	-	-	-	C,R	-	-	-
16.FL	n.d.m4.	-	-	-	-	-	C	R	-	-
17.FILEACT	-	-	-	-	-	-	-	C,R	-	-
18.FILEDN	-	-	-	-	-	-	-	C,R	-	-
19.FILECAL	-	-	-	-	-	-	-	C,R	-	-
20.FILEINT2	-	-	-	-	-	-	-	C,R	-	-
21.FILEINT	-	-	-	-	-	-	-	C,R	-	-

A: append ; C: create ; R: read ; W: write

CHAPITRE 12 : PROGRAMME DE SERVICE

1. SPECIFICATIONS

Dans ce chapitre nous allons brièvement présenter un programme de service pour notre outil. Ce programme sert à mettre à jour le fichier catalogue des modèles. Il permet d'imprimer le contenu du fichier catalogue, d'ajouter un modèle à la fin du fichier, d'enlever un modèle du catalogue et de libèrer l'accès à un modèle auquel tout accès est interdit.

1.1. DESCRIPTION

1.1.1. Entrèes

- fichier catalogue des modèles, (fcata - mocata);
- fichier intermèdiaire du catalogue des modèles, (fica -).

1.1.2. Sorties

- fichier intermèdiaire du catalogue des modèles;
- fichier catalogue des modèles mis à jour.

1.1.3. Traitements

- cfr.: code Pascal.

1.2. CODE PASCAL CORRESPONDANT

cfr.: programme UPDATING

DIMENSIONNEMENT DE RESEAUX DSA

MANUEL D'UTILISATION DE L'OUTIL AUTOMATIQUE

Version : 0.1 - avril 1982

Rèfèrence : " SIZING A DSA NETWORK "
(sales aid 80 - 034 rèvisè)

TABLE DES MATIERES PAGE

1. INTRODUCTION GENERALE 1

2. OBJECTIFS DE L'OUTIL 2

2.1. DESCRIPTION DU RESEAU 2

2.2. RESULTATS 2

3. ETUDES PRELIMINAIRES REQUISES 4

3.1. INTRODUCTION ET PLAN DES ETUDES 4

3.2. ETUDE DES APPLICATIONS DE L'UTILISATEUR 4

3.2.1. TYPES DE TRANSACTIONS 4

3.2.2. TYPES D'AGENTS 6

3.3. ETUDE DE LA TOPOLOGIE DU RESEAU 8

3.3.1. CLASSES DE SITES 8

3.3.2. LIENS ENTRE CLASSES DE SITES 10

3.3.2.1. MODE DE COMMUNICATION EQUIDISTRIBUE OU "E" 11
3.3.2.2. MODE DE COMMUNICATION HIERARCHIQUE OU "H" 11
3.3.2.3. MODE DE COMMUNICATION IDENTIQUE OU "I" 12

PAGE

3.3.3. NATURE DES LIENS RETENUS 12

4. INFORMATIONS GENERALES SUR LE PROGRAMME 14

5. UTILISATION DU PROGRAMME 16

5.1. ACCES AU PROGRAMME 16

5.2. AIDES FOURNIES A L'UTILISATEUR 16

5.2.1. SAUT DE PAGE 16

5.2.2. AIDE AUTOMATIQUE 16

5.2.3. POINT D'INTERROGATION 17

5.2.4. MAJUSCULES ET MINUSCULES 17

5.3. GRANDES LIGNES DU FONCTIONNEMENT DE L'OUTIL 18

5.3.1. APPEL 18

5.3.2. INTRODUCTION 18

5.3.3. IDENTIFICATION DU MODELE 20

5.3.4. GENERALITES 21

5.3.4.1. REMARQUES 21
5.3.4.2. ENCHAINEMENT 22
5.3.4.3. MODES D'EXECUTION DES MODULES DU PROGRAMME 22
5.3.4.4. MODIFICATIONS 23
5.3.4.5. LIENS 23
5.3.4.6. DATANETS 23

6. FORMULAIRES 24

PAGE

7. EXEMPLE CONCRET D'UTLISATION DE L'OUTIL 28

MANUEL D'UTILISATION

1. INTRODUCTION GENERALE

L'outil que vous utilisez est une version automatisée du CII - HB sales aid " SIZING A DSA NETWORK " (80-034 révisé).

En partant des données fournies par une analyse fonctionnelle d'une application utilisateur, l'outil permet de dimensionner :

- les liens du réseau primaire;
- les noeuds composés par des ordinateurs de réseau du type datanet DN 7100.

L'outil guide l'utilisateur d'une étape à l'autre, en assurant l'exploitation de toutes les combinaisons logiquement possibles.

Le modèle entré par l'utilisateur est stocké en mémoire secondaire, ce qui permet des redéfinitions aisées. Ainsi l'utilisateur peut simuler des modèles différents pour son réseau.

2. OBJECTIFS DE L'OUTIL

2.1. DESCRIPTION DU RESEAU

L'utilisateur décrit son réseau. Cette modélisation du réseau est basée sur les données fournies par une analyse fonctionnelle de l'ensemble des applications de l'utilisateur qui doivent être mises en oeuvre sur le réseau. Ainsi l'utilisateur n'est pas obligé de se familiariser avec un vocabulaire spécial.

Les informations requises se situent à deux niveaux distincts :

- Le niveau "application"

  Les applications sont analysées dans une optique de réseau, c'est-à-dire on s'intéresse aux transactions générées et aux agents qui les initialisent.

- Le niveau "réseau"

  L'utilisateur définit les différents sites de son réseau.

2.2. RESULTATS

Le programme assiste l'utilisateur dans la collecte des données et il fournit :

- un résumé des données saisies concernant
  - les types de transactions;
  - les types d'agents;
  - les classes de sites;

- des propositions de liens logiquement possibles entre des sites et il accepte les liens retenus par l'utilisateur;

- un dimensionnement des liens (en nombre de lignes, capacité de ces lignes et taux d'utilisation);

- par site, une proposition d'un modèle de datanet DN 7100 (3 modèles sont possibles : DN 7102; DN 7102 + DCE 7104; DN 7103) avec la charge du processeur central du datanet. La charge sera présentée :
  - globalement
  - par réseau (primaire, secondaire)
  - pour la connexion datanet-ordinateur hôte (s'il y en a une).

3. ETUDES PRELIMINAIRES REQUISES

3.1. INTRODUCTION ET PLAN DES ETUDES

Dans ce paragraphe nous exposons les études à faire par l'utilisateur avant d'utiliser l'outil de dimensionnement. Ces études vont déboucher sur une collecte de données pour laquelle l'utilisation de formulaires est vivement recommandée. Ces formulaires sont présentés au chapitre 6 de ce manuel.

Les études sont divisées en deux parties :

- la première partie est relative aux applications de l'utilisateur sans prendre en compte les aspects "hardware";
- la deuxième partie s'occupe de la topologie du réseau sur lequel les applications seront mises en oeuvre.

3.2. ETUDE DES APPLICATIONS DE L'UTILISATEUR

3.2.1. TYPES DE TRANSACTIONS

L'utilisateur doit d'abord décrire ce que le réseau doit acheminer.

Il décrit les types de transactions possibles, c'est-à-dire les unités de travail en termes de processus utilisés et de messages échangés.

La description des types de transactions de l'application utilisateur est facilitée par l'utilisation du formulaire "transaction types definition" présenté au chapitre 6 de ce manuel.

Pour chaque type de transactions identifié, l'utilisateur détermine tous les types de processus

nècessaires au traitement de la transaction considèrèe.

Il faut remarquer que chaque type de processus ainsi spècifiè est indistribuable, c'est-à-dire le processus ne peut pas être implantè partiellement sur un site et partiellement sur un autre.

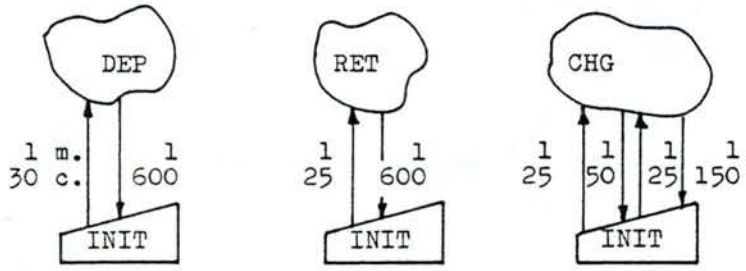
Pour chaque type de transactions il faut dèterminer les transmissions, c'est-à-dire il faut dèterminer le nombre de messages et leur longueur en caractères, transmis entre agent initialisant la transaction et type(s) de processus; entre types de processus et entre type(s) de processus et agent initialisant.

EXEMPLE

Soit une application bancaire. Une analyse a dèterminè 3 unitès de travail, c'est-à-dire 3 tyes de transaction :

1) DEPOT
2) RETRAIT
3) CHANGE

L'analyse nous fait en même temps dècouvrir les processus utilisès pour traiter ces transactions :



transaction type id.	process types employed	transmissions			
		sending proc.typ.	receiving proc.typ.	nbr of mess.	length (char)
DEPOT	DEP	INIT DEP	DEP INIT	1 1	30 600
RETRAIT	RET	INIT RET	RET INIT	1 1	25 600
CHANGE	CHG	INIT CHG	CHG INIT	2 1 1	25 50 150

3.2.2. TYPES D'AGENTS

L'utilisateur décrit ensuite qui initialise les transactions.

Il définit les types d'agents, c'est-à-dire les "utilisateurs - terminal" vus par l'analyse fonctionnelle de l'ensemble des applications.

Exemple: pour une application bancaire on distingue :

- le type "Caissier"
- le type "Guichetier"
- le type "Bureau".

Chaque type d'agents est caractérisé par son (ses) profil(s) d'utilisation (d'initialisation) de transactions.

Par profil nous entendons l'ensemble des types de transactions que l'agent initialise durant une certaine période de la journée de travail.

Nous nous sommes limités à un maximum de 3 profils. Le premier profil étant relatif aux heures dites "normales", le deuxième et le troisième aux heures dites de "pointe" et aux évènements saisonniers.

Le premier profil est obligatoire, tandis que les deux autres sont optionnels.

Pour chaque type d'agents et pour chaque profil utilisé, il faut déterminer les types de transactions initialisés et indiquer pour chaque type la part qu'il représente dans l'ensemble des transactions initialisées.

La description des types d'agents est facilitée par l'utilisation du formulaire " agent type definition " présenté au chapitre 6 du manuel.

EXEMPLE

agent type id.	work load	transactions usage profile			
		transaction	pr.1	pr.2	pr.3
CAISSIER	T	DEPOT	40		10
		RETRAIT	50		70
		CHANGE	10		20
BUREAU	T	DEPOT	100		0
GUICHET	T	DEPOT	10		0
		RETRAIT	90		100

pas utilisé

3.3. ETUDE DE LA TOPOLOGIE DU RESEAU

3.3.1. CLASSES DE SITES

L'entreprise désirant un réseau pour supporter son (ses) application(s), doit être vue comme un ensemble structuré de sites.

Chaque site est un centre de traitement (au sens large du terme) situé dans un lieu géographique donné.

Pour faciliter la collecte de ces données, on introduit la notion de CLASSES de sites.

Tous les sites d'une même classe présentent :

- la même configuration (hardware);
- les mêmes processus;
- les mêmes types d'agent (nombre ; intensité et procédure).

Ainsi tous les résultats seront identiques pour chaque élément appartenant à une classe.

La collecte des données est facilitée par l'utilisation du formulaire "site-classes definition" présenté au chapitre 6 de ce manuel.

Pour chaque classe distinguée, il faut spécifier le nombre de sites dans cette classe.

Par classe de sites il faut préciser la configuration hardware, c'est-à-dire il faut spécifier les composants. Les composants peuvent être des :

- DPS 8
- DPS 7
- DN 7100
- MINI 6 / DSS

Par classe de sites il faut déterminer les processus implantés. Les processus peuvent uniquement être implantés sur un ordinateur hôte (c'est-à-dire un ordinateur offrant des services localement et à d'autres systèmes à travers le réseau primaire) ou sur un satellite (c'est- à-dire un ordinateur rendant des services à un nombre d'utilisateurs locaux dans son réseau secondaire et communiquant avec d'autres ordinateurs hôtes et satellites via le réseau

primaire).

L'utilisateur doit ensuite spécifier les types d'agents présents au niveau de chacun des sites de la classe. Il faut indiquer leur nombre, leur intensité moyenne de travail exprimée en transcations par heure (sans faire une discrimination entre les différents types de transactions initialisés) et le protocole utilisé.

Il est très probable qu'il y ait dans un réseau complexe des sites qui sont identiques d'un point de vue configuration hardware, processus et types d'agents présents sauf que l'activité des agents peut varier. Au lieu de regrouper tous ces sites dans une même classe, nous avons retenu la possibilité de subdiviser une classe en sous-classes ayant chacune la même configuration hardware, les mêmes processus et les mêmes types d'agents mais une activité distincte.

Au point 3.2.2. nous avons introduit la notion de "profil"; nous avons vu que l'utilisateur peut définir jusqu'à 3 profils différents. A un profil donné correspond un problème et donc une solution globale. C'est pourquoi il faut définir les classes de sites autant de fois qu'il y a des profils car il est probable que l'activité change sur un site avec le profil.

REMARQUE

Il existe une relation entre intensité de travail et profil d'utilisation des transactions.

Pour un agent initialisant une transaction TA :

intensité(S) * profil(TA) = nombre de transactions TA par heure pour cet agent localisé au site S

Supposons que l'agent X initialise deux types de transaction TA et TB et que son travail soit réparti de la facon suivante :

20 % de TA
80 % de TB

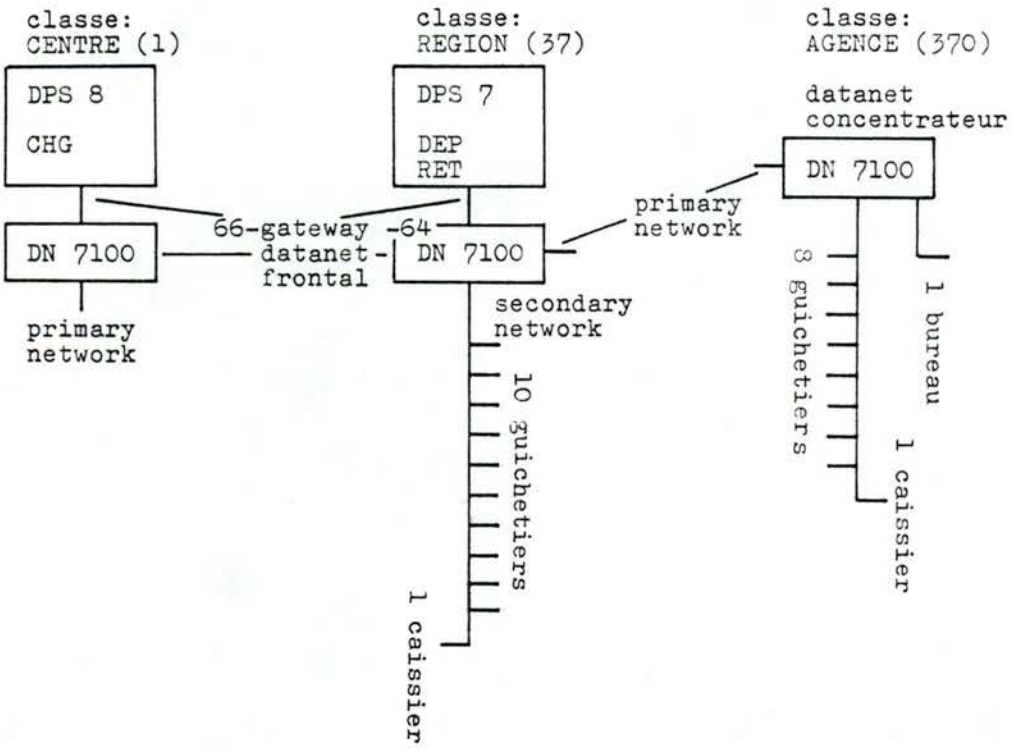
Si l'intensité de travail est de 50 transactions par heure, l'agent X initialisera :

10 transactions du type TA
40 transactions du type TB.

EXEMPLE

Supposons que nous utilisions un seul profil (heures normales). Supposons que nous ayons découvert les classes de sites suivantes. (Consultez également les exemples

précédents).



class id.	nbr	configu-ration comp.	process on the site	initiating agents at the site			
				type id.	nbr	intens.	proc.
CENTRE		DPS8 DN 7100	CHG				
REGION		DPS7 DN 7100	RET DEP	GUICHET. CAISSIER	10 1	40 80	VIP VIP
AGENCE		DN 7100		GUICHET. CAISSIER BUREAU	8 1 1	35 70 20	VIP VIP ASY

3.3.2. LIENS ENTRE CLASSES DE SITES

Après avoir défini les classes de sites, l'utilisateur est amenené à choisir les liens entre les classes de sites.

Le programme fournit tous les liens logiquement possibles et l'utilisateur sélectionne les liens désirés en spécifiant le mode de communication et la nature du lien.

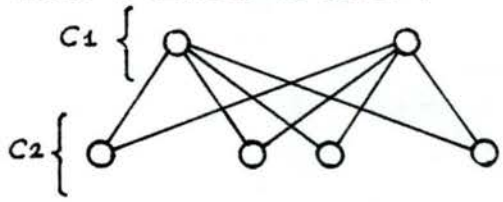
Ainsi l'utilisateur définit un schéma de son réseau.

Nous allons rapidement passer en revue les différents modes de communication.

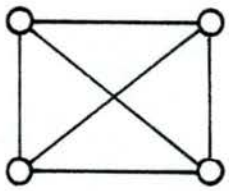
3.3.2.1. MODE DE COMMUNICATION EQUIDISTRIBUE OU "E"

Le mode peut être envisagé entre 2 classes de sites, ou entre sites d'une même classe.

* entre 2 classes de sites :



* entre les sites d'une même classe :

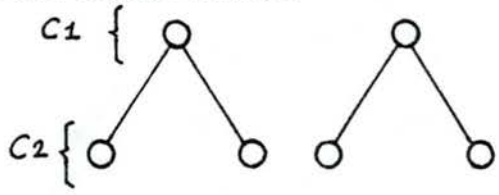


L'activateur activera le type de processus indifféremment sur n'importe quel site de la classe d'un point de vue géographique. Les processus activés peuvent être différents pour chaque initialisation de la transaction.

La charge de travail est supposée être distribuée de facon égale sur les sites de la classe considérée.

3.3.2.2. MODE DE COMMUNICATION HIERARCHIQUE OU "H"

Un activateur sur un site donné d'une classe donnée activera toujours le même processus localisé sur un site d'une classe donnée.



Comme la charge de travail est de nouveau

uniformément distribuée , il s'en suit que la classe du site supportant l'activateur contient au moins autant de sites que la classe supportant le processus récepteur.

3.3.2.3. MODE DE COMMUNICATION IDENTIQUE OU "I"

L'activateur et le serveur (processus traitant) se trouvent sur le même site physique. La transaction à traiter est lancée à partir du réseau secondaire.

Si les activations d'une transaction ne sont pas toutes traitées de la même façon il faut indiquer les proportions des différents modes utilisés.

EXEMPLE

70 % des activations de la transaction sont traitées en mode hiérarchique;
30 % de façon équidistribuée;

possibilités :

1) à l'intérieur d'une même classe
   mode I : 70 %
   mode E : 30 %

2) entre deux classes de sites
   mode H : 70 %
   mode E : 30 %

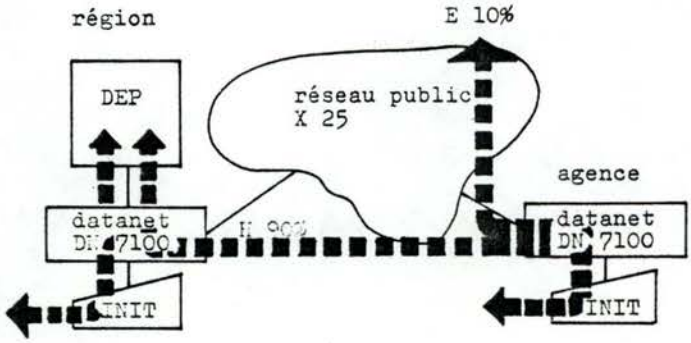
3.3.3. NATURE DES LIENS RETENUS

Pour chaque lien, l'utilisateur a le choix entre :

- lien point-à-point (HDLC);
- lien X25 - public
- lien X25 - privé

EXEMPLE

Reprenons l'exemple de l'application bancaire :



Le programme proposera pour la transaction DEPOT :

```
for DEPOT from INIT REGION to DEP REGION
mode ?
> I
proportion ?
> 100

for DEPOT from INIT AGENCE to DEP REGION
mode ?
> E
proportion ?
> 10
link ?
> u (public X25)
        90 % in mode H

etc.
```

les lignes précédées de ">" indiquent les réponses de l'utilisateur.

Les résultats de cette définition sont :

transaction	site émetteur	site récepteur	mode	%	messages nb.	messages long	lien
DEPOT	région	région	I	100	1	30	
			I	100	1	600	
	agence	région	E	10	1	30	U
			H	90	1	30	U
	région	agence	E	10	1	600	U
			H	90	1	600	U

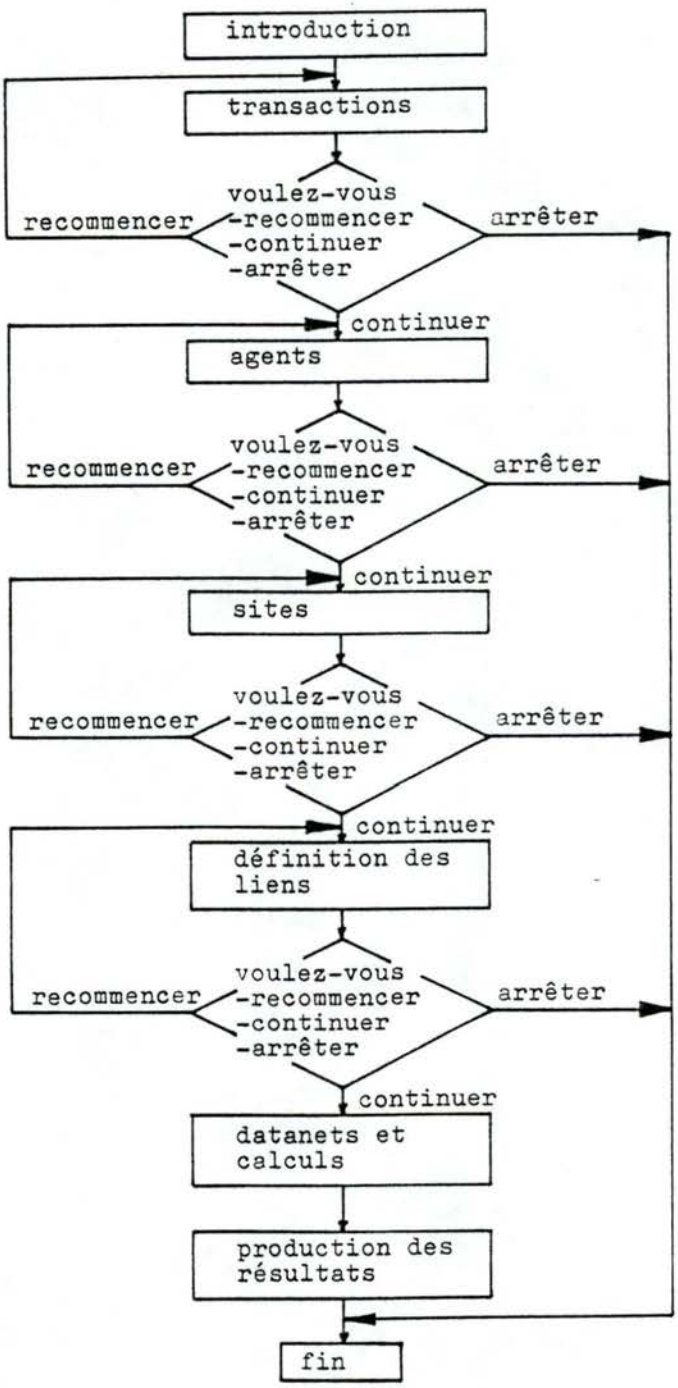
4. INFORMATIONS GENERALES SUR LE PROGRAMME

Le programme, d'un point de vue utilisateur, peut être subdivisé en 6 parties :

- introduction et identification du modèle;
- collecte des données : transactions;
- : agents;
- : sites;
- : définition des liens;
- : datanets et calculs.

Les données collectées sont mémorisées em mémoire secondaire. Le programme mémorise également l'état d'avancement du travail sur le modèle de l'utilisateur. Ainsi l'utilisateur a la possibilité de quitter le programme entre deux parties. Le schéma suivant présente les possibilités d'exécution du programme.

schéma d'exécution :



5. UTILISATION DU PROGRAMME

5.1. ACCES AU PROGRAMME

L'accès au programme est fonction de son implantation physique. La version que nous avons développée sera mise en oeuvre sur un DPS8 sous TSS.

Les modalités de connexion et d'accès seront précisées par l'équipe chargée de la maintenance de l'outil.

5.2. AIDES FOURNIES A L'UTILISATEUR

5.2.1. SAUT DE PAGE

L'exécution du programme se faisant à partir d'un télétype , des sauts de page sont prévus dans un souci de bonne présentation des résultats. L'utilisateur peut ne pas désirer ces sauts de page (certaines imprimantes ne les acceptant pas), c'est pourquoi ils sont optionnels. La question suivante sera posée au début du programme :

Do you want form feed option ?
Type Y or N

Si la réponse est oui (Y), les sauts de page seront effectués dans tout le programme. Sinon ils seront remplacés par un saut de ligne.

5.2.2. AIDE AUTOMATIQUE

Pour les utilisateurs non initiés, un système d'aide automatique a été conçu. La question suivante sera posée au début du programme:

Do you want help ?
Type Y or N

Si la réponse est oui (Y), une introduction générale sera imprimée et des explications seront fournies à la première occurence de chaque question.

5.2.3. POINT D'INTERROGATION

A toute question l'utilisateur peut répondre " ? " afin d'avoir quelques lignes d'explication sur la réponse attendue. Cette possibilité sera rappelée au début du programme par la remarque suivante :

Explanations to individual questions are available by typing ?

5.2.4. MAJUSCULES ET MINUSCULES

Les réponses aux questions peuvent se faire indifféremment en minuscules ou en majuscules avec la remarque suivante : Si le nom du modèle et son mot de passe associé sont écrits en majuscules (p.ex.), il faut toujours donner le nom et le mot de passe en majuscules sinon le programme est incapable d'identifier le modèle.

5.3. GRANDES LIGNES DU FONCTIONNEMENT DE L'OUTIL

5.3.1. APPEL

L'appel au programme (pour la version ci-présentée) se fait par : /monitor (cfr 5.1.) .

Les deux premières questions posées sont celles du saut de page et de l'aide automatique. Il convient de remarquer qu'au début de l'exécution du programme, l'utilisateur a la possibilité d'ajuster le papier.

5.3.2. INTRODUCTION

Nous allons présenter les textes généraux servant d'introduction.

entête :

CII - HONEYWELL BULL

TERMINALS & NETWORKING
MARKETING SUPPORT

...

TOOL ON SUBJECT : SIZING A DSA NETWORK
REFERENCE : CII-HB SALES AID 80-034

Si l'aide automatique est demandée, une introduction au système sera fournie.

GENERAL INTRODUCTION

The tool you are using, is an automated version of the revised CII - HB Sales Aid 80 - 034 " SIZING A DSA NETWORK ".

By processing the data, provided by a functional ana-

lysis of a user's application, the tool permits to size :

- the links of the primary network
- the nodes composed by datanets DN 7100

The tool leads the user on, one step at a time, carefully ensuring that all possible combinations (logical point of view) are considered.

The network model entered will be saved in order to allow easy redefinitions of the model. Thus the user may simulate different models for his network.

See also "Sizing a DSA network" Computer Aided Sales Aid (CASA) user's guide.

INTRODUCTION FOR NOVICE USER

The user will be asked to define his network. This network modelization is almost entirely based on the data provided by a functional analysis of the user's application that has to be implemented on the network. Therefore, the user will not be obliged to get accustomed to a special vocabulary.

In order to reduce the amount of data to be collected, certain definitions have been introduced :

- TRANSACTIONS :

  The user will first be asked to define the basic transactions used. Each transaction will be processed by one or several processes somewhere in the system, and these are named.
  This processing will give rise to transmissions of messages between initiator and process(es).

- AGENTS :

  Next the user defines agents who initiate these transactions.

- SITES :

  The user has to define sites, this task being simplified by regrouping identical sites (configuration , processes) into classes. The agents local to a site (if any), and the processes implanted at that site (if any) are specified.

- LINKS :

The user is given all logically possible links and he selects those that are required.

The tool sizes the required links of the primary network and the nodes composed by datanets.

The user will be promted for all required information in a logical manner.

5.3.3. IDENTIFICATION DU MODELE

Cette identification commence par une spécification du travail de l'utilisateur : s'il veut créer un nouveau modèle ou s'il veut travailler sur un ancien modèle. Cette spécification est faite en répondant à la question suivante :

Did you already enter a model of your network ?
Type Y or N

Ensuite l'utilisateur doit répondre à la suite des questions suivantes :

Enter model name of your network,
(6 char, no spaces)

exemple : reseau

Il faut donc entrer un nom de 6 caractères, ce nom ne peut pas contenir des blancs.

Enter password associated to the model,
(8 char max)

exemple : dsa

Suite à ces données, 5 évènements sont possibles .

Il peut y avoir l'impression d'un des 4 textes suivants :

1 - Network model under this name already exists, if you want to continue now, you erase your previous network model.

Ce message se termine par la question :

Do you want to continue ?
Type Y or N

2 - No such network model.

3 - Network model under this name, already exists, choose an other name for your network model.

4 - You have no access to the referenced network model.

Si aucun de ces messages n'apparaît, l'accès au modèle spécifié est permis.

Le nombre d'essais d'accès est limité à 3.

Le message : "no access given, please refer to user guide" indique que l'exécution du programme se termine.

Le message : "network model : ------ is busy, try access later" indique qu'un utilisateur est en train de travailler sur le modèle en question. Pour des raisons d'intégrité des données, l'accès concurrentiel à un même modèle n'est pas permis.

Si l'accès a été permis et si cet accès est possible, alors un bref rappel est donné :

You may use lowercase or uppercase letters.

5.3.4. GENERALITES

5.3.4.1. REMARQUES

Dans la suite nous nous limitons à faire des remarques d'ordre général en ce qui concerne le fonctionnement du programme. Il ne sert à rien de commenter une exécution complète qui serait trop détailllée ou un extrait de dialogue "utilisateur - outil" qui serait incomplèt et incompréhensible. Nous recommandons à l'utilisateur de faire des essais sur un

petit modèle expérimental. En demandant l'assistance automatique durant l'exécution, l'utilisateur pourra aboutir rapidement à des résultats acceptables. En effet notre outil est au maximum "self-explaining", c'est-à-dire il fournit à chaque question un minimum de commentaires facilitant la compréhension. Pour des problèmes de compréhension de la méthode sous-jacente à l'outil, veuillez consulter le sales aid ou un responsable.

5.3.4.2. ENCHAINEMENT

Le schéma d'exécution du programme présenté ci-dessus, montre qu'il y a moyen de reprendre un module qu'on vient de quitter, d'aller au module suivant ou bien de quitter le programme.

L'utilisateur peut réaliser ce choix en répondant à la question suivante :

```
Do you want to go to next step (G),
            or rerun same step (R),
            or exit            (E)?
```

L'enchaînement se fera en fonction de la réponse fournie.

5.3.4.3. MODES D'EXECUTION DES MODULES DU PROGRAMME

Nous avons prévu deux modes d'exécution possibles. Cette remarque s'applique aux collectes de données relatives aux transactions, agents, sites et liens. Le premier mode d'exécution est relatif à la création d'un nouveau modèle. L'utilisateur est guidé à travers la collecte des données et il va déboucher sur la possibilité de modification des informations entrées. Le deuxième mode est relatif à l'utilisation d'un ancien modèle. Ici l'utilisateur peut faire imprimer les données déjà entrées avant de procéder à des modifications éventuelles.

5.3.4.4. MODIFICATIONS

Lors des modifications, l'utilisateur doit savoir que ces modifications se font ligne par ligne et dans le cas particulier d'une insertion l'utilisateur est obligé de fournir une valeur pour chaque attribut de la ligne, sinon le programme ne peut pas s'exécuter correctement. Pour des raisons de cohérence, des modifications partielles des liens ne sont pas permises.

5.3.4.5. LIENS

De tous les liens logiquement possibles, l'utilisateur ne retient que les liens nécessaires, les liens non désirés sont ignorés en tapant directement CR après la proposition du lien.

5.3.4.6. DATANETS

Des questions qui spécifient clairement ce que l'utilisateur doit fournir comme données seront affichées. Ces données servent à compléter les informations collectées dans les parties précédentes.

6. FORMULAIRES

Nous allons présenter ci-dessous les 3 formulaires recommandés comme support à la collecte des données relatives aux types de transactions, aux types d'agents et aux classes de sites.

formulaire : " **TRANSACTION TYPE DEFINITION** "

TRANSACTION TYPES DEFINITION
NETWORK MODEL - NAME :

TRANSACTION TYPES					
TRANSACTION TYPE ID	PROCESS TYPES EMPLOYED	TRANSMISSIONS			
		SENDING PROC. T.	RECEIVING PROC. T.	NBER OF MESS.	LENGTH (char)

formulaire : " AGENT TYPE DEFINITION "

AGENT TYPES DEFINITION

NETWORK MODEL - NAME :

AGENT TYPES					
AGENT TYPE ID.	WORKLOAD TYPE	TRANSACTION USAGE PROFILES			
		TRANSACTIONS INITIATED	PROF. 1	PROF. 2	PROF. 3

formulaire : " SITE-CLASSES DEFINITION "

SITE - CLASSES DEFINITION

NETWORK MODEL - NAME :

SITE - CLASSES							
CLASS ID	NBER OF SITES	CONFIGURATION	PROCESSES ON SITE	INITIATING AGENTS AT SITE			
		COMPONENTS	TYPE - ID	TYPE-ID	NBER	INT.	PROC.

7. EXEMPLE CONCRET D'UTILISATION DE L'OUTIL

Nous allons présenter l'application de l'outil à un cas concret.

```
*/monitor

  Do you want form feed option ?
  Type Y or N
n

  Adjust form now, if ready type CR



  CII  -  HONEYWELL BULL

  TERMINALS & NETWORKING
    MARKETING  SUPPORT

          ...

  TOOL ON SUBJECT : SIZING A DSA NETWORK
  REFERENCE       : CII-HB SALES AID



  Do you want help ?
  Type Y or N
n



  Explanations  to individual questions  are   available
  by typing  ?



  Did you already entered a model of your network ?
  Type Y or N
n



  Enter model name of your network (5 char, no spaces)
DEMNS



  Enter password associated to the model (8 char max)
fin

  You may use Lowercase or uppercase letters
```

```
TRANSACTION TYPE DEFINITION
***************************

**************************************************

  transaction type ?
?


  12 alphanumeric characters (max)

  if no more data to be entered, type CR

  transaction type ?
depot
      process type ?
dep
            sending process type ?
init
            receiving process type ?
dep
            number of messages ?
1
            length of message ? (characters)
5o
  invalid data : non signed integer required
50
            sending process type ?
dep
            receiving process type ?
init
            number of messages ?
1
            length of message ? (characters)
600
            sending process type ?
?

  Example :
  If a transaction is initiated by an agent from
  a terminal, type init for sending process.
  The receiving process will be the one specified
  under "process type".


  12 alphanumeric characters (max)

  if no more data to be entered, type CR

            sending process type ?
      process type ?
?

  a transaction may use one or se-
  veral processes, one process min


  12 alphanumeric characters (max)

  if no more data to be entered, type CR
```

```
**************************************************

  transaction type ?
retrait
        process type ?
ret
                sending process type ?
init
                receiving process type ?
ret
                number of messages ?
1
                length of message ? (characters)
25
                sending process type ?
ret
                receiving process type ?
init
                number of messages ?
1
                length of message ? (characters)
600
                sending process type ?
        process type ?

**************************************************

  transaction type ?
change
        process type ?
cng
                sending process type ?
init
                receiving process type ?
chg
                number of messages ?
2
                length of message ? (characters)
25
                sending process type ?
chg
                receiving process type ?
init
                number of messages ?
1
                length of message ? (characters)
50
                sending process type ?
chg
                receiving process type ?
init
                number of messages ?
1
                length of message ? (characters)
200
                sending process type ?
        process type ?
```

```
cheques
        process type ?
cng
                sending process type ?
init
                receiving process type ?
chq
                number of messages ?
1
                length of message ? (characters)
100
                sending process type ?
cnq
                receiving process type ?
init
                number of messages ?
1
                length of message ? (characters)
200
                sending process type ?
        process type ?

**************************************************

  transaction type ?

  NETWORK MODEL-NAME  : DEMONS
```

****

TRANSACTION TYPES

TRANSACTION	PROCESS-TYPES	TRANSMISSIONS SEND PROC	RECEIV PROC	NUM MESS	LENGTH	LINE NMBR
depot	dep	init	dep	1	50	1
		dep	init	1	600	2
retrait	ret	init	ret	1	25	3
		ret	init	1	600	4
change	chq	init	chq	2	25	5
		chq	init	1	50	6
				1	200	7
cneques	chq	init	chq	1	100	8
		cng	init	1	200	9

```
  do you want to insert, modify, delete or print the transactions ?
?

  enter I, M, D, P or CR if you want to end

  do you want to insert, modify, delete or print the transactions ?

  Do you want to go to next step (G),
              or rerun same step (R),
              or exit            (E)?
```

```
AGENT TYPES
***********



is profile  2 used ?
?

Profile 1 concerns normal hours
Profile 2 concerns peak hours
Profile 3 concerns seasonal events
Profile 2 and 3 are optional

type Y or N

is profile  2 used ?
n
is profile  3 used ?
n

**************************************************

agent type ?
guichetier
      workload type ?
t
      transactions used in profile  1
            transaction name ?
depot
                  % ?
30
            transaction name ?
retrait
                  % ?
70
            transaction name ?

**************************************************

agent type ?
caissier
      workload type ?
t
      transactions used in profile  1
            transaction name ?
depot
                  % ?
20
            transaction name ?
retrait
                  % ?
60
            transaction name ?
change
                  % ?
20
            transaction name ?

**************************************************
```

```
agent type ?
bureau
      workload type ?
t
      transactions used in profile  1
            transaction name ?
cheques
                  % ?
100
            transaction name ?

**************************************************

agent type ?


NETWORK MODEL-NAME  : DEMONS


*****************************************************************

                        AGENT TYPES

*****************************************************************
AGENT-TYPE  WORK            TRANSACTION USAGE PROFILES        LINE
            ***************************************************
IDENTIFIER  LOAD TRANS INIT  PROFILE 1 PROFILE 2 PROFILE 3    NBER
*****************************************************************
guichetier    T  depot              30                          1
                 retrait            70                          2
*****************************************************************
caissier      T  depot              20                          3
                 retrait            60                          4
                 change             20                          5
*****************************************************************
bureau        T  cheques           100                          6
*****************************************************************


do you want to insert, modify, delete or print the agents ?

Do you want to go to next step (G),
            or rerun same step (R),
            or exit            (E)?
g
```

```
  SITE CLASSES
  ************


**************************************************

  class id ?
centre
      CONFIGURATION : component ?
DPS8
      CONFIGURATION : component ?
DN7100
      CONFIGURATION : component ?
?

  Possible components :
           DPS7
           DPS8
           MINI6
           DN7100

  However computations for MINI 6 are not
  performed. The network task is done  by
  a datanet which is assumed.


  12 alphanumeric characters (max),
  followed by "cr"

  If no more data to be entered, type CR

       CONFIGURATION : component ?
       PROCESSES ON THE SITE : type id ?
cnq
       PROCESSES ON THE SITE : type id ?
cnq
       PROCESSES ON THE SITE : type id ?
       In the class  centre        ,are there agents of
       the same type with different characteristics ?
       type Y or N
n
            number of sites in this class ?
1
            INITIATING AGENTS AT THE SITE
                 type id of agent ?

**************************************************

  class id ?
region
      CONFIGURATION : component ?
DPS7
      CONFIGURATION : component ?
DN7100
      CONFIGURATION : component ?
       PROCESSES ON THE SITE : type id ?
dep
       PROCESSES ON THE SITE : type id ?
ret
       PROCESSES ON THE SITE : type id ?
       In the class  region        ,are there agents of
       the same type with different characteristics ?
       type Y or N
n
            number of sites in this class ?
2
            INITIATING AGENTS AT THE SITE
                 type id of agent ?
guichetier
                 number of agents ?
7
                 intensity ? (transactions per hour)
30
                 procedure used ?
vip
  invalid data : non signed integer required
?


  Possible procedures :

  0 = no procedure
  1 = ASY
  2 = VIP
  3 = MINI 6 / PVE
  4 = RCI (for  66-level only)
  5 = BSC 2780 (for 66-level only)
  6 = BSC 3270 (for 66-level only)
  7 = PAD
  8 = DCU 7010 QUESTAR/T
  9 = FILE TRANSFER

  enter an integer between 0 and 9, corresponding
  to the procedure used

  procedure used ?
2
            INITIATING AGENTS AT THE SITE
                 type id of agent ?
caissier
                 number of agents ?
2
                 intensity ? (transactions per hour)
40
                 procedure used ?
2
            INITIATING AGENTS AT THE SITE
                 type id of agent ?
bureau
                 number of agents ?
1
                 intensity ? (transactions per hour)
10
                 procedure used ?
1
            INITIATING AGENTS AT THE SITE
                 type id of agent ?
```

```
*****************************************************************************

  LINK BETWEEN SITES FOR PROFILE  1

*****************************************************************************
                                                          MESSAGES
TRANSACTION  SENDER-SITE  RECEIV-SITE  MODE  PROPORT    ***************** LINK
                                                         NUMBER     LENGTH
**********************T*****************************************************
depot        region       region        i      100        1          50
                                        i      100        1         600
             agence       region        h      100        1          50    u
             region       agence        h      100        1         600    u
*****************************************************************************
retrait      region       region        i      100        1          25
                                        i      100        1         600
             agence       region        h      100        1          25    u
             region       agence        h      100        1         600    u
*****************************************************************************
change       region       centre        h      100        2          25    u
             centre       region        h      100        1          50    u
                                        h      100        1         200    u
*****************************************************************************
cheques      region       centre        h      100        1         100    u
             centre       region        h      100        1         200    u
*****************************************************************************


  Do you want to redefine ALL the links for profile 1 ?
  type Y or N
n
  Do you want to go to next step (G),
              or rerun same step (R),
              or exit            (E)?
g
```

```
  Do you use default value for X25 packets
  for public network : 128 char ?
  type Y or N
y



*****************************************************************


*************************
*                       *
*  Site :  centre       *
*                       *
*************************


  enter the total number of lines connected to datanet
16


  Primary network
  ***************


  Do you use basic control (1), full or recovery control (2)
   or the multipath option (3) ?
2



  Host connection
  ***************


  Do you know the number of administrative records per second ?
  Type Y or N
y
  Enter record number
2
  Is NAD connected to a host logfile on the same site ?
  Type Y or N
y
```

```
*************************
*                       *
*  Site :  region       *
*                       *
*************************


 enter the total number of lines connected to datanet
12


 Primary network
 ***************


 Do you use basic control (1), full or recovery control (2)
  or the multipath option (3) ?
2


 Secondary network
 *****************


 do you use default value for buffer size : 200 char ?
 type Y or N
n
 buffer size ? (char)
500
 Enter number of special characters used (normal value = 2)
 for agent bureau        using asynchronous protocol
3


 Host connection
 ***************


 Do you know the number of administrative records per second ?
 Type Y or N
y
 Enter record number
1
 Is NAD connected to a host logfile on the same site ?
 Type Y or N
y
```

```
*************************
*                       *
*  Site :  agence       *
*                       *
*************************


 enter the total number of lines connected to datanet
8


 Primary network
 ***************


 Do you use basic control (1), full or recovery control (2)
  or the multipath option (3) ?
2


 Secondary network
 *****************


 do you use default value for buffer size : 200 char ?
 type Y or N
n
 buffer size ? (char)
300
```

```
NETWORK MODEL-NAME  : DEMONS


*****************************************************
*                                                   *
*  RESULTS FOR SITE  :  agence        FOR PROFILE :  1  *
*                                                   *
*****************************************************


Datanet used  :  1  DN7102

CPU load for primary network    :    1 %

CPU load for secondary network  :    2 %

CPU load for host connection    :    0 %

Datanet total CPU load          :    3 %


NETWORK MODEL-NAME  : DEMONS


*****************************************************
*                                                   *
*  RESULTS FOR SITE  :  region        FOR PROFILE :  1  *
*                                                   *
*****************************************************


Datanet used  :  1  DN7102

CPU load for primary network    :    1 %

CPU load for secondary network  :    3 %

CPU load for host connection    :    3 %

Datanet total CPU load          :    7 %


NETWORK MODEL-NAME  : DEMONS
```

```
*****************************************************
*                                                   *
*  RESULTS FOR SITE  :  centre        FOR PROFILE :  1  *
*                                                   *
*****************************************************


Datanet used  :  1  DN7102

CPU load for primary network    :    1 %

CPU load for secondary network  :    0 %

CPU load for host connection    :    6 %

Datanet total CPU load          :    7 %


NETWORK MODEL-NAME  : DEMONS


PHYSICAL LINKS BETWEEN SITES FOR PROFILE :  1

*****************************************************************

Kind of link : PUBLIC X25
Between  region       and  centre        for link supporting
hierarchic (H) traffic, there are :

                 84  messages   per hour
               7800  characters per hour

You need    1 line(s) of 1200 bps.  Utilization rate   2 %

*****************************************************************

Kind of link : PUBLIC X25
Between  agence       and  region        for link supporting
hierarchic (H) traffic, there are :

                288  messages   per hour
              91050  characters per hour

You need    1 line(s) of 1200 bps.  Utilization rate  17 %

*****************************************************************
Do you want to go to next step (G),
            or rerun same step (R),
            or exit            (E)?
```