الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة سعيدة د. مولاي الطاهر

كلية التكنولوجيا

قسم: الإعلام الآلي

# Mémoire de Master

Spécialité : Master 2 Réseaux Informatiques et Systèmes Réparties (RISR)

## Thème

# Vérification et configuration de pare-feu par model checking.

**Présenté par :**

Hocini Zohra

HAMADENE Imène

**Dirigé par :**

Mr. ADJIR NOUREDINNE

Promotion 2021 - 2022

# *Remerciements*

*Avant tout ; nous remercions Dieu de nous avoir aidé à faire notre thème. Nous tenons à adresser nos remerciements à notre encadreur Monsieur ADJIR NOUREDINNE qui nous a aidé à élaborer ce projet.*

*Nos remerciements à tous les membres de jury pour l'honneur qu'ils nous ont fait en acceptant d'examiner ce travail. Nos remerciements à nos enseignants qui ont contribué à notre formation. Nos remerciements à tous ceux qui nous ont aidé de près ou de loin à réaliser ce mémoire qui nous a aider à élaborer ce travail.*

*Merci.*

# *Dédicace*

*Je dédie ce Modest travail :*

*Aux deux personnes les plus chères de ma vie, ma mère et mon père, que Dieu prolonge leur vie.*

*En mémoire de mon oncle et de ma grand-mère, que Dieu les bénisse, qui m'ont toujours poussé et motivé dans mes études.*

*À mes chers frères Ahmed, Youcef, mes sœurs Ahlem et Rahma et mon neveu Adem.*

*À la personne la plus chère à mon cœur Hadj qui m'a soutenu tout mon temps.*

*Et aussi à mes cousins, qui sont mes frères.*

*A mes très chers A mon binôme : Hamadene Imane je les remercie beaucoup de m'avoir aidé*

*À tous les membres de ma famille et toute personne qui porte le nom*

*HOCINI*

*À tous mes amis et à toutes les personnes qui occupent une place dans mon cœur.*

*Hocini Zohra*

*Je dédie ce modeste travail :*

*Aux deux êtres les plus chers à Mon cœur auxquels je dois mon existence :*

*Mon père et ma mère : vous qui étaient toujours à mes*

*Côtés pour me soutenir et m'encourager à me battre sans jamais m'arrête à mi-chemin ; que dieu vous protège.*

*A mes grands-parents.*

*A mes chères mes sœurs : Mokhtaria, Ikram, Meriem*

*A mes frères : Mokhtar, Ben Ahmed, Ben Oumer.*

*A mes oncles et mes tantes.*

*À tous les membres de ma famille et toute personne qui porte le nom*

*Hamadene*

*A mes très chers A mon binôme : HOCINI ZOHRA je les remercie beaucoup de m'avoir aidé*

*A tous mes amis la promotion 2ème année Master RISR*

*2021/2022*

*À tous mes amis et à toutes les personnes qui occupent une place dans mon cœur.*

*Hamadene IMANE*

## ملخص

تلعب جدران الحماية دورا أساسيا في تعزيز السياسات الأمنية لشبكة الإنترنت. ومع ذلك ، فإن تكوينها يتطلب عادة تدخلا يدويا ، وهو مصدر رئيسي للثغرات الأمنية. لذلك ، هناك حاجة إلى حلول تلقائية للكشف عن التناقضات في تكوينات جدار الحماية. في هذه المذكرة ، نقترح طريقة للمساعدة في تكوين جدران الحماية ، استنادا إلى التقنيات الرسمية مثل فحص النماذج. هذا النهج يجعل من الممكن التحقق تلقائيا من تشغيل النظام باستخدام UPPAAL أداة نمذجة النظام والتحقق منه في الوقت الفعلي.



## Abstract

Firewalls play an essential role in strengthening the security policies of the Internet network. However, their configuration usually requires manual intervention, a major source of security vulnerabilities. Therefore, automated solutions are needed to detect inconsistencies in firewall configurations. In this dissertation, we propose approaches to help configure firewalls, based on formal techniques such as model checking. This approach makes it possible to automatically verify the operation of the system using UPPAAL, a real-time system modeling and verification tool.



## Résumé

Les pares-feux jouent un rôle essentiel dans le renforcement des politiques de sécurité du réseau L'Internet. Cependant, leur configuration nécessite généralement une intervention manuelle, Une source majeure de failles de sécurité. Par conséquent, des solutions automatisées sont nécessaires pour détecter les incohérences dans les configurations de pare-feu. Dans ce mémoire, nous proposons des approches d'aide a la configuration des pares-feux, Basé sur des techniques formelles telles que la vérification de modèle. Cette approche permet de vérifier automatiquement le fonctionnement du système en utilisant UPPAAL, un outil de modélisation et de vérification système en temps réel.

# Sommaire

Remerciements…… a
Dédicaces…… b
Résumé…… c
Abréviations…… d
Liste des tableaux …… e
Liste des figures …… f
Introduction Générale…… g

Chapitre I : MECANISMES DE MODELISATION ET VERIFICATION FORMELLES.
I.1 Introduction …… 03
I.2 Model checking …… 03
I.2.1 Définition du model checking …… 03
I.2.2 Extensions du model checking …… 06
I.2.3 Explosion combinatoire …… 06
I.2.4 Choix du model checker…… 07
I.2.5 UPPAAL…… 08
I.2.6 Architecture de UPPAAL …… 10
I.3 Formalismes de modélisation …… 11
I.3.1 Automates temporisés (Timed Automata : TA) …… 11
I.3.2 Définition formelle …… 11
I.3.3 Problème de l'atteignabilité …… 12
I.4 Formalisme de spécification …… 12
I.4.1 La logique temporelle arborescente CTL …… 12
I.4.2 la Logique Temporelle Linéaire LTL…… 15
I.5 Les Avantages et Les Inconvenant De Model Checking …… 16
I.6 Conclusion …… 17

Chapitre II : SECURITE DES RESEAUX INFORMATIQUES.
II.1 Introduction…… 18
II.2 Concepts de base…… 18
II.2.1 Les types de réseau…… 19
II.2.2 Architecture d'un réseau …… 20
II.2.3 Menaces réseau …… 26
II.2.4 Politique de sécurité …… 27

II.3 Systèmes de détection d'intrusions …… 28
II.3.1 Les types des systèmes de détection d'intrusion …… 29
II.3.2 Caractéristiques d'un système de détection d'intrusion…… 31
II.3.3 L'architecture d'un IDS…… 31
II.3.4 Les méthodes de détections…… 33
II.4 Pare feu (firewall) …… 35
II.4.1 Les type de Pare-feu…… 35

# Sommaire


II.4.2 Emplacement d'un Pare-feu…… 36
II.4.3 Fonctions d'un Pare feu …… 37
II.4.4 Différences clés entre le Firewall et le Serveur Prox..... 41
II.4.5 Les limites des firewalls…… 43

II.5 Renforcement des politiques sur les réseaux informatiques …… 43
II.5.1 Approche algébrique…… 44
II.5.2 Autres approches …… 46

II.6 Conclusion…… 49

CHAPITRE III : MODELISATION Et VERIFICATION FORMELLES D'UN RESEAU INFORMATIQUE A PARE-FEUX
III.1 Introduction. …… 50
III.2 Représentation graphique d'un réseau a pare-feu …… 50
III.3 Modélisation du comportement …… 52
III.3.1 Automate "Nœud générateur"…… 52
III.3.2 Automate " firewall "…… 54
III.3.3 Automate "network"…… 55
III.3.4 Automate ''Host " …… 56
III.3.5 la simulation …… 57

III.4 Vérification formelle de la cohérence des pares-feux…… 57
III. 4.1 Propriétés assurant le bon fonctionnement du modèle…… 58
III.4.2 Cohérence de la politique de sécurité globale du réseau…… 59
III.4.3 Propriété de vérification …… 59
III.5. Conclusion …… 60
Conclusion générale …… 61
Référence…… 62

# Liste des Abréviations

**BDD :** Binary Decision Diagram (Les diagrammes de décision binaires)

**APNBDD**: Automatic Petri Net Binary Decision Diagram**.**

**AT :** Timed Automata (Automates Temporisés).

**BMC:** Bounded model checking (Modele checking borné).

**MCS:** Model checking symbolique (Vérification symbolique du modèle).

**TCTL :** Timed Computation Tree Logic.

**CTL :** computation tree logic (La logique temporelle arborescente)

**LTL :** linear temporal logic (la Logique Temporelle Linéaire)

**GUI :** Graphical User Interface (l'interface graphique utilisateur).

**LT:** Logique Temporelle.

**W:** weak until.

**G**: globally(globalement).

**F**: the **f**uture (le **f**uture).

**R:** ***r****elease* .

**IDS :** Intrusion Detection System (les systèmes de détection d'intrusion).

**TCP**: *Transmission Control Protocol*.

**PAN :** Personal Area Network (réseau personnel).

**LAN:** Local Area Network (réseau local)**.**

**MAN :** Metropolitan Area Network (réseau métropolitain).

**WAN:** Wide Area Network (réseau étendu).

**GAN:** Global Area Network (réseau global)**.**

**VPN :** Virtuel Private Network (réseau privé virtuel)

**PDU :** Protocol Data Unit (Unité de données de protocole).

# Liste des Abréviations

**Data :** champ de données.

**UDP :** User Datagram Protocol.

**IP :** Internet Protocol.

**OSI:** Open Systems Interconnection**.**

**IHL :** Internet Header Length (Longueur de l'en-tête Internet) **.**

**TOS:** Type of Service.

**MPLS:** Multi-Protocol Label Switching**.**

**NIDS:** Network-based Intrusion Detection System**.**

**HIDS:** Host-based Intrusion Detection System**.**

**FTP :** File Transfer Protocol (Protocole de transfert de fichiers).

**Http :** HyperText Transfer Protocol**.**

**ICMP :** Internet Control Message Protocol (Protocole de message de contrôle sur Internet)

**CPU:** Central Processing Unit.

**Dos:** Disk Operating System**.**

**SSH:** Secure Shell.

**CERT:** Computer Emergency Response Team.

**CCS :** Calculus of Communicating Systems

**CMN:** Calculus for Monitored Network**.**

**LM :** logique propositionnelle

**Firmato :** Firewall Management Toolkit.

**MLD** : Le langage de définition de modèle.

**FDD :** firewall decision Diagram (Diagramme de décision d'un pare-feu).

**HYTECH** : Documentation technique hypertextuelle et hypermédial**.**

**KRONOS:** Kintetsu Rapid Operated New Original System

# Liste des Abréviations

**Ip_src :** l'adresse IP source

**Ip_dest :** l'adresse IP destination

# Liste des Tableaux

**Chapitre I :** MECANISMES DE MODELISATION ET VERIFICATION FORMELLES.

**Chapitre II** : SECURITE DES RESEAUX INFORMATIQUES.

Tableau II.1 : des exemples de règles de pare-feu.....39

Tableau II.2 : Un pare-feu consistant f.....48

**CHAPITRE III :** MODELISATION Et VERIFICATION FORMELLES D'UN RESEAU INFORMATIQUE A PARE-FEUX

Tableau III .1 : Structure d'un paquet.....54

Tableau III.3: Propriétés vérifiant le bon fonctionnement du modèle.....58

# Liste des figures

**Chapitre I** : MECANISMES DE MODELISATION ET VERIFICATION FORMELLES.

Figure I.1 : Principe du model checking ..... 4
Figure I.2 : Procédure du Model Checking ..... 5
Figure I.3 : Comparaison des performances de UPPAAL, HYTECH et KRONOS ..... 7
Figure I.4: Interface d'édition de modèle ..... 9
Figure 1.5 Illustration de certaines formules CTL. ..... 14
Figure 1.6 la Logique Temporelle Linéaire. ..... 16

**Chapitre II** : SECURITE DES RESEAUX INFORMATIQUES.

Figure II.1 **:** Les types de réseau. ..... 20
Figure II.2 **:** la topologie en bus ..... 21
Figure II.3 la topologie en étoile ..... 21
Figure II.4: la topologie en anneau ..... 22
Figure II.5: la topologie en maillée ..... 22
Figure II.6 : Structure d'un paquet. ..... 23
Figure II.7 **:** Système de détection d'intrusion réseau. ..... 29
Figure II.8 : Système de détection d'intrusion hôte. ..... 30
Figure II.9 : L'architecture d'un IDS. ..... 32
Figure II.10 **:** Emplacement d'un Pare-feu a la frontière **.** ..... 36
Figure II.11 : Emplacement d'un Pare-feu au centre d'un réseau**.** ..... 37
Figure II.12**:** Fonctionnement d'un Pare feu. ..... 37
Figure II.13: serveur proxy ..... 41
Figure II.14: Firewall. ..... 42
Figure II.15 Approche algébrique pour la sécurisation des réseaux informatiques. ..... 45
Figure II.16 Opérateur du monitorage sélectif ..... 45
Figure II.17 Diagramme de décision du pare-feu f ..... 49

**CHAPITRE III** : MODELISATION Et VERIFICATION FORMELLES D'UN RESEAU INFORMATIQUE A PARE-FEUX

Figure III.1: Représentation graphique d'un réseau ..... 51

# Liste des figures

Figure III.2: Automate générateur. ..... 52
Figure III.3: Automate générateur en UPPAAl ..... 53
Figure III.4: la simulation d'automate générateur. ..... 53
Figure III.5 : Automate firewall ..... 54
Figure III.5 : Automate network ..... 55
Figure III.5 : Automate Host ..... 56
Figure III.7 : Exemple de simulation du modele ..... 57

# INTRODUCTION GENERALE

La sécurité des réseaux informatiques est devenue un problème majeur dans la société moderne. En effet les réseaux informatiques intrinsèquement vulnérables sont indispensables à toutes les activités entrepreneuriales et Tous horizons : Banques Assurances Hôpitaux etc. Au cours de la dernière décennie nous avons assisté à l'essor fulgurant d'Internet qui a entièrement connecté notre monde numérique permettant Accéder à une mine d'informations disponibles sur le web. Cependant cette ouverture sur l'extérieur a ouvert des portes à ceux qui prennent plaisir à piller les systèmes informatiques pour infiltrer les réseaux. Et procéder à la destruction malveillante ou au vol d'informations confidentielles.

Il est donc primordial de mettre en place des barrières pour protéger les données critiques circulant sur les réseaux privés. Fais ça Plusieurs solutions ont été déployées telles que : l'authentification le chiffrement l'antivirus les systèmes de détection d'intrusion (IDS) et les pares-feux.

La solution la plus largement adoptée est un pare-feu. En effet cet outil Constitue la pierre angulaire de la sécurisation des connexions entre les réseaux privés et publics. La fonction principale du pare-feu est de contrôler le flux d'informations au sein du réseau. Le pare-feu est Il se caractérise par une configuration basée sur des règles qui décide de rejeter ou d'accepter des paquets circulant sur le réseau en fonction de ses attributs (par exemple port de protocole source-destination).

Ce mémoire est structuré comme suit :

- Un premier chapitre Engagé dans la modélisation et la validation des techniques utilisées dans Le cadre de ce mémoire, notamment la technique de model-checking ;
- Un deuxième chapitre présente les réseaux informatiques, leurs différentes architectures, leurs composants ainsi que leurs mécanismes de sécurisation les plus répandus : les IDS et les pares-feux ;
- Un troisième chapitre montre comment modéliser et vérifier par model checking, la cohérence des pares-feux d'un réseau, par rapport à des objectifs de sécurité ;
- Enfin nous terminerons notre travail par une conclusion générale.

# I.1.Introduction :

Le processus de conception du système passe essentiellement par trois étapes à savoir : la conception le développement et la vérification. La dernière étape est très importante car elle détecte les erreurs et assure systèmes avant qu'ils ne soient exploités. En effet il permet la détection d'erreurs à moindre coût. Les méthodes formelles sont fortement recommandées pour le développement de systèmes logiciels critiques pour la Sécurité.

Ce chapitre porte sur une étude bibliographique sur le model checking. La section 1.2 d´écrit le modèle checking (1. Définition du model checking, 2. Extensions du model checking, 3. Explosion combinatoire, 4. Choix du model checker, 5. Les types d'UPPAAL). La section 1.3 présente les Formalismes de modélisation les plus utilisés en entrée des model checkers à savoir les automates temporises, les automates temporises à cout et les automates temporises de jeux. La section 1.4 présente les logiques temporelles qui permettent de spécifier les propriétés attendues d'un système. La section 1.5 Les Avantages et les Inconvenant De Model Checking. 1.6 Conclusion.

# I.2. Model checking (vérification du modèle) :

## I.2.1 Définition du model checking :

Le Model Checking désigne une famille de techniques de vérification automatique des systèmes dynamiques (souvent d'origine informatique ou électronique). Il s'agit de vérifier algorithmiquement si un modèle donné, le système lui-même ou une abstraction du système, satisfait une spécification, souvent formulée en termes de logique temporelle.

On peut distinguer deux aspects du model checking :

Il peut s'agir de démontrer qu'une certaine classe de propriétés, ou une certaine logique, est décidable, ou que sa décision appartient à une certaine classe de complexité.

Il peut s'agir de rechercher des algorithmes efficaces sur des cas intéressants en pratique, de les implémenter, et de les appliquer à des problèmes réels.

Les premiers travaux sur le model checking de formules de logique temporelle ont été menés par Edmund M. Clarke et E. Allen Emerson en 1981, ainsi que par Jean-Pierre Queille et Joseph Sifakis en 1982. Clarke, Emerson et Sifakis se sont vu attribuer le Prix Turing 2007 pour leurs travaux sur le model checking.

(Techno-science) (figure **I.**1)

(GUEFFAZ, M. (2012))

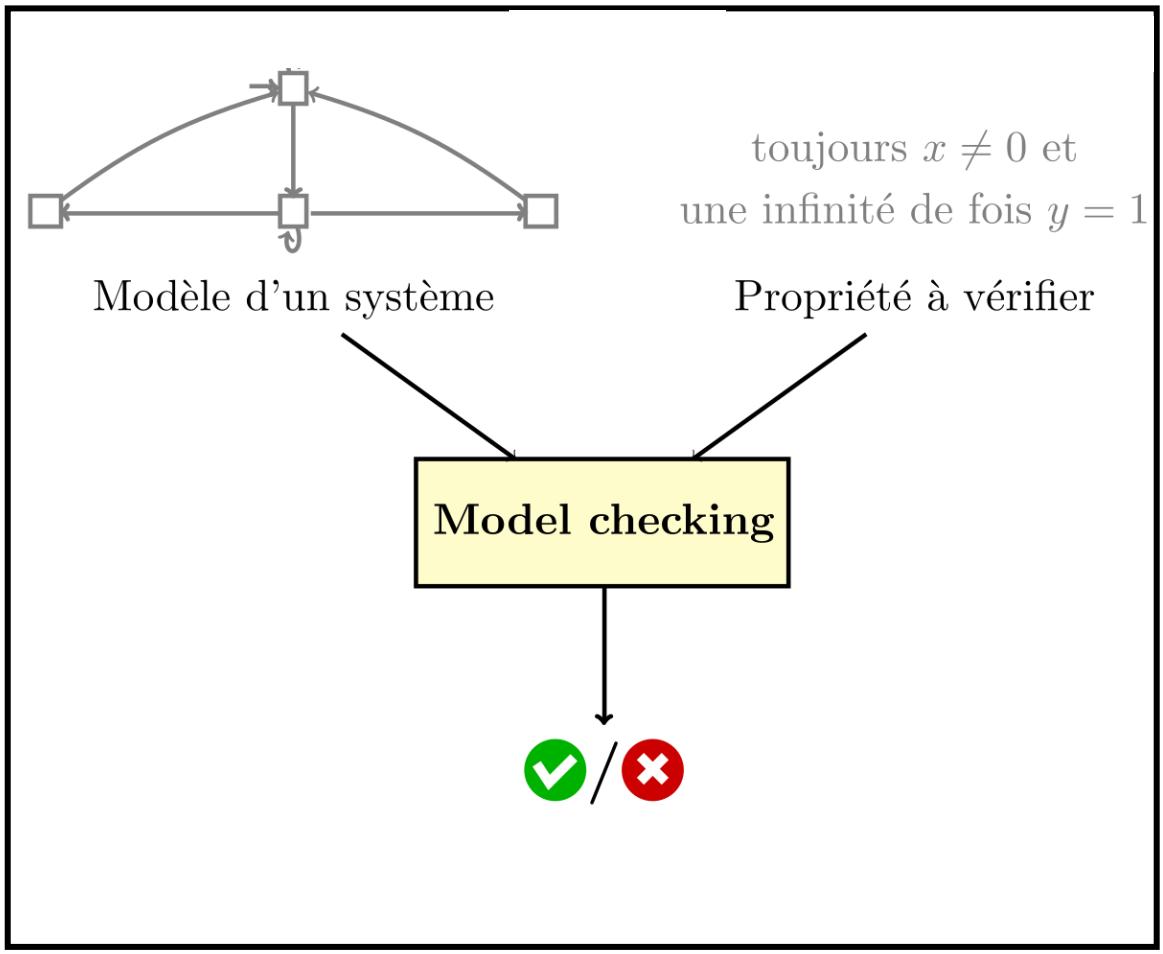



**Figure I.1:** Principe du *model checking*.

En informatique, la vérification de modèles, ou model checking en anglais, est le problème suivant : vérifier si le modèle d'un système (souvent informatique ou électronique) satisfait une propriété. Par exemple, on souhaite vérifier qu'un programme ne se bloque pas, qu'une variable n'est jamais nulle, etc. Généralement, la propriété est écrite dans un langage, souvent en logique temporelle. La vérification est généralement faite de manière automatique. Sur le plan pratique, la vérification de modèles est devenue, au niveau industriel, la méthode de vérification de code et de systèmes matériels la plus populaire et la plus utilisée aujourd'hui

(Wikipédia)

La vérification de tout système en utilisant la technique de model checking passe par trois étapes comme l'explique la Figure **I** .2 :

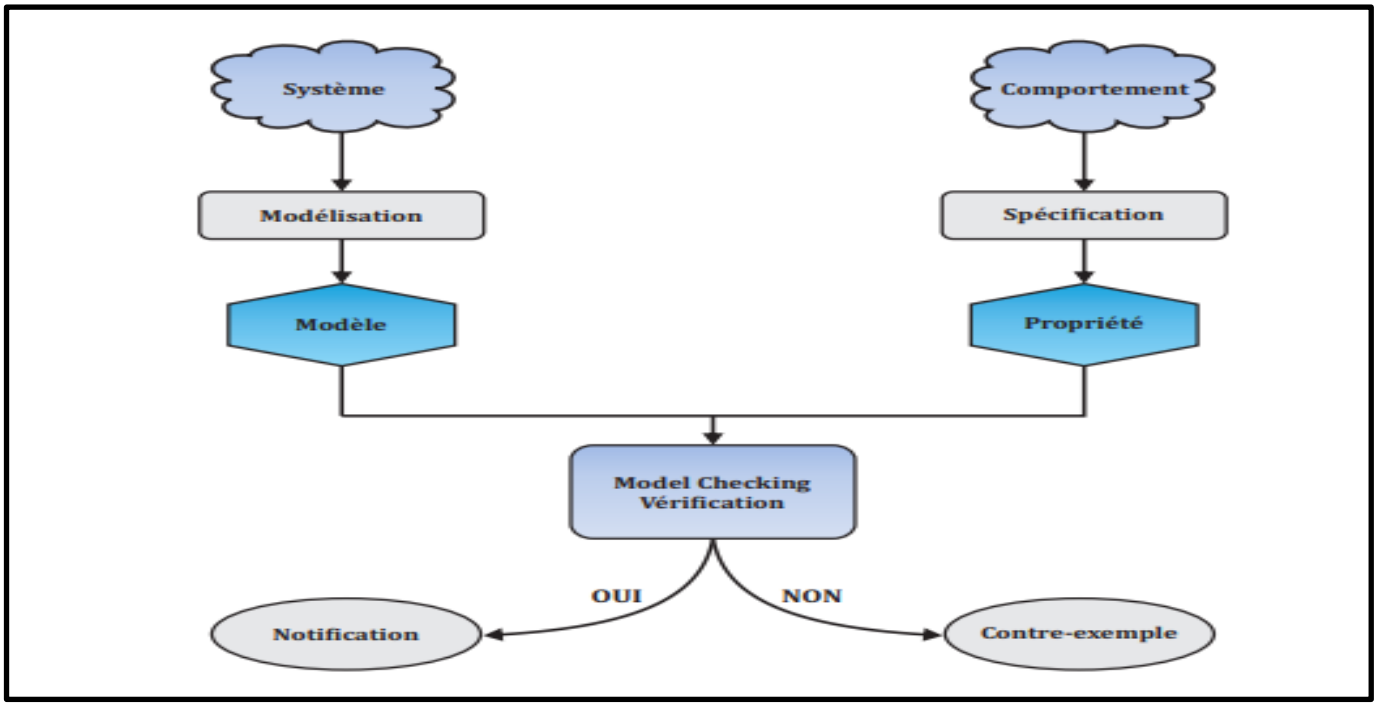



**Figure I.2 :** Procédure du Model Checking

**Modélisation :** Cette étape consiste à générer l'abstraction du système sous la forme d'un modèle. Le modèle doit être suffisamment expressif pour pouvoir décrire sans ambiguïté la spécificité du Système 6. Il s'agit de transformer des systèmes sous la forme de structures de données finies qui peuvent être expliquées par des algorithmes de vérification de modèles. Parmi ces structures de données, nous citons : Automatic Petri Net Binary Decision Diagram (BDD), etc.

**Spécification :** cette étape traduit les exigences (propriétés attendues) Système dans le Formalisme. Il existe différents types d'exigences : Sécurité, vivacité, accessibilité, équité et non blocage, etc.

**Vérification :** Cette étape consiste à mettre en œuvre un algorithme qui prend en entrée le modèle (le résultat de l'étape de modélisation) et la spécification (le résultat de l'étape de spécification). Le vérificateur de modèle renvoie "true" si la spécification est vérifiée et fournit des contre-exemples (traces d'erreurs), le cas échéant. La validation du modèle est basée sur l'exploration de l'espace d'états, donc je pourrais avoir un problème d'explosion combinatoire. Dans ce cas, il est nécessaire de revenir à l'étape initiale (modélisation et validation) pour réviser le modèle et la spécification pour réduire le modèle et surmonter ce problème. Pour pallier cette explosion des combinaisons d'états, plusieurs techniques de vérification ont été proposées. Les sections suivantes traitent de ce problème et proposent des solutions pour le résoudre

### I.2.2 Extensions du model checking :

Pour s'attaquer aux systèmes infinis - Sous classes décidables

• automates temporisés [groupe TEMPORISÉ],

• systèmes bien structurés [groupe INFINI]

- Mais on ne peut traiter automatiquement la majorité des modèles intéressants

• abstractions finies (modélisation difficile, faux négatifs, etc.)

• semi-algorithmes, qui fonctionnent bien en PRATIQUE [groupe INFINI]

### I.2.3 Explosion combinatoire :

La principale limitation du model checking est le problème d'explosion combinatoire des états (Clarke et al., (2012)). De nombreuses recherches ont été conduites pour trouver une solution à ce problème. Ce volet présente quelques solutions (Gueffaz, (2012)) :

**. L'abstraction du modèle** : peut se faire en ignorant quelques états du modelé, ce qui réduit le nombre de transitions et par conséquent la taille du modèle. Il y a aussi l'abstraction par « fusion d'états ». Cette méthode consiste à regrouper un ensemble d'états ayant des caractéristiques communes en un seul état.

**. Les diagrammes de décision binaires** : sont très utiles dans la vérification formelle car ils permettent de

Représenter le comportement d'un système sous forme d'une formule ensembliste.

**. Model checking symbolique (Vérification symbolique du modèle) :** Cette approche comprend Déterminer l'ensemble des états accessibles et les points fixes de l'ensemble des états satisfaire les propriétés données. Un ensemble d'états donné.

**. Vérification à la volée :** du système et simultanément à vérifier la spécification sur cet espace d'états. ii) y compris les calculs prédécesseurs de l'état cible et vérifiez si l'état initial se trouve dans l'ensemble de calcul.

**. Modelé checking borné (Bounded model checking : BMC) :** Un modèle de vérification borné est une solution proposée pour atténuer le problème d'explosion combinatoire. En d'autres termes BMC consiste à trouver des contre-exemples de longueur k. La vérification du modèle limité ne permet pas la validation des propriétés ne pas.

**. Technique d'ordre partiel :** Cette technique peut être considérée comme une abstraction par restriction**.**

### I.2.4 Choix du model checker :

Dans ce mémoire une attention particulière est accordée aux modèles de validation quantitative. Leur analyse est basée sur des recherches sur les protocoles d'exclusion mutuelle de Fisher. En effet d'après la Figure 1.3 UPPAAl prend en charge la validation du modèle de recherche pour un maximum de 8 processus ce qui échoue avec HYTECH et KRONOS après 5 et 4 processus respectivement.

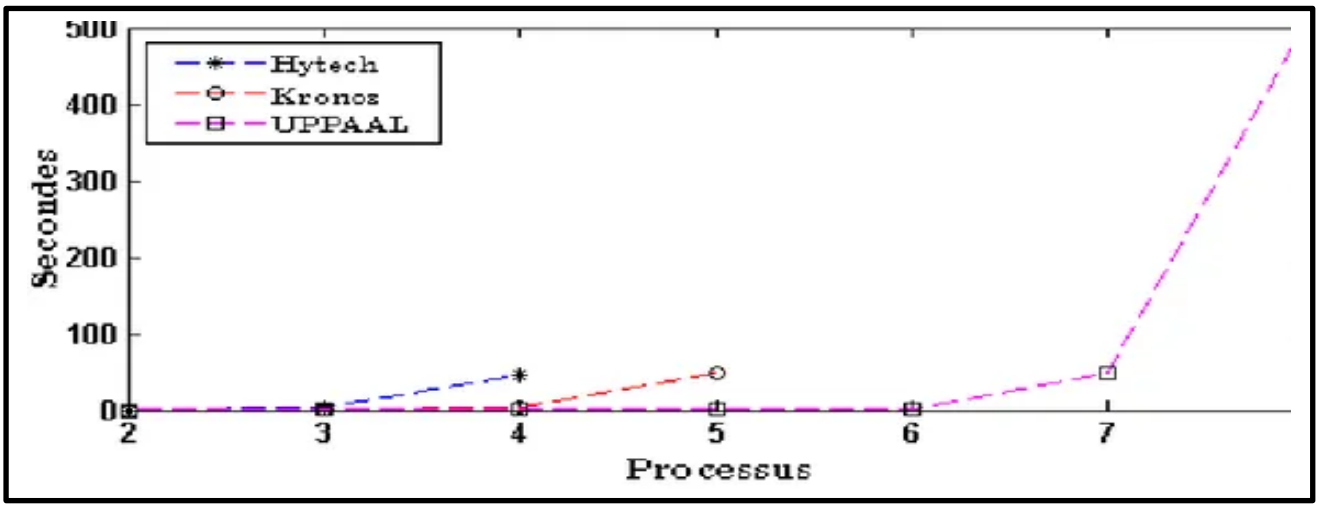



**Figure I.3 :** Comparaison des performances de UPPAAL, HYTECH et KRONOS

**Comparaison des performances de UPPAAL, HYTECH et KRONOS :**

**UPPAAl :**

Est, aujourd'hui, l'outil d'analyse de systèmes temporisés le plus abouti : il est régulièrement mis à jour et une interface graphique conviviale assure une prise en main aisée du logiciel. Les modèles manipulés par l'outil sont une variante équivalente du modèle des automates temporisés. De nombreuses études de cas sont présentées dans la littérature. En particulier, UPPAAl a été utilisé avec succès pour analyser un protocole audio/vidéo de chez Bang & Olufsen. La présence D'un bug dans le protocole était connue mais les concepteurs ne réussissaient pas à déterminer l'origine du problème. UPPAAl a permis d'une part de mettre en évidence le bug mais également de montrer qu'une version modifiée du protocole était correcte. Les auteurs modélisent un bus de terrain (destiné à un réseau de contrôle dans une industrie) : des erreurs de conception et d'implémentation furent découvertes et des améliorations ont été proposées. Un

contrôleur de boîte de vitesses de voiture a été également analysé avec succès à l'aide de cet outil. (Philipe schnocbelen, 2000)

**Hytech :**

Permet de manier des modèles plus généraux que les automates temporisés. Ainsi, le model checker Hytech permet d'analyser des systèmes décrits comme des automates hybrides linéaires Il s'agit d'un des rares outils permettant de manier des modèles aussi expressifs. En contrepas tie, la terminaison de l'analyse n'est pas assurée et la complexité des algorithmes utilisés est plus importante que pour les outils décrits précédemment. La maintenance d'Hytech n'est plus assurée par son équipe de développement et son interface utilisateur, en mode texte, n'est pas particulièrement conviviale. (Philipe schnoebelen, 2000)

**Kronos :**

Est un model-cheker se la logique TCTL. II décide si un automate (appelé graphe) temporisé, donné sous forme textuelle, vérifier une propriété exprimée par une formule TCTL. A partir d'un système à plusieurs automates composants, Kronos calcule l'automate correspondant au produit synchronisé. (Philipe schnoebelen, 2000).

### I.2.5 UPPAAL :

UPPAAL est un outil disposant d'une interface graphique pour : définir des réseaux d'automates temporisés, simuler leur comportement, et vérifier des propriétés CTL sur le réseau d'automate)

UPPAAL est un outil de vérification de systèmes temps réel

Développé par Uppsala University et Aalborg University Composé de deux parties

- Une interface graphique écrite en JAVA
- Un moteur de vérification de modèles écrit en C++
- Peut être exécuté sur la même machine que l'interface ou sur un serveur puissant

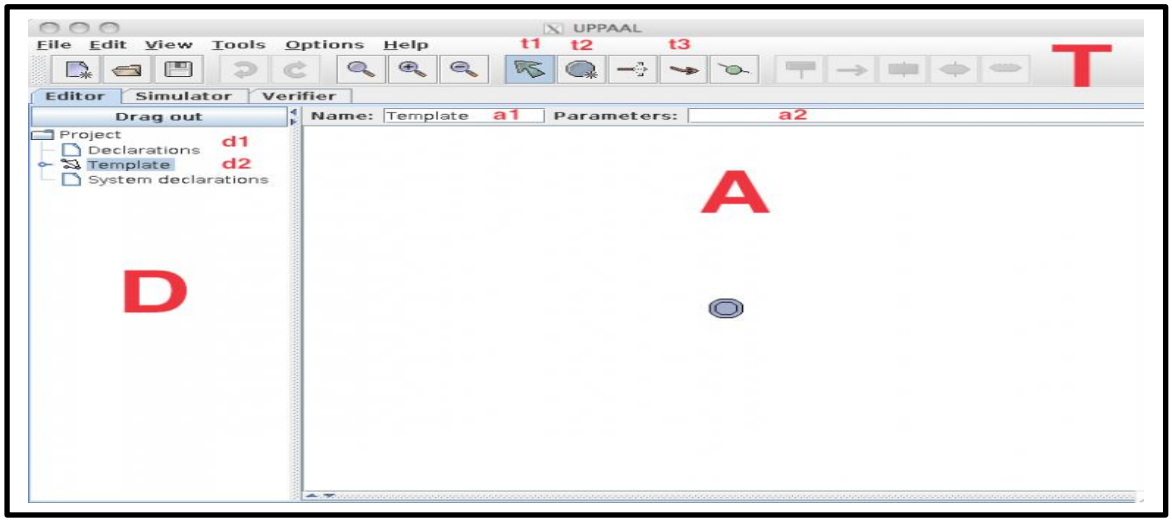



Figure I.4: Interface d'édition de modèle

La figure 1 détaille le rôle de chaque zone de l'interface graphique.

Les déclarations (D) : les variables, types, constantes et événements, et canaux de synchronisation (type chan, comme par exemple l'événement push) sont déclarés dans la section Déclarations du système (marqueur d1) si la déclaration est globale, de déclaration de l'automate sinon (marqueur d2 – vous devez développer le modèle en cliquant sur la puce à gauche du nom de l'automate pour accéder à sa déclaration).

- Edition d'automate (A) : il est possible de définir des automates graphiquement. Les transitions peuvent être complétées par : des gardes, des « synchronisation », des mises à jour de variables. La convention de couleur donne la nature de chaque expression : vert = garde, bleu clair = synchro, bleu foncé = mise à jour.

- La barre d'icône : elle rassemble essentiellement les icônes servant à l'édition graphique du modèle. En particulier, vous y trouverez des outils qui permettent de sélectionner des parties de l'automate (t1) pour les déplacer, les copier / supprimer, d'ajouter des états (t2), ou des transitions (t3). Par défaut, l'outil de sélection (t1) est activé. Notez qu'une fois activée, les autres outils restent actifs (par exemple l'outil d'ajouts d'états (t2) va rester actif après avoir été sélectionné). Il faut cliquer sur l'outil de sélection (t1) pour réactiver cet outil…

( *Fièrement propulsé par WordPress)*

### I.2.6 Architecture de UPPAAL :

L'outil est séparé en deux composants : l'interface graphique utilisateur (GUI en anglais) - le client et le moteur de model-checking le serveur. Le GUI est écrit en Java est déployé facilement sur différentes plateformes alors que le moteur est recompilé pour chaque plateforme. Nous nous référons ici à l'architecture du moteur la partie critique pour les performances de l'outil. Les structures de données du moteur sont conçues autours d'une architecture centrée sur un flot de données qui forme un pipeline. Les données qui circulent à travers les composants de filtre sont des états ou des états ainsi qu'une transition. (Behrmann et al., 2006)

Les différents composants sont les sources où les états crées (typiquement pour l'état initial), les drains où les états disparaissent (typiquement pour évaluer une expression), les tampons où les états sont poussés et enlevés et les filtres où les états sont poussés et expédiés aux composants suivants. De plus, une pompe montre où la boucle principale d'atteignable ou de vivacité est exécutée. Les bénéfices d'utiliser des composants de pipeline sont la flexibilité, la réutilisation de code et l'efficacité. La flexibilité vient de la possibilité d'échanger un composant pour un autre pour configurer le pipeline selon les options, typiquement pour utiliser différentes structures de stockage qui implémentent une exploration exacte, une sous approximation ou une sur-approximation. Une telle configuration dynamique nous permet de sauter complètement des étapes du pipeline si elles ne sont pas nécessaires comme par exemple les traces. La réutilisation de code vient de la réutilisation des composants à travers les différents pipelines d'atteignable ou de vivacité. (Behrmann et al., 2006) Les composants communs qui sont utilisés dans les pipelines d'atteignabilité ou de vivacité sont les suivants :

- transition qui calcule quelles transitions peuvent être prises à partir des états entrants, - successeur qui prend les transitions,

- délai qui calcule le délai possible dans un état borné par son invariant,

- extrapolation + réduction des horloges actives qui applique l'extrapolation en fonction des constantes maximales des horloges du modèle en même temps que la réduction des horloges actives. En fait, donner localement-00 pour la constante maximale d'une horloge a pour effet de libérer les contraintes de cette horloge avec l'algorithme d'extrapolation. (Larsen et al., 1997)

## I.3 Formalismes de modélisation :

Généralement, le model checking utilise une logique temporelle afin de sp´ecifier les propriétés d´écrivant les requis du système ´étudie. Une logique temporelle est un langage de spécification qui permet d'exprimer des propriétés faisant intervenir la notion d'ordonnancement dans le temps. Une logique temporelle permet d'exprimer des propriétés de type « après une instruction i un ´événement j se produira » en utilisant des opérateurs temporels. La logique temporelle linéaire permet de d´écrire le comportement des systèmes qui ´évoluent linéairement avec le temps.

En effet, LTL permet d'exprimer des propriétés portant sur une séquence d'´états donnée appelée exécution. Dans ce mémoire, nous nous intéressons a la logique temporelle CTL, étant la logique de base supportée par tous les outils que nous avons utilisés.

### I.3.1 Automates temporisés (Timed Automata : TA)

En théorie des automates, un automate temporisé est un automate fini doté d'un ensemble fini d'horloges à valeurs réelles. Au cours d'un calcul de l'automate, les valeurs des horloges augmentent toutes à la même vitesse. Dans les transitions de l'automate, les valeurs d'horloges sont comparées à des entiers. Ces comparaisons constituent des *gardes* qui activent ou inhibent les transitions et imposent ainsi des contraintes aux comportements de l'automate. Les horloges peuvent être réinitialisées. Les automates temporisés ont été introduits par Alur et Dill en 1994[1]. Les auteurs ont reçu, pour cet article, le prix Alonzo Church 2016 de l'European Association for Theoretical Computer Science.

Les automates temporisées peuvent être utilisés pour modéliser et analyser le comportement temporel de systèmes informatiques, par exemple, des systèmes ou des réseaux opérant en temps réel. Des méthodes pour vérifier les propriétés de sûreté et de vivacité ont été développées et étudiées depuis l'introduction des automates temporisées en 1994.

Un automate temporisé accepte des mots temporisés — des suites infinies où une valeur réelle de moment d'occurrence est associée à chaque symbole**.**

### I.3.2 Définition formelle :

Un automate temporisé est A = (Σ, S, S0, X, I, T) avec 1. Σ un alphabet fini. 2. S un ensemble fini de places (locations). 3. S0 ⊂ S un ensemble de places initiales. 4. X un ensemble fini de variables (horloges). 5. I : S → C (X) des invariants de places. 6. T ⊂ S × Σ × C (X) × 2 X × S un ensemble de transitions d'actions. Chaque

e =< s, a, φ, λ, s ’ > ∈ T correspond à une transition entre la place s et la place s ’, gardée par la contrainte φ, étiquetée par la lettre a, et qui remet les variables de λ ⊂ X à 0.

### I.3.3 Problème de l’atteignabilité :

Dans la suite on s’intéresse aux problèmes suivants :

- Problème de l’atteignabilité :

1. Données : (a) A un automate temporisé, (b) Q f ⊂ Q un sous-ensemble d’états.

2. Réponse : Oui s’il existe une trajectoire de A qui atteint Q f, Non sinon.

- Problème du calcul des états atteignables :

1. Données : (a) A un automate temporisé.

2. Réponse : l’ensemble Q R des états atteints par les trajectoires**.**

## I.4 Formalisme de spécification :

En règle générale la vérification de modèles utilise une logique temporelle pour spécifier les propriétés qui décrivent les exigences du système à l'étude. La logique temporelle est un langage de spécification qui exprime Propriétés impliquant le concept de planification juste à temps. La logique temporelle peut exprimer des propriétés de type "l'événement j se produira après l'instruction i" en utilisant des opérateurs temporels. La logique temporelle linéaire permet de décrire le comportement du système qui évolue linéairement dans Le temps.
En fait LTL permet d'exprimer des propriétés associées à une séquence donnée d'états appelés exécutions. Dans cet article la logique temporelle CTL qui nous intéresse est donnée par Tous les outils que nous utilisons.

### I.4.1 La logique temporelle arborescente CTL :

En informatique théorique, notamment en vérification formelle, **CTL*** (prononcé *CTL star* en anglais) est une logique temporelle. C'est une généralisation de la logique du temps arborescent **(en)** (CTL : *computation tree logic*) et de la logique temporelle linéaire (LTL : *linear temporal logic*). Elle combine librement les

quantificateurs sur chemins et les opérateurs temporels. CTL* est de ce fait une logique arborescente. La sémantique des formules CTL* repose sur une structure de Kripke.

---

**NB :**

Une **structure de Kripke** est un modèle de calcul, proche d'un automate fini non déterministe, inventé par Saul Kripke. Elle est utilisée par exemple dans le *model checking* pour représenter le comportement d'un système. C'est un graphe orienté dont les nœuds représentent les états accessibles du système et dont les arcs représentent les transitions entre les états. Une fonction d'étiquetage fait correspondre à chaque état un ensemble de propositions logiques vraies dans cet état. Les logiques temporelles sont généralement interprétées dans des structures de Kripke. L'existence de certains chemins dans le graphe est alors considérée comme une éventualité de réalisation de formules.

---

Les formules CTL sont exprimées à l'aide du formalisme suivant :

• Les propositions atomiques : p, q, . . ., true, false qui s'interprètent directement sur un état.

• Les connecteurs booléens : ¬ (négation), ∧ (et logique), ∨ (ou logique), ⇔ (´équivalence), ⇒ (implication).

• Les connecteurs temporels : EF p | EG p | E p U q | EX p | AF p | AG p | A p U q | AX p avec E et A sont des quantificateurs de chemins.

La syntaxe d'une formule CTL est alors la suivante :

ϕ, ψ : := p | q | ¬ϕ | ∧ψ | ϕ ∨ ψ | ϕ ⇔ ψ | ϕ ⇒ ψ | EF ϕ | EG ϕ | E ϕ U ψ | EX ϕ | AF ϕ | AG ϕ | A ϕ U ψ | AX ϕ avec ϕ et ψ sont deux formules CTL.

Les formules CTL peuvent être interprétées comme suit :

• EF ϕ : Il existe une séquence d'états (operateur E) qui mène à un état où la propriété ϕ est vérifiée (operateur F).

• AF ϕ : Pour toute séquence d'états (opérateur A), il existe un état où la propriété ϕ est vérifiée (opérateur F).

• EG ϕ : Il existe une séquence d'états (opérateur E) où la propriété ϕ est toujours vérifiée (opérateur G).

• AG ϕ : Pour toute séquence d'´états (opérateur A), la propriété ϕ est toujours vérifiée (opérateur G).

• AX ϕ : La propriété ϕ est vérifiée sur tous les états (opérateur A) immédiatement successeurs de l'état courant (opérateur X).

• EX ϕ : Il existe une s´séquence d'états (opérateur E) où il existe un état dont l'état successeur (opérateur X) satisfait la propriété ϕ.

• A ϕ U ψ : Pour toute s´séquence d'états (opérateur A) la propriété ϕ est vérifiée jusqu'à ce que la propriété ψ soit satisfaite.

• E ϕ U ψ : Il existe une séquence d'états (opérateur E) sur laquelle la propriété ϕ est vérifiée jusqu'à ce que la propriété ψ soit satisfaite.

Certaines formules CTL sont illustrées par la Figure

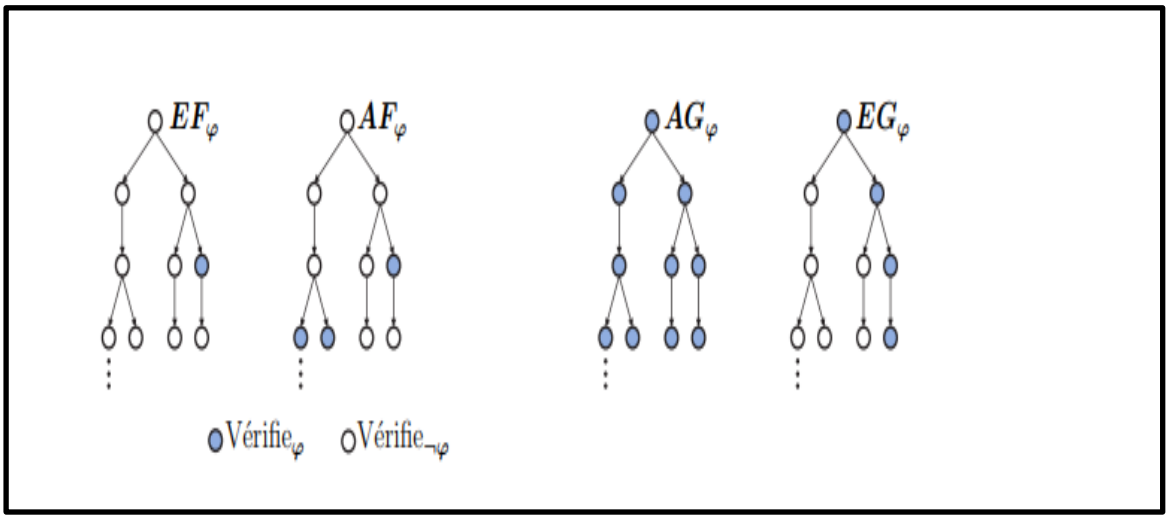



**Figure I.5** Illustration de certaines formules CTL**.**

Formellement, CTL est interprétée sur les états d'un système de transitions (qui caractérise les arbres issus de ces états).

**Définition 1 (Système de transitions étiqueté S** (voir Arnold, 1990)) **:**

Un système de transitions S est un tuple (Q, q0, A, →, L) tel que :

• Q est un ensemble (fini) d'états,

• q0 ∈ Q est l'Etat initial,

• A est un alphabet (fini) d'actions,

• →⊆ Q × A × Q est la relation de transition de S. On note $q \xrightarrow{a} q' \text{ si } (q, a, q') \in \rightarrow$ ,

• $L : Q \rightarrow 2^{AP}$ où AP est un ensemble de propositions atomiques. L(q) donne l'ensemble des propositions que satisfait l'état q.

**Définition 2 (Chemin dans un système de transitions)** (voir Cassez, 2003) :

Un chemin dans un système de transitions est une séquence d'états σ = q0, q1, q2, ... telle que (qi, qi+1) ∈→ pour tout i > 0.

**Définition 3 (Sémantique de CTL)** (voir Cassez, 2003) : Soit q un état de S. La sémantique de CTL est définie inductivement par :

• ϕ ∈ AP, q |= ϕ ⇐⇒ p ∈ L(q),

• q |= ϕ V ϕ 0 ⇐⇒ q |= ϕ ou q |= ϕ 0,

• q |= ¬ϕ ⇐⇒ q 6|= ϕ,

• q |= AX ϕ ⇐⇒ ∀ q 0 tel que (q, q 0) ∈→, q 0 |= ϕ,

• q |= EX ϕ ⇐⇒ ∃ q 0 tel que (q, q 0) ∈→, q 0 |= ϕ,

• q |= AX ϕ ⇐⇒ ∀ σ un chemin tel que σ (0) = q, ∃ j tel que σ(j) |= ϕ,

• q |= EF ϕ ⇐⇒ ∃ un chemin σ tel que σ (0) = q et ∃ j tel que σ(j) |= ϕ,

• q |= AG ϕ ⇐⇒ ∀ σ un chemin tel que σ (0) = q, ∀ j, σ(j) |= ϕ,

• q |= EG ϕ ⇐⇒ ∃ σ tel que σ (0) = q et ∀ j, σ(j) |= ϕ,

• q |= A (ϕ U ϕ0) ⇐⇒ ∀ σ tel que σ (0) = q, ∃ j tel que σ(j) |= ϕ 0 et ∀ k tel que k < j, σ(k) |= ϕ,

• q |= E (ϕ U ϕ0) ⇐⇒ ∃ un chemin σ tel que σ (0) = q, ∃ j tel que σ(j) |= ϕ 0 et ∀ k tel que k < j, σ(k) |= ϕ.

### I.4.2 la Logique Temporelle Linéaire LTL :

En logique, la **logique temporelle linéaire** (**LTL**) est une logique temporelle modale avec des modalités se référant au temps. En LTL, on peut coder des formules sur l'avenir d'un chemin infini dans un système de transitions, par exemple une condition finira par être vraie, une condition sera vraie jusqu'à ce qu'une autre devienne vraie, etc. Cette logique est plus faible que la logique CTL*, qui permet d'exprimer des conditions sur des ramifications de chemins et pas seulement sur un seul chemin. LTL est aussi parfois appelée *logique temporelle propositionnelle*, abrégé *LTP* (*PTL* en anglais). La logique temporelle linéaire (LTL) est un fragment de S1S (pour second-order with 1 successor), la logique monadique du second ordre avec un successeur.

LTL a d'abord été proposé pour la vérification formelle des programmes informatiques par Amir Pnueli en 1977.

Syntaxe :

La LTL est construite à partir d'un ensemble fini de variables propositionnelles *AP*, opérateurs logiques ¬ et ∨, et des opérateurs temporels modaux **X** (certaines notations utilisent **O** ou **N**) et **U**. Formellement, l'ensemble des formules de LTL sur *AP* est inductivement défini comme suit :

- Si p ∈ *AP* alors p est une formule de LTL ;
- Si ψ et ϕ sont des formules de LTL alors ¬ψ, ϕ ∨ ψ, **X** ψ, et ϕ **U** ψ sont des formules de LTL.[5]

**X** est lu comme *suivant* (*Ne**x**t* en anglais) et **U** est lu comme *jusqu'à* (***u**ntil* en anglais). On peut ajouter d'autres opérateurs, non nécessaires mais qui simplifient l'écriture de certaines formules.

Par exemple, les opérateurs logiques ∧, →, ↔, vrai et faux sont généralement ajoutés. Les opérateurs temporels ci-dessous le sont également :

- **G** pour toujours (**g**lobalement (***g****lobally* en anglais)) ;
- **F** pour finalement (dans le **f**utur (*in the **f**uture* en anglais)) ;
- **R** pour libération (***r****elease* en anglais) ;
- **W** pour faible jusqu'à (***w****eak until* en anglais).

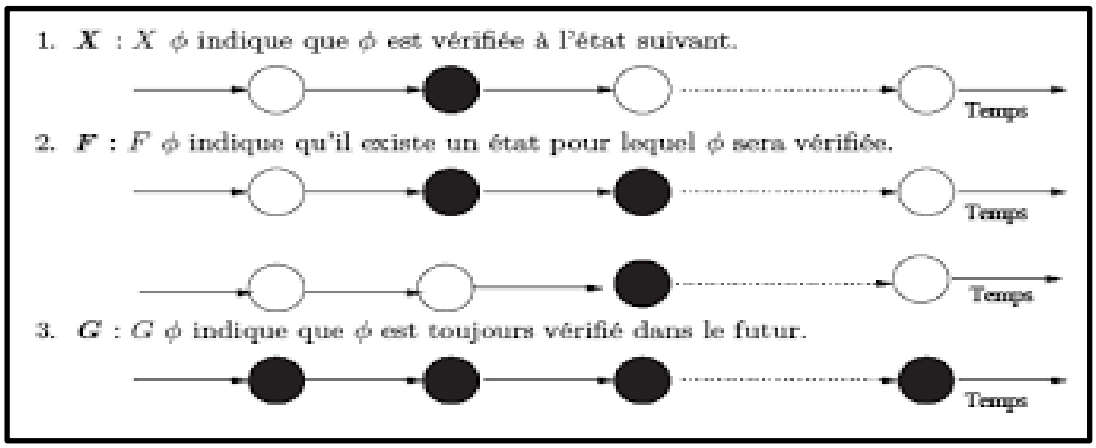



**Figure I.6** la Logique Temporelle Linéaire

## I.5 Les Avantages et Les Inconvenant De Model Checking :

L'avantage essentiel du model-checking sur les méthodes fondées sur le test et la simulation est l'exhaustivité : une propriété valide sur le modèle le sera dans absolument toutes les situations alors qu'un scénario réel de fonctionnement a pu être oublié dans une procédure de test. Par ailleurs, lorsqu'une propriété n'est pas vérifiée dans un modèle, les outils existants fournissent une trace d'exécution du modèle démontrant que la propriété n'est pas satisfaite. Il est alors plus facile de déterminer l'origine de l'erreur et de la corriger.

En contrepartie, l'effectivité du model checking est limitée par la taille des systèmes qu'on peut analyser en pratique. En effet, les modèles ont, même pour des systèmes relativement simples, des tailles importantes et on parle du problème de l'explosion combinatoire. L'analyse du modèle est alors impossible en raison de la complexité importante des calculs nécessaire. Dans de telles situations, l'utilisateur doit simplifier le modèle et proposer une abstraction qui préserve les propriétés à vérifier. (Zennon, 2004)

## I.6 Conclusion :

Nous avons consacré ce chapitre aux techniques de vérification qui nous intéressent dans le cadre de ce mémoire, à savoir : le model checking. Nous avons présente le principe du model checking, ses extensions aux aspects temporels et architecture du UPPAAl, ainsi qu'une revue de littérature des solutions proposées pour contrecarrer le problème d'explosion combinatoire. Parmi les formalismes utilises par le model checking, nous avons retenu et présente les automates temporisés, la logique temporelle et CTL et LTL. Finalement, nous avons d´écrit les outils de vérification par model checking. Et les avantages et les inconvénients.

## II.1. Introduction :

Les télécommunications et les réseaux sont devenus indispensables à toutes les activités entrepreneuriales et dans tous les domaines de la vie moderne : banques assurances hôpitaux etc. Malgré les efforts des experts Les réseaux informatiques sécurisés font l'objet de nombreuses attaques de la part de divers pirates. En fait presque tous les jours depuis l'avènement d'Internet nous avons lu et entendu des informations sur des entreprises subissant des pertes financières importantes. Peut subir des pertes dues à des failles de sécurité dans ses systèmes d'information. Au cours des dernières décennies plusieurs outils et approches techniques ont été développés pour améliorer la sécurité Les réseaux informatiques tels que les systèmes de détection d'intrusion (IDS) (Biermann et al. 2001) et les pares-feux (Cheswick et al. 2003). IDS est le système d'alerte du réseau. Ils permettent de détecter tout Activité indésirable et fournir aux administrateurs réseau les informations nécessaires pour empêcher les intrusions et être en mesure de suivre les attaquants si nécessaire. Les pares-feux quant à eux permettent d'inspecter et de filtrer le trafic sur une politique de sécurité donnée. Les pares-feux ont été largement déployés pour protéger les réseaux. Cependant la configuration de ces outils s'avère être une tâche très lourde sujette à de nombreuses anomalies pouvant Interférer avec l'application des politiques de sécurité sur le réseau.

Le reste de ce chapitre est organisé comme suit : La section 2.2 expose quelques concepts de base sur les réseaux informatiques. La section 2.3 présente les systèmes de détection d'intrusion (IDS) ainsi que leurs différents types. Nous y abordons également les différentes approches de détection d'intrusions. La section 2.4 est dédiée à la présentation des pares-feux ainsi que les différents types d'anomalies qui peuvent compromettre leur bon fonctionnement. La section 2.5 explore quelques techniques de détection d'anomalies liées à la mauvaise configuration des pares-feux alors que la section 2.6 passe en revue quelques solutions proposées dans la littérature pour résoudre les problèmes de configuration de pare-feu.

## II.2 Concepts de base :

Nous donnons un aperçu sur l'architecture des réseaux et les menaces informatiques qui leurs sont connexes. Enfin, nous explorons la notion de politiques de sécurité.

## II.2.1 Les types de réseau :

Un réseau informatique ou plus simplement un réseau fait référence à un nombre quelconque de systèmes informatiques indépendants qui sont reliés entre eux pour que l'échange de données soit réalisable. Pour cela, en plus d'une connexion physique, il doit aussi exister une connexion logique des systèmes en réseau. Cette dernière est produite par des protocoles réseau spécifiques comme par exemple le protocole TCP (*Transmission Control Protocol*, littéralement protocole de contrôle de transmissions). Même seulement deux ordinateurs reliés entre eux peuvent être considérés comme un réseau.

Les réseaux sont mis en place dans le but notamment de transférer des données d'un système à un autre ou de fournir des ressources partagées comme par exemple les serveurs, les bases de données ou une imprimante sur le réseau. Il est possible selon la taille et la portée du réseau informatique de différencier et de catégoriser les réseaux. Voici ci-dessous les principales catégories de réseaux informatiques:

- Personal Area Network (PAN) ou réseau personnel
- Local Area Network (LAN) ou réseau local
- Metropolitan Area Network (MAN) ou réseau métropolitain
- Wide Area Network (WAN) ou réseau étendu
- Global Area Network (GAN) ou réseau global

La connexion physique qui relie ces types de réseau peut être câblée (filaire) ou bien réalisée à l'aide de la technologie sans fil. Bien souvent les réseaux de communication physique constituent le fondement de plusieurs réseaux logiques, appelés VPN (*Virtuel Private Network*, ou réseau privé virtuel en français). Ceux-ci utilisent un moyen de transmission physique commun, par exemple un câble de fibre optique et, lors du transfert des données, sont assignés à des réseaux virtuels logiquement différents au moyen d'un logiciel de VPN créant un tunnel (ou logiciel de tunneling).

Chaque type de réseau a été développé pour des domaines d'application spécifiques, un réseau est basé sur des techniques et des normes propres apportant différents avantages et limites.

(Redes de Comunicação)

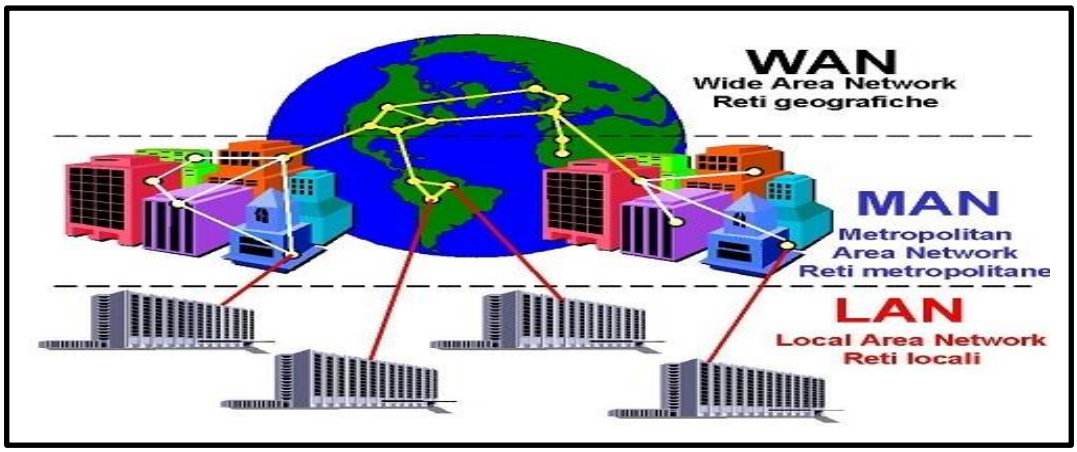



**Figure II.1 :** Les types de réseau.

## II.2.2 Architecture d'un réseau :

Les réseaux informatiques se composent d'un grand nombre de différents types d'appareils. Il existe trois types d'éléments de réseau (Pujolle et Salvatori, 2010) :

• Des équipements terminaux ou des systèmes informatiques tels que des ordinateurs, des postes de travail, des serveurs, des périphériques, etc. ;

• Dispositifs d'interconnexion tels que routeurs et commutateurs.

• Supports de communication tels que câbles, fibres optiques et lignes de communication. Se propager. Un réseau peut avoir plusieurs architectures appelées topologie pour définir les connexions parmi ses différents appareils.

Il existe plusieurs topologies de réseau (BILLAMI et BENDAHMANE (2014)). Dans cette section, nous nous référons aux topologies les plus couramment utilisées :

**Le bus :**

Tous les nœuds sont connectés en parallèle et chaque message est reçu par tous les nœuds. Le mot bus désigne littéralement la ligne physique qui relie les machines. Cette structure a plusieurs avantages : la défaillance d'un nœud n'entraine pas l'interruption de la communication avec les autres et il nécessite moins de connexions. Il est le plus facile à implémenter. En revanche, il y a plusieurs inconvénients :

- Deux machines ne peuvent pas communiquer en même temps
- En cas de panne du câble, le réseau n'existe plus
- La vitesse de transmission est faible

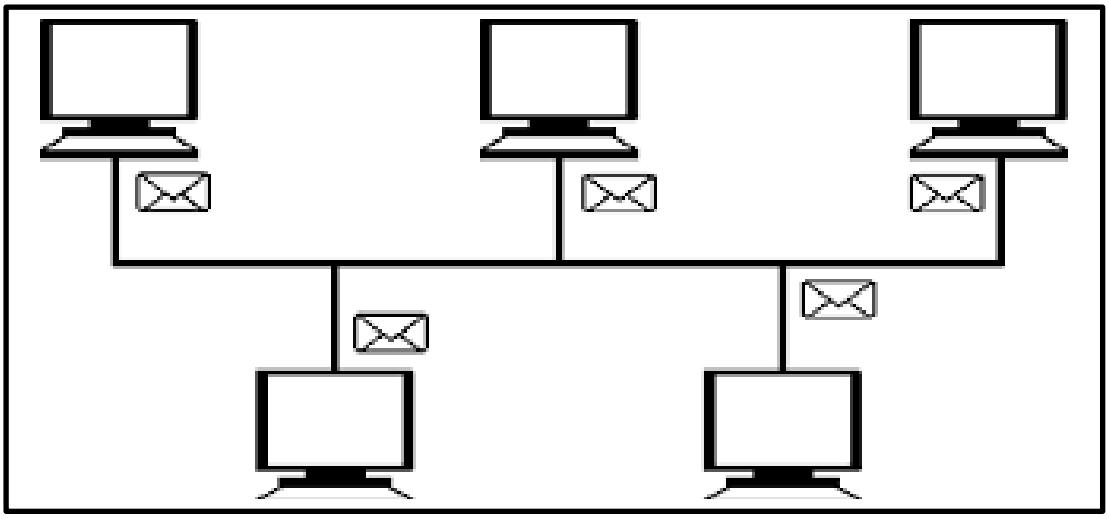


**Figure II.2 :** la topologie en bus

**L'étoile :**

C'est une structure basée sur un élément central, souvent un switch, d'où partent les connexions. Dans ce cas, deux machines peuvent émettre simultanément. Cette structure permet d'ajouter un nouveau nœud au réseau sans interrompre le service et la défaillance d'un nœud n'interrompt pas la communication entre les autres nœuds. Cependant, si l'élément central ne fonctionne plus, le réseau ne fonctionne plus.

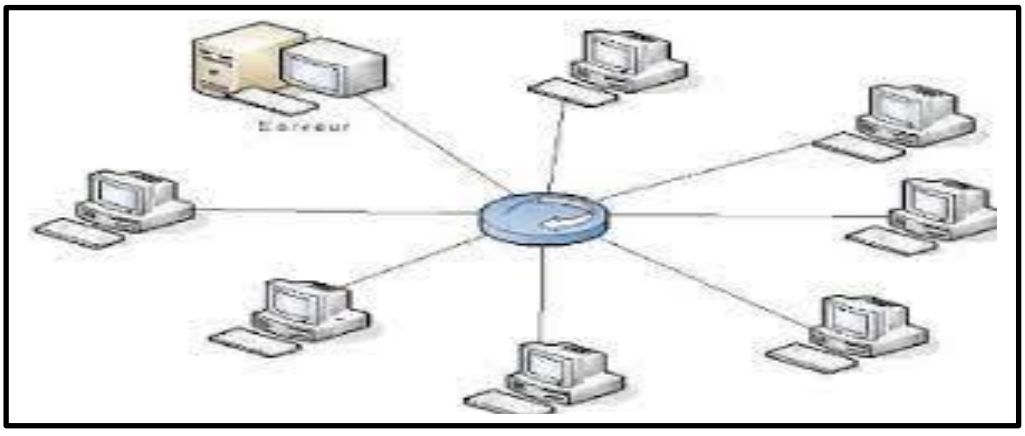


**Figure II.3 :** la topologie en étoile

**L'anneau :**

Cette structure permet de relier en cascade les machines d'un réseau. Elle présente plusieurs avantages:

- Le signal est régénéré à chaque nœud donc on peut couvrir des distances importantes
- Elle permet de mettre en place des priorités

Mais aussi plusieurs inconvenient:

- On ne peut pas étendre le réseau sans l'interrompre (on rompt le cercle)
- Si l'un des composants ne fonctionne plus, c'est le réseau entier qui est hors service

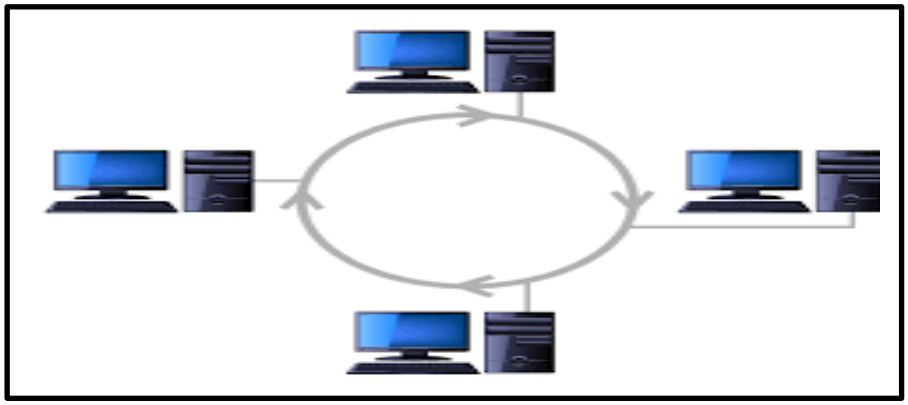


**Figure II.4:** la topologie en anneau

**Topologie maillée :**

Dans cette topologie, chacun des nœuds doit être relié. Il existe donc de multiples chemins entre deux nœuds du réseau. C'est donc une configuration peu sensible aux pannes mais couteuse et difficile à mettre en place.

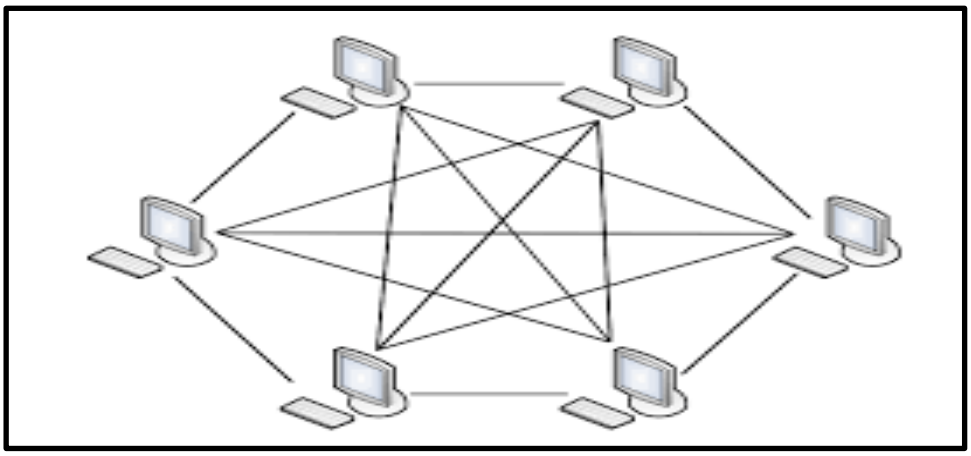


**Figure II.5** : la Topologie en maillée

Le paquet IP est formé de deux grandes parties :

- L'entête du paquet, généralement d'une taille de 20 octets, constitue le PCI du protocole. C'est là que sont inscrites toutes les informations du protocole (adresse, segmentation, options, etc.).

- La partie « data », ou champ de données, d'une taille maximum de : (65536 octets) – (les octets d'entête et d'options). Elle véhicule la PDU de couche supérieure (généralement un segment TCP ou UDP).

**Format général de l'entête du paquet :**

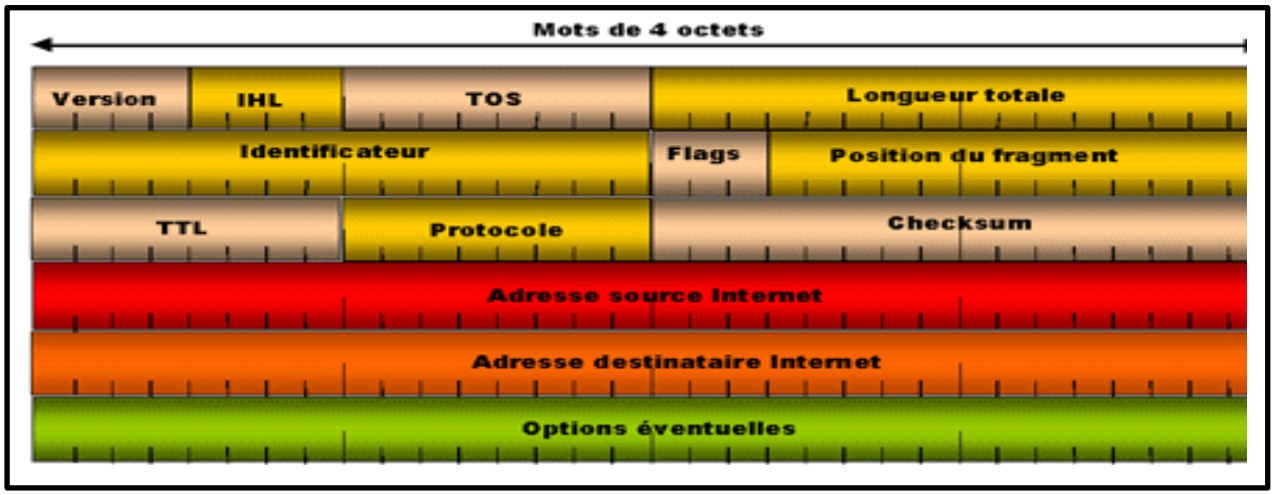



**Figure II.6** : Structure d'un paquet.

L'entête IP, comme tout le paquet IP est structurée en mots de 4 octets. C'est une des raisons qui a conduit a refusé la normalisation d'IP dans le modèle OSI, qui préconise une structure en mots de 1 octet.

**Structure en mots de 4 octets** veut dire que le nombre d'octets d'un paquet IP est toujours un multiple de 4 octets. Si on doit placer un nombre d'octets non multiple de 4 dans le paquet (65 par exemple), on procède à un complément, encore appelé « **bourrage** » pour les moins élégants, ou « **padding** » pour les cultivés ! Cette opération consiste à ajouter un nombre d'octets supplémentaires formés de 0, qui vont compléter jusqu'à un multiple de 4. Pour notre exemple, on ajouterai 3 octets pour obtenir 68 octets (68/4 = 17 mots de 4 octets).

L'entête IP standard n'utilise pas le champ « Option », elle a donc une longueur de 20 octets (5 mots de 4 octets) dans 99% des cas.

Le champ d'option peut avoir une longueur très variable. Nous le présentons succinctement, plus loin.

**Rôles des champs de l'entête :**

- **Version**

Ce champ code sur 4 bits, la version du protocole. La version actuelle est la version 4 (0100). Une version 6 est annoncée depuis bientôt 6 ans ! On commence à en voir certaines implémentations, mais le chemin est encore long. Elle introduit un nouveau format d'adressage permettant d'affecter une adresse IP à n'importe quoi sur terre, avec plusieurs milliards d'adresses possibles … Peut-être un jour en parlerons-nous … Mais il faudra que je me documente sur la chose …

- **Internet Header Length (IHL)**

Longueur de l'en-tête Internet, exprimée en mots de 4 octets. Cette longueur étant codée sur 4 bits, elle peut avoir comme valeur maximum binaire 1111 soit 15. Comme elle indique un nombre de mots de 4 octets, on en déduit que la taille maximum de l'entête IP est de 15*4 = 60 octets.

- **Type Of Service (TOS)**

Ce champ donne des indications aux équipements qu'il traverse sur son niveau de priorité et sa classe de service. Défini sur 8 eb (1 octet), il est divisé en 5 parties :

- Les bits 0 à 2 indiquent la « **précédence** « . Autrement dit la priorité d'acheminement par rapport aux autres paquets. Cette partie du champ permet de définir 8 niveaux de priorité d'acheminement.
- Le bit 3 indique si le paquet peut supporter une attente de traitement normale ou faible.
- Le bit 4 indique si le paquet doit être acheminé en **priorité** sur des liens à hauts débits ou non
- Le bit 5 indique si le paquet doit être acheminé sur des liens très fiables (faible taux d'indisponibilité et d'erreurs) ou normaux
- Les bits 6 et 7 sont réservés et donc inutilisés.

Quoiqu'il en soit, ce champ est rarement utilisé. Avec l'arrivée de nouvelles technologies comme le MPLS (Multi Protocol Label Switching) on commence à utiliser le champ TOS bits 0 à 2 pour définir des classes de services qui sont exploitées par MPLS. Certaines applications utilisaient déjà ce champ TOS pour définir des priorités d'acheminement de leurs paquets dans le réseau.

- **Total Length**

Longueur totale du datagramme, exprimée en octets, en-tête et données comprises. Ce champ étant codé sur 2 octets, la longueur maximale d'un paquet IP est donc de 65 536 octets (0 à 65535).

- **Identification, Flags et Fragment Offset**

Ces trois champs servent à gérer le mécanisme de fragmentation (ou segmentation si vous préférez) du paquet IP. En effet, il est possible qu'un paquet IP n'entre pas entièrement dans une

trame de niveau 2 parce que sa taille est trop importante. Il sera nécessaire de fragmenter le paquet pour le véhiculer dans plusieurs trames.

Cependant en raison de la nature non connectée du protocole, la fragmentation est assez complexe à gérer. Nous étudierons cet aspect plus complétement dans un chapitre suivant.

Pour le moment retenez qu'un mot complet de 4 octets est réservé à la gestion de la fragmentation.

- **Time To Live**

Toujours en raison du fait que le protocole IP fonctionne en mode non connecté, est capable d'implémenter des techniques de routage dynamique susceptible de lui permettre de changer de route pour acheminer ses paquets, il est possible qu'un paquet se perde dans le réseau. Pour éviter qu'il tourne indéfiniment, on lui affecte une durée de vie. Durée de vie du datagramme. Cette valeur est décrémentée toutes les secondes ou à chaque passage dans une passerelle. Si cette valeur est à 0, le datagramme est mis au rebut.

- **Protocol**

Ce champ codé sur un octet, identifie le protocole de niveau supérieur transporté dans le champ de données du paquet IP. Il permet au destinataire, en analysant ce champ, de savoir à quel protocole de niveau supérieur (niveau 4 en général : TCP ou UDP) il doit transmettre le contenu du datagramme.

- **Checksum**

Le checksum est le champ de contrôle d'erreur. Il est calculé uniquement sur l'en-tête. Le principe consiste à faire la somme des valeurs des octets de l'entête et à inscrire le résultat dans l'octet de checksum. Le récepteur effectue la même opération, si la valeur trouvée est identique, il n'y a pas d'erreur. Dans le cas contraire, le paquet est rejeté.

---

**ATTENTION** : IP possède un mécanisme de détection d'erreurs, mais pas de correction d'erreurs. Autrement dit, il jette le paquet, mais n'informe personne et ne cherche pas à générer une répétition du paquet (réémission par l'émetteur). Les couches supérieures devront gérer elles-mêmes cette perte de paquet et s'occuper des demandes de réémissions éventuelles.

Comme certains champs de l'entête peuvent varier dans le temps, le checksum est recalculé par chaque équipement traversé par le paquet.

---

- **Adresses destination et source**

Les champs d'adresses sont chacun codés sur 4 octets. Il existe donc, en théorie, 2 puissances 32 adresses possibles (4 294 967 296 adresses). Dans le chapitre consacré à l'adressage, nous étudierons précisément ce format.

Le champ adresse Source indique l'adresse IP de la machine qui émet les données, le champ adresse Destination indique l'adresse IP de la machine à qui sont envoyées les données.

- **Options**

Je vous l'ai déjà indiqué, ce champ est assez peu utilisé. Il permet de mettre en œuvre des mécanismes évolués comme le routage par la source (l'émetteur indique par où doit passer le paquet) ou l'enregistrement de routes (le paquet enregistre par où il est passé pour arriver à destination).

La longueur totale de la partie options doit être un multiple de quatre octets. Il y a donc une possibilité de « padding » (bourrage) à la fin de la section options.

(Jean-Luc GATOUX)

## II.2.3 Menaces réseau :

Une menace informatique représente le type d'actions susceptibles de nuire dans l'absolu à un système informatique. En termes de sécurité informatique les menaces peuvent être le résultat de diverses actions. Ce sont des adversaires déterminés capables de monter une attaque exploitant une vulnérabilité.

**Les types de menaces :**

- Menaces accidentelles : Ils sont liés à des erreurs involontaires des utilisateurs du poste, perte accidentelle de données, dégradation ou destruction involontaire de matériel, copies illicites de logiciels….
- Menaces intentionnelles : Une menace intentionnelle est une action exécutée par une entité pour violer la sécurité de l'information et l'utilisation non autorisée des ressources. Les menaces intentionnelles peuvent être passives ou active.
- Menaces passives : Une menace passive constitue à écouter le trafic du réseau (ou de la machine) cible, donc l'interface de la machine attaquante (du pirate) est en mode écoute. L'objectif est de découvrir et capturer des trames du réseau cible pour y rechercher des informations particulières : clés de cryptages, mots de passe ou données. Elle se réalise grâce à des outils tels que : les sniffer, les scanners.

- Menaces actives : Contrairement à une menace passive, ici l'attaquant n'est plus en mode écoute. Elles consistent à modifier des données ou des messages, à s'introduire dans des équipements réseau ou à perturber le bon fonctionnement de ce réseau, et aussi à interroger le réseau (ou la machine) cible, contourner le dispositif de sécurité existant par diverses méthodes tels que : Déni de Service et Virus.

## II.2.4 Politique de sécurité :

Une politique de sécurité informatique est une stratégie visant à maximiser la sécurité informatique d'une entreprise. Elle est matérialisée dans un document qui reprend l'ensemble des enjeux, objectifs, analyses, actions et procédures faisant parti de cette stratégie.

- À distinguer de la *charte informatique*, qui est un document de recommandations concernant la bonne utilisation des technologies informatiques, et qui est destiné aux employés de l'entreprise
- Ce document est unique et personnalisé, car établi en tenant compte du fonctionnement, de l'environnement, de la composition du système d'information de l'entreprise et des enjeux et des risques informatiques qui lui sont propres

La sécurité du système d'information en général

La mise en place d'une politique de sécurité informatique n'est que l'une des nombreuses mesures possibles pour assurer la sécurité du système d'information de l'entreprise. A titre d'exemple, voici quelques-unes des meilleures pratiques en matière de sécurité informatique pour une entreprise :

- Une bonne maintenance du parc informatique
- une responsibilities du personnel
- La formation du personnel aux bonnes pratiques informatiques
- L'utilisation d'outils permettant d'être prêt face aux attaques informatiques (tels que antivirus, antispam, pare-feu etc.)
- Le contrôle des accès Internet de l'entreprise
- Le contrôle des accès aux informations de l'entreprise, et notamment aux informations sensibles
- L'hébergement des données dans des environnements sécurisés et monitorés
- La mise en place de sauvegardes adaptées, sécurisées, redondées

Mettre en place une politique de sécurité informatique : les bonnes pratiques

Vous trouverez ci-dessous quelques-unes des meilleures pratiques à observer lors de l'élaboration de votre politique de sécurité informatique :

1. Désigner un responsable informatique, qui sera en charge de l'élaboration et de la mise en place de cette politique de sécurité.
2. Définir le périmètre et les objectifs de la politique de sécurité informatique, à des fins d'efficience et de mesure des résultats.
3. Effectuer une analyse de l'existant, matériel et logiciel, et tenir à jour un registre de l'ensemble des éléments qui composent le système d'information. Ce registre est important lors des modifications des composants de la configuration informatique. En cas d'incident, il peut permettre aux équipes IT de trouver l'origine du problème.
4. Effectuer une analyse des risques informatiques, au regard du préjudice possible et de la probabilité d'occurrence de l'incident.
5. Déterminer les moyens nécessaires pour la réduction des risques et la prise en charge des incidents, qu'il s'agisse de moyens matériels ou humains.
6. Définir les procédures adaptées, notamment en matière de gestion des incidents, ou de gestion de la continuité d'activité.
7. Rédiger une charte informatique, à l'attention des collaborateurs.
8. Communiquer sur la politique de sécurité informatique auprès de l'ensemble de l'entreprise.

(IVISION)

## II.3 Systèmes de détection d'intrusions :

Un IDS (Intrusion Detection System) est un ensemble de composants logiciels et/ou matériels dont la fonction principale est de détecter et analyser des activités anormales ou suspectes sur la cible analysée (un réseau ou un hôte). Il permet ainsi d'avoir une connaissance sur les tentatives réussies comme échouées des intrusions. Certains termes sont souvent employés quand on parle d'IDS :

• Faux positif : une alerte provenant d'un IDS, mais qui ne correspond pas à une attaque réelle.

• Faux négatif : une intrusion réelle qui n'a pas été détectée par l'IDS.

### II.3.1 Les types des systèmes de détection d'intrusion :

À cause de la diversité des attaques que mettent en œuvre les pirates, la détection d'intrusion doit se faire à plusieurs niveaux. Il existe donc différents types d'IDS :

- **Les NIDS (Network-based Intrusion Detection System) :** Les NIDS sont des IDS dédiés aux réseaux. Ils comportent généralement une machine qui écoute sur le segment de réseau à surveiller, un capteur et un moteur qui réalise l'analyse du trafic afin de détecter les intrusions en temps réel. Un NIDS écoute donc tout le trafic réseau, puis l'analyse et génère des alertes si des paquets semblent dangereux. (Voir figure II.7)

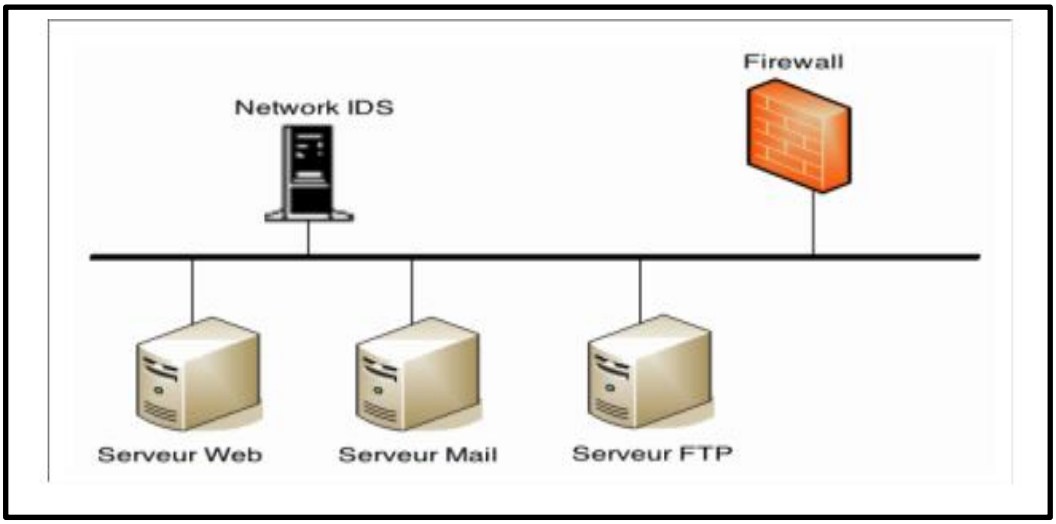



**Figure II.7 :** Système de détection d'intrusion réseau.

L'implantation d'un NIDS sur un réseau se fait de la façon suivante : des capteurs (souvent de simples hôtes) sont placés aux endroits stratégiques du réseau et génèrent les alertes s'ils détectent une attaque. Ces alertes sont envoyées à une console sécurisée qui les analyse et les traite éventuellement. Cette console est généralement située sur un réseau isolé qui relie uniquement les capteurs et la console. On peut placer les capteurs dans deux endroits différents :

• **A l'intérieur du pare-feu** : Si les capteurs se trouvent à l'intérieur du pare-feu, il sera plus facile de dire si le pare-feu a été mal configuré et nous pouvons ainsi savoir si une attaque est venue par ce pare-feu.

• **A l'extérieur du pare-feu** : Les capteurs placés à l'extérieur du pare-feu servent à la détection et l'analyse d'attaques. Il offre l'avantage d'écrire dans les logs, ainsi l'administrateur voit ce qu'il doit modifier dans la configuration du pare-feu.

**Quelques exemples des NIDS :**

- NetRanger [http://www.cisco.com].
- Dragon [http://www.securitywizards.com].
- NFR [http://www.nfr.net].
- Snort [http://www.snort.org].
- DTK [http://all.net/dtk/dtk.html].

- **Les HIDS (Host-based Intrusion Detection System) :** Les systèmes de détection d'intrusion basés sur l'hôte analysent exclusivement l'information concernant cet hôte. Comme ils n'ont pas à contrôler le trafic du réseau mais seulement les activités d'un hôte ils se montrent habituellement plus précis sur les types d'attaques. De plus, nous remarquons immédiatement l'impact sur la machine concernée comme par exemple si un utilisateur l'attaquait avec succès. Ces IDS utilisent deux types de sources pour fournir une information sur l'activité : les logs et les traces d'audit du système d'exploitation. Chacun a ses avantages : les traces d'audit sont plus précises et détaillées et fournissent une meilleure information alors que les logs qui ne fournissent que l'information essentielle sont plus petits. (Voir figure II.8)

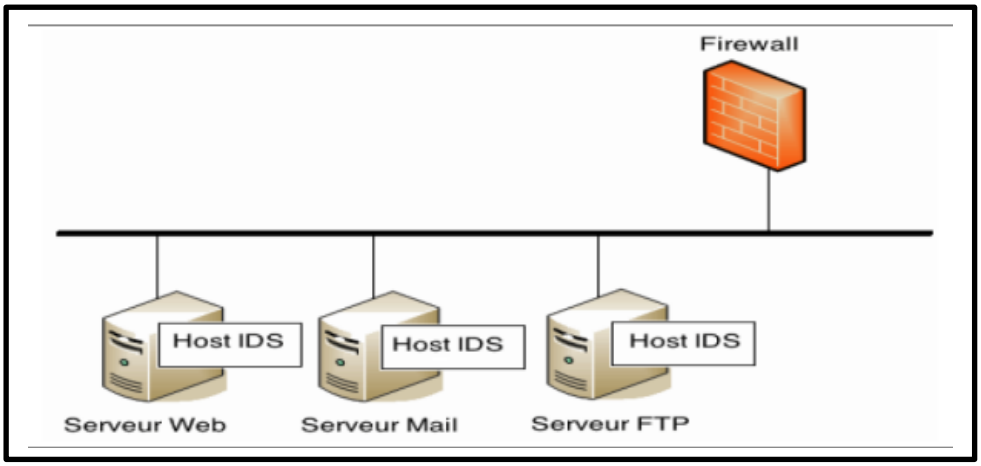



**Figure II.8** : Système de détection d'intrusion hôte.

Quelques exemples des HIDS :

- Tripwire [http://www.tripwire.com/products/index.cfml].
- SWATCH [http://freshmeat.net/redir/swatch/10125/url_homepage/swatch].

 DragonSquire [http://www.enterasys.com/ids/squire/].

 Tiger [http://freshmeat.net/redir/tiger-audit/30581/url_homepage/tiger].

- **IDS hybride :** Les systèmes de détections d'intrusions hybrides, rassemblent les caractéristiques de plusieurs systèmes de détections différents. En pratique, on trouve la combinaison des NIDS et HIDS qui permettent de surveiller le réseau et l'hôte. Les sondes agissent comme un NIDS ou un HIDS Il permet de réunir les informations de diverses sondes placées sur le réseau. L'exemple le plus connu dans le monde Open-Source est Prelude.
  - Cet IDS permet de stocker dans une base de données des alertes provenant de différents systèmes relativement variés.
  - Utilisant Snort comme NIDS, et d'autres logiciels tels que Samhain en tant que HIDS, il permet de combiner des outils puissants tous ensemble pour permettre une visualisation centralisée des attaques.

## II.3.2 Caractéristiques d'un système de détection d'intrusion :

Parmi les caractéristiques souhaitables trouvées dans un système de détection d'intrusion nous pouvons citer :

- Résister aux tentatives de corruption, c'est-à-dire, il doit pouvoir détecter s'il a subi lui-même une modification indésirable.
- Utiliser un minimum de ressources de système sous surveillance.
- S'adapter au cours du temps aux changements du système surveillé et du comportement des utilisateurs.
- Être facilement configurable pour implémenter une politique de sécurité spécifique d'un réseau.

## II.3.3 L'architecture d'un IDS :

Cette section décrit les trois composants qui constituent classiquement un système de détection d'intrusions. La Figure II.9 illustre les interactions entre ces trois composants.

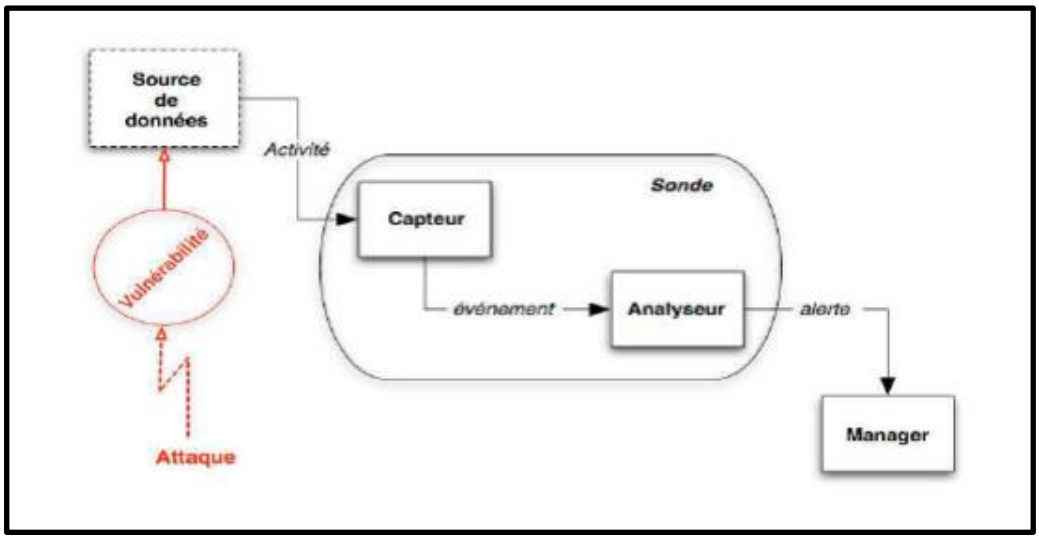



**Figure II.9** : : L'architecture d'un IDS.

- **Capteur :** Le capteur observe l'activité du système par le biais d'une source de données et fournit à l'analyseur une séquence d'événements qui renseignent de l'évolution de l'état du système. Le capteur peut se contenter de transmettre directement ces données brutes, mais en général un prétraitement est effectué. On distingue classiquement trois types de capteurs en fonction des sources de données utilisées pour observer l'activité du système : les capteurs système, les capteurs réseau et les capteurs applicatifs.

- **Analyseur :** L'objectif de l'analyseur est de déterminer si le flux d'événements fourni par le capteur contient des éléments caractéristiques d'une activité malveillante.

- **Manager :** Le manager collecte les alertes produites par le capteur, les met en forme et les présente à l'opérateur. Éventuellement, le manager est chargé de la réaction à adopter qui peut être :

• Confinement de l'attaque, qui a pour but de limiter les effets de l'attaque.

• Eradication de l'attaque, qui tente d'arrêter l'attaque.

• Recouvrement, qui est l'étape de restauration du système dans un état sain.

• Diagnostic, qui est la phase d'identification du problème.

## II.3.4 Les méthodes de détections :

Pour bien gérer un système de détection d'intrusions, il est important de comprendre comment celui-ci fonctionne. Une question simple se pose sur la détection d'un tel système et sur le critère de différencier un flux contenant une attaque d'un flux normal. Ces questions nous ont amenés à étudier le fonctionnement interne d'un IDS. De là, nous en avons déduit deux techniques mises en place dans la détection d'attaques. La première consiste à détecter des signatures d'attaques connues dans les paquets circulant sur le réseau. La seconde, consiste quant à elle, à détecter une activité suspecte dans le comportement de l'utilisateur. Ces deux techniques, aussi différentes soient-elles, peuvent être combinées au sein d'un même système afin d'accroître la sécurité.

**1.L'approche par scénario (mésuse détection) :**

Cette technique s'appuie sur la connaissance des techniques utilisées par les attaquants pour déduire des scénarios typiques. Elle ne tient pas compte des actions passées de l'utilisateur et utilise des signatures d'attaques (ensemble de caractéristiques permettant d'identifier une activité intrusive : une chaîne alphanumérique, une taille de paquet inhabituelle, une trame formatée de manière suspecte…). Plusieurs analyses peuvent être utilisées pour la méthode de détection par scénario :

- **Recherche de motifs (pattern matching) :** La méthode la plus connue et la plus à facile à comprendre. Elle se base sur la recherche de motifs (chaînes de caractères ou suite d'octets) au sein du flux de données. L'IDS comporte une base de signatures où chaque signature contient les protocoles et port utilisés par l'attaque ainsi que le motif qui permettra de reconnaître les paquets suspects

Le principal inconvénient de cette méthode est que seules les attaques reconnues par les signatures seront détectées. Il est donc nécessaire de mettre à jour régulièrement la base de signatures. Un autre inconvénient est que les motifs sont en général fixes. Or une attaque n'est pas toujours identique à 100 %. Le moindre octet différent par rapport à la signature provoquera la non-détection de l'attaque. Pour les IDS utilisant cette méthode, il est nécessaire d'adapter la base de signatures en fonction du système à protéger. Cela permet non seulement de diminuer les ressources nécessaires et donc augmenter les performances ; mais également réduire considérablement le nombre de fausses alertes et donc faciliter le travail des administrateurs réseau qui analyseront les fichiers d'alertes.

- **Recherche de motifs dynamiques** : Le principe de cette méthode est le même que précédemment, mais les signatures des attaques évoluent dynamiquement. L'IDS est de ce fait doté de fonctionnalités d'adaptation et d'apprentissage.

- **Analyse de protocoles** : Cette méthode se base sur une vérification de la conformité des flux, ainsi que sur l'observation des champs et paramètres suspects dans les paquets. L'analyse protocolaire est souvent implémentée par un ensemble de préprocesseurs, où chaque préprocesseur est chargé d'analyser un protocole particulier (FTP, HTTP, ICMP…). Du fait de la présence de tous ces préprocesseurs, les performances dans un tel système s'en voient fortement dégradées. L'intérêt fort de l'analyse protocolaire est qu'elle permet de détecter des attaques inconnues, contrairement au pattern Matching qui doit connaître l'attaque pour pouvoir la détecter.
- **Analyse heuristique et détection d'anomalies :** Le but de cette méthode est une analyse intelligente, de détecter une activité suspecte ou toute autre anomalie. Par exemple : une analyse heuristique permet de générer une alarme quand le nombre de sessions à destination d'un port donné dépasse un seuil dans un intervalle de temps prédéfini.

**2. L'approche comportementale (Anomalie Detection) :** Cette technique consiste à détecter une intrusion en fonction du comportement passé de l'utilisateur. Pour cela, il faut préalablement dresser un profil utilisateur à partir de ses habitudes et déclencher une alerte lorsque des événements hors profil se produisent. Cette technique peut être appliquée non seulement à des utilisateurs, mais aussi à des applications et services. Plusieurs métriques sont possibles : la charge CPU, le volume de données échangées, le temps de connexion sur des ressources, la répartition statistique des protocoles et applications utilisés, les heures de connexion... Cependant elle possède quelques inconvénients :

- **Peu fiable** : tout changement dans les habitudes de l'utilisateur provoque une alerte ;
- **Nécessite une période de non-fonctionnement** pour mettre en œuvre les mécanismes d'auto-apprentissage : si un pirate attaque pendant ce moment, ses actions seront assimilées à un profil utilisateur, et donc passeront inaperçues lorsque le système de détection sera complètement mis en place.
- **L'établissement du profil doit être souple** afin qu'il n'y ait pas trop de fausses alertes : le pirate peut discrètement intervenir pour modifier le profil de l'utilisateur afin d'obtenir après plusieurs jours ou semaines, un profil qui lui permettra de mettre en place son attaque sans qu'elle ne soit détectée. Plusieurs approches peuvent être utilisées pour la méthode de détection comportementale :

- **Approche probabiliste** : Des probabilités sont établies permettant de représenter une utilisation courante d'une application ou d'un protocole. Toute activité ne respectant pas le modèle probabiliste provoquera la génération d'une alerte.
- **Approche statistique** : Le but est de quantifier les paramètres liés à l'utilisateur : taux d'occupation de la mémoire, utilisation des processeurs, valeur de la charge réseau, nombre d'accès à l'Intranet

par jour, vitesse de frappe au clavier, sites les plus visités… Cette méthode est très difficile à mettre en place. Elle n'est actuellement présente que dans le domaine de la recherche, où les chercheurs utilisent des réseaux neuronaux pour tenter d'avoir des résultats convaincants.

## II.4 Pare-feu(firewall) :

Un pare-feu (appelé aussi coupe-feu, garde-barrière ou firewall en anglais), est un système permettant de protéger un ordinateur ou un réseau d'ordinateurs des intrusions provenant d'un réseau tiers (notamment internet). Le pare-feu est un système permettant de filtrer les paquets de données échangés avec le réseau, il s'agit ainsi d'une passerelle filtrante comportant au minimum les interfaces réseaux suivantes : une interface pour le réseau à protéger (réseau interne) ; une interface pour le réseau externe.

### II.4.1 Les type de Pare-feu :

- **Le pare-feu sans état**

Le pare-feu sans état est le plus ancien dispositif de filtrage réseau. Il agit au niveau de la couche réseau et transport du modèle OSI, le standard de communication entre les systèmes informatiques. Ce pare-feu inspecte chaque paquet et lui accorde le passage uniquement s'il répond à une liste de règles prédéterminées, qui se basent sur l'adresse IP source et destinataire, le numéro de port source et destinataire, et les protocoles de couche 3 et 4. Le problème, c'est que la configuration des pares-feux sans état est complexe, et que ce dispositif ne permet pas d'obtenir une finesse de filtrage très évoluée. Il est donc en train de devenir obsolète, même s'il est encore utilisé sur certains systèmes d'exploitation et routeurs.

- **Le pare feu avec état**

Les pares-feux avec état sont plus exigeants que leurs prédécesseurs sans état : ils vérifient systématiquement la conformité des paquets à une connexion en cours, en regardant si chaque paquet d'une connexion constitue bien la suite du précédent. Le pare-feu avec état a également de la mémoire : il garde une trace des sessions et des connexions, et réagit en cas d'anomalie. Contrairement à son prédécesseur, il permet ainsi de se protéger de certaines attaques Dos.

- **Le pare-feu applicatif**

Le pare-feu applicatif appartient à la dernière génération de firewalls. Il fonctionne au niveau de la couche application : par exemple, chaque requête de type http est filtrée par un processus proxy http. Le pare-feu rejette ainsi toutes les requêtes non conformes aux spécifications du protocole.

Ce type de pare-feu est plus sûr que le pare-feu avec état, mais est très gourmand en temps de calcul dès que le débit est important. Dans les années à venir, il devrait gagner en performance**.**

- **Le pare-feu identifiant**

Le pare-feu identifiant réalise des associations entre l'IP et les utilisateurs, et permet de suivre l'activité réseau par utilisateur. Les règles de filtrage sont ainsi définies en fonction de l'identification, et non des adresses IP. Selon les pares-feux, plusieurs méthodes différentes sont utilisées, dont l'identification par authpf, les pares-feux entièrement basés sur l'identité, ou la création de règles dynamiques prenant en compte l'identité de l'utilisateur et de son poste, ainsi que son niveau de sécurité informatique

(IPE Informatique)

## II.4.2 Emplacement d'un Pare-feu :

L'emplacement d'un pare-feu est l'une des tâches importantes d'un administrateur réseau. Ce choix dépend de la fonction du pare-feu. Ce dernier peut être place `a la frontière de sorte qu'il protège le réseau local de toutes les connexions venant d'Internet. La situation correspondante est présentée par la Figure II.10

(Fall, 2010)(Adnane EL KABBAL (2005))

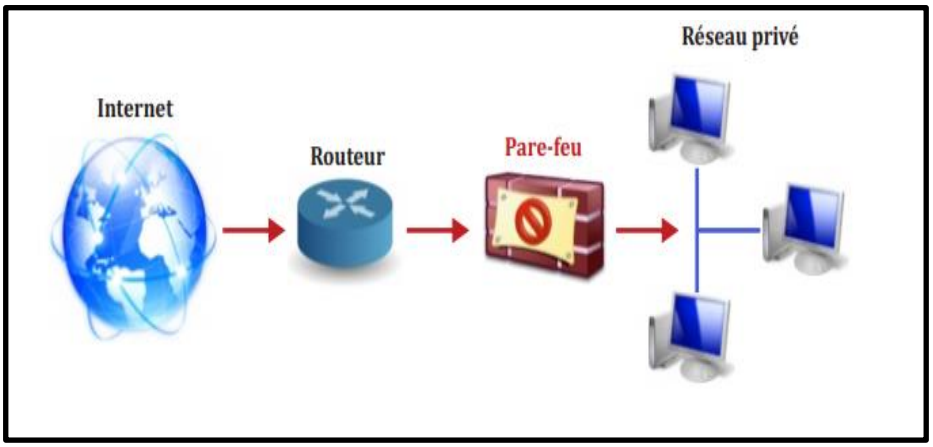



**Figure II.10 :** Emplacement d'un Pare-feu a la frontière.

Un pare-feu peut aussi être placé à l'intérieur d'un réseau pour le diviser en plusieurs sous-réseaux et contrôler les différents accès entre eux. La disposition correspondante est représentée par la Figure II.11

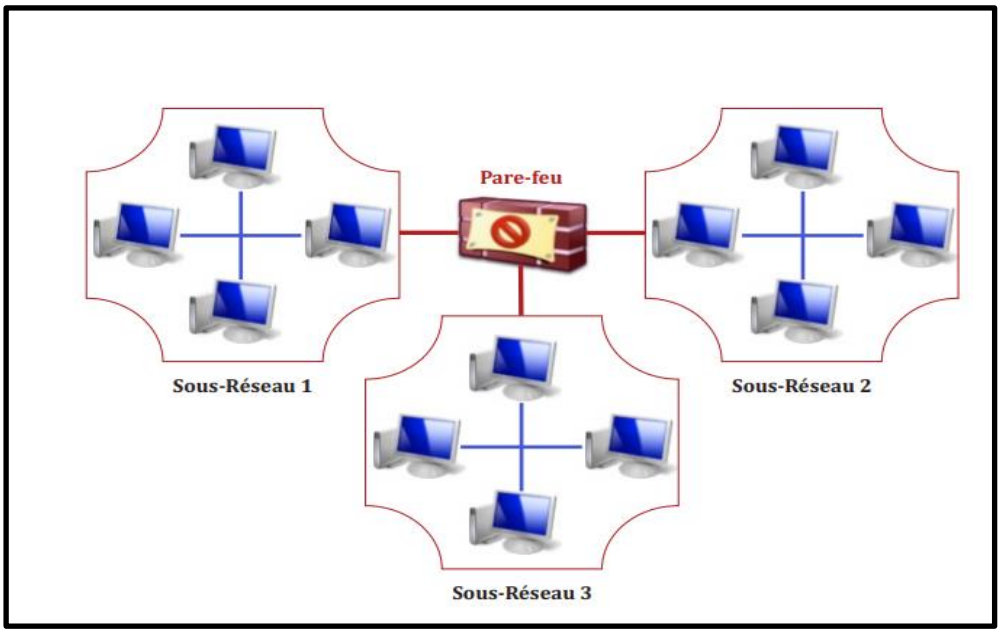



**Figure II.11 :** Emplacement d'un Pare-feu au centre d'un réseau**.**

## II.4.3 Fonctions d'un Pare-feu :

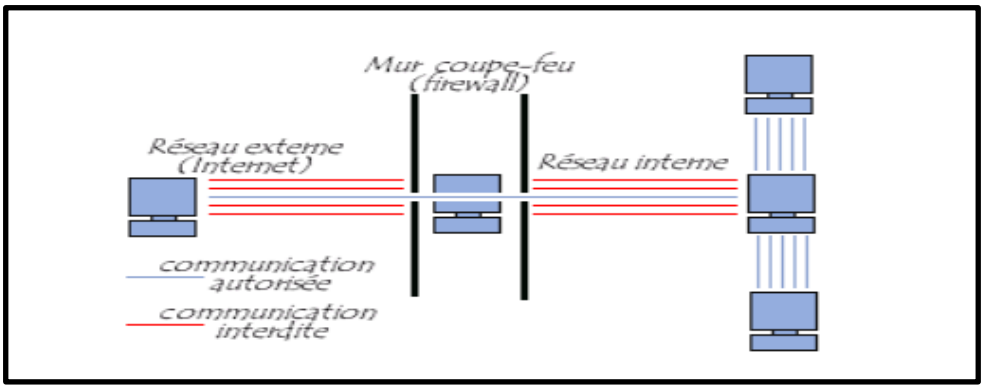



**Figure II.12:** Fonctionnement d'un Pare feu.

Un système pare-feu contient un ensemble de règles prédéfinies permettant :

- D'autoriser la connexion (*allow*);
- De bloquer la connexion (*deny*) ;
- De rejeter la demande de connexion sans avertir l'émetteur (*drop*).

L'ensemble de ces règles permet de mettre en œuvre une méthode de filtrage dépendant de la **politique de sécurité** adoptée par l'entité. On distingue habituellement deux types de politiques de sécurité permettant :

- Soit d'autoriser uniquement les communications ayant été explicitement autorisées (c'est le Principe du moindre privilège) ;
- Soit d'empêcher les échanges qui ont été explicitement interdits.

La première méthode est sans nul doute la plus sûre, mais elle impose toutefois une définition précise et contraignante des besoins en communication.

❖ **Le filtrage simple de paquets**

Un système pare-feu fonctionne sur le principe du filtrage simple de paquets (en anglais « *stateless packet filtering* »). Il analyse les en-têtes de chaque paquet de données (*datagramme*) échangé entre une machine du réseau interne et une machine extérieure.

Ainsi, les paquets de données échangée entre une machine du réseau extérieur et une machine du réseau interne transitent par le pare-feu et possèdent les en-têtes suivants, systématiquement analysés par le firewall :

- Adresse IP de la machine émettrice ;
- Adresse IP de la machine réceptrice ;
- Type de paquet (TCP, UDP, etc.) ;
- Numéro de port (rappel: un port est un numéro associé à un service ou une application réseau).

Les adresses IP contenues dans les paquets permettent d'identifier la machine émettrice et la machine cible, tandis que le type de paquet et le numéro de port donnent une indication sur le type de service utilisé.

Le tableau ci-dessous donne des exemples de règles de pare-feu :

| Règle | Action | IP source | IP dest | Protocol | Port source | Port dest |
|---|---|---|---|---|---|---|
| 1 | Accept | 192.168.10.20 | 194.154.192.3 | tcp | any | 25 |
| 2 | Accept | any | 192.168.10.3 | tcp | any | 80 |
| 3 | Accept | 192.168.10.0/24 | any | tcp | any | 80 |
| 4 | Deny | any | any | any | any | any |

**Tableau II.1** des exemples de règles de pare-feu

Les ports reconnus (dont le numéro est compris *entre 0 et 1023*) sont associés à des services courants (les ports 25 et 110 sont par exemple associés au courrier électronique, et le port 80 au Web). La plupart des dispositifs pare-feu sont au minimum configuré de manière à filtrer les communications selon le port utilisé. Il est généralement conseillé de bloquer tous les ports qui ne sont pas indispensables (selon la politique de sécurité retenue).

Le port 23 est par exemple souvent bloqué par défaut par les dispositifs pare-feu car il correspond au protocole Telnet, permettant d'émuler un accès par terminal à une machine distante de manière à pouvoir exécuter des commandes à distance. Les données échangées par Telnet ne sont pas chiffrées, ce qui signifie qu'un individu est susceptible d'écouter le réseau et de voler les éventuels mots de passe circulant en clair. Les administrateurs lui préfèrent généralement le protocole SSH, réputé sûr et fournissant les mêmes fonctionnalités que Telnet.

- **Le filtrage dynamique**

Le filtrage simple de paquets ne s'attache qu'à examiner les paquets IP indépendamment les uns des autres, ce qui correspond au niveau 3 du modèle OSI. Or, la plupart des connexions reposent sur le protocole TCP, qui gère la notion de session, afin d'assurer le bon déroulement des échanges. D'autre part, de nombreux services (le FTP par exemple) initient une connexion sur un port statique, mais ouvrent dynamiquement (c'est-à-dire de manière aléatoire) un port afin d'établir une session entre la machine faisant office de serveur et la machine cliente.

Ainsi, il est impossible avec un filtrage simple de paquets de prévoir les ports à laisser passer ou à interdire. Pour y remédier, le système de **filtrage dynamique de paquets** est basé sur l'inspection des couches 3 et 4

du modèle OSI, permettant d'effectuer un suivi des transactions entre le client et le serveur. Le terme anglo-saxon est « **stateful inspection** » ou « *stateful packet filtering* », traduisez « *filtrage de paquets avec état* ».

Un dispositif pare-feu de type « stateful inspection » est ainsi capable d'assurer un suivi des échanges, c'est-à-dire de tenir compte de l'état des anciens paquets pour appliquer les règles de filtrage. De cette manière, à partir du moment où une machine autorisée initie une connexion à une machine située de l'autre côté du pare-feu ;

L'ensemble des paquets transitant dans le cadre de cette connexion seront implicitement acceptés par le pare-feu.

Si le filtrage dynamique est plus performant que le filtrage de paquets basique, il ne protège pas pour autant de l'exploitation des failles applicatives, liées aux vulnérabilités des applications. Or ces vulnérabilités représentent la part la plus importante des risques en termes de sécurité.

- **Le filtrage applicatif**

Le filtrage applicatif permet de filtrer les communications application par application. Le filtrage applicatif opère donc au niveau 7 (couche application) du modèle OSI, contrairement au filtrage de paquets simple (niveau 4). Le filtrage applicatif suppose donc une connaissance des protocoles utilisés par chaque application.

Le filtrage applicatif permet, comme son nom l'indique, de filtrer les communications application par application. Le filtrage applicatif suppose donc une bonne connaissance des applications présentes sur le réseau, et notamment de la manière dont elle structure les données échangées (ports, etc.).

Un firewall effectuant un filtrage applicatif est appelé généralement « passerelle applicative » (ou « proxy »), car il sert de relais entre deux réseaux en s'interposant et en effectuant une validation fine du contenu des paquets échangés. Le proxy représente donc un intermédiaire entre les machines du réseau interne et le réseau externe, subissant les attaques à leur place. De plus, le filtrage applicatif permet la destruction des en-têtes précédant le message applicatif, ce qui permet de fournir un niveau de sécurité supplémentaire.

Il s'agit d'un dispositif performant, assurant une bonne protection du réseau, pour peu qu'il soit correctement administré. En contrepartie, une analyse fine des données applicatives requiert une grande puissance de calcul et se traduit donc souvent par un ralentissement des communications, chaque paquet devant être finement analysé.

Par ailleurs, le proxy doit nécessairement être en mesure d'interpréter une vaste gamme de protocoles et de connaître les failles afférentes pour être efficace.

Enfin, un tel système peut potentiellement comporter une vulnérabilité dans la mesure où il interprète les requêtes qui transitent par son biais. Ainsi, il est recommandé de dissocier le pare-feu (dynamique ou non) du proxy, afin de limiter les risques de compromission.

## II.4.4 Différences clés entre le Firewall et le Serveur Proxy :

Quelle est la différence entre Proxy et Firewall ? – Les deux sont des composants de sécurité réseau. Dans une certaine mesure, ils sont similaires dans un sens qu'ils limitent ou bloquent les connexions vers et depuis votre réseau, mais ils le font de différentes manières.
Les Firewall peuvent bloquer les ports et les programmes qui tentent d'accéder sans autorisation à votre ordinateur, tandis que les Serveurs Proxy masquent essentiellement votre réseau interne sur Internet. Il fonctionne comme un Firewall en ce sens qu'il empêche votre réseau d'être exposé à Internet en redirigeant les requêtes Web si nécessaire.

- **Définition de Serveur Proxy :**

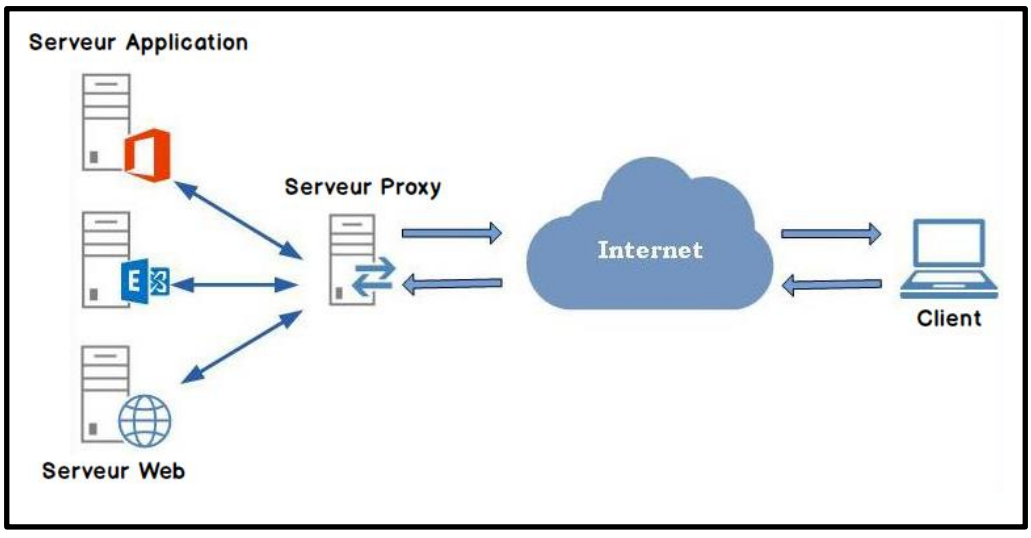



**Figure II.13:** serveur proxy**.**

Un serveur proxy est un serveur situé entre une application cliente, telle qu'un navigateur Web, et un serveur réel. Il intercepte toutes les demandes adressées au serveur réel pour voir s'il peut répondre aux demandes lui-même. Sinon, il transmet la demande au serveur réel.Le serveur proxy peut exister sur le même ordinateur qu'un Firewall ou sur un serveur distinct, qui transfère les demandes via le Firewall.

L'avantage d'un serveur proxy est son cache peut servir tous les utilisateurs. Si un ou plusieurs sites Internet sont fréquemment demandés, ceux-ci se trouvent probablement dans le cache du proxy, ce qui améliore le temps de réponse de l'utilisateur. Un proxy peut également enregistrer ses interactions, ce qui peut être utile pour le dépannage.

- **Définition de Firewall:**

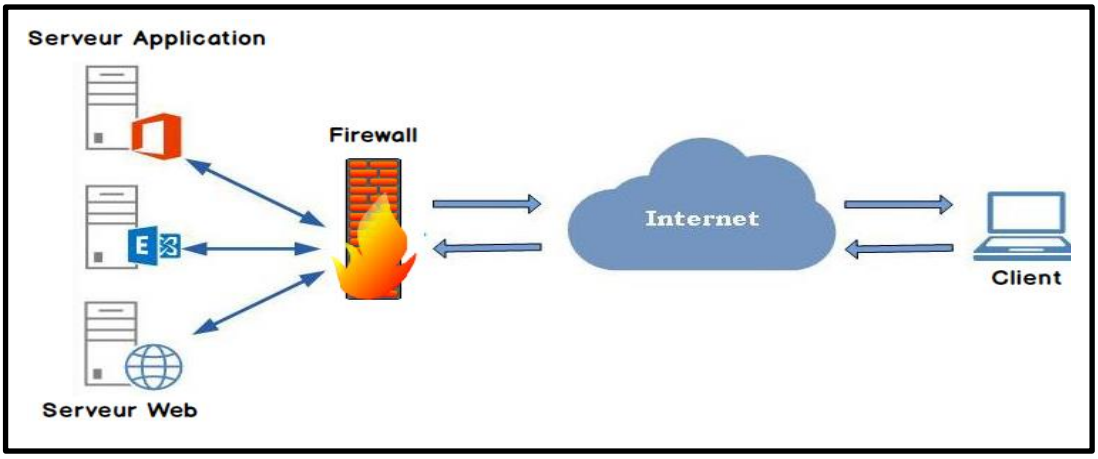



**Figure II.14:** Firewall.

Un Firewall est utilisé pour assurer la sécurité d'un réseau privé. Les Firewalls bloquent les accès non autorisés à ou depuis des réseaux privés et sont souvent utilisés pour empêcher les utilisateurs Web non autorisés ou les logiciels illicites d'accéder aux réseaux privés connectés à Internet. Un Firewall peut être implémenté à l'aide de matériel, de logiciels ou d'une combinaison des deux.

**Différences clés entre le Firewall et le Serveur Proxy :**

- Un Firewall bloque les connexions tandis qu'un Serveur Proxy facilite les connexions
- Un Serveur Proxy peut également servir de Firewall
- Les Firewalls existent souvent en tant qu'interface entre un réseau public et privé, alors que des Serveurs Proxy peuvent également exister avec des réseaux publics des deux côtés.
- Un Firewall est utilisé pour protéger un réseau interne contre les attaques tandis qu'un Serveur Proxy est utilisé pour l'anonymat et pour contourner les restrictions

### II.4.5 Les limites des firewalls :

Un système pare-feu n'offre bien évidemment pas une sécurité absolue, bien au contraire. Les firewalls n'offrent une protection que dans la mesure où l'ensemble des communications vers l'extérieur passe systématiquement par leur intermédiaire et qu'ils sont correctement configurés. Ainsi, les accès au réseau extérieur par contournement du firewall sont autant de failles de sécurité. C'est notamment le cas des connexions effectuées à partir du réseau interne à l'aide d'un modem ou de tout moyen de connexion échappant au contrôle du pare-feu.

De la même manière, l'introduction de supports de stockage provenant de l'extérieur sur des machines internes au réseau ou bien d'ordinateurs portables peut porter fortement préjudice à la politique de sécurité globale.

Enfin, afin de garantir un niveau de protection maximal, il est nécessaire d'administrer le pare-feu et notamment de surveiller son journal d'activité afin d'être en mesure de détecter les tentatives d'intrusion et les anomalies. Par ailleurs, il est recommandé d'effectuer une veille de sécurité (en s'abonnant aux alertes de sécurité des CERT par exemple) afin de modifier le paramétrage de son dispositif en fonction de la publication des alertes.

La mise en place d'un firewall doit donc se faire en accord avec une véritable politique de sécurité.

## II.5 Renforcement des politiques sur les réseaux informatiques :

Lacasse a défini une nouvelle algèbre de processus basée sur CCS. Cette algèbre qui est destinée à la sécurisation formelle des réseaux, est munie d'un opérateur de surveillance qui permet de contrôler les entrées et les sorties d'un processus, ou d'un sous-processus, à l'image d'un pare-feu dans un réseau informatique. L'algèbre permet donc de modéliser des réseaux munis de moniteurs, et également, n'importe quel système communicant devant être contrôlé par des moniteurs. L. Pene et K.Adi ont présenté une nouvelle approche pour spécifier et vérifier les politiques de sécurité distribuées mises en application dans les pare-feu distribués. Ils ont démontré par une étude de cas comment leur nouveau calcul qui est inspiré du calcul ambient peut être employé pour aborder le problème de la détection de conflit dans les pares-feux distribués. Le but de leur contribution est d'identifier comment les paquets légitimes peuvent être arrêtés d'atteindre leur destination et les paquets potentiellement nocifs peuvent passer. Nous constatons que

Lacasse et L.pene ont focalisé leur travaux sur les politiques de sécurité mises en application sur les pare-feu hors que les politiques de sécurités peuvent être appliquées sur des domaines plus vastes. T.Mechri a proposé une technique formelle permettant de configurer automatiquement un réseau donné afin qu'il respecte une politique de sécurité donnée. En d'autres termes, étant donné un réseau informatique $N$, et une politique de sécurité Φ, il a introduit une technique formelle qui produit automatiquement un autre réseau $N'$ tel que $N'$Φ, $N$, et $N'$ se comportent d'une façon équivalente (par rapport à une définition d'équivalence donnée). On effet, il a défini une nouvelle algèbre de processus permettant de mieux préciser et analyser un réseau surveillé. Il a défini aussi un opérateur $\otimes$ qui produit à partir d'un réseau initial $N$ et une politique de sécurité Φ une autre version du réseau, notée $N \otimes \Phi$, configurée d'une manière que la politique de sécurité soit toujours respectée. Un autre système de renforcement de politique de sécurité a été présenté par Gloria et Pugliese . Leur système traite des questions liées à la mobilité de code dans les systèmes distribués, tels que le commerce électronique et les services bancaires en ligne. Le type est expressément conçu pour le processus de calcul $\mu$−$Klaim$ , un calcul de processus distribués et mobiles. Il permet l'attribution de différents privilèges aux utilisateurs sur différentes ressources, qui imite le comportement d'un système réel. Cela rend le système pratique, comme en témoigne leur modèle de système de gestion de compte bancaire. Martins a introduit un système de type différent applicable aux réseaux complexes qui ont besoin d'une meilleure sécurité. Il a utilisé D$\pi$ pour contrôler la migration de code entre les nœuds de grands systèmes distribués. La topologie du réseau se reflète par des politiques de sécurité partagés dans les sites web. Le résultat est un système qui est libre de toute violation des règles lors de l'exécution.

(Marfall N'Diaga Fall, (2010))

## II.5.1 Approche algébrique :

Approche algébrique T.Mechri a proposé dans (voir Mechri, 2007) une approche algébrique basée sur des méthodes formelles pour la configuration automatique et la sécurisation des réseaux informatiques conformément a une politique de sécurité bien déterminée. Cette approche consiste a définir une nouvelle algèbre de processus (Calculus for Monitored Network, CMN) pour modéliser le réseau et une logique propositionnelle (LM) pour sp´ecifier des politiques de sécurité. L'objectif est de définir un opérateur de renforcement noté $\otimes$ permettant de générer, a partir d'un réseau et une politique de sécurité donnée, une autre version sécuritaire du réseau. La Figure II.15 illustre l'objectif de l'approche.

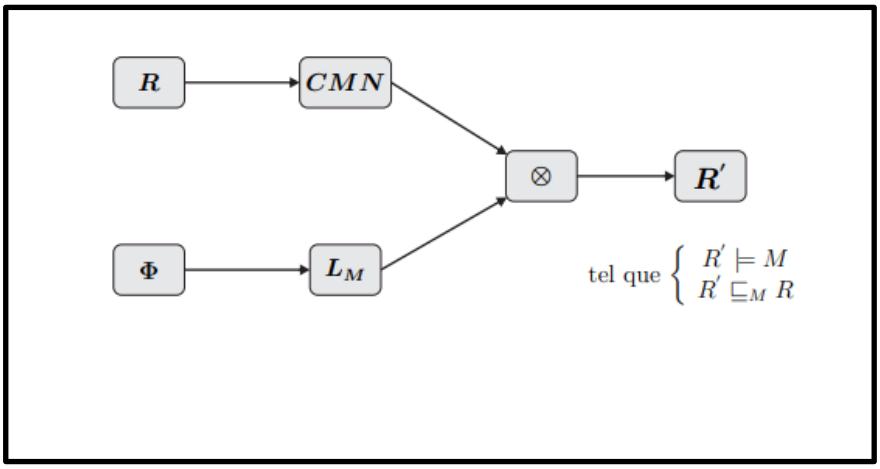



**Figure II.15** Approche algébrique pour la sécurisation des réseaux informatiques

Une version améliorée de cet opérateur est obtenue par le développement d'une technique de monitorage sélectif permettant de contrôler l'insertion des moniteurs dans les composantes du réseau. Le nouvel opérateur de renforcement noté ⊗s consiste à générer automatiquement, à partir d'un réseau R, une politique de sécurité M et un ensemble de composantes du réseau S, un nouveau réseau R' respectant M dans lequel les moniteurs sont placés uniquement sur les éléments de S. La Figure II.16 présente un schéma illustratif du principe de fonctionnement de l'opérateur ⊗s

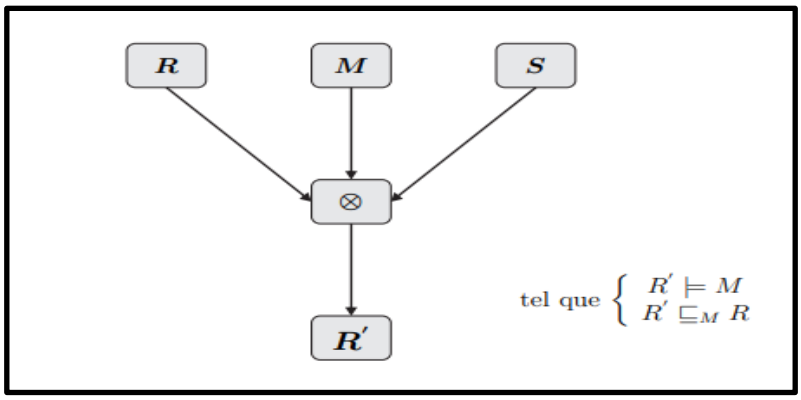



**Figure II.16** Opérateur du monitorage sélectif

L'approche proposée par Mechri est une approche rigoureuse. Il a fourni une extension du langage de spécification des pares-feux proposé par Mechri. En effet, le but est de définir un opérateur de renforcement permettant de générer une configuration sécuritaire et optimale d'un réseau. La valeur ajoutée de ce travail, par rapport au travail T. Mechri, est l'introduction de la notion du cout relatif à la

qualité de service d'un réseau afin de déterminer le coût d'une configuration.
Cependant, la définition du cout d'une configuration est basée uniquement sur le nombre de règles dans un pare-feu. Cette définition est loin d'être pratique et ne faisant pas intervenir concrètement les couts matériels lies à une configuration.

## II.5.2 Autres approches :

D'autres approches ont été proposées afin de renforcer la sécurité sur les réseaux informatiques a savoir :

- **Firmato (Firewall Management Toolkit)**

Firmat est un outil aux fonctionnalités similaires au filtrage de posture. Cependant, il est basé sur un modèle entité-relation et un langage (MDL) se servant de ce modèle, tous deux dédiés à la description des réseaux et des propriétés de sécurité. Firmato dispose également d'un compilateur générant automatiquement les règles de filtrage des pares-feux ainsi qu'un illustrateur pour leur visualisation. Le principe de fonctionnement de Firmato peut se résumer comme suit : Après la définition et la spécification de la politique de sécurité en MDL, cette version en MDL est ensuite utilisée par l'analyseur pour générer un modèle entité-relation. Ce dernier est transformé par le compilateur en fichiers de configuration de pare-feu garantissant une mise en œuvre correcte de la politique. En effet, le principe de fonctionnement du compilateur est basé sur une notion de rôle. Un rôle est la description des opérations autorisées entres les hôtes selon leurs fonctions. Les différentes composantes de Firmato peuvent être décrites comme suit :

**1. Le modèle entité-relation** : Cette entité contient sous une forme unifiée, une vue globale de la politique de sécurité ainsi que la topologie du réseau.

**2. Le langage de définition de modèle (MDL)** : MDL est un langage simple pour spécifier aussi bien une politique de sécurité que la topologie d'un réseau. Il permet ainsi de définir une instance du modèle entité-relation.

**3. Le Compilateur** : Le compilateur permet de transformer le modèle entité-relation en fichiers de configuration spécifiques aux pares-feux. Il ignore la structure du réseau et se focalise sur les définitions des rôles, des groupes de rôles et de leurs assignations aux groupes d'hôtes. À partir de ces informations, il déduit les groupes d'hôtes qui devraient avoir des règles autorisant certains services entre eux. La question de la passerelle qui pourra appliquer cette règle reste donc ignorée. Ainsi, le compilateur génère une base de règles centralisée contenant toutes les règles nécessaires pour mettre en œuvre la politique.

**4. Le distributeur** : La base de règles générée par le compilateur doit être distribuée sur chacune des passerelles du réseau avec une personnalisation appropriée selon les interfaces. Pour s'assurer que la politique de sécurité est correctement renforcée, une règle concernant une paire d'hôtes doit être incluse dans chacune des passerelles se trouvant sur chaque chemin reliant les deux hôtes. Afin d'éviter de faire des hypothèses sur les protocoles de routage, la stratégie adoptée consiste à reproduire la base de règles sur toutes les interfaces de chaque passerelle. Toutefois, la direction du domaine des règles doit être définie car c'est important pour réduire les attaques d'usurpation. Pour ce faire, le pare-feu examine la direction dans laquelle le paquet s'introduit dans l'interface de la passerelle et la compare avec la direction du domaine de la règle. Si le paquet tente de quitter l'interface pour une zone adjacente, il ne sera autorisé que lorsque la direction de la règle est entrante ou double. De même, un paquet ne sera autorisé à entrer dans l'interface depuis une zone adjacente que lorsque la direction de la règle est sortante ou double. Le distributeur utilise un algorithme pour déterminer les directions et permettre ainsi de diminuer le nombre de règles ayant une direction double afin d'être le plus précis possible.

**5. L'illustrateur** : L'illustrateur donne une représentation graphique des règles du pare-feu et permet ainsi de faire une rétro-ingénierie pour extraire ou retrouver la politique à partir de la configuration. Il prend en entrée le modèle généré par le compilateur et le transforme en un graphe visualisant la structure des groupes d'hôtes (host-groups) et des services autorisés. Il crée ainsi une visualisation de la politique comme on pourrait le voir sur une seule interface. Le débogage devient donc plus facile. On peut également retrouver la politique à partir de ce résultat.

- **Diagramme de décision d'un pare-feu (FDD)** Dans, Gouda et al. Présentent une méthode pour la conception et l'ordonnancement des séquences de règles dans un pare-feu afin que ce dernier soit cohérent, complet et consistant. La cohérence indique que les règles sont correctement ordonnées, la complétude signifie que chaque paquet doit satisfaire au moins une règle dans le pare-feu et la consistance est le fait que le pare-feu est dépourvu de toute redondance entre ses règles. Pour atteindre un tel objectif, Gouda et al. Commencent par concevoir un diagramme de décision de pare-feu, FDD (firewall decision Diagram) dont la cohérence et la complétude peuvent être vérifiées systématiquement (par un algorithme). On applique alors une série d'algorithmes à ce diagramme pour générer, réduire et simplifier les règles de pare-feu tout en maintenant la cohérence et la complétude du FDD original.

**Définition (Diagramme de décision d'un pare-feu):** Un diagramme de décision f d'un pare-feu est un arbre ayant les cinq propriétés suivantes :

1. Il n'y a qu'un nœud dans f n'ayant pas d'arêtes entrantes. C'est la racine de f. Les nœuds de f qui n'ont pas d'arêtes sortantes sont les nœuds terminaux de f.

2. Chaque nœud v a une étiquette F(v) tel que :

$$F(v) = \begin{cases} \{F_1, ...F_d\} & si\ v\ n'est\ pas\ un\ noeud\ terminal, \\ DS, & Si\ v\ est\ un\ noeud\ terminal. \end{cases}$$

Où D est un ensemble de décision.

3. Chaque arête e dans f à une étiquette I(e) tel que si e est une arête sortante du nœud v, alors I(e) est un sous-ensemble non vide de D(F(v)).

4. Un chemin direct de la racine à un nœud terminal dans f est appelé un chemin de décision de f. Deux nœuds sur un même chemin de décision ne peuvent avoir la même étiquette.

5. E(v) désigne l'ensemble des arêtes sortantes d'un nœud v et satisfait les deux conditions suivantes :

(a) ***La cohérence :*** $I(e) \cap I(e') = \emptyset$ *pour toutes arêtes* $e$ *et* $e'$ *distinctes dans* $E(v)$,
(b) ***la complétude :*** $\bigcup_{e \in E(v)} I(e) = D(F(v))$.

Exemple Soit f le pare-feu du Tableau II.2

| | | | |
|---|---|---|---|
| $r_1$ : | $S \in [4,7]$ | $\wedge\ D \in [6,8]$ | $\rightarrow a$ |
| $r_2$ : | $S \in [4,7]$ | $\wedge\ D \in [2,5] \cup [9,9]$ | $\rightarrow d$ |
| $r_3$ : | $S \in [4,7]$ | $\wedge\ D \in [1,1] \cup [10,10]$ | $\rightarrow a$ |
| $r_4$ : | $S \in [3,3] \cup [8,8]$ | $\wedge\ D \in [2,9]$ | $\rightarrow d$ |
| $r_5$ : | $S \in [3,3] \cup [8,8]$ | $\wedge\ D \in [1,1] \cup [10,10]$ | $\rightarrow a$ |
| $r_6$ : | $S \in [1,2] \cup [9,10]$ | $\wedge\ D \in [1,10]$ | $\rightarrow a$ |

**Tableau II.2** Un pare-feu consistant f

Le diagramme de décision équivalent est représenté par la Figure **II.17**

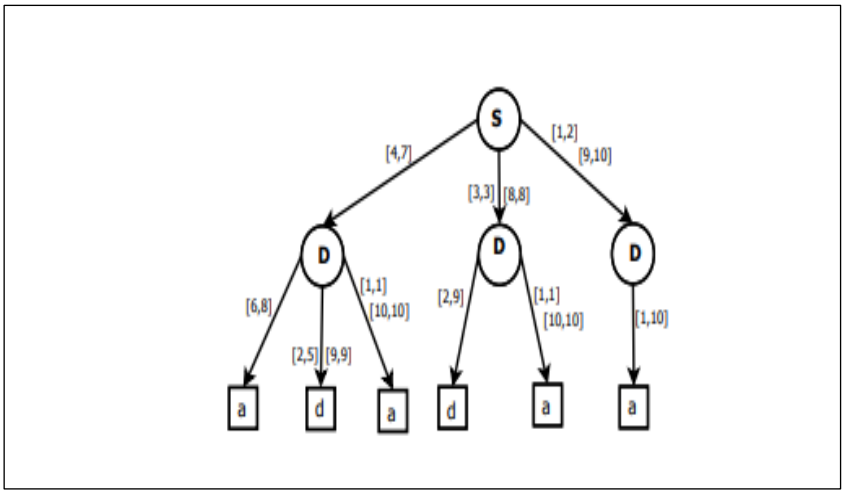



**Figure II.17** Diagramme de décision du pare-feu f

(ABDOUNE Mohamed Oulhadj HAMIDOUCHE Ahcène)

## II.6 Conclusion :

Dans ce chapitre, nous avons présenté un bref aperçu sur quelques concepts de base lies à la sécurité informatique (les types, architecture …..). Les différentes topologies d'un réseau informatique (i.g. anneau, bus, étoile, maillé, etc.) ont été également présentées. Après avoir exposé les menaces informatiques, nous avons introduit la notion de politiques de sécurité dans les réseaux.

Dans un deuxième volet, nous avons passé en revue deux principales techniques utilisées pour la sécurisation de réseaux informatiques : i) les systèmes de détection d'intrusions IDS et ii) les pares-feux. Etant l'objet de notre travail, nous avons exploré les différents types et fonctionnalités des pares-feux et Différences clés entre le Firewall et le Serveur Proxy, Les limites des firewalls.

Afin de corriger ces renforcer automatiquement une politique de sécurité dépourvue de toute incohérence et les Approche algébrique, Autres approches.

## III.1 Introduction :

Dans le chapitre présent, nous proposons un exemple de modele formel permettant de vérifier et configurer les pares-feux par modele checking.

Le reste du chapitre est organisé comme suit : La section III.2 donne une représentation graphique d'un réseau informatique a pares-feux. La section III.3 présente Modélisation du comportement. La section III.4 présente Vérification formelle de la cohérence des pares-feux.

## III.2 Représentation graphique d'un réseau a pare-feu :

Un réseau informatique est constitué d'un ensemble de nœuds interconnectes. Chaque pare-feu contrôle la réception du nœud dont il est responsable et incarne une politique de sécurité. Une politique de sécurité d'un pare-feu est une liste de règles ordonnées. Ces règles reflètent les objectifs et les mesures prises par un expert de sécurité afin d'assurer la sécurité sur un réseau.

La mise en œuvre d'une politique de sécurité dépend de la philosophie de la sécurité adoptée par l'administrateur du réseau.

Les règles de filtrage d'un pare-feu sont définies dans une liste spécifique appelée liste de contrôle d'accès (ACL). Chaque règle est définie comme suit :

**Rule :< Condition >⇒< Decision >**

Ou « Condition » est une expression booléenne qui associe à chaque paquet une valeur booléenne en se basant sur un ensemble de paramètres de filtrage. La decision d'une règle appartient à l'ensemble {accept, deny} dépendamment de la valeur de vérité de la condition, ou « accept » et « deny » signifient respectivement que le paquet en question est accepté ou refusé par le processus de filtrage.

Formellement, un réseau représenté par un graphe étiqueté est défini par un tuple G =
(N, F, R, A, L), où :

• N = N0, N1, ..., $N_n$ est un ensemble d'identificateurs de nœuds (n + 1 est le nombre de Nœuds),

• F = F0, F1, ..., $F_n$ est un ensemble d'identificateurs de pares-feux,

• A est une fonction partielle N ×N → F, qui associe, `a tout couple de nœuds connectés,
Un pare-feu,

• R = R0, R1, ..., $R_k$ est un ensemble d'identificateurs de règles (k + 1 est le nombre de Règles),

• Soit $2^R$ un ensemble de tous les sous-ensembles de R. L est une fonction F → $2^R$ qui

Associe à un pare-feu un sous-ensemble de règles appartenant à $2^R$

A titre illustratif, la Figure III.1 montre un exemple d'un réseau a pares-feux représentés

Par graphe étiqueté, où N = {N0, N1, N2, N3, N4}, F = {F0, F1, F2, F3, F4}, A et L sont

Définies par :

• A (N1, N0) = A (N2, N0) = A (N3, N0) = F0,

• A (N0, N1) = A (N2, N1) = F1,

• A (N0, N2) = A (N1, N2) = F2,

• A (N0, N3) = A (N4, N3) = F3,

• A (N3, N4) = F4

• L(F0) = R1, R2

• L(F1) = R3, R4

• L(F2) = R2

• L(F3) = R1, R5

• L(F4) = R1

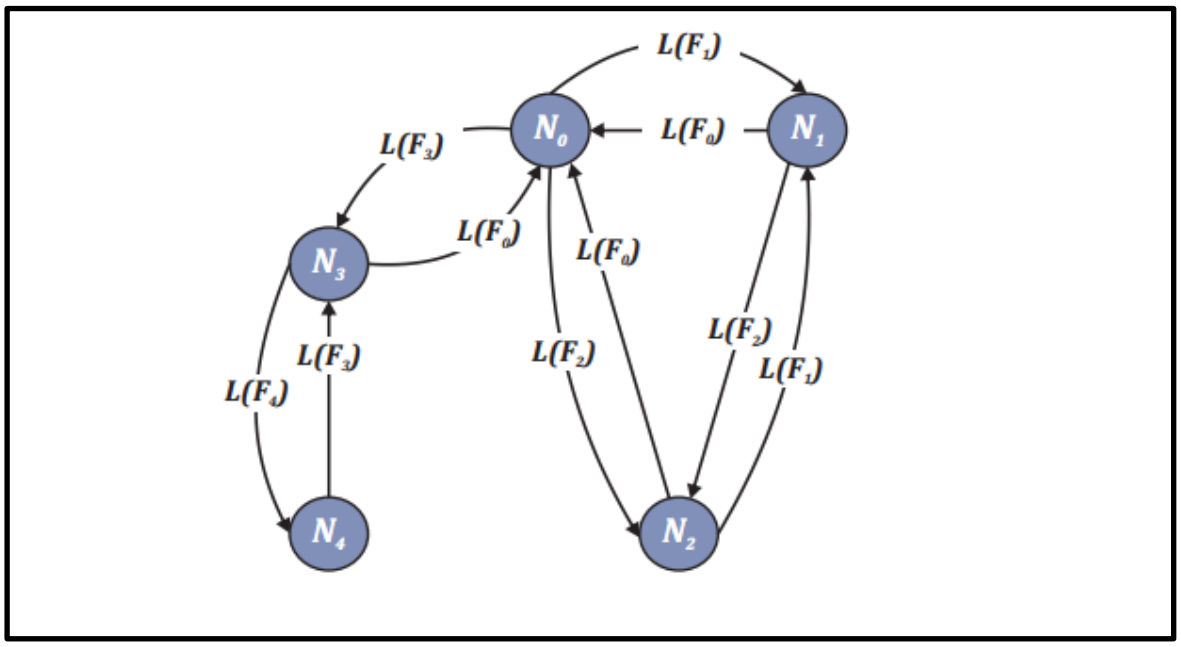



**Figure III.1:** Représentation graphique d'un réseau.

Chaque nœud $N_i$ peut recevoir un paquet d'un autre nœud $N_i$si et seulement s'il existe

Un arc reliant $N_i$et $N_j$. Le pare-feu $F_j$ = A ($Ni, Nj$ ) accepte le paquet considère en se basant Sur sa liste de contrôle d'accès L ($F_j$). La decision d'un pare-feu suit la loi de "first match". En effet, la liste L ($F_j$) est explorée en ordre d´encroisant du haut vers le bas jusqu'à atteindre Une règle acceptant le paquet. Dans le cas où il n'existe pas une règle de correspondance, le Paquet est rejeté.

En général, les nœuds d'un réseau sont connectés selon différentes topologies (anneau, bus, Étoile et maillée)

## III.3 Modélisation du comportement :

Notre objectif consiste à élaborer une étude comportementale d'un réseau informatique Un état est défini par les valeurs courantes de toutes les variables du système ainsi que les invariants sur les horloges définis sur cet état. Une transition franchie à partir d'un état, si sa garde est satisfaite. Le franchissement d'une transition implique l'exécution de son bloc d'actions permettant ainsi de changer l'Etat du système**.**

(KHORCHANI, B., HALLE, S. et VILLEMAIRE, R. (2012)).

### III.3.1 Automate "Nœud générateur" :

L'automate, présente dans la Figure III.2, permet de spécifier la topologie du réseau en se basant sur sa taille et son taux de connectivité. La procédure define_topology () crée initialement un réseau ayant une topologie en anneau en fonction de la taille du réseau (nombre de nœuds). Ensuite, en fonction du taux de connectivité du réseau défini comme une variable globale, cette procédure calcule et ajoute les connexions afin de concevoir la topologie maillée choisie. La procédure define_topology () permet également de spécifier les règles de configuration des différents pares-feux d´employés sur le réseau. Elle définit la fonction partielle A qui associe, a toute paire de nœuds connectés, le pare-feu correspondant.

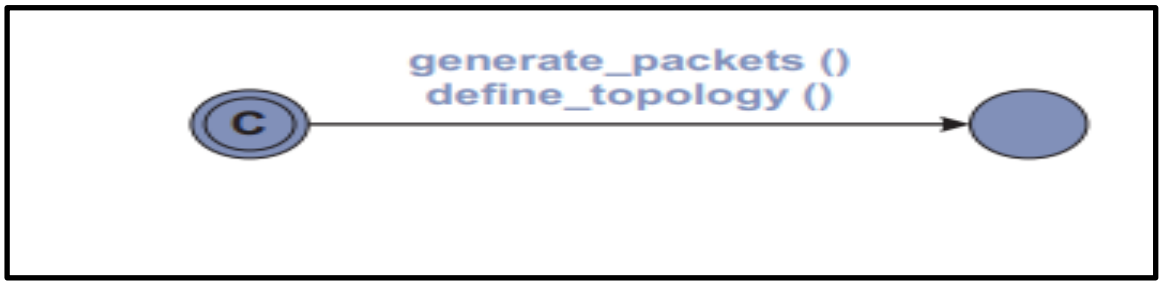



**Figure III.2:** Automate générateur.

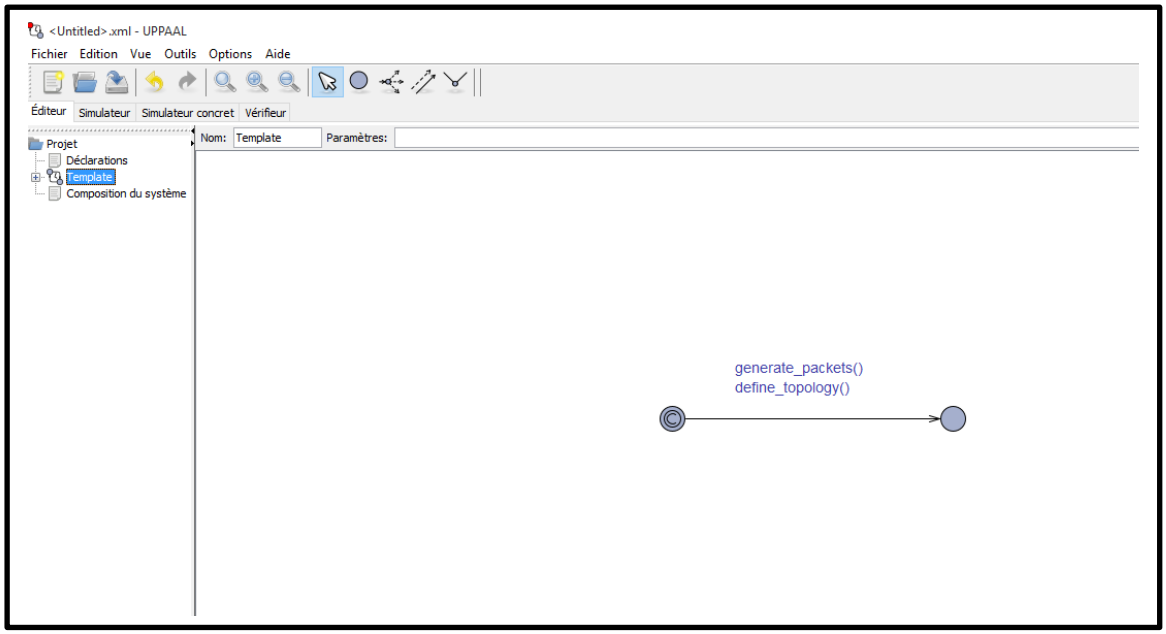



**Figure III.3:** Automate générateur en UPPAAl

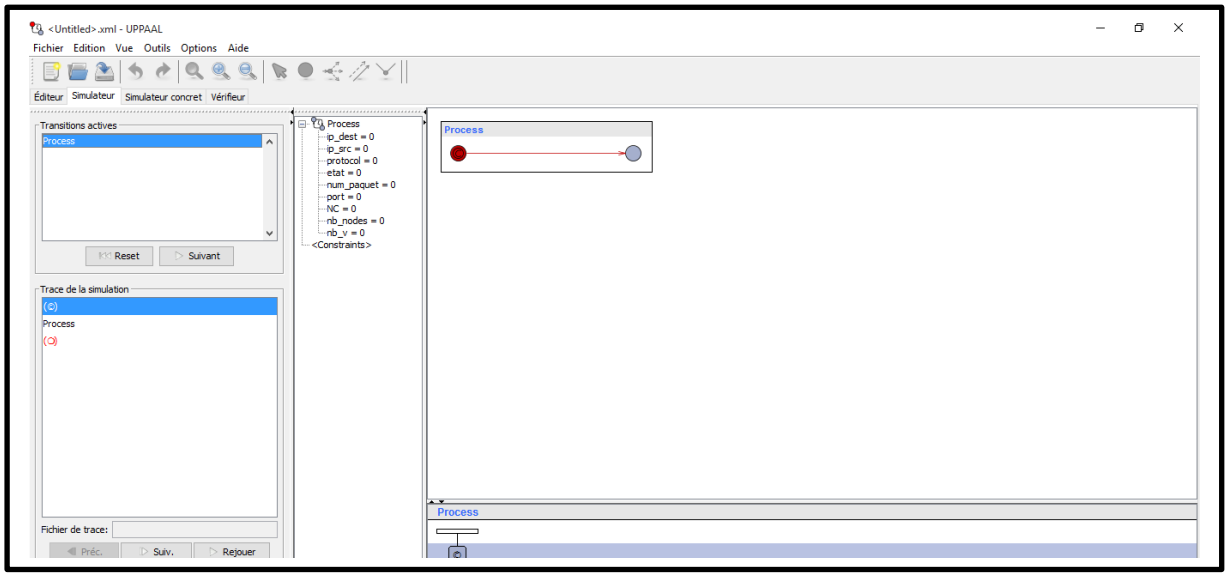



**Figure III.4:** la simulation d'automate générateur.

L'automate "Nœud_générateur" permet aussi de définir une liste exhaustive et figée contenant tous les types de paquets non redondants pouvant être générés en fixant les sources et les destinations dans le réseau à contrôler. Cette liste est générée à l'aide de la fonction generate_packets (). La structure d'un paquet est représentée dans le tableau III.1. Les attributs Ip_src, Ip_dest, port et Protocol sont des données statiques relatives aux paquets en transit et état sont des données dynamiques qui changent en fonction du parcours de paquets dans le réseau.

| Attributs | Definition |
|---|---|
| **int num_paquet** | L'identifiant du paquet. |
| **int ip_src** | L'adresse IP source du paquet. |
| **int ip_dest** | L'adresse IP destination du paquet. |
| **int protocol** | Le protocole (TCP ou UDP). |
| **int port** | Le port (80 ou 25). |
| **int etat** | L'état courant du paquet (0 à l'état initial, 1 si le paquet est acheminé vers la destination avec succès, -1 si le paquet est rejeté et -2 s'il est bloqué dans un nœud terminal). |

**Tableau III .1** : Structure d'un paquet.

### III.3.2 Automate " firewall " :

L'automate, illustre par la Figure 3.3, à comme rôle d'accepter ou de refuser les paquets selon les règles spécifié par l'expert il est paramétré par un identifiant unique " nt (network number) "

La communication entre l'automate firewall et l'automate network est assurée par les canaux de synchronisation accept [NBnetwork] et deny[NBnetwork]

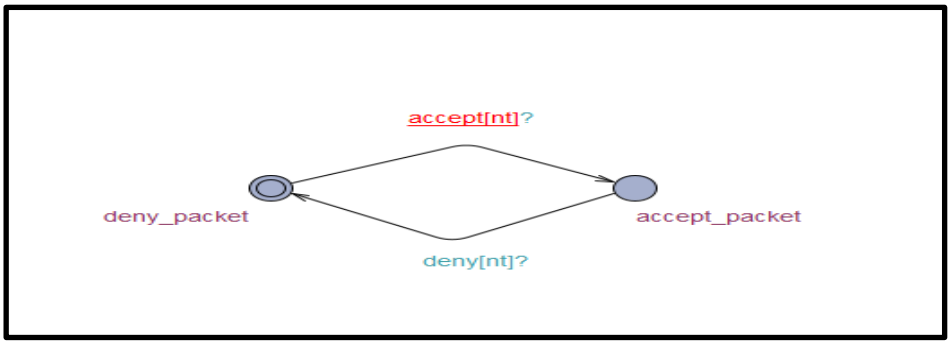



**Figure III.5**: Automate firewall

Ces gardes sont les règles des pare-feu prescrits dans les ACLs et spécifiées par la fonction L. La decision prise par un pare-feu concernant un paquet donné est définie par une fonction basée sur des conditions.

Le paquet est envoyé si et seulement si la fonction allow send il n'a déjà pas été rejeté ou bloqué dans un nœud terminal il y a une connexion dans le réseau entre son emplacement courant et sa prochaine destination.

Si la fonction allow packetAccept est à « true », la procédure update paquet est appelée pour mettre à jour le statut du paquet en fonction des règles du pare-feu qui contrôle le trafic entre la position actuelle et la prochaine destination du paquet.

Si la fonction packetAccept est à « false », le paquet sera abandonné et la procédure Skip ne sont pas bloqués dans un nœud terminal.

### III.3.3 Automate "network " :

L'état initial de l'automate c'est la préparation à la réception des paquets.
Apres la réception des paquets il attend la décision de firewall, s'il accepte il envoie les paquets au nœud de réception sinon il les bloque.

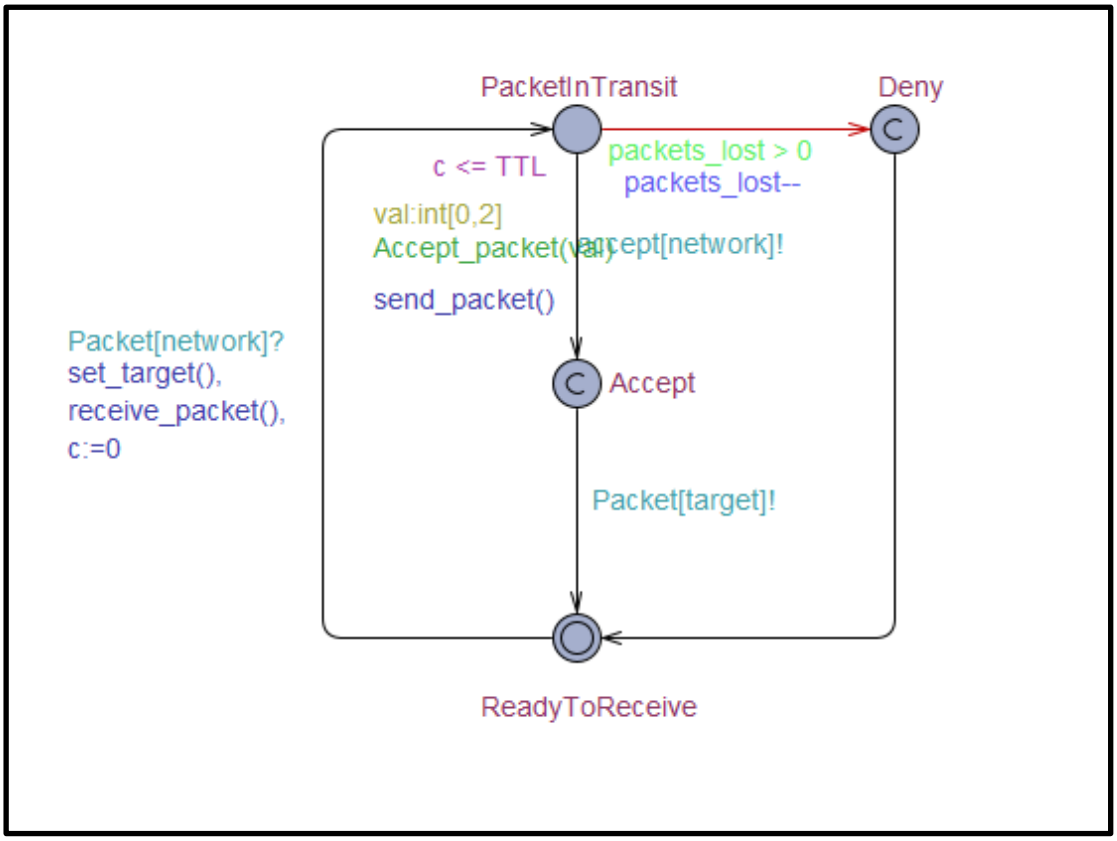



**Figure III.6**: Automate network

### III.3.4 Automate ''Host '' :

L'état initial de l'automate c'est la préparation à la réception des paquets.
Apres la réception des paquets il attend la décision de firewall, s'il accepte il envoie les paquets au nœud de réception sinon il les bloque.
Le noud Host1 envoie des paquets au nœud Host2 . La technique de model
Checking permet d'explorer tous les chemins qui lient les deux nœuds Host1 et Host2 en vérifiant que l'état du paquet (accepte ou rejeté) est le même quel que soit le chemin emprunté.

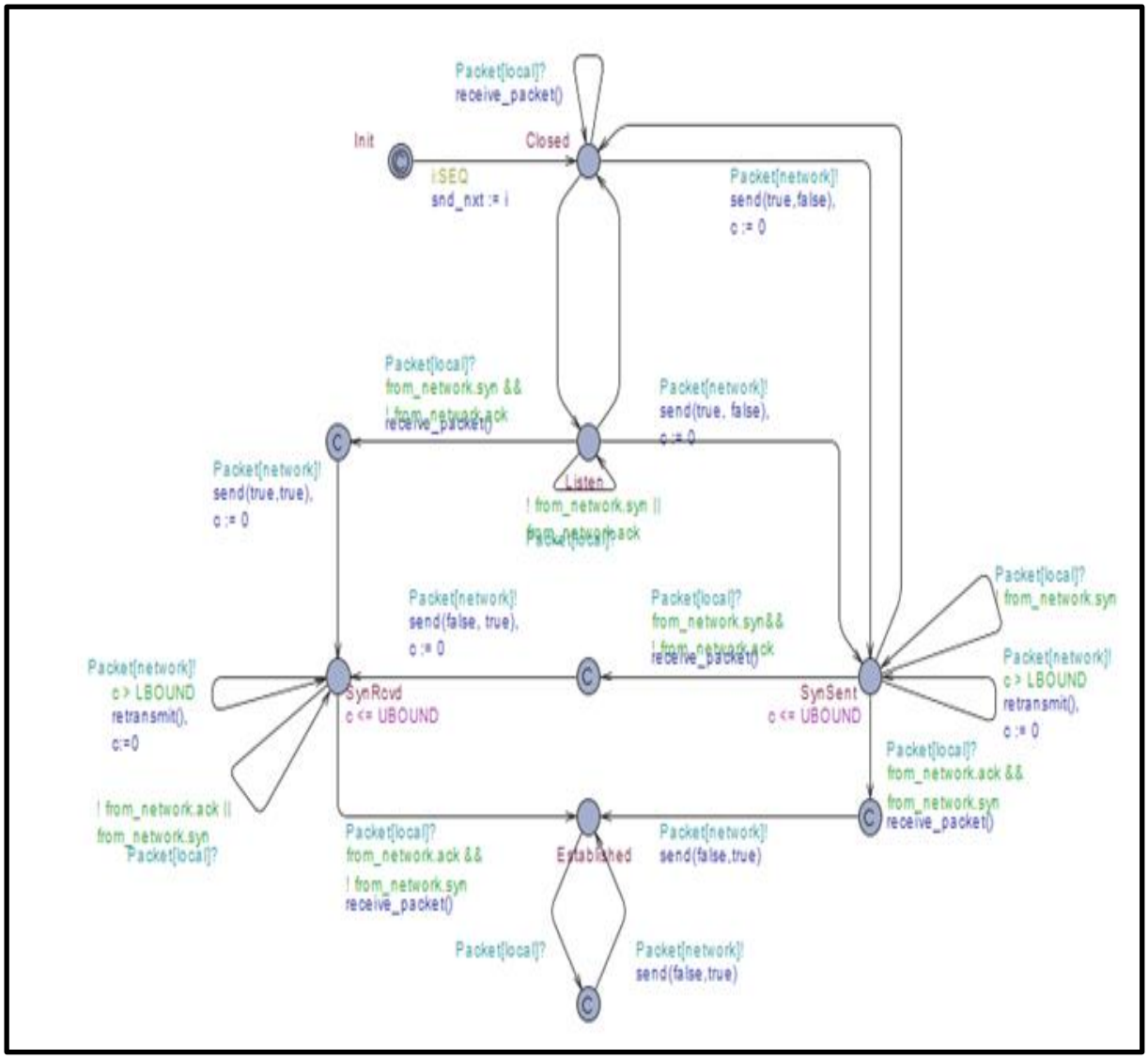



**Figure III.7**: Automate Host

### III.3.5 la simulation :

La Figure III.7 présente un exemple de simulation du réseau ci-dessus, dans le simulateur UPPAAL. Le Simulateur affiche l'ensemble de paquets en transit ainsi que leurs données statiques et dynamiques. Par Exemple, le paquet 0 a été accepté (état=1) par le pare-feu responsable de sa destination .

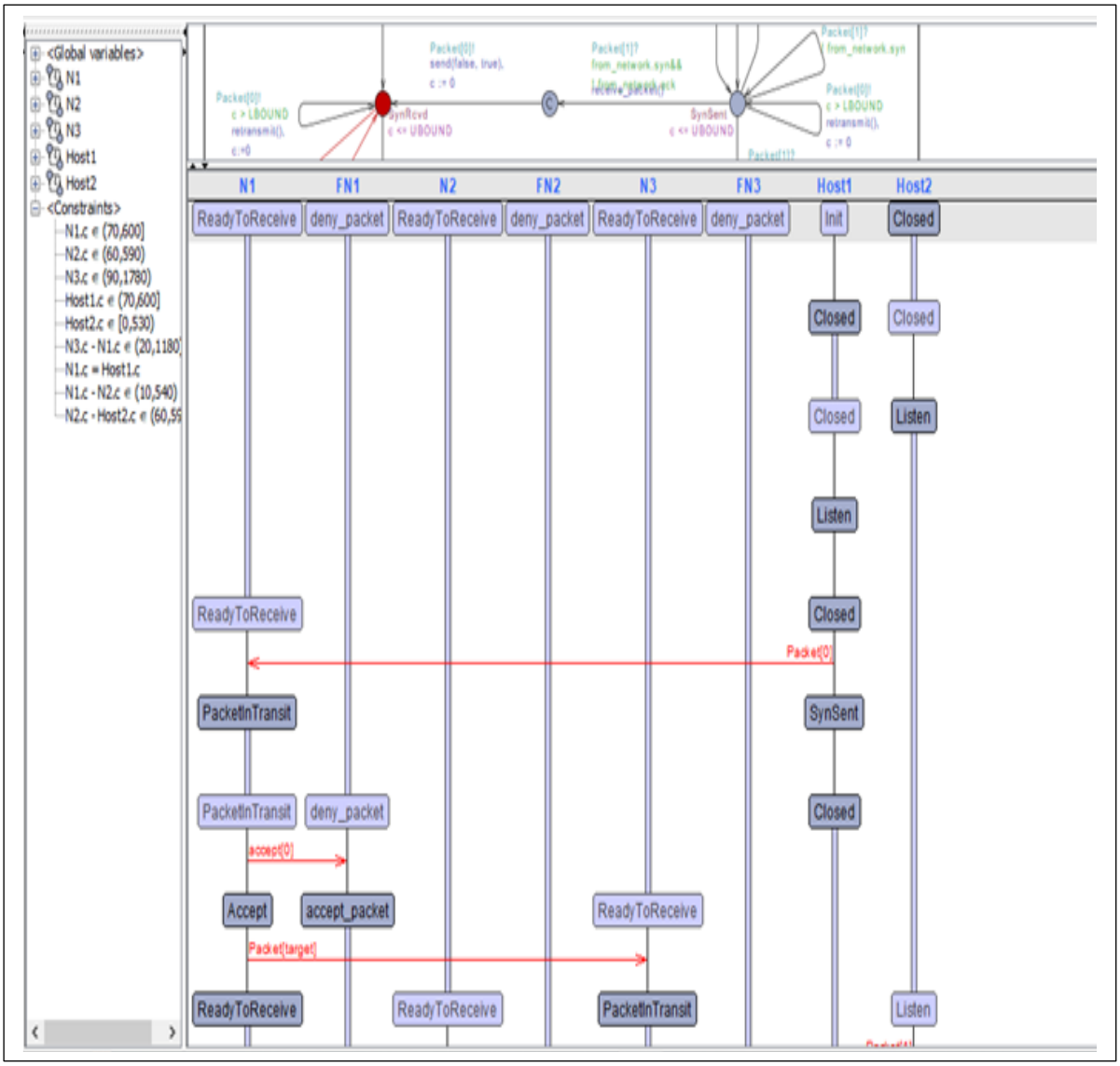



**Figure III.7** : Exemple de simulation du modele.

## III.4 Vérification formelle de la cohérence des pares-feux :

Pour ce faire, nous avons détermine un ensemble de propriétés permettant d'analyser le comportement de réseau. Dans ce qui suit, nous analysons ces 3 classes de propriétés

## III. 4.1 Propriétés assurant le bon fonctionnement du modèle :

Ce groupe de propriétés a été envisagé pour valider le bon fonctionnement du modèle. Le tableau III.3 présente la syntaxe ainsi que les résultats de vérification du modèle vis-à-vis de ces propriétés. La propriété P2 stipule que si l'état out est activé alors tous les paquets sont traites avec suces. L'activation de l'état out annonce la fin d'exécution du modèle. Les propriétés P4 et P5 quant à elles, vérifient la capacité de modèle à utiliser les listes de contrôle d'accès en vue de rejeter les paquets suspects et accepter ceux qui ont accès selon les règles établies

| **Propriétés** | **Syntaxe** | **Résultats** |
|---|---|---|
| **P0:** L'absence des deadlocks | A[] not deadlock | Non-Satisfaite |
| **P1 :** L'absence des deadlocks avant la terminaison du traitement de tous les paquets. | A[](deadlock imply end proc == true) | Satisfaite |
| **P2 :** L'état "out" est active alors le modele a fini le traitement des paquets**.** | A[](for all( i : int[0, nb_p − 1])(end proc == true imply packets[i].is rejected ‖ packets[i].is blocked ‖ (packets[i].is accepted)) | Satisfaite |
| **P3 :** Un paquet peut transiter par un nœud intermédiaire | E<>exists(i:int[0,nb_p−1])(packets[i].nb v > 2) | Satisfaite |
| **P4** : Un paquet parmi les paquets générés peut être rejeté par l'une des listes d'accès. | E <> exists (i: int[0, nb p − 1])packets[i].is− rejected) | Satisfaite |
| **P5 :** Un paquet parmi les paquets générés peut etre accepté par l'une des listes d'accès. | E <> exists (i: int[0, nb p − 1])(packets[i].is− accepted) | Satisfaite |
| **P6 :** Un paquet peut se boquer dans un nœud terminal**.** | E <> exists (i: int[0, nb p − 1])(packets[i].is− blocked) | Satisfaite |
| **P7 :** Les paquets générés peuvent etre achemines sans blocage a leurs destinations | E <> (exists (i : int[0, nb p − 1])(packets[i].is− accepted && packets[i].ip_dest == packets[i]. NC)) | Satisfaite |

**Tableau III.2** Propriétés vérifiant le bon fonctionnement du modèle

### III.4.2 Cohérence de la politique de sécurité globale du réseau :

La cohérence de la politique de sécurité globale du réseau est une propriété fondamentale

Dans la spécification de notre modèle. Cette propriété a été exprimée en utilisant la syntaxe suivante :

**E <> exists (i: int [0, nb p − 1]) packets[i].is rejected imply node(Ip_dest).decision(i) == −1)**

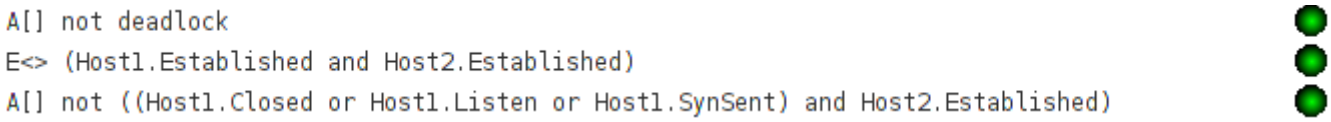
La satisfaction de cette propriété permet de montrer que si un paquet devrait etre accepte par sa destination, il ne doit jamais etre rejeté par un nœud de transit sur un chemin qui

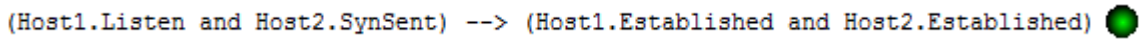
Mené à cette destination. Cette propriété permet de vérifier qu'il n'existe pas des règles en

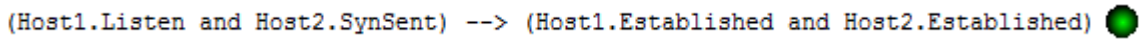
Aval du flux qui sont ombragées par d'autres en amont du flux.

### III.4.3 Propriété de vérification :

La syntaxe d'une telle propriété est :



## III.5. Conclusion :

Dans ce chapitre, nous avons proposé une approche formelle qui permet de vérifier qualitativement le comportement de bout en bout de la sécurité dans un réseau (end-to-end security behavior). Plus précisément, cette approche permet de vérifier la cohérence des pares-feux distribues en identifiant les anomalies inter-pare-feux, en particulier, l'incohérence de croisement de chemins. L'approche proposée est basée sur la technique de model checking et elle a été implémentée en utilisant l'outil UPPAAL.

# CONCLUSION GENERALE

Dans le présent mémoire, nous nous sommes intéressés à l'utilisation des méthodes formelles comme cadre solide d'aide à la configuration et à la validation des réseaux informatiques. Un réseau informatique est un ensemble d'équipements reliés entre eux pour offrir des services très diversifiés tels que la communication entre les différents utilisateurs du réseau et le partage des ressources. L'un des défis majeurs des réseaux informatiques est d'assurer la sécurité des données.

Plusieurs dispositifs, tels que les IDS et les pares-feux, ont été mis en œuvre pour contrôler les échanges d'information et d´détecter les intrusions ou menaces susceptibles d'entraver la sécurité des données. Parmi les mécanismes de sécurité les plus largement d´éployés, ce m´emoire s'est intéressé aux pares-feux et plus précisément à la conception d'approches formelles d'aide à la configuration de pares-feux.

L'approche permet d'analyser le comportement qualitatif d'un réseau. Elle s'appuie sur la technique de model checking de l'outil UPPAAL, pour vérifier formellement la consistance d'un réseau a pares-feux de bout en bout (end-to-end security behavior) vis- a-vis d'un objectif de sécurité du réseau. Elle permet aussi de détecter les incohérences de configuration des pares-feux distribués, notamment, l'incohérence de croisement de chemins. Pour atténuer le problème d'explosion combinatoire inhérent au model checking, nous avons proposé deux modèles spécifiant le comportement d'un réseau. Afin de valider l'impact de ces abstractions sur les performances du processus de vérification, nous avons élaboré une étude expérimentale qui montre en fonction de la taille et de la topologie du réseau, le temps et l'espace mémoires nécessaires à la vérification des propriétés. Les résultats expérimentaux montrent un gain significatif en temps et en utilisation m´emoire. En effet, la deuxième abstraction permet de vérifier des réseaux de plus grandes tailles que la première.

# Références bibliographiques

**Techno-science :** (Ensemble dans lequel coopèrent institutions, chercheurs et ingénieurs afin de mettre en œuvre, pour des applications précises, les ressources de la science et de la technique).

**WIKIPEDIA.**

**GUEFFAZ, M. (2012). Scale Sem :** model checking et web sémantique. Thèse de doctorat, Université de Bourgogne.

**Clarke et al., (2012) :** article sur le problème d'explosion combinatoire des états.

**Gueffaz, (2012) :** article sur les solutions d'explosion combinatoire.

**Philipe schnocbelen, (2000) :**PDF sur la Comparaison des performances de UPPAAL, HYTECH et KRONOS**.**

**Fièrement propulsé par WordPress :** platform Interface d'édition de modèle **(model checking).**

**Behrmann et al., (2006) :** article sur l'architecture de UPPAAL**.**

**Larsen et al., (1997) :** article sur l'Architecture de UPPAAL.

**Zennon, (2004) :** cet article sur Les Avantages et Les Inconvenant De Model Checking**.**

**Biermann et al. (2001) :** cet article sur améliorer la sécurité Les réseaux informatiques tels que les systèmes de détection d'intrusion (IDS**).**

**Cheswick et al. (2003) :** cet article sur les pares-feux.

**Redes de Comunicação :** site official pour les Réseaux de communication**.**

**Pujolle et Salvatori, (2010) :** cet article sur les types de réseau**.**

**BILLAMI et BENDAHMANE (2014) :** cet article sur les topologies de réseau**.**

**Jean-Luc GATOUX :** cet PDF sur la Structure d'un paquet.

**IVISION : est** spécialisée dans l'infogérance et l'hébergement informatique professionnel et propose ses services à destination des TPE et des PME.

# Références bibliographiques

**GOUDA, M. G. et LIU, X.-Y. (2004).** Firewall design : Consistency, completeness, and compactness. Distributed Computing Systems, 2004. Proceedings. 24th International Conference on. IEEE, 320–327

**IPE informatique :** IPE est une entreprise informatique à Paris. Nos équipes vous apportent conseils personnalisés couvrant l'ensemble de vos projets informatiques : maintenance informatique, infogérance, sauvegarde de données, sécurité informatique, plan de reprise d'activité à Paris et en Ile-de-France**.**

**Fall, (2010) :** PDF sur l'emplacement du pare-feu**.**

**Marfall N'Diaga Fall, (2010) :** Sécurisation formelle et optimisée de réseaux informatiques.

**Adnane EL KABBAL (2005) :** mémoire sur un système de type pour l'analyse des pare_ feu.

**ABDOUNE Mohamed Oulhadj HAMIDOUCHE Ahcène** : Mémoire de fin d'étude sur Approche algébrique pour la prévention d'intrusions.

**JEFFREY, A. et SAMAK, T. (2009):** Model checking firewall policy configurations. Policies for Distributed Systems and Networks, 2009. POLICY 2009. IEEE International Symposium on. IEEE, 60–67.

**KHORCHANI, B., HALLE, S. et VILLEMAIRE, R. (2012).** Firewall anomaly detection ´ with a model checker for visibility logic. Network Operations and Management Symposium (NOMS), 2012 IEEE. IEEE, 466–469.

# Références bibliographiques