UNIVERSITE **P**AUL **S**ABATIER

Master 2 Informatiques et Télécommunications

Parcours : Réseaux et Télécoms

Rapport de Stage

Apprentissage hors ligne de chroniques pour les réseaux de communications autonomes

Khalil ZOUAOUI

Laboratoire : LAAS-CNRS

Responsable de stage : Audine SUBIAS

Encadré par : Audine SUBIAS, José Lisandro AGUILAR CASTRO

Adresse : 7 Av. Colonel Roche 31077 Cedex 4 Toulouse, France

LAAS-CNRS

Toulouse, 12 Septembre 2011

Résumé

Le travail réalisé au cours de ce stage porte sur la proposition d'une approche basée sur la reconnaissance de chroniques dans le but de réaliser une estimation de la situation de communication dans un système en réseau. La difficulté principale de ce type d'approche est la mise en place des chroniques, la solution proposée dans ce travail est l'obtention des chroniques par apprentissage hors ligne.

Chaque chronique ou motif temporel traduit l'évolution du système de communication au cours du temps, et l'obtention de ces motifs constitue l'objectif principal de notre travail. Dans ce cadre, nous avons proposé de construire ces chroniques à partir des données historiques du système. Au cours de ce stage nous avons développé un algorithme d'apprentissage hors ligne de chroniques et nous avons appliqué le test de reconnaissance des motifs obtenus sur différentes situations du système de communication.

Abstract

The work done during this internship focuses on the proposal of an approach based on the recognition of chronicle in order to make an estimate of the situation of communication in a networked system. The main difficulty of this approach is the development of chronicle, the solution proposed in this work is obtaining the chronicle by off-line learning.

Each chronicle or temporal pattern reflects the temporal evolution of the communication system over the time, and obtaining these patterns is the main objective of our work. In this context, we proposed to construct the chronicle from the historical data of the system. During this internship we have developed an off-line learning algorithm of chronicle and we applied the test of recognition of patterns obtained on different situations of the communication system.

Avant d'entamer ce rapport de master, je tiens à adresser mes sincères remerciements à tous les membres du LAAS-CNRS et à mes enseignants et responsables du master Informatique et Télécommunications de l'Université Paul Sabatier qui ont contribué au bon déroulement de ce projet.

Je tiens à remercier particulièrement, Monsieur José Lisandro AGUILAR CASTRO et Madame Audine SUBIAS, pour m'avoir donné l'opportunité de pouvoir travailler sur ce sujet de recherche mais aussi bien pour toute l'aide et le soutien qu'ils m'ont apporté tout au long de mon stage. Je les remercie également de m'avoir offert l'opportunité de découvrir le monde de la recherche à travers mon stage de master.

Je souhaite aussi vivement remercier Monsieur Christophe CHASSOT, pour ses remarques pertinentes sur mon travail, pour ses conseils avisés et les réflexions que nous avons pu mener tout au long de ces mois. Je lui suis également reconnaissant pour sa confiance et ses nombreux encouragements.

J'exprime ma profonde reconnaissance à Madame Louise TRAVE-MASSUYES, pour m'avoir fait profiter de son expérience et pour l'intérêt qu'elle a porté à mon travail.

Mes remerciements vont aussi au stagiaire Code DIOP avec qui j'ai partagé certaines tâches de ce travail et échangé des idées dans le cadre de mon projet.

J'adresse mes plus sincères remerciements à tous mes chèr(e)s collègues de travail (Rim, Nouha, Amine, Aymen, Mahdi, Hatem, Ghada et Emna) pour l'ambiance amicale et pour les moments de détentes partagés.

Enfin, qu'il me soit permis d'exprimer ma reconnaissance à mon cher ami Moutie, qui a partagé avec moi cette année universitaire et qui a été mon soutien le plus efficace, à ma chère Ines et à toute ma famille pour leurs sacrifices, leur soutien sans faille et leur compréhension.

Table des matières

Table de figures

Introduction Générale

La nature dynamique des futurs réseaux mobiles (sans fil, ad hoc, ...) ouvre des défis importants, notamment pour satisfaire les exigences de qualité de services (QoS) des applications qui y seront distribuées. L'évolution des ressources réseaux d'une part et des besoins applicatifs d'autre part, exigent la disponibilité de services et protocoles de communication capables de s'adapter de manière proactive ou réactive et autonome. Pour guider l'auto-adaptation de tels services et protocoles, les futurs systèmes de communication devront disposer d'une estimation explicite de la situation de communication.

Une approche envisagée pour réaliser cette estimation est une approche basée sur la reconnaissance de chroniques. En effet, une chronique est un motif temporel qui décrit l'évolution d'un système donné au cours du temps. Une situation ou un état du système étudié est reconnu si la chronique qui le décrit a été identifiée dans la phase de surveillance du comportement du système.

La difficulté principale dans ce type d'approche est la mise en place des chroniques. Une des solutions possibles est d'obtenir les chroniques par apprentissage. La mise en place et le test de cette approche fait l'objet de ce travail, au cours duquel nous allons construire des chroniques par apprentissage décrivant certains états d'un système de communication. La deuxième phase du travail est le test de reconnaissance des chroniques obtenues.

Le présent rapport s'articule au tour de quatre chapitres. Le premier chapitre introduit le contexte dans lequel s'est déroulé ce stage : organisme d'accueil et présentation générale du sujet. Le deuxième chapitre présente l'état de l'art à travers la description du concept de chronique et l'approche de diagnostic à base de reconnaissance de chroniques. Le troisième chapitre détaille la solution que nous proposons et les nouveaux concepts utilisés dans l'implémentation de l'algorithme d'apprentissage. Le quatrième chapitre est consacré au test de reconnaissance de chroniques générées par notre algorithme. Finalement, nous terminerons par une conclusion où nous décrivons quelques perspectives à notre travail.

Chapitre 1

Présentation Générale

I- Contexte du stage
II- Présentation du sujet

Au cours de ce chapitre nous allons situer le stage dans son cadre général puis nous présenterons l'organisme d'accueil. Ensuite, nous allons présenter le contexte du sujet ainsi que l'objectif et les problématiques qu'il présente.

1.1 Contexte du Stage

Ce stage s'inscrit dans le cadre de la deuxième année de mastère Informatique et Télécommunication parcours Réseaux et Télécommunications, à l'Université Paul Sabatier Toulouse III. J'ai été amené à réaliser ce stage de 6 mois au sein du Centre National de la Recherche Scientifique français (CNRS) au Laboratoire d'Analyse et d'Architectures des Systèmes (LAAS) dans le groupe Diagnostic Supervision et Conduite (DISCO).

1.1.1 Organisme d'accueil

Le LAAS est une unité propre de recherche du CNRS créée en 1967 situé sur le campus scientifique de Rangueil, à Toulouse en Haute-Garonne. Avec plus de 600 chercheurs, ingénieurs, techniciens et administratifs, c'est le plus gros laboratoire CNRS. Ses activités s'articulent autour de 4 pôles qui animent l'activité de 19 groupes de recherche rassemblée sous le terme « systèmes » : elles vont de la robotique aux microsystèmes, en passant par la sécurité informatique, la sûreté de fonctionnement et les nanotechnologies.

1.1.2 Groupe d'accueil

Le groupe DISCO développe des outils de Diagnostic et Supervision pour les systèmes dynamiques complexes, pouvant inclure l'homme. Ce groupe met l'accent sur le caractère qualitatif de la supervision sans ignorer les aspects continus des systèmes. De ce fait, les systèmes hybrides et l'interface entre signaux continus et leur interprétation sous forme plus abstraite à événements discrets correspondent à une grande partie de ses travaux. La nature qualitative des connaissances et l'incertitude entachant les données amènent les membres de DISCO à faire appel à des formalismes qualitatifs et symboliques trouvant leurs origines en Intelligence Artificielle comme le Raisonnement Qualitatif et la modélisation floue et neuronale, de même qu'à des méthodes de classification et d'apprentissage. Les axes de recherche du groupe DISCO sont :

- Diagnostic et décision — Approches à base de modèles
- Surveillance et Supervision par des méthodes d'apprentissage

1.1.3 Cadre du stage

Ce stage s'inscrit dans le cadre du projet DAISY (Diagnosis for AdaptIve Strategies in collaborative Systems) mené en collaboration entre deux groupes de recherche au sein du LAAS qui sont le groupe DISCO et le groupe OLC (Outils et Logiciel pour la Communication). DAISY s'intéresse à l'auto-adaptation des systèmes de communication autonomes et s'inscrit dans le programme scientifique ADREAM (Architectures Dynamiques Reconfigurables pour systèmes Embarqués Autonomes Mobiles) qui a comme axe principal l'intelligence ambiante.

1.2 Présentation du sujet

1.2.1 Contexte général

Une des solutions pour satisfaire les exigences de qualité de service repose sur la disponibilité de services et protocoles de communication capables de s'adapter de manière proactive ou réactive et autonome. Pour guider l'auto-adaptation de tels services et protocoles, les futurs systèmes de communication devront disposer d'une estimation explicite de la situation de communication. L'approche envisagée pour réaliser cette estimation est une approche basée sur la reconnaissance de chroniques. La difficulté principale de ce type

d'approche est la mise en place des chroniques. Une des solutions est d'obtenir les chroniques par apprentissage.

Notre contribution a été réalisé dans le cadre du projet DAISY qui a pour but de développer un système de détection et d'analyse de la dégradation de QoS d'un système de communication en utilisant des chroniques (motifs temporels), afin de guider l'auto-mécanisme d'adaptation du système de communication, et dans le cadre de ce projet l'architecture suivante a été proposée :

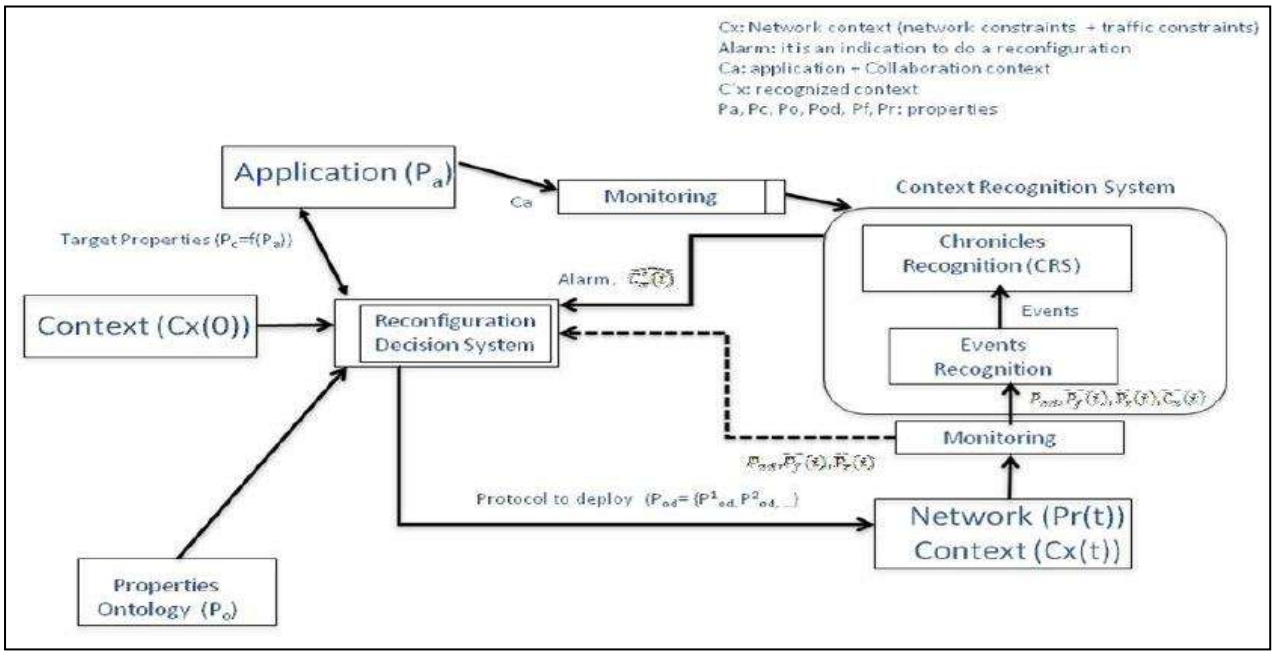


Figure 1 : Architecture du système de communication autonome proposé par DAISY

Dans cette architecture, le système de décision de reconfiguration est un mécanisme de raisonnement basé sur les propriétés du réseau et utilise comme entrées les exigences de QoS des applications, le contexte actuel et les alarmes générées par le système de reconnaissance.

Le système de reconnaissance est un système de supervision et d'analyse qui détecte les problèmes de performance du système de communication, afin de permettre au système de communication la possibilité d'auto-configuration, d'auto-optimisation, d'autoprotection, d'auto-adaptation et d'autoréparation.

1.2.2 Objectif et problématique

L'approche envisagée pour réaliser une estimation sur l'état du réseau est une approche basée sur la reconnaissance de chroniques. Les chroniques sont des motifs temporels

décrivant des comportements ou situations devant être reconnues. La difficulté principale de ce type d'approche est la mise en place des chroniques. Une des solutions est d'obtenir les chroniques par apprentissage. Ce stage porte sur ce problème d'apprentissage des chroniques. L'objectif est d'étudier les différentes approches d'apprentissage de chroniques existant dans la littérature de manière à proposer ensuite une approche adaptée au problème d'auto-adaptation des protocoles de communication.

1.3 Conclusion

Au cours de ce chapitre nous avons donné le cadre général du stage puis nous avons présenté le contexte du sujet ainsi que l'objectif et les problématiques qu'il présente.

Chapitre 2

État de l'art

I- Les chroniques
II- Diagnostic à base de reconnaissance de chroniques
III- Apprentissage
IV- Application

Au cours de ce chapitre, nous allons définir le concept de chronique en premier lieu, puis l'approche de diagnostic à base de reconnaissance de chronique. Finalement, nous présenterons quelques applications de cette approche à travers des exemples issus de divers domaines.

2.1 Les chroniques

Dans cette partie nous présentons le formalisme « Chronique » en s'appuyant sur la définition donnée par C. Dousson [1].

2.1.1 Définition

Une chronique est un motif temporel destiné à représenter un schéma d'évolution d'un système, il est composé principalement d'un ensemble d'événements et de contraintes temporelles entre leurs dates d'occurrences. Une chronique C est présentée comme suit : $C = \{S, T\}$ avec S l'ensemble des événements et T l'ensemble des contraintes. Soit l'exemple suivant :

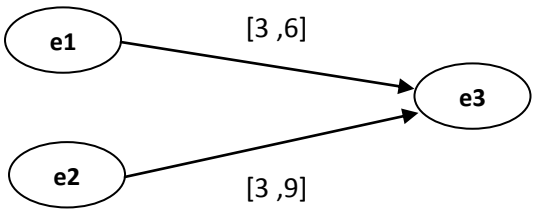


Figure 2 : Exemple de chronique simple

Cette chronique décrit le scénario suivant : « L'événement e3 » se produit après 3 à 6 unités de temps de « l'événement e1 » et 3 à 9 unités de temps après « l'événement e2 ». Dans cet exemple nous avons $C = \{S, T\}$ avec :

- $S = \{e1, e2, e3\}$
- $T = \{\, t_1 < t_3 \;\; avec \;\; 3 \leq t_3 - t_1 \leq 6 \;,$
  $t_2 < t_3 \;\; avec \;\; 3 \leq t_3 - t_2 \leq 9 \;\}$

Les séquences d'événements suivantes sont des exemples qui satisfont les contraintes de la chronique C :

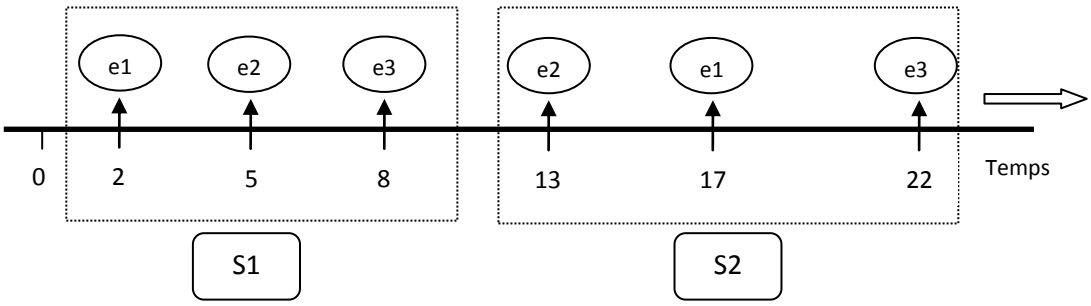


Figure 3 : Exemple de flux d'événements

Les séquences « *S1* » et « *S2* » sont appelées instances de la chronique C.

Un modèle de chronique peut inclure aussi des assertions et des actions à exécuter lorsque le système peut conclure que l'environnement a bien suivi l'évolution décrite par cette chronique, c'est-à-dire que la reconnaissance a eu lieu ou bien qu'une est envisageable dans un certain délai fixé par l'utilisateur [1].

2.1.2 Langage de chroniques

Il existe plusieurs langages de chronique tels que celui définit par O. Bertrand [3] et dans lequel les événements sont présentés par des variables et les relations entre ces événements sont des opérateurs. Dans le cadre de notre stage nous nous sommes appuyés sur le langage d'expression de chroniques proposé par C. Dousson [1]. Dans ce langage, la construction de chroniques se fait à l'aide d'un ensemble de prédicats.

i. Le prédicat ***event***

Syntaxe : *event* $(a\,, t)$

Correspond à l'occurrence d'un événement a à une date donnée t

ii. Le prédicat ***noevent***

Syntaxe : *noevent* $(a\ , [t_1, t_2[\ )$

Correspond à la négation du prédicat *event* sur un intervalle de temps donné, c.-à-d. aucun événement a n'a eu lieu entre t_1 et t_2.

iii. Le prédicat ***occurs***

Syntaxe : *occurs* $((n, m), a\ , [t_1, t_2[\ )$

C'est un prédicat introduit récemment [2], il traduit l'apparition d'un événement a, n à m fois entre les instants t_1 et t_2.

A titre d'exemple la chronique de la Figure 2 est décrite de la manière suivante:

Chronicle C

$\{$ *event* $(e1, t_1)$

event $(e2, t_2)$

event $(e3, t_3)$

$t_3 - t_1\ in\ [3,6]$

$t_3 - t_2\ in\ [3,9]\ \}$

2.2 Diagnostic à base de reconnaissance de chroniques

2.2.1 Principe de la reconnaissance

Considérons :

i. Un ensemble d'événements datés, cet ensemble constitue les données d'entrée du système de reconnaissance.

ii. Un ensemble de contraintes sur le domaine. Dans le cas de reconnaissance de chronique, ces contraintes sont des intervalles temporels entre les dates d'occurrences des événements.

iii. Un ensemble de formules ou schéma général. Dans notre cas, ce sont les modèles de chroniques sans tenir compte des contraintes temporelles entre leurs dates d'occurrence.

Reconnaitre, c'est expliquer les événements d'entrée (i) à l'aide des schémas temporels (iii) tout en respectant les contraintes du domaine (ii) [1].

Revenant sur l'exemple précédent (figure 2) de la chronique C , l'ensemble $S = \{e1, e2, e3\}$ représente le schéma général (iii), l'ensemble $T = \{ t_1 < t_3,\ t_2 < t_3 \}$ représente les contraintes du domaine (ii), et finalement les séquences *S1* et *S2* (figure 3) représentent les données d'entrée du système de reconnaissance (i).

2.2.2 Système de reconnaissance de chroniques

Une chronique est un ensemble d'événements avec des contraintes sur leurs dates d'occurrence, chaque chronique caractérise une situation donnée en se basant sur les conséquences observées. Dans notre contexte, une situation anormale du système de communication est caractérisée par un ensemble de chroniques, chaque chronique décrit la situation d'intérêt selon l'évolution du système. Par exemple, une situation de perte de données a plusieurs causes telles que la congestion du réseau, la perte de connectivité, le changement d'interface d'accès ou aussi les pertes dues au bruit. Chacune de ces sous-situations ou états doit être représenté par au moins une chronique.

Un système de reconnaissance de chronique est donc constitué d'une base de chroniques, ce système reçoit un flux d'événements observés et vérifie si tout ou une partie de ce flux correspond à une ou plusieurs chroniques de la base. Une chronique est dite reconnue, si tous les événements qui la construisent sont observés tout en respectant les contraintes temporelles entre leurs dates d'occurrences.

2.3 Apprentissage

La mise en place de la base de chroniques apparait clairement comme un point délicat. Plusieurs approches existent notamment des approches par apprentissage. Ce point constitue l'objet de notre travail et nous y reviendrons dans la partie contribution. L'état de l'art des

algorithmes d'apprentissage a été fait dans des travaux antérieurs et l'approche adoptée dans ce stage m'a été proposée suite aux résultats de ces études. Un aperçu de ces travaux sera présenté dans le troisième chapitre.

2.4 Applications

Le concept de chronique a été utilisé dans le projet AUSTRAL afin d'analyser une séquence d'alarmes émise par les postes dans un réseau français de distribution moyenne tension [4]. Il est également utilisé dans le projet GASPAR, pour analyser les alarmes déclenchées par les équipements dans un réseau de télécommunications [5]. Les chroniques on été exploitées aussi dans certains sous-tâches du projet WITAS [6, 7] pour représenter les comportements de voitures dans le trafic routier supervisé.

De nouvelles applications à base de reconnaissance temporelle ont été proposées dans le domaine médical comme le suivi des symptômes de l'hépatite [8] ou la détection d'arythmie cardiaque [10]. Récemment les chroniques ont été utilisées dans le cadre du diagnostic des défauts dans les services web [4]. Un aperçu plus complet des applications de reconnaissance de chroniques est présenté dans [11].

2.5 Conclusion

Dans ce chapitre nous avons présenté le concept « chronique » en premier lieu, et l'approche de reconnaissance de ces motifs temporels en deuxième lieu. Finalement, nous avons donné un aperçu sur les domaines d'application des chroniques au travers de quelques projets et travaux récents. Dans le chapitre suivant, nous allons présenter notre contribution et les travaux réalisés pour adapter cette approche à notre besoin.

Chapitre 3

Contribution

I- Méthodologie
II- Prétraitement
III- Classification
IV- Algorithme d'apprentissage de chronique

Dans de ce chapitre nous allons présenter notre contribution et les travaux réalisés pour adapter l'approche d'apprentissage de chronique pour l'identification et la reconnaissance des certaines situations du fonctionnement d'un système en réseau. Notre objectif est de construire des chroniques par apprentissage, nous intéresserons en particulier à la possibilité de mettre en place des chroniques paramétrées de manière à prendre en compte le contexte dans lequel les communications sont établies. Dans la suite de ce chapitre nous présenterons la méthodologie de notre travail puis nous détaillerons les étapes de notre algorithme d'apprentissage de chroniques.

3.1 Méthodologie

L'objectif de notre travail est de construire des chroniques à travers plusieurs flux de données caractérisant une situation particulière du système de communication. Dans ce stage nous avons considéré la situation de perte de paquets par congestion du réseau, donc nous voulons caractériser cette situation par une ou plusieurs chroniques. Notre travail est divisé en quatre grandes étapes présentées dans le schéma suivant et expliquées par la suite.

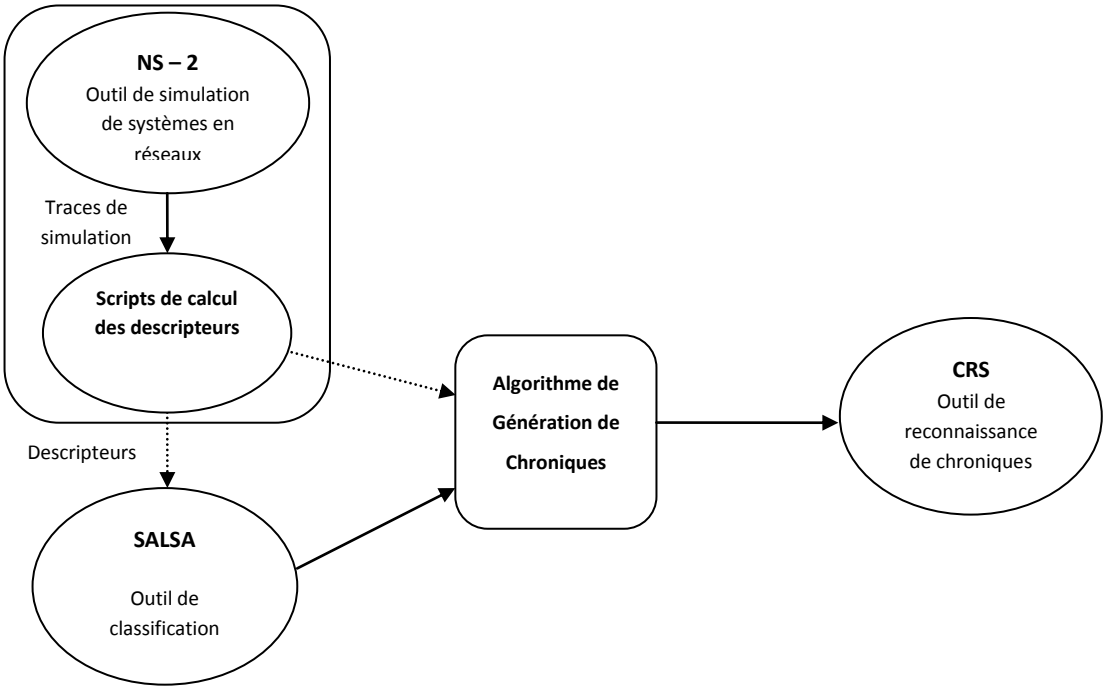


Figure 4 : Principe général de la méthode d'apprentissage de chroniques

Dans une première étape nous avons utilisé l'outil NS-2 (Network Simulator) pour simuler le comportement d'un système en réseau dans lequel se produit un problème de congestion entraînant la perte de paquets. Les fichiers de traces contenant les résultats de simulation ont ensuite été traités de manière à en extraire les informations caractéristiques de la situation étudiée que sont les descripteurs et leurs valeurs et à constituer des fichiers de données, ces deux sous étapes constituent la phase de prétraitement.

Dans une deuxième étape nous avons classé les fichiers de données à l'aide de l'outil de classification de données SALSA (Situation Assessement using LAMDA claSsification Algorithm) développé au sein du groupe DISCO du LAAS-CNRS. Cette classification donne une représentation en termes de classes du comportement de la situation de congestion analysée.

Une troisième étape est l'étape d'apprentissage des chroniques proprement dite. A partir des résultats des étapes précédentes les différents éléments permettant de définir les chroniques sont mis en place, à savoir les événements et les contraintes temporelles.

Enfin, la quatrième étape est une étape de test qui s'appuie sur l'outil de reconnaissance de chroniques CRS (Chronicle Recogniton System).

3.2 Prétraitement

La première phase dans notre travail consiste à préparer les informations caractéristiques de la situation de congestion du réseau, et pour se faire nous avons procédé en deux étapes :

3.2.1 Simulation de la situation de congestion du réseau

En utilisant l'outil Network Simulator V2 et le langage OTCL (Object Tools Command Language), nous avons simulé un réseau filaire composé de quatre nœuds, dont deux émetteurs (nœud 0 et nœud 1), un intermédiaire (nœud 2) et un récepteur (nœud 3) comme indiqué sur les deux figures suivantes :

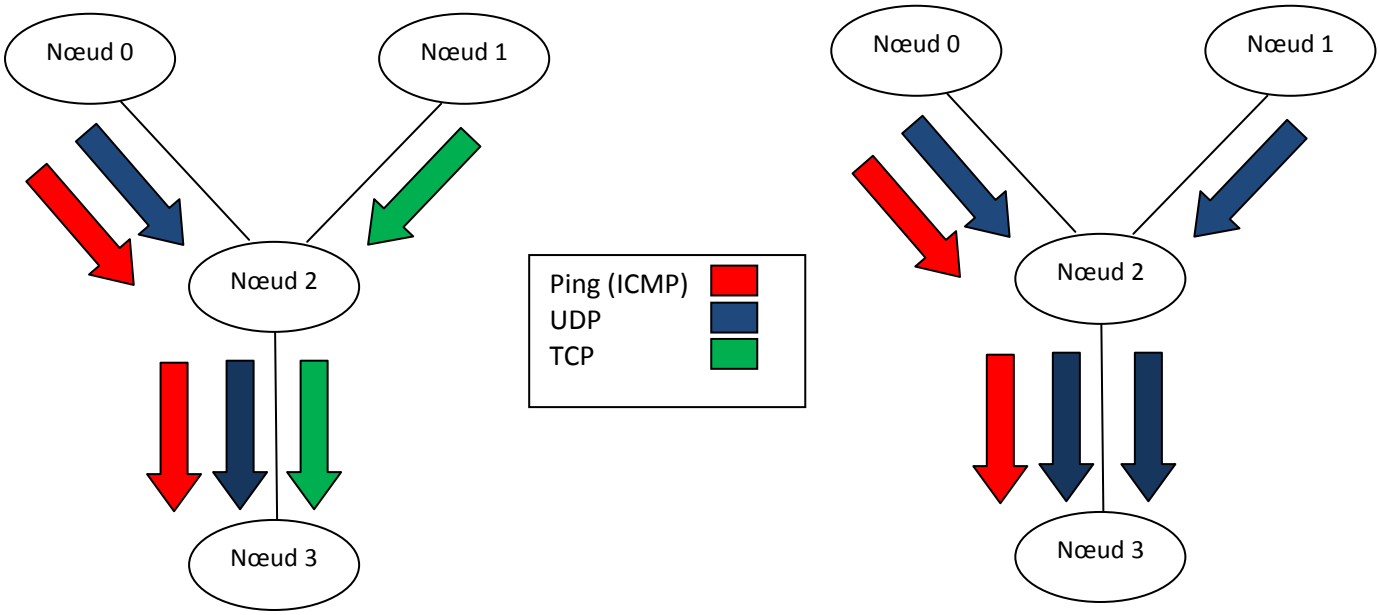


Figure 5 : Architecture physique du réseau simulé

En effet, nous avons simulé deux scénarios identiques avec deux variantes au niveau du flux généré par le nœud 1. Ce nœud génère un flux TCP[1] (*Transmission Control Protocol*) dans une première simulation, et un flux UDP[2] (*User Datagram Protocol*) dans la deuxième. Les résultats de ces deux simulations sont très différents. En effet, TCP assure le contrôle de congestion, c.-à-d. essai de régler la situation du réseau quand une congestion se produit et

[1] TCP : *Transmission Control Protocol.* Protocole orienté connexion de la couche transport du modèle TCP/IP
[2] UDP : *User Datagram Protocol*. Protocole non orienté connexion de la couche transport du modèle TCP/IP

baisse la fréquence d'émission. Par contre, un nœud UDP continue la diffusion sans tenir compte des pertes provoquées. La simulation avec deux flux TCP générés par le nœud 0 et le nœud 1 n'a provoqué aucune perte grâce au mécanisme de contrôle de congestion et la retransmission des paquets perdus que le protocole TCP assure, pour cela nous n'avons pas traité ce troisième cas.

Les deux scripts OTCL développés pour ces deux simulations décrivent le scénario suivant :

- Durée de la simulation 20 secondes
- « Nœud 0 » émet un paquet Ping (ICMP[3]) de taille 64 octets chaque 0.1 seconde
- « Nœud 0 » et « Nœud 1 » génèrent des flux UDP/TCP (*figure 5 à gauche*) ou UDP/UDP (*figure 5 à droite*) en trois temps de la simulation
- Un premier flux UDP/TCP ou UDP/UDP qui commence à $t = 0.5\ s$ et se termine à $t = 5.5\ s$
- Un deuxième flux UDP/TCP ou UDP/UDP de $t = 7\ s$ à $t = 12.5\ s$
- Un troisième et dernier flux UDP/TCP ou UDP/UDP de $t = 14\ s$ à $t = 18\ s$

La congestion du réseau simulé est provoquée par les flux UDP/TCP ou UDP/UDP émises respectivement par le nœud 0 et le nœud 1, donc le scénario précédent représente trois situations successives de congestion. Les paquets Ping issues du « Nœud 0 » sont les capteurs de notre système. Nous allons suivre les traces des paquets Ping uniquement et nous dégagerons par la suite les descripteurs de notre système en état de congestion à travers ces traces.

3.2.2 Extraction des descripteurs du système

3.2.2.1 Filtrage des données

L'outil NS-2 fournit comme résultat de simulation un fichier de traces collectées à partir des quatre nœuds et la structure de ce fichier est la suivante :

[3] ICMP : *Internet Control Message Protocol.* Protocole de niveau 3 du modèle OSI qui permet le contrôle des erreurs de transmission.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
Action	Instant	Nsrc	Ndest	Type	Taille	Flag	IDflux	@src	@dest	N°Seq	IDpaquet	Timestamp

Figure 6 : Structure du fichier de traces généré par NS-2

Ce fichier contient des informations sur l'ensemble de tous les paquets échangés au cours de la simulation. Tout d'abord, et pour extraire les informations concernant le flux ICMP (paquets Ping), il suffit d'utiliser le champ numéro (4) et faire un filtrage sur les paquets de type Ping. Dans le cas ou nous avons plusieurs émetteurs de paquets Ping, nous pouvons utiliser les champs adresse source (8) et adresse destinataire (9) en plus du champ (4).

Par la suite, nous appliquons un deuxième filtrage sur les champs (2) et (3), nœud source et nœud destination. En effet, notre objectif principal est de détecter les pertes des paquets Ping, pour cela, nous allons placer tout d'abord nos capteurs dans le nœud destinataire et utiliser le champ identifiant du paquet (11) pour détecter les pertes. Ensuite nous allons nous placer du côté du nœud source et utiliser les informations d'acquittement pour la détection de pertes.

A ce stade là, nous avons obtenu deux fichiers de traces différents. Le premier contient des informations sur le flux ICMP collectées du côté du nœud destinataire (récepteur final, nœud 3) et le deuxième contient des informations sur le flux ICMP collectées du côté du nœud source (nœud 0).

3.2.2.2 Scripts de calculs des descripteurs du système

Comme nous l'avons dit précédemment, l'objectif de ce travail en simulation est de déterminer des informations caractéristiques ou descripteurs de la situation de pertes de paquets par congestion que nous souhaitons au final modéliser sous forme de chroniques.

Les descripteurs que nous avons choisis sont des paramètres classiques au niveau de la QdS (Qalité de service) des réseaux. Il s'agit du pourcentage de pertes, du débit, du délai de bout en bout et de la gigue. Pour calculer ces descripteurs nous allons considérer les hypothèses suivantes :

- Le trafic ICMP (paquets PING) est supposé régulier (64 octets chaque 0.1 seconde)
- Les horloges des différents nœuds sont synchronisées
- Chaque paquet contient un numéro de séquence unique dans son flux
- Chaque acquittement porte le numéro de séquence du paquet acquitté
- Chaque paquet contient un champ estampille temporaire (11) qui indique le temps de son émission
- Chaque acquittement comporte les informations sur le temps de son envoie (estampille temporaire) et le temps de la réception du paquet acquitté

Les scripts de calculs que nous avons développés ont pour objectif pour chacun des fichiers de traces mis en place auparavant, de générer un fichier contenant les différentes valeurs prises par les quatre descripteurs de notre système.

Deux questions se posent alors : comment calculer la valeur de chaque descripteur en se basant sur les informations obtenues ? A quel instant ou sous quelles conditions allons-nous faire ce calcul ?

Pour répondre à la première question nous avons développés deux scripts de calculs en utilisant le langage Perl, le premier fait le calcul du côté récepteur et le deuxième le fait du côté de l'émetteur. Vous trouverez à la suite de ce paragraphe une description détaillée de ces deux scripts.

Par ailleurs, et pour répondre à la deuxième question, deux approches ont été proposées. La première approche consiste à faire le calcul sur une fenêtre temporelle de *X* secondes, la deuxième qui est celle adoptée dans notre travail, consiste à calculer les quatre descripteurs sur une fenêtre glissante de *X* paquets transmis avec succès. Dans ce stage nous avons décidé de démarrer avec la seconde mais une des perspectives de travail sera de voir les résultats obtenus en utilisant la première méthode.

i. Script de calcul coté récepteur :

- Instant de calcul = instant de réception du dixième paquet
- Calcul moyen des dix derniers paquets reçus : Fenêtre glissante de 10 paquets
- Délai de transit d'un paquet = date de réception – date d'émission (estampille temporaire)

- Délai de transit = Moyenne des délais des 10 derniers paquets reçus
- Gigue = La moyenne sur une fenêtre de la différence entre délai de transit du paquet (n) et délai de transit du paquet (n-1)
- Délai d'une fenêtre = différence entre instant de réception du dernier paquet et l'instant de réception du premier paquet de la fenêtre
- Débit = Somme des tailles des paquets reçus dans une fenêtre (10 paquets) sur le délai de cette fenêtre
- Nombre de pertes += Numéro de séquence du paquet reçu – Numéro de séquence du paquet précédent – 1 (sur une fenêtre de 10 paquets)
- Pourcentage de pertes = Nombre de Pertes / Nombre de paquets envoyés * 100

ii. Script de calcul coté émetteur :

- Instant de calcul = instant de réception du dixième paquet Echo (Acquittement)
- Calcul moyen des dix derniers acquittements reçus : Fenêtre glissante de 10 acquittements
- Gigue = La moyenne sur une fenêtre de la différence entre délai de transit du paquet (n) et délai de transit du paquet (n-1)
- Délai de transit = Moyenne des délais des 10 derniers paquets acquittés
- Délai d'une fenêtre = différence entre instant de réception du dernier paquet et instant de réception du premier paquet de la fenêtre
- Débit = Somme des tailles des paquets acquittés dans une fenêtre (10 paquets) sur le délai de cette fenêtre
- Nombre de pertes += Numéro de séquence de l'acquittement reçu – Numéro de séquence de l'acquittement précédent – 1 (sur une fenêtre de 10 acquittements)
- Pourcentage de pertes = Nombre de Pertes / Nombre de paquets envoyés * 100

La structure du fichier résultant est donnée dans la figure suivante, chaque ligne est appelé un ***« individu »*** ;

[0]	[1]
N° Individu	Instant

[0]	[1]	[2]	[3]
gigue	Délai	Débit	% perte

Figure 7 : Structure du fichier résultant du script de calcul

3.3 Classification

Cette étape va nous permettre de caractériser le comportement de notre système à travers les données historiques que nous avons dégagées lors de la phase de prétraitement. Pour cela nous allons utiliser l'outil de classification SALSA développé au sein du groupe DISCO du LAAS-CNRS.

3.3.1 L'outil SALSA

SALSA [12] met en œuvre une stratégie pour la détection et le diagnostic à partir des données et en utilisant la méthode de classification floue LAMDA (*Learning Algorithm for Multivariate Data Analysis*) [13]. Cette stratégie est basée sur l'analyse d'informations disponibles issues des capteurs du processus surveillé, et consiste en deux étapes différentes mais non-indépendantes. Une étape hors ligne d'apprentissage dans laquelle des données historiques sont analysées et traitées pour caractériser le comportement du système, une deuxième étape de reconnaissance dans laquelle le comportement du système obtenu précédemment et les données en ligne sont utilisées pour déterminer l'état courant attendu du processus. Donc SALSA permet d'une part de suivre l'évolution temporelle du fonctionnement du système, d'autre part de réaliser la détection des dysfonctionnements en s'appuyant sur le principe de déviation du comportement observé par rapport au comportement prévu. Dans notre travail nous n'avons besoin que de la partie apprentissage hors ligne de manière à obtenir une classification du comportement de congestion à partir des quatre descripteurs que nous avons définis.

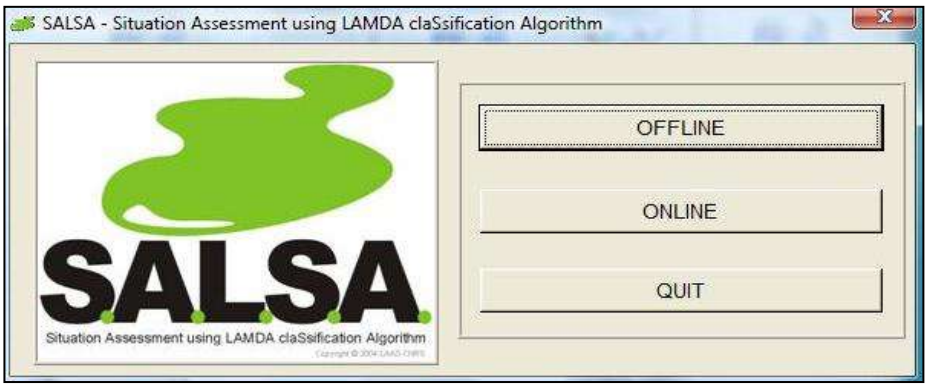


Figure 8 : Interface d'accueil SALSA

3.3.2 Résultats de classification

Le fichier résultant de chacun des deux scripts de calcul développé dans la phase de prétraitement sera l'entrée de SALSA. Rappelons que nous avons deux simulations différentes (UDP/ TCP et UDP/UDP) et que de chaque simulation nous avons dégagé deux fichiers, un du côté récepteur et un du côté émetteur. Chaque fichier contient des lignes de valeurs de nos quatre descripteurs (% perte, débit, délai, gigue) chaque ligne constitue un « Individu », donc au total nous avons quatre fichiers. Les deux fichiers de chaque simulation représentent une même situation et portent les mêmes valeurs (calculées de façon différentes), donc le résultat de classification de ces deux fichiers est logiquement identique. De ce fait nous allons choisir un de ces deux fichiers pour faire la classification et par la suite nous allons effectuer deux classifications au lieu de quatre, une pour la simulation avec les flux UDP/TCP et une pour la simulation UDP/UDP.

3.3.2.1 Classification UDP/UDP

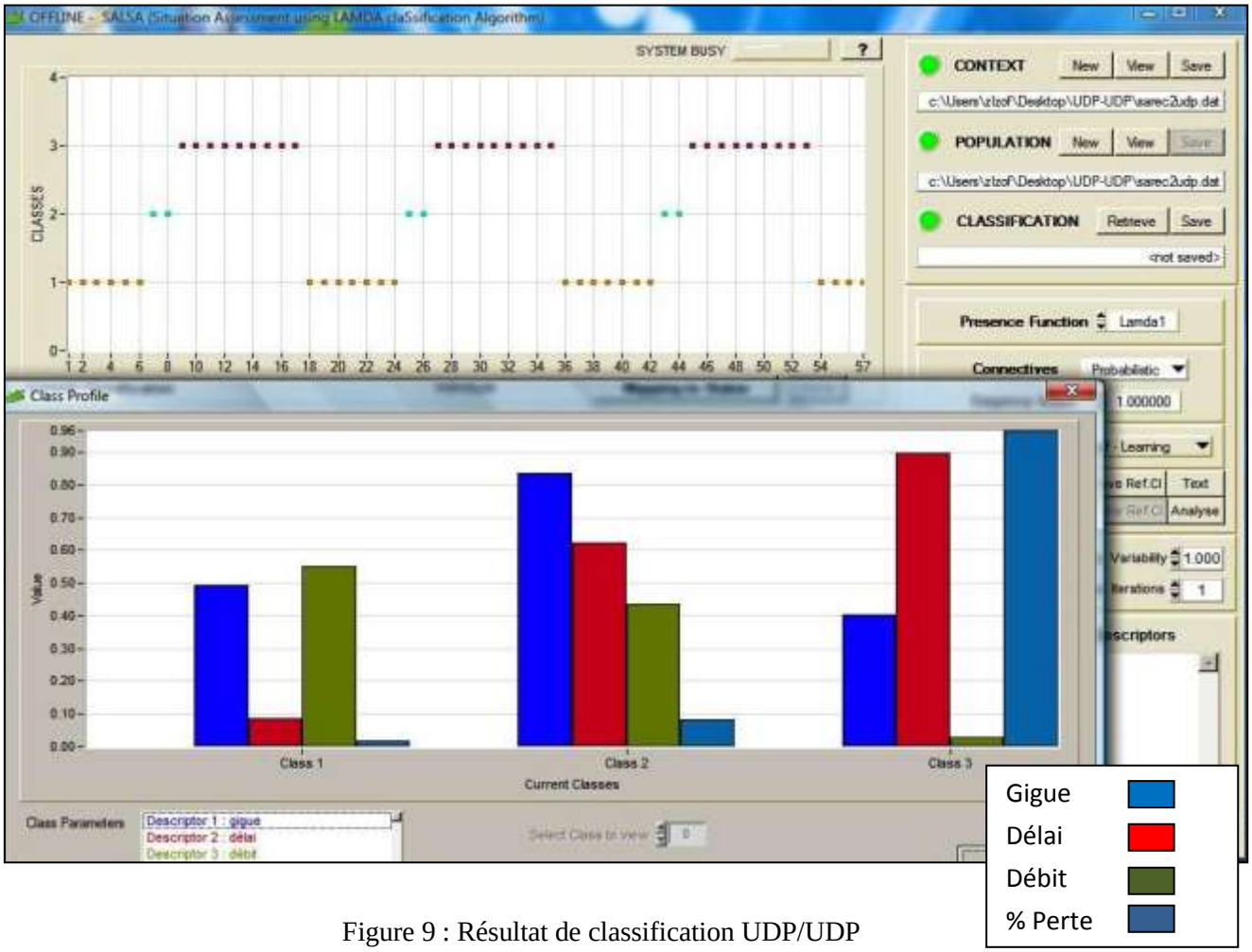


Figure 9 : Résultat de classification UDP/UDP

La figure 9 donne les résultats de la classification obtenue dans le cadre de la simulation UDP/UDP. Le profil de chacune des classes (partie inférieures de la figure 9) est donné avec des valeurs normalisées des quatre descripteurs. Cette figure (partie supérieure) montre également le résultat de classification des 57 individus résultant de la simulation UDP/UDP et du script de calcul. Chacun de ces individus est associé à une classe, toutefois, chaque individu possède un degré d'appartenance à chaque classe comme indiqué dans la figure suivante :

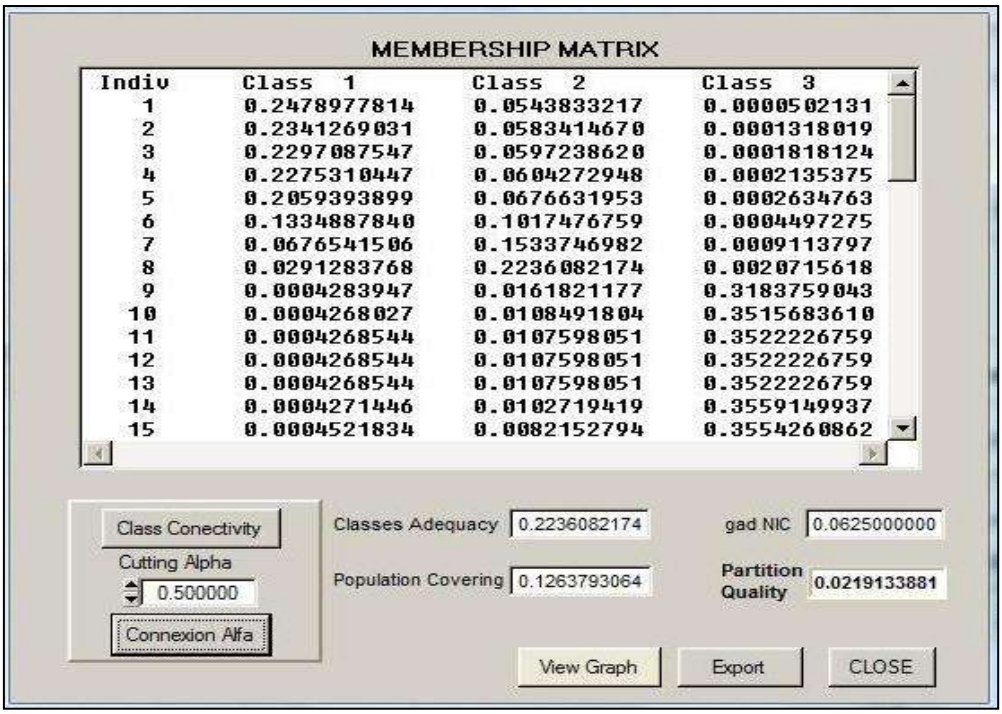


Figure 10 : Matrice d'appartenance

Quand un individu est associé à une classe, son degré d'appartenance à cette classe est le plus grand de ses degrés d'appartenance à toutes les autres classes. Les degrés d'appartenances sont représentés aussi par le graphe suivant :

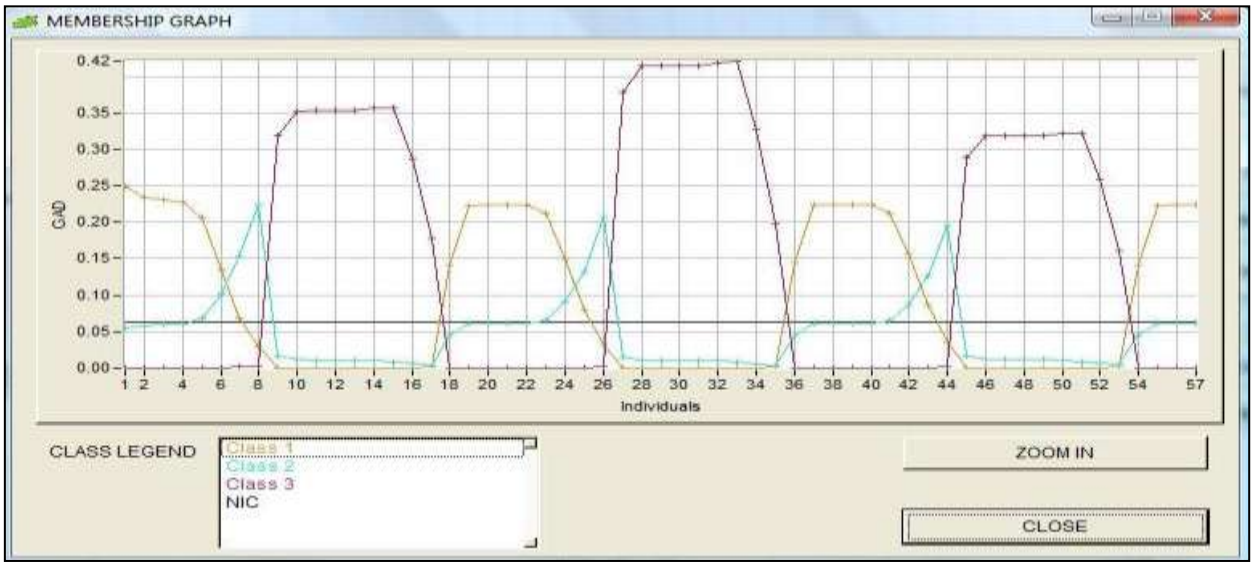


Figure 11 : Graphe d'appartenance

3.3.2.2 Classification UDP/TCP

Figure 12 : Résultat de classification UDP/TCP

La simulation avec les flux UDP/TCP et le script de calcul a donné 173 individus. La classification de ces individus a donné quatre classes différentes. Comme nous l'avons dit précédemment, chaque individu a un degré d'appartenance supérieur ou égale à zéro, à chacune de ces quatre classes.

3.3.3 Notations et définitions

Nous allons donner ici la définition des concepts de *"classes"* et d*' "individus"* que nous avons introduits auparavant.

i. Un individu : Représente un vecteur (*délai*, *débit*, *gigue*, *%perte*) calculé à un instant t_i, noté $indiv_i$ avec i son identifiant, et caractérisé par :

- $\forall\ C_j\ \forall\ indiv_i\ \exists\ \mu_{i_j} \geq 0$ degré d'appartenance de $indiv_i$ à C_j *où* C_j est la classe j à laquelle appartient $indiv_i$ selon SALSA
- $\forall\ indiv_i\ , \exists\ C_j :\ indiv_i \in C_j \Leftrightarrow \forall\ C_k\ ,\ \mu_{i_j} \geq \mu_{i_k}$

ii. Une classe : Résultat de SALSA, noté C_j, et caractérisée par :

- $\mu_{max\ j}$ degré d'appartenance maximal à la classe C_j
- $\mu_{min\ j}$ degré d'appartenance minimal à la classe C_j
- $\mu_{moy\ j}$ la moyenne des degrés d'appartenances à la classe C_j
- $\#indiv_j$ nombre d'individus appartenant à la classe C_j

Un troisième concept dégagé du résultat de classification par SALSA est le « motif primitif », ce concept est défini par :

iii. Un motif primitif : Ensemble d'individus successifs appartenant à une même classe :

- $\{indiv_k\}_{k\in\mathbb{N}^*}$ *tel que* $\forall\ k, indiv_k \in C_\alpha$

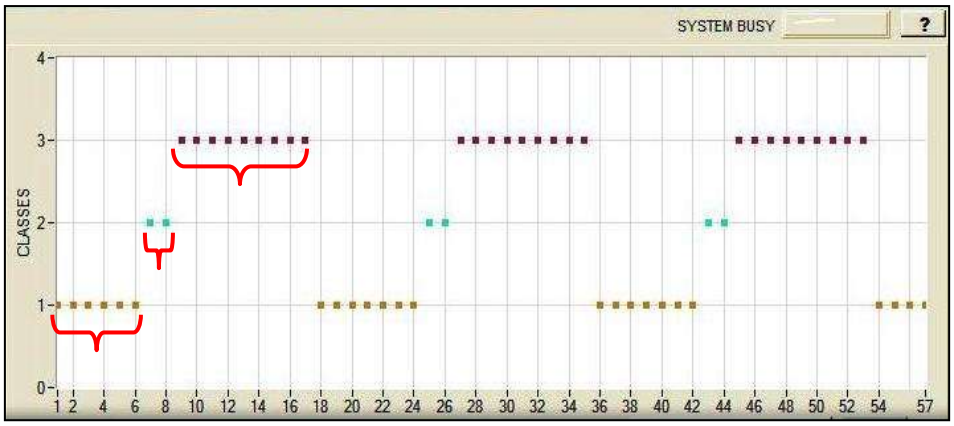


Figure 13 : Représentation graphique d'un motif primitif

La figure ci-dessus représente la classification des individus issus de la simulation UDP/UDP, et à titre d'exemple, le motif primitif de la classe C_2 est le suivant : $\{indiv_7, indiv_8\}$

3.3.4 Perspective de classification

SALSA est un outil qui offre à l'utilisateur plusieurs possibilités de réglage de manière à obtenir lors de l'apprentissage la classification qui lui semble la plus pertinente compte tenu de la connaissance experte qu'il a sur le système analysé. Ces réglages portent sur différents paramètres de la méthode de classification LAMDA à la base de SALSA.

L'objectif de notre stage n'étant pas une étude approfondie de la méthode de classification LAMDA nous ne détaillerons pas ici ces paramètres de réglages. Au cours de notre stage nous avons travaillé avec des chercheurs du groupe DISCO de manière à obtenir les résultats de classification obtenus.

Lors de ce travail nous avons analysé également les résultats obtenus à partir d'une classification plus sensible générant un nombre de classes important. La figure 14 donne un exemple d'une telle classification permettant de générer 14 classes dans le cas de la simulation UDP/TCP.

Figure 14 : Résultat de classification UDP/TCP avec des paramètres différents

En changeant les paramètres de SALSA nous avons changé la sensibilité de la classification. Cependant le problème dans une classification "sensible" c'est qu'il est possible d'avoir des individus mal classés, c.-à-d. des individus qui ne suivent pas la logique de l'évolution décrite par la classification. Un tel individu n'a pas était bien classé, donc son traitement a aboutit à la création d'une nouvelle classe à laquelle il a été associé.

L'idée que nous avons proposée est d'utiliser les résultats d'une telle classification et d'appliquer un algorithme de reclassification permettant d'éliminer les individus mal classés. Pour atteindre ce but, cet algorithme doit éliminer les classes C_ε ayant les caractéristiques suivantes :

- $\#indiv_\varepsilon \ll Moyenne_{\#indiv}$

  Avec $\#indiv_\varepsilon = nombre\ d'individus\ appartenant\ à\ la\ classe\ C_\varepsilon$

  $Moyenne_{\#indiv} = \sum_j \#indiv_j / \#C_{total}$

  Représente le nombre moyen des individus dans chaque classe

  Et

  $$\#C_{total} = nombre\ de\ classes$$

- $\mu_{moy_\varepsilon} \ll Moyenne_{\mu_{moy}}$

  Avec

$$Moyenne_{\mu_{moy}} = \sum_i \mu_{moy_i} / \#C_{total}$$

  Représente la moyenne de tous les degrés d'appartenance moyennes

- Fréquence d'apparition dans le temps ≪ Fréquence moyenne d'apparition

- $\forall\ indiv_i \in C_\varepsilon$, $\exists$ une classe C_j tel que

$$\mu_{i_\varepsilon} \cong \mu_{i_j}$$

3.4 Algorithme d'apprentissage de chroniques

La partie la plus importante dans notre travail est de développer un algorithme d'apprentissage qui permet de traduire l'évolution du fonctionnement de notre système en des modèles de chroniques. Comme nous l'avons expliqué précédemment, cet apprentissage se fait à l'aide des données historiques de notre système, les résultats de classification SALSA seront l'entrée de notre algorithme qui contient cinq étapes.

La première étape de l'algorithme est de définir et détecter les transitions d'état du système simulé, chaque transition représente un événement. L'identification de ces événements constitue la deuxième étape du traitement. La troisième étape est l'extraction des séquences d'événements, chaque séquence traduit la situation d'intérêt de notre système sous forme d'événements datés. La quatrième étape est la construction de la chronique à travers les séquences obtenues précédemment. La dernière étape est la mise à jour de la base de chroniques construites.

3.4.1 Détection des transitions

3.4.1.1 Définition d'une transition

Une transition dans SALSA est un passage d'une classe à une autre, dans notre cas ce passage exprime un changement d'état du système de communication. La figure suivante montre les transitions de la classification des individus issue de la simulation UDP/UDP :

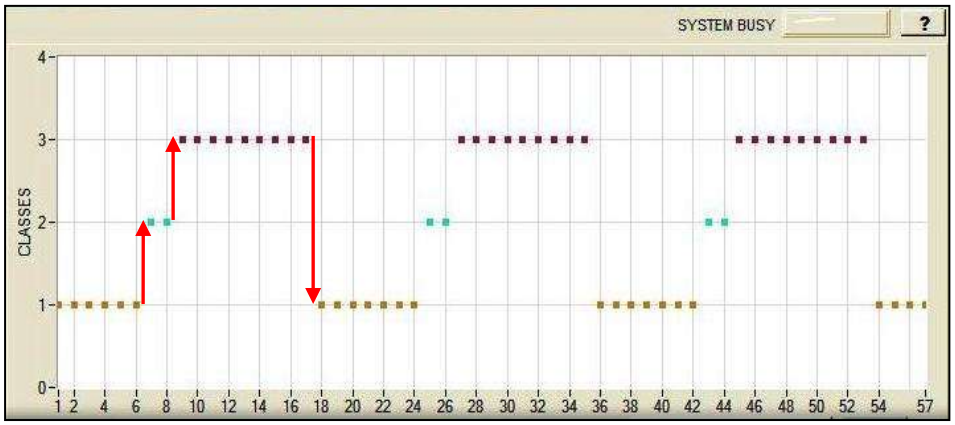


Figure 15 : Représentation graphique d'une transition

Une première transition de la classe C_1 à la classe C_2 est présentée par $indiv_6$ et $indiv_7$, la première transition de C_2 à C_3 est traduite par $indiv_8$ et $indiv_9$ et finalement la première transition C_3, C_2 est présentée par $indiv_{17}$ et $indiv_{18}$.

3.4.1.2 Etude de la zone de transition

D'après les résultats de SALSA, une transition est toujours présentée par deux individus, cela veut dire qu'un changement dans l'état de notre système est brusque ou instantané. De manière à pouvoir considérer des changements d'état progressifs et intégrer ainsi davantage l'aspect temporel dans l'approche d'apprentissage, nous allons "élargir" les zones de transitions.

Pour cela nous proposons d'étudier les degrés d'appartenance des individus et de détecter les individus qui peuvent appartenir à deux classes différentes. Ces individus constitueront la zone de transition et nous les appellerons par la suite des individus transitoires.

Notre approche est à base de seuil, en effet, la première étape est de définir un seuil d'appartenance pour chaque classe. La deuxième étape est de comparer pour chaque individu

son degré d'appartenance à une classe avec le seuil de cette classe, si le degré est supérieur au seuil nous estimons que cet individu peut appartenir à la classe concernée.

Soit la classe C_α , $\forall\ indiv_i \notin C_\alpha$

Si $\mu_{i_\alpha} \geq Seuil_\alpha$ *Alors* *Ajouter* C_α *aux appartenances de* $indiv_i$

➔ $indiv_i$ Devient ***transitoire***

Un $\boldsymbol{Seuil_\alpha}$ d'une classe C_α dépend principalement de ses degrés d'appartenances maximal et minimal ; μ_{max_α} et μ_{min_α} (définis dans la partie classification), mais aussi doit dépendre du nombre d'individus de cette classe $\#indiv_\alpha$. En effet, élargir la zone de transition veut dire que certaines classes vont perdre des individus considérés comme définissant la zone de transition. Par conséquent, une classe ayant peu d'individus peut disparaitre si son seuil est petit. Pour éviter ce genre de problème nous avons défini le seuil suivant :

$$Seuil_\alpha = \mu_{max_\alpha} - \left(\frac{\mu_{max_\alpha} - \mu_{min_\alpha}}{\#indiv_\alpha}\right)$$

En effet on a :

$$\begin{cases} \lim\limits_{\#indiv_\alpha \to \infty} Seuil_\alpha = \mu_{max_\alpha} \\ \lim\limits_{\#indiv_\alpha \to 1} Seuil_\alpha = \mu_{min_\alpha} \end{cases}$$

Par la suite, en utilisant la règle précédente il y aura peu de chance que des individus de la classe C_2 puissent appartenir à C_1 et des individus de C_1 vont être assignés à C_2 et donc ajoutés à la zone de transition qui sera élargie au détriment de la classe C_1 comme le montre la figure suivante :

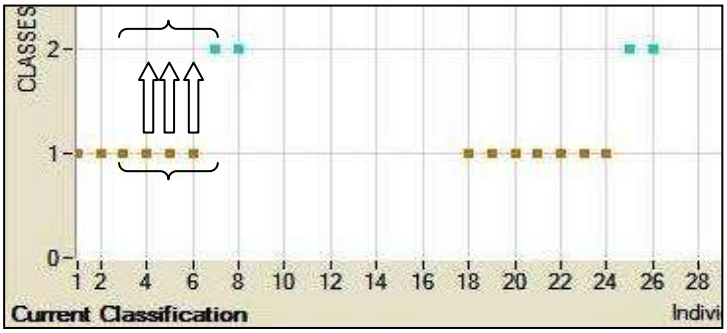


Figure 16 : Etude de la zone de transition

La transition $C_1\,C_2$ représentée par $indiv_6$ et $indiv_7$ est maintenant représentée par $\{indiv_3, indiv_4, indiv_5, indiv_6, indiv_7\}$

Le résultat de cet algorithme est l'élargissement des zones de transitions. Précédemment représentée par deux individus, une transition avait une date de début et une date de fin très proches. Après avoir appliqué l'approche du seuil et l'algorithme d'appartenance, une transition possède des dates de début et de fin bien différents.

3.4.1.3 Détection de la transition

A l'issue de l'étape précédente chaque individu sera caractérisé par un attribut booléen traduisant son caractère d'individu transitoire ou pas.

Un individu ayant l'attribut $Transitoire = True$ sera au milieu d'une transition, les deux limites de cette dernière seront deux individus, $indiv_{debut}$ et $indiv_{fin}$ ayant tout les deux l'attribut $Transitoire = False$, *($indiv_3$ et $indiv_7$ dans l'exemple de la figure 16)*, et de plus :

$$Si \quad indiv_{debut} \in C_d \quad alors \quad indiv_{fin} \in C_f \quad avec \quad C_d \neq C_f$$

3.4.2 Identification des événements

Un événement noté $event_i$ est issu d'une transition. En effet un événement est défini par l'individu de début et l'individu de fin d'une transition et à partir de ces deux informations nous pouvons extraire les caractéristiques ou attributs suivants :

- t_{d_i} : date de début de l'événement
- t_{f_i} : date de fin de l'événement
- t_i : date de l'événement, représente la moyenne entre t_{d_i} et t_{f_i}
- C_{d_i} : classe de départ
- C_{a_i} : classe d'arrivée

Grâce à l'étude de la zone de transition, chaque événement possède une date de début et une date de fin bien différentes, ce résultat sera très utile et même indispensable pour la phase de mise à jour de la base de chroniques. Mais pour l'instant, nous n'utiliserons pas les

attributs t_d (date de début) et t_f (date de fin), pour cela nous noterons un événement de la façon suivante :

$$event\ (e_{ij}^{k}, t)$$

Avec :

- ij : L'indice de l'événement qui dépend uniquement du couple (C_d, C_a)
- k : L'indice de l'événement qui donne sa position dans une séquence ordonnée de plusieurs événements
- t : La date de l'événement

De manière à alléger la notation, nous utiliserons e_{ij} si nous nous intéressons uniquement aux classes de départ et d'arrivée de l'événement et e^k quand nous nous intéresserons uniquement à l'indice de position d'un événement dans une séquence donnée.

3.4.3 Extraction des séquences d'événements

Une séquence est un ensemble d'événements datés, successifs et ordonnés dans le temps. L'étape d'extraction des séquences consiste à trouver les séquences d'événements qui décrivent la situation d'intérêt. En effet, notre algorithme peut traiter un jeu de données qui présente une succession de la même situation, dans notre cas les données d'entrée représentent le résultat de simulation de trois situations de congestion successives, donc notre algorithme doit détecter trois séquences.

Ces séquences, comme elles décrivent une même situation, devront être plus ou moins identiques. Le critère de ressemblance entre les séquences d'événements constitue l'idée de base de cette phase.

L'algorithme d'extraction de séquence consiste à distinguer et à extraire les séquences à partir de leurs événements de début et de fin. Cet algorithme se répète dans un traitement récursif jusqu'au dernier événement de l'ensemble d'entrée.

Cet algorithme prend en compte aussi la possibilité d'avoir deux séquences qui traduisent la même situation mais qui ont des événements de début ou de fin différents. En effet et comme nous l'avons expliqué précédemment, un événement est caractérisé par une classe de départ et une classe d'arrivée, en utilisant ces deux attributs nous pouvons étudier la ressemblance ou la similarité entre les événements que nous définissons de la façon suivante :

Soit l'événement e_{xy} , avec x la classe de départ et y la classe d'arrivée. S'ils existent deux événements successifs e_{xn} et e_{ny} , avec x la classe de départ du premier événement et y la classe d'arrivée du deuxième événement, nous dirons que ces deux événements sont similaires à e_{xy} et nous pouvons considérer l'égalité suivante :

$$e_{xy} = e_{xn} , e_{ny}$$

L'algorithme d'extraction de séquences est donné ci-dessous.

- Entrée : Ensemble d'événements datés ordonnés dans le temps
- Sortie : Une ou plusieurs séquences d'événements
- Traitement :
  - $e^d = dernier\ event$
  - $e^p = premier\ event$
  - $de \quad e^{p+1} \quad jusqu\ à\ e^{d-1} \quad faire$

$si\ (\ e^i == \ e^d)\ faire$

{

$si \quad similarité \quad entre \quad e^{i+1}\ et \quad e^p\ est \quad Vrai \quad alors\ "sequence"$

}

$si\ (e^i == \ e^p)\ faire$

{

$si \quad similarité \quad entre\ e^{i-1}\ et \quad e^d \quad est \quad Vrai \quad alors\ "sequence"$

}

En appliquant cet algorithme sur les événements identifiés à partir de la simulation TCP/UDP nous avons trouvé les trois séquences suivantes :

$$Seq_1 = \{ e_{12},\ e_{21},\ e_{13},\ e_{32},\ e_{21},\ e_{13},\ e_{31},\ e_{12},\ e_{24},\ e_{41} \}$$

$$Seq_2 = \{ e_{13},\ e_{32},\ e_{23},\ e_{31},\ e_{13},\ e_{31},\ e_{12},\ e_{21} \}$$

$$Seq_3 = \{ e_{13},\ e_{31},\ e_{12},\ e_{24},\ e_{42},\ e_{21},\ e_{13},\ e_{31},\ e_{12},\ e_{24},\ e_{41} \}$$

Avec les événements identifiés de la simulation UDP/UDP nous avons dégagé trois séquences identiques :

$$Seq_1 = \{ e_{12},\ e_{23},\ e_{31} \}$$

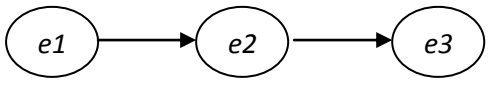
$$Seq_2 = \{ e_{12},\ e_{23},\ e_{31} \}$$

$$Seq_3 = \{ e_{12},\ e_{23},\ e_{31} \}$$

3.4.4 Construction de chronique

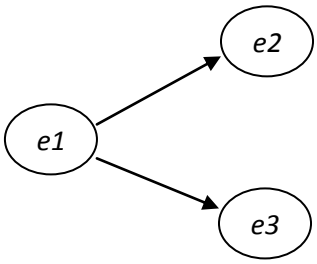
Après avoir identifié les séquences d'événements, nous passons à la construction de la chronique qui traduit la situation de perte par congestion du système de communication. La chronique générée par l'algorithme à partir des séquences d'événements extraites auparavant, doit principalement reconnaitre ces séquences. Les étapes de la construction sont les suivantes :

i. Trouver les événements qui se répètent dans toutes les séquences et dans le même ordre. Ces événements prendront leurs places dans la chronique en cours de construction sans aucune conjonction et seront présentés comme une séquence simple.



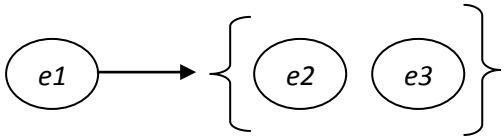
e1, e2, e3

ii. Repérer les événements qui se répètent dans toutes les séquences mais dans un ordre différents, ces événements seront reliés entre eux avec la conjonction « Et ». Pour cette étape nous avons besoin de connaitre la position de chaque événement figurant dans toute les séquences par rapport aux autres événements qui eux aussi se répètent dans toutes les séquences, nous noterons cet attribut : pos_i



e1, (e2 Et e3)

iii. Repérer les événements qui se répètent dans la plupart des séquences et trouver s'ils existent des évènements dans les autres séquences qui leurs ressemblent. Le critère de ressemblance entre les événements est le même que celui utilisé dans la phase d'extraction des séquences. Ces événements seront rattachés par la conjonction « Ou ».



e1, (e2 Ou e3)

La notion de séquence et le ***« Et »*** sont déjà définis dans les travaux de C. Dousson [1], dans le cadre de ce travail nous avons introduit le ***« Ou »***. L'algorithme de construction de chronique résultant est le suivant :

- Entrée : Ensemble de séquences d'événements
- Sortie : Une chronique
- Traitement :
  - $pour\ chaque\ \ seq_j$
  - $pour\ chaque\ e^i\ \ faire$

    {

    $si\ (\ e^i\ \in\ seq_j\ \&\&\ pos_i == x\ \ quelque\ soit\ \ j)$

    $alors\ ajouter\ "e^i"\ à\ la\ chronique$

    $sinon\ si\ \ (\ \ e^i\ \in\ seq_j\ quelque\ soit\ \ j\ \&\&\ \exists\ \ pos_i\ !=x\ )$

    $alors\ ajouter\ "\ \boldsymbol{Et}\ \ e^i\ "\ à\ \ la\ chronique$

    }

  - $pour\ chaque\ e^i\ non\ encore\ ajouté\ à\ \ la\ chronique\ faire$

{

$si \quad (Similarité\ entre\ e^i\ et\ \ e^j,\ e^{j+1}\ est\ Vrai)$

$alors\ ajouter\ "\ e^i\ \ \boldsymbol{Ou}\ e^j,\ e^{j+1}"$ à $la\ chronique$

}

Reprenons les séquences résultantes de la simulation TCP/UDP et appliquons l'algorithme de construction de chronique :

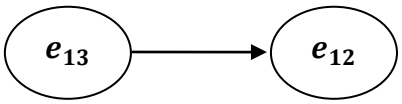
$Seq_1 = \{\ e_{12},\ \ e_{21},\ \ e_{13},\ \ e_{32},\ \ e_{21},\ \ e_{13},\ e_{31},\ e_{12},\ e_{24},\ \ e_{41}\ \}$

$Seq_2 = \{\ e_{13},\ \ e_{32},\ \ e_{23},\ \ e_{31},\ \ e_{13},\ \ e_{31},\ \ e_{12},\ \ e_{21}\ \}$

$Seq_3 = \{\ e_{13},\ \ e_{31},\ \ e_{12},\ \ e_{24},\ \ e_{42},\ \ e_{21},\ \ e_{13},\ \ e_{31},\ \ e_{12},\ \ e_{24},\ \ e_{41}\ \}$

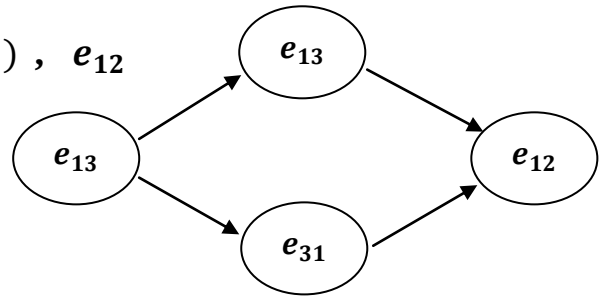
Le premier traitement dans notre algorithme consiste à trouver les événements qui se répètent dans toutes les séquences et dans le même ordre (en rouge), l'événement e_{13} est le premier événement qui se répètent dans les trois séquences, e_{12} garde toujours la position quatre. Le résultat de ce traitement est le suivant :

$\boldsymbol{e_{13}\ ,\ e_{12}}$



Le deuxième traitement consiste à dégager les événements qui se répètent dans toutes les séquences mais qui ne gardent pas le même ordre (en vert), après cette étape la chronique en cours de construction devient :

$\boldsymbol{e_{13}\ ,\ (e_{13}\ et\ e_{31})\ ,\ e_{12}}$



La troisième et dernière partie de l'algorithme consiste à étudier la similarité entre les événements restants, pour se faire notre algorithme suit ce traitement :

$Seq_1 = \{\ e_{12},\ \ e_{21},\ \ \sout{e_{13}},\ \ e_{32},\ \ e_{21},\ \ \sout{e_{13}},\ \sout{e_{31}},\ \sout{e_{12}},\ e_{24},\ \ e_{41}\ \}$

$Seq_2 = \{\ \sout{e_{13}},\ \ e_{32},\ \ e_{23},\ \ \sout{e_{31}},\ \ \sout{e_{13}},\ \ e_{31},\ \sout{e_{12}},\ \ e_{21}\ \}$

$Seq_3 = \{\ \sout{e_{13}},\ \ e_{31},\ \ e_{12},\ \ e_{24},\ \ e_{42},\ \ e_{21},\ \ \sout{e_{13}},\ \ \sout{e_{31}},\ \ \sout{e_{12}},\ \ e_{24},\ \ e_{41}\ \}$

Les événements barrés sont les événements déjà ajoutés à la chronique, maintenant nous allons étudier la similarité entre les sous-séquences d'événements qui ont le même ordre d'apparence avec les événements barrés, les trois premières sous-séquences (en bleu) sont :

$SSeq_1 = \{ \ e_{32}, \ e_{21} \}$

$SSeq_2 = \{ \ e_{32}, \ e_{23} \}$

$SSeq_3 = \{ \ e_{31}, \ e_{12}, \ e_{24}, \ e_{42}, \ e_{21} \}$

La similarité entre les événements (en vert) est vérifiée , les trois sous-séquences suivantes (en rouge) sont :

$SSeq_1 = \{ \ e_{24}, \ e_{41} \}$

$SSeq_2 = \{ \ e_{21} \}$

$SSeq_3 = \{ \ e_{24}, \ e_{41} \}$

La similarité entre les événements de ces sous-séquences est aussi vérifiée , la chronique finale qui traduit la situation de perte par congestion dans un système en réseau communicant deux flux différents UDP et TCP, est la suivante :

$$e_{13} \ , (e_{32} \ \boldsymbol{Ou} \ e_{31}, e_{12} \ ) , (e_{13} \ \boldsymbol{Et} \ e_{31}) \ , e_{12} \ , \ (e_{21} \ \boldsymbol{Ou} \ e_{24}, e_{41} \ )$$

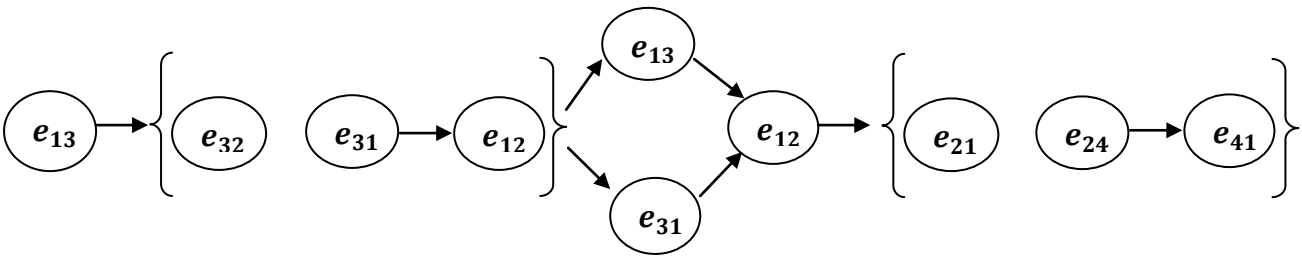


Figure 17 : Chronique Finale TCP/UDP

Après avoir défini la structure de la chronique il faut préciser les contraintes temporelles entre les événements. Le tableau ci-dessous donne les intervalles temporels entre les dates d'occurence de chaque couple d'événements successifs définissant les séquences obtenues dans le cas de la simulation UDP/TCP.

Séquence: 1		Séquence: 2		Séquence: 3	
e_{12}	1.6518397	e_{13}	1.6152239	e_{13}	0.0770998
e_{21}	0.8582244	e_{32}	0.7053919	e_{31}	2.2652245
e_{13}	0.2973919	e_{23}	0.32505608	e_{12}	0.7053919
e_{32}	0.0123445	e_{31}	0.11147117	e_{24}	0.32505608
e_{21}	0.0203331	e_{13}	0.5684805	e_{42}	0.11147308
e_{13}	0.0049012	e_{31}	0.8855686	e_{21}	0.41636658
e_{31}	0.0112566	e_{12}	1.2041273	e_{13}	0.73225594
e_{12}	0.37378407	e_{21}		e_{31}	1.3058243
e_{24}	0.4060003			e_{12}	0.15356255
e_{41}				e_{24}	0.684845
				e_{41}	

La déduction des contraintes temporelles pour la chronique finale se fait à partir des données représentées dans le tableau précédent (la différence entre les dates d'occurrence des événements), prenons l'exemple suivant :

$$event\ (e_{13}, t_{13})$$
$$event\ (e_{32}, t_{32})$$

$$t_{32} - t_{13} \quad in \quad [\mathbf{0.298}, \mathbf{1.615}]$$

Dans la suite nous allons arrondir les bornes des intervalles temporelles à 0.5 seconde prés, l'exemple précédent sera :

$$t_{32} - t_{13} \quad in \quad [\mathbf{0}, \mathbf{2}]$$

Le langage de chronique défini par C.Dousson [1] et l'outil CRS ne supportent pas la nouvelle syntaxe « ***Ou*** » que nous avons introduite. Mais une chronique contenant un « ***Ou*** » entre ses événements peut être divisée en deux chroniques. La chronique résultante de la situation de congestion par les flux TCP/UDP contient deux fois la syntaxe « ***Ou*** », elle sera donc divisée en quatre sous chroniques :

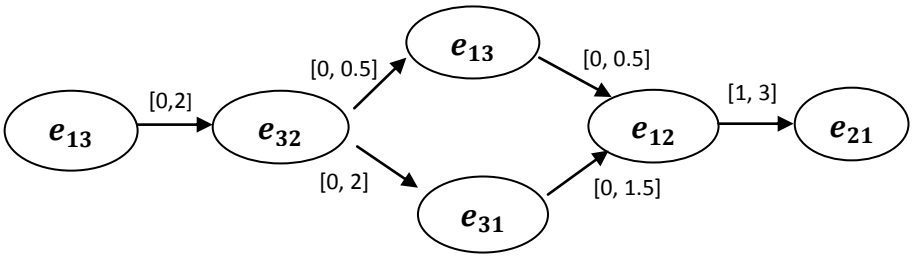


Figure 18: Sous Chronique-1 TCP/UDP

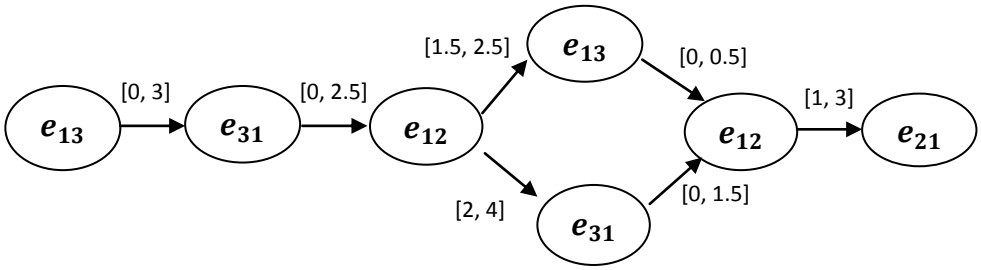


Figure 19 : Sous Chronique-2 TCP/UDP

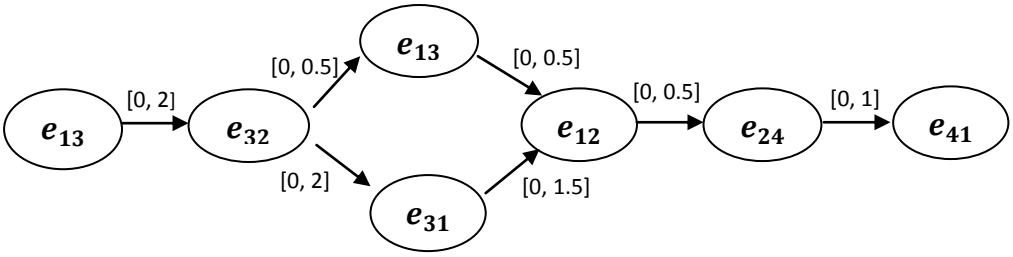


Figure 20 : Sous Chronique-3 TCP/UDP

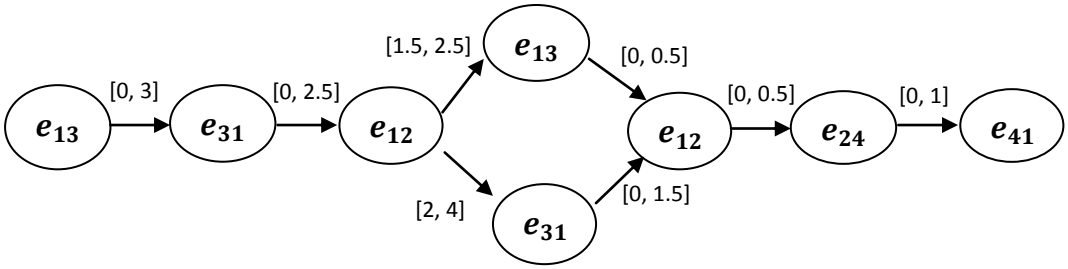


Figure 21 : Sous Chronique-4 TCP/UDP

La chronique résultante de la simulation de perte de paquets par congestion d'un système de communication échangeant deux flux UDP est la suivante :

e_{12} , e_{23} , e_{31}

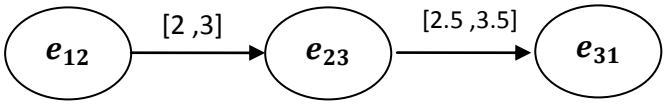


Figure 22 : Chronique Finale UDP/UDP

Les contraintes temporelles de la chronique UDP/UDP sont déduites à partir du tableau ci-dessous

Séquence: 1		Séquence: 2		Séquence: 3	
e_{12}	2.438512	e_{12}	2.8430243	e_{12}	2.3490238
e_{23}	3.10672	e_{23}	3.3547192	e_{23}	2.85672
e_{31}		e_{31}		e_{31}	

3.4.5 M.A.J. de la base de chroniques

Cette étape consiste à mettre à jour les chroniques générées par notre algorithme, ce processus de mise à jour s'exécutera au moment du test de reconnaissance. En effet, c'est un apprentissage en ligne au cours duquel chaque chronique existante dans la base de l'outil de reconnaissance pourra être modifiée.

3.4.5.1 Concepts utilisées

Dans cette étape nous nous intéresserons aux dates de début et de fin de chaque événement, l'approche proposée est inspirée de l'article [3] de T.Guyet et R.Quiniou et basée sur les concepts suivants :

1) La signature symbolique d'un motif temporel : C'est la séquence ordonnée de ses événements.

Par exemple, soit le motif temporel suivant :

$$M = \{event(A, [1,3]) \; ; \; event(B, [4,5])\}$$

Avec [1,3] = $[t_{d_A}, t_{f_A}]$ et [4,5] = $[t_{d_B}, t_{f_B}]$

Sa signature symbolique est : $Sym_M = \{A, B\}$

2) La couverture d'un motif temporel :

Un motif temporel $\boldsymbol{M}$ couvre un exemple $\boldsymbol{S}$ noté $\boldsymbol{S \preccurlyeq M}$ si la signature symbolique de $\boldsymbol{S}$ contient toute la signature symbolique de $\boldsymbol{M}$.

Soit l'exemple suivant :

$$\boldsymbol{S} = \{event(A, [1.1\,, 3.4]) \; ; \; event(C, [1.5\,, 5]); \; event(B, [4.1\,, 5.3\,])\}$$

Nous avons $\boldsymbol{M}$ couvre $\boldsymbol{S}$ $(\boldsymbol{S \preccurlyeq M})$ et la valeur de couverture de $\boldsymbol{S}$ par $\boldsymbol{M}$ est :

$$\boldsymbol{cov_S^M} = \frac{(3 - 1.1) * (5 - 4.1)}{max\,(1 * 2\,, 1.3 * \; 1.2)} \approx 0.62$$

La valeur de couverture peut être nulle dans deux cas :

- Si $\boldsymbol{M}$ ne couvre pas $\boldsymbol{S}$
- Si une paire des événements en commun ne se chevauche pas (pas d'intersection dans les intervalles de temps des événements communs)

3.4.5.2 Algorithme de M.A.J.

En effet, cette étape n'est pas encore achevée. L'idée générale de l'algorithme de mise à jour consiste à appliquer le concept de couverture entre les chroniques existantes dans la base de l'outil de reconnaissance et les motifs temporels ou séquences d'événements observées lors du test de reconnaissance. Un motif observé couvre une chronique existante avec une valeur supérieure à un seuil donné remplacera cette chronique ou sera ajouté à la base de chroniques.

Le concept de similarité entre les événements peut être aussi intégré dans l'algorithme de mise à jour.

3.5 Conclusion

Dans de ce chapitre nous avons d'abord présenté la méthodologie de notre travail, ensuite nous avons détaillé les différentes étapes de l'algorithme d'apprentissage qui constitue l'objet de notre contribution. Nous passons maintenant au test de reconnaissance des chroniques générées.

Chapitre 4

Test de Reconnaissance

I- Données de test
II- Résultats
III- Analyse et discussion

Dans ce chapitre nous allons détailler les données utilisées dans le test de reconnaissance et donner la structure finale des chroniques générées par notre algorithme d'apprentissage. En suite, nous allons présenter les résultats de reconnaissance de chacun des flux testé, et nous finissons par une analyse et discussion autour des résultats obtenus.

4.1 Données de test

Le système de reconnaissance de chronique CRS prends en entrée deux fichiers, le premier contient une description algorithmique de la chronique à tester (*.crs) et le deuxième contient un flux d'événements (*.evts). Pour chaque cas (UDP/UDP ou TCP/UDP) nous allons tester plusieurs flux d'événements, chaque flux représente le résultat d'une simulation d'une situation inconnue avant le test de reconnaissance. Dans la suite de cette partie nous présentons en détail les chroniques générées par notre algorithme et les flux d'événements utilisés dans le test de reconnaissance.

4.1.1 Les chroniques

4.1.1.1 UDP/UDP

chronicle UDP/UDP

{

event(e12,t12)

event(e23,t23)

event(e31,t31)

t23 - t12 in [2,3]

t31 - t23 in [2.5, 3.5]

}

4.1.1.2 TCP/UDP

chronicle TCP /UDP-1 *{* *event(e13,t13)* *event(e32,t32)* *event(e31,t31)* *event(e12,t12)* *event(e21,t21)* *t31 - t13 in [0,3]* *t12 - t31 in [1.5, 2.5]* *t21 - t12 in [1,3]* *}*	*chronicle TCP/UDP-2* *{* *event(e13,t13)* *event(e31,t31)* *event(e12,t12)* *event(e21,t21)* *t31 - t13 in [0,3.5]* *t12 - t31 in [1.5, 2.5]* *t21 - t12 in [1,3]* *}*

chronicle TCP/UDP-3 *{* *event(e13,t13)* *event(e32,t32)* *event(e31,t31)* *event(e12,t12)* *event(e24,t24)* *event(e41,t41)* *t31 - t13 in [0,3]* *t12 - t31 in [0, 2.5]* *t24 - t12 in [0,0.5]* *t41 - t24 in [0,1]* *}*	*chronicle TCP /UDP-4* *{* *event(e13,t13)* *event(e31,t31)* *event(e12,t12)* *event(e24,t24)* *event(e41,t41)* *t12 - t31 in [1.5, 2.5]* *t24 - t12 in [0,0.5]* *t41 - t24 in [0,1]* *}*

4.1.2 Les flux d'événements

4.1.2.1 UDP/UDP

Flux1 : Résultat d'une simulation de la situation de congestion, représente trois situations de congestion

Flux 2 : Résultat d'une simulation d'une situation normale du système de communication, c.-à-d. sans congestion

Flux 3 : Résultat d'une simulation de la situation de congestion avec diminution des capacités des liens, présente trois situations de congestion

Flux 4 : Résultat d'une simulation d'une situation normale du système de communication, c.-à-d. sans congestion mais avec une coupure des liens et par la suite une perte de paquets

Les flux testés sont présentés dans le tableau suivant, les événements en couleur rouge sont les événements reconnus, ceux en bleu représentent une similarité avec certains événements de la chronique testée (cela sera expliqué dans l'analyse des résultats)

Flux1	Flux2	Flux3	Flux4
(e12, 0.16)	(e12, 3.07)	(e12, 4.26)	(e12, 1.07)
(e23, 2.30)	(e21, 3.27)	(e23, 7.75)	(e21, 3.27)
(e31, 5.02)		(e31, 10.07)	(e12, 9.07)
(e12, 6.65)		(e12, 10.21)	(e21, 11.27)
(e23, 9.53)		(e23, 11.13)	(e12, 15.07)
(e31, 12.10)		(e32, 13.02)	(e21, 16.27)
(e13, 13.40)		(e23, 14.51)	
(e31, 19.07)		(e32, 15.16)	
		(e23, 17.77)	
		(e32, 19.22)	

4.1.2.2 TCP/UDP

Flux 5 : Résultat d'une simulation de la situation de congestion, représente trois situations de congestion

Flux 6 : Résultat d'une simulation d'une situation normal du système de communication, c.-à-d. sans congestion

Flux 7 : Résultat d'une simulation de la situation de congestion avec diminution des capacités des liens, présente trois situations de congestion

Flux 8 : Résultat d'une simulation d'une situation normal du système de communication, c.-à-d. sans congestion mais avec une coupure des liens et par la suite il y a une perte de paquets

Les flux testés sont présentés dans le tableau suivant, les événements en couleur rouge sont les événements reconnus, ceux en bleu représentent une similarité avec certains événements de la chronique testée (cela est expliqué dans l'analyse des résultats)

Flux5	Flux6	Flux7	Flux8
(e12, 1.22)	(e12, 4.57)	(e12, 3.07)	(e12, 1.77)
(e31, 2.37)	(e21, 14.42)	(e13, 3.55)	(e21, 3.17)
(e13, 4.88)		(e31, 5.25)	(e12, 9.77)
(e31, 5.09)		(e12, 7.01)	(e21, 11.17)
(e12, 7.72)		(e23, 7.11)	(e12, 15.77)
(e23, 8.49)		(e32, 7.20)	(e21, 17.17)
(e31, 8.77)		(e24, 7.22)	
(e13, 10.46)		(e41, 7.72)	
(e31, 11.68)		(e13, 10.62)	
(e12, 18.71)		(e32, 12.31)	
(e21, 19.90)		(e23, 13.89)	
(e13, 25.97)		(e32, 14.73)	
(e31, 26.70)		(e23, 16.13)	
(e13, 28.65)		(e32, 16.92)	
(e31, 29.47)		(e23, 17.62)	
		(e32, 18.59)	
		(e23, 19.60)	

4.2 Résultats de reconnaissance

4.2.1 UDP/UDP

Flux1 : Deux reconnaissances

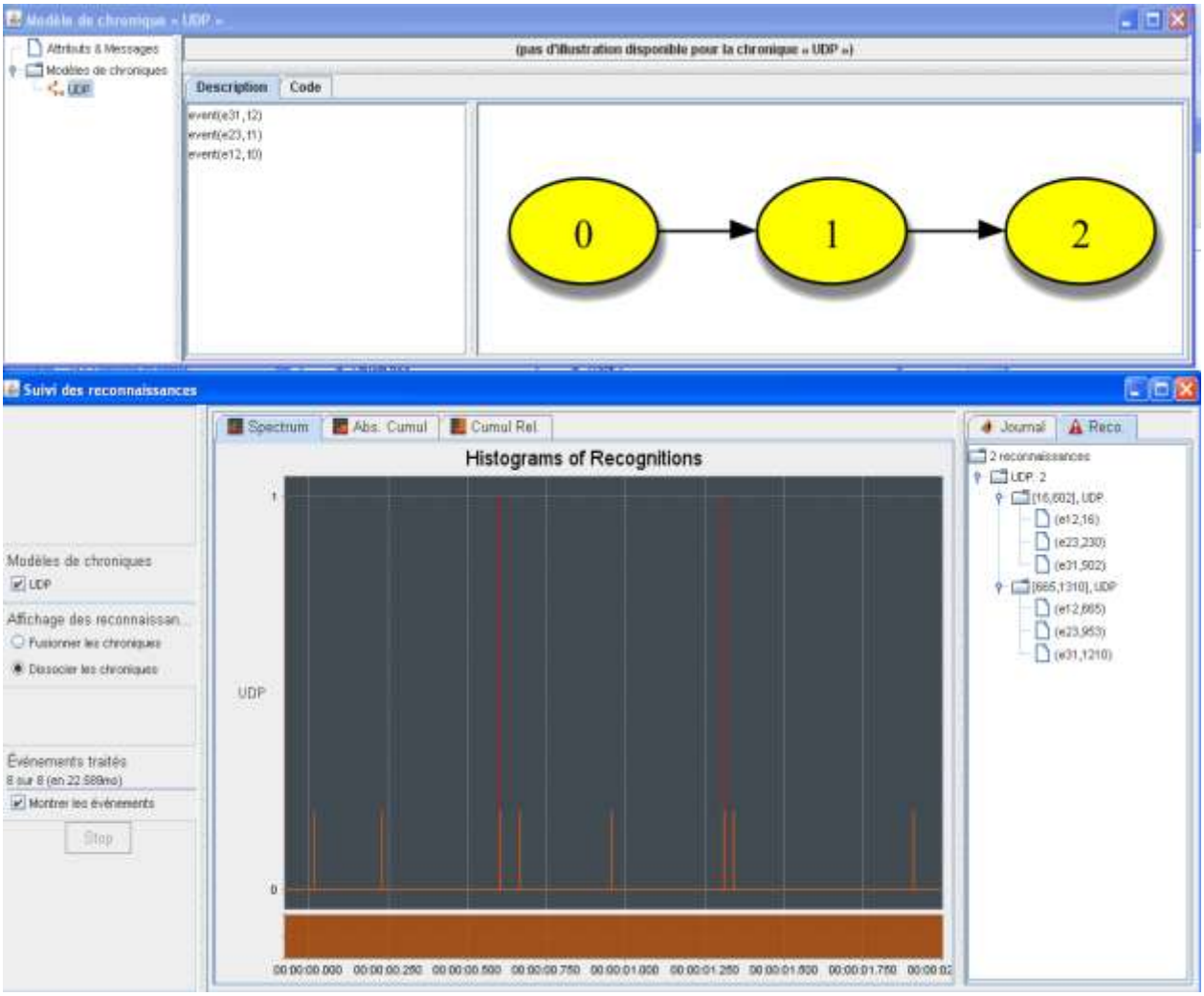


Figure 23 : Résultat de reconnaissance - Flux 1

La figure ci-dessus montre le résultat de CRS, le test appliqué sur le flux 1 a soulevé deux reconnaissances de la chronique traduisant la congestion avec les flux UDP /UDP. L'histogramme de reconnaissance montre les deux reconnaissances par des lignes rouges, les petites lignes orangées sont les événements du flux 1. Les événements reconnus (en rouge dans les tableaux précédents) sont présentés à droite de la figure et la chronique testée est présentée en haut de la figure.

La première reconnaissance commence à $t = 0.16\,s$ avec l'événement e_{12} et se termine à $t = 5.02\,s$ avec l'événement e_{31}. Tandis que la deuxième reconnaissance commence à $t = 6.65\,s$ et fini à $t = 12.10\,s$.

Flux 2 : Pas de reconnaissance

Flux 3 : Une seule reconnaissance

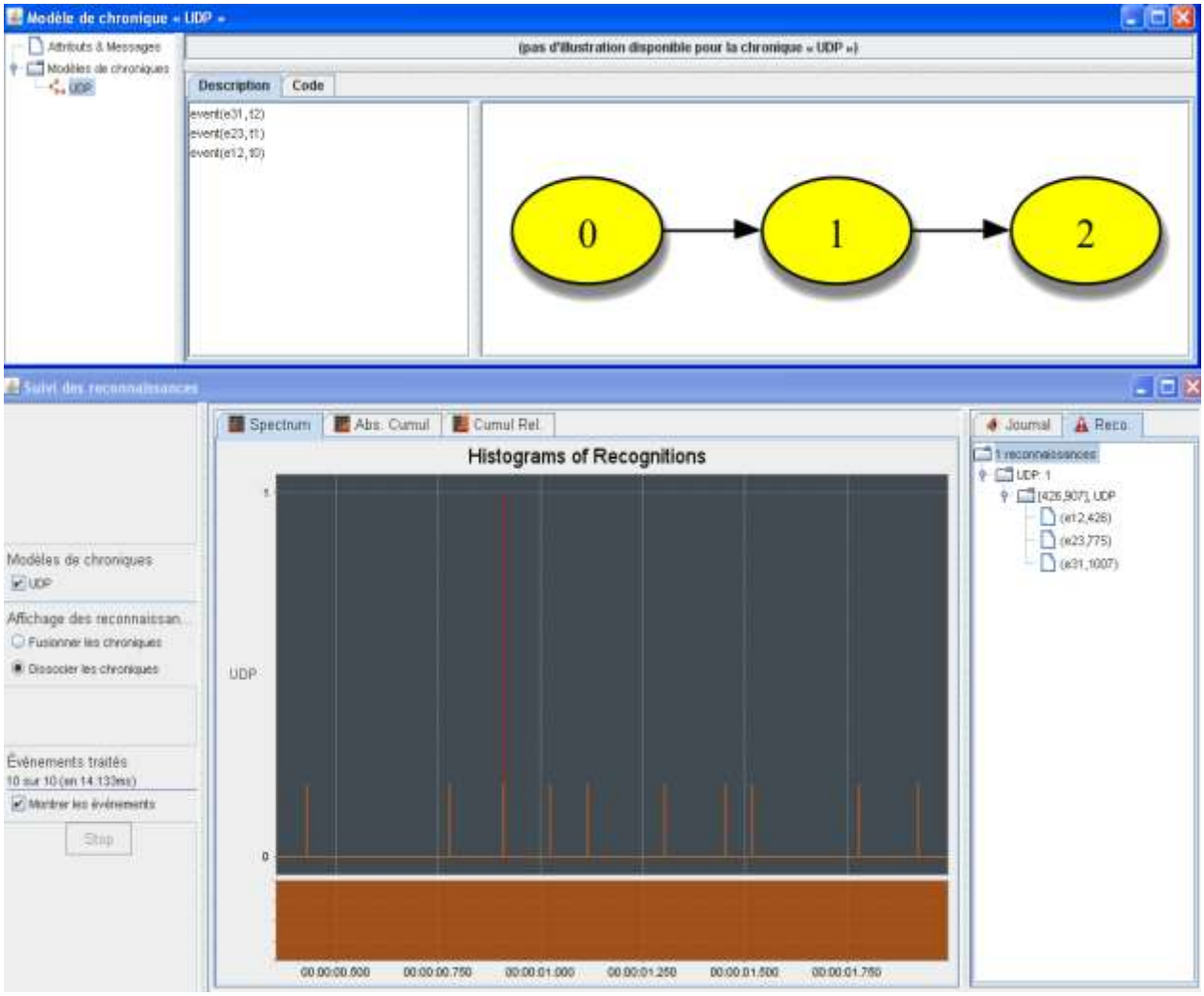


Figure 24 : Résultat de reconnaissance - Flux 2

La reconnaissance déclenchée dans le test du deuxième flux commence à $t = 4.26\,s$ avec l'événement e_{12} et se termine à $t = 10.07\,s$ avec l'événement e_{31}.

Flux 4 : Pas de reconnaissance

4.2.2 TCP/UDP

Flux 5 : Une seule reconnaissance

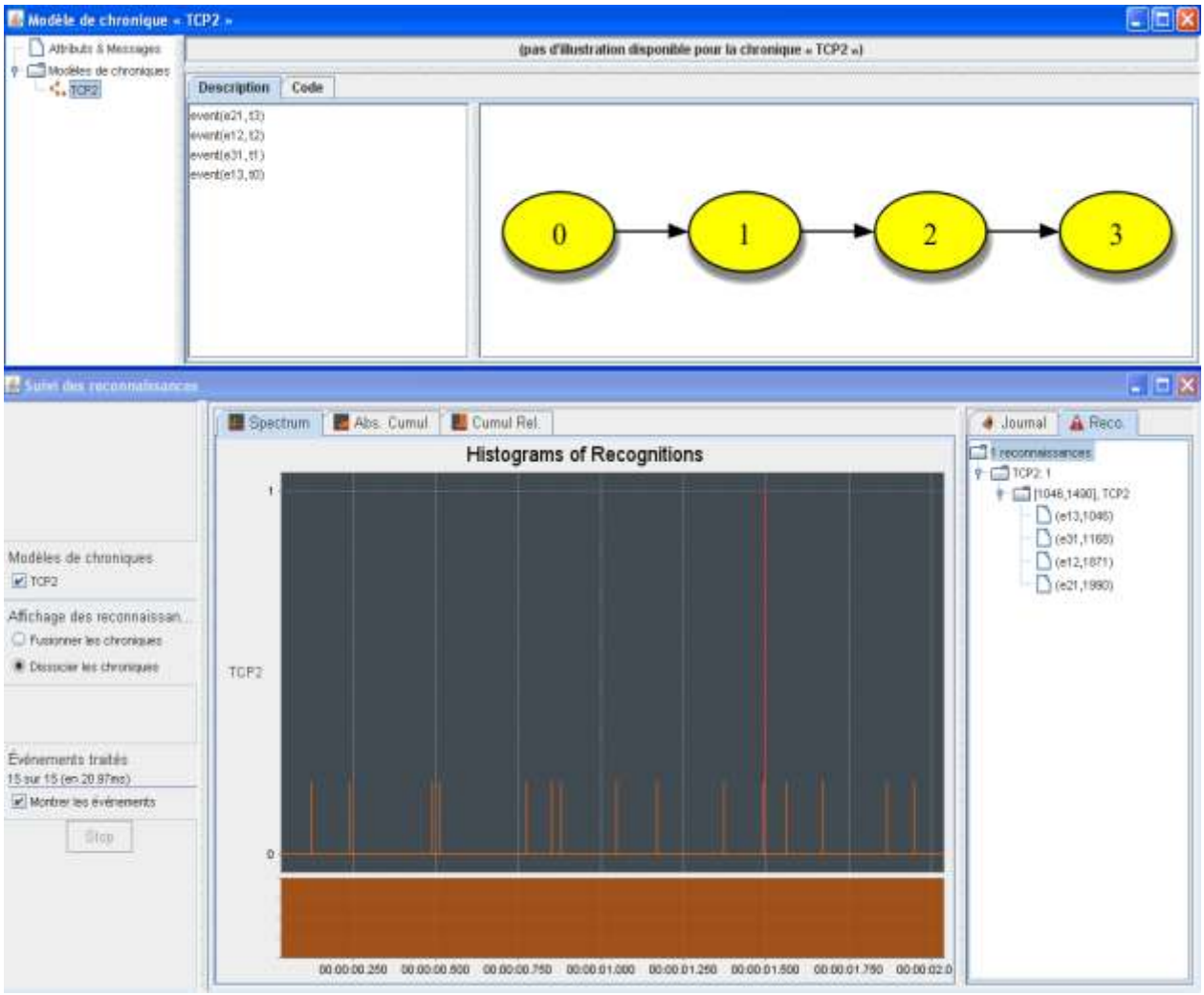


Figure 25 : Résultat de reconnaissance - Flux 5

Le test de reconnaissance appliqué sur le flux numéro 5 et la sous-chronique TCP/UDP–2, a déclenché une reconnaissance à $t = 10.46\ s$ avec l'événement e_{13} qui se termine à $t = 19.90\ s$ avec l'événement e_{21}.

Flux 6 : Pas de reconnaissance

Flux 7 : Une seule reconnaissance

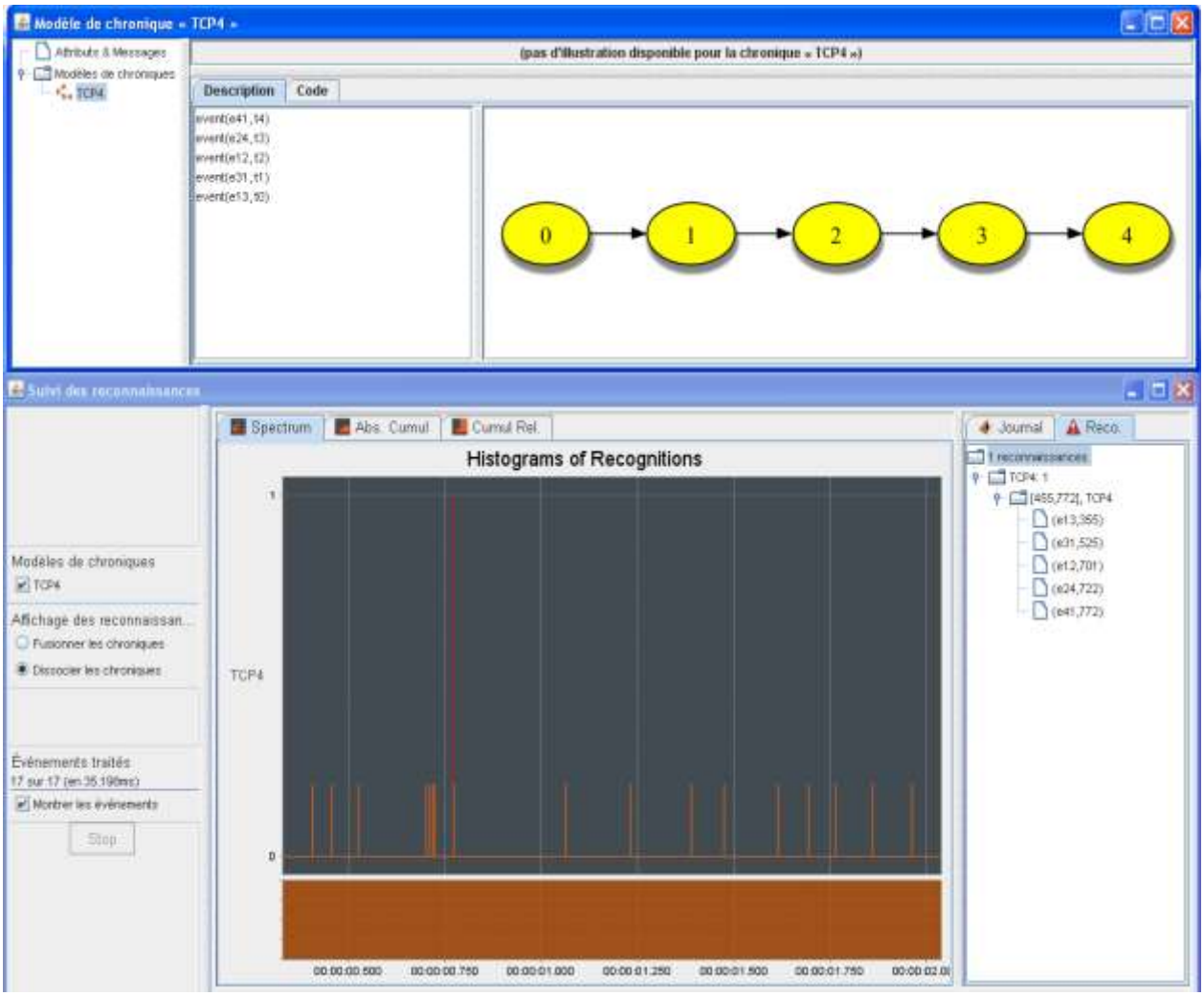


Figure 26 : Résultat de reconnaissance - Flux 6

Le test de reconnaissance appliqué sur le flux numéro 6 et la sous-chronique TCP/UDP–4, a déclenché une reconnaissance à $t = 3.55\,s$ avec l'événement e_{13}, qui se termine à $t = 7.72\,s$ avec l'événement e_{41}.

Flux 8 : Pas de reconnaissance

4.3 Analyse et Discussion

Les résultats de test de reconnaissance sont prometteurs, en effet, et à l'aide de l'outil de reconnaissance CRS, nous avons pu identifier les flux qui représentent des situations de congestion. D'autre part, aucune reconnaissance n'a été déclenchée dans le test des flux qui ne représentent pas notre situation d'intérêt. Par ailleurs, le pourcentage de reconnaissance varie entre 33% et 67%, et le pourcentage global est d'environ 41.5% des situations de

congestion simulées, en effet, sur trois situations de congestion dans chaque flux testé nous n'avons pu reconnaître qu'une ou deux situations (flux 1, 3, 5 et 7). Ce résultat peut être expliqué par les flux ou échantillons utilisés dans l'apprentissage mais aussi par le processus de reconnaissance et les flux utilisés dans le test.

Les échantillons utilisés dans l'apprentissage hors ligne de chroniques ne sont pas nombreux, en effet nous avons utilisé une seule simulation, c.-à-d. un seul scénario. En outre cette simulation contient seulement trois fois la situation qui nous intéresse. Il sera donc intéressant d'essayer d'autres simulations et d'autres scénarios comme celui utilisé dans les flux de test 4 et 8, et par la suite de multiplier le nombre d'échantillons utilisés dans l'apprentissage de chroniques.

D'autre part, certains résultats de test montrent des reconnaissances non complètes ou partielles. Reprenons le test du flux 1 avec la chronique UDP, nous pouvons constater qu'une troisième reconnaissance pourra être déclenchée. En effet, le dernier événement dans ce flux est similaire aux deux derniers événements de la chronique testée ; e_{21} et e_{23}, e_{31} . Ce même cas apparait avec le flux 5 et la sous chronique TCP-2, (ces événements sont présentés en couleur bleu dans les tableaux des flux de test). Si le système de reconnaissance de chroniques applique un processus de mise à jour et d'apprentissage continu, une telle situation pourra être reconnue. D'autre part, la durée de la simulation de test et le choix de l'instant pour appliquer la situation d'intérêt au cours de cette simulation sont des paramètres importants. En effet, il ne faut pas employer une congestion au début ou à la fin de la simulation car une telle situation sera mal caractérisée et même interrompue par la fin de la simulation.

4.4 Conclusion

Dans ce chapitre nous avons présenté les données utilisées dans le test de reconnaissance et les structures finales des chroniques générées par notre algorithme d'apprentissage. Ensuite, nous avons présenté les résultats et finalement nous avons donné une analyse et une discussion autour des résultats obtenus.

Conclusion et Perspectives

Nous avons présenté dans ce rapport une méthode basée sur la reconnaissance de motifs temporels, pour la détection et la reconnaissance de certaines situations et états d'un système en réseau communicant des données applicatives via les protocoles de la couche transport. Chaque motif temporel ou chronique traduit l'évolution du système de communication au cours du temps, et l'obtention de ces motifs constitue l'objectif principal de notre travail. Dans ce cadre, nous avons proposé une nouvelle approche de génération de chroniques basée sur un apprentissage hors ligne, c.-à-d la construction de ces chroniques est faite à l'aide de données historiques. Dans le cadre de ce stage, nous avons été intéressés par la reconnaissance d'une situation particulière de notre système, cette situation est la perte de paquets par congestion. Les chroniques traduisant cette situation et générées par apprentissage prennent en compte le type de flux échangé par les nœuds du système.

Notre approche est basée sur trois grandes étapes, la première est l'étape de prétraitement au cours de laquelle nous avons simulé la situation d'intérêt et nous avons extrait les descripteurs de notre système. La deuxième étape est la classification des descripteurs et finalement la troisième étape est l'étape d'apprentissage de chronique proprement dit qui est la plus importante dans notre travail. En effet, la contribution de ce travail est le développement d'un algorithme d'apprentissage de chronique, cet algorithme comprend à son tour cinq phases. La première phase est la définition d'une zone de transition d'état du système de communication étudié et pour laquelle nous avons proposé une approche à base de seuil. Cette phase a pour but de bien exploiter le passage d'un état courant de notre système à un autre afin d'avoir la date précise du changement. La deuxième phase est l'identification des événements ainsi que leurs dates exactes, un événement est par définition un changement d'état du système. La troisième étape est l'extraction des séquences d'événements, chaque séquence présente l'évolution temporelle du système en passant d'une situation normale à la situation d'intérêt puis son retour à la situation initiale. Le flux d'événements utilisé comme échantillon d'apprentissage peut présenter une oscillation d'état du système, ce qui veut dire que la situation qui nous intéresse peut se présenter plusieurs fois dans l'échantillon d'apprentissage. Le but de cette troisième phase est donc d'extraire chaque flux décrivant la situation d'intérêt pour l'utiliser comme échantillon propre de la situation à apprendre. La quatrième phase est la construction de la chronique à partir des séquences

extraites dans l'étape qui précède, chaque chronique décrit la situation de perte de paquet par congestion d'une façon plus générale que celle des séquences, autrement dit, chaque séquence utilisée dans l'apprentissage représente une instance de la chronique générée. La cinquième et dernière phase est la phase de mise à jour des chroniques obtenues. Le processus de mise à jour se fait en ligne lors du test de reconnaissance. Cette phase ne fait pas partie de l'apprentissage hors ligne, pour cela elle ne fait pas partie de notre travail, néanmoins, nous avons défini quelques concepts qui pourront être utiles pour ce processus d'apprentissage en ligne.

Après avoir défini et développé l'algorithme d'apprentissage hors ligne de chroniques, nous avons procédé à un test de reconnaissance en utilisant les chroniques générées par notre algorithme, et des flux de test représentant plusieurs situations d'un système en réseau dont notre situation étudiée qui est la perte par congestion. Les résultats de ces tests sont prometteurs, en effet, dans chaque flux présentant un certains nombre de situations de pertes par congestion, 33% à 67% de ces situations ont été reconnues. En outre, aucune reconnaissance n'a été déclenchée dans le test d'un flux qui ne présente pas la situation d'intérêt.

Plusieurs travaux et modifications pourront améliorer les résultats de l'algorithme d'apprentissage développé au cours de ce stage, les perspectives envisageables touchent la plupart des étapes d'apprentissage :

Tout d'abord lors de la phase de prétraitement les échantillons d'apprentissage pourraient être issus de plusieurs simulations. En effet, le comportement du réseau évolue en changeant certains paramètres de simulation tel que le nombre de nœuds, les capacités des liens d'accès et notamment le type des flux communiqués. Prendre des échantillons de plusieurs simulations pourrait sans doute améliorer la qualité d'apprentissage.

Prenons maintenant les scripts de calcul des descripteurs du système. Etant donné que l'aspect temporel est très important dans notre approche il serait intéressant de faire un calcul des descripteurs sur une fenêtre temporelle au lieu d'une fenêtre glissante de X paquets transmis avec succès.

L'étape de classification pourrait également être revue. Dans notre approche nous avons utilisé un outil de classification existant mais d'autres outils seraient tout à fait envisageables.

Finalement, un algorithme de mise à jour de la base de chroniques pourra être introduit comme un processus d'apprentissage en ligne au cours du test de reconnaissance. La notion de similarité que nous avons utilisée pourra sans doute être utilisée dans ce cadre.

Références

[1] C. Dousson. “Suivi d’évolution et reconnaissance de chroniques. PhD Thésis, Université Paul-Sabatier”, 1994.

[2] C.Dousson and O. Bertrand. “Extending and Unifying Chronicle Representation with Event Counters”. In European Conference on Artificial Intelligence, 2002

[3] T.Guyet and R.Quiniou. “Mining temporal patterns with quantitative intervals”. INRIA, DREAM Team

[4] X. Le Guillou, M-O. Cordier, S. Robin and L. Rozé. “Surveillance de chorégraphies de Web Services basees sur WS-CDL”. RJCIA'09 (9e Rencontres des Jeunes Chercheurs en Intelligence Artificielle), Hammamet, Tunisie, mai, 2009

[5] S. Bibas, M. O. Cordier, P. Dague, F. Lévy, and L. Rozé, “Scenario generation for telecommunication network supervision,” Workshop on AI in Distributed Information Networks, Aug. 1995. Montréal, Québec, Canada.

[6] P. Doherty, G. Granlund, K. Kuchcinski, E. Sandewall, K. Nordberg, E. Skarman, et J. Wiklund, “The WITAS unmanned aerial vehicle project,” in Proc. of the 14th ECAI, (Berlin, Germany), pp. 747–755, Werner Horn, Aug. 2000.

[7] F. Heintz, “Chronicle recognition in the WITAS UAV project – a preliminary report,” Swedish AI Society Workshop (SAIS2001), 2001.

[8] J. Gamper and W. Nejdl, “Proposing measurements in dynamic systems”. Proc. of the 14th IJCAI, pp. 784–790, Aug. 1995.

[9] M. Dojat, N. Ramaux, and D. Fontaine, “Scenario recognition for temporal reasoning in medical domains”. Artificial Intelligence in Medicine, pp. 139–155, 1998.

[10] G. Carrault, M. Cordier, R. Quiniou, M. Garreau, J. Bellanger, and A. Bardou, "A model-based approach for learning to identify cardiac arrhythmias," Artificial Intelligence in Medicine and Medical Decision Making, vol. 1620, pp. 165–174, 1999. W. Horn et al. editors.

[11] M. O. Cordier and C. Dousson, "Alarm Driven Monitoring Based on Chronicles," in Proc. of the 4th Symposium on Fault Detection Supervision and Safety for Technical Processes (SAFEPROCESS), (Budapest, Hungary), pp. 286–291, IFAC, A.M. Eldemayer., June 2000."

[12] T. Kempowsky, "Surveillance de procédés à base de méthodes de classification: Conception d'un outil d'aide pour la détection et le diagnostic des défaillances. " PhD Thésis, Institut National des Sciences Appliquées de Toulouse, 2004.

[13] C. V. Isaza Narvaez, "Diagnostic par techniques d'apprentissages floues : conception d'une méthode de validation et d'optimisation des partitions. " PhD Thésis, Institut National des Sciences Appliquées de Toulouse, 2007.