Août 2005

Evaluation de performances avec TAU Generation 2

VAN WAMBEKE NICOLAS
LAAS-CNRS
Groupe OLC

Version 1.1

SOMMAIRE

Introduction

Les développements récents autours du langage de modélisation objet UML (« Unified Modelling Language ») suscitent un intérêt croissant pour ces techniques dans de nombreux domaines tels que les systèmes temps réel, les architectures distribués ou encore les protocoles de communication.

C'est dans ce contexte que la version 2.0 du standard UML propose une intégration partielle de techniques de modélisation existantes dans ces nouveaux domaines comme SDL (« System Description Language ») afin de permettre une meilleure adaptation de la technique universelle de modélisation utilisée dans le monde du logiciel à ces disciplines pointues de la technique actuelle.

De nombreux outils proposent déjà une possibilité de vérifier le comportement d'un modèle décrit en UML sous la forme de machine à état et diagrammes d'architecture, au minimum, par la simulation. Cette fonctionnalité est d'un intérêt très grand pour les domaines précédemment cités de par la criticité des mécanismes mis en œuvre d'une part et comme un outil de détection des erreurs de conception avant même de débuter le processus de développement.

Dans le cadre de la modélisation UML de protocoles de communication, le travail de thèse [EXP03] propose un cadre de modélisation décrit en UML pour les protocoles de transport dynamiquement composables est proposé. Ce modèle a fait l'objet de plusieurs études et extensions comme le présentent les travaux [ARM05] de M. Armando.

Ce document présente comme il est possible de modifier le comportement de l'outil de vérification de modèle UML TAU Generation 2 édité par la société Télélogic ([Telelogic]) pour permettre d'exploiter les données fournies dans le but d'effectuer une mesure de performances obtenues à l'aide d'outils tierces (scripts shell unix, utilitaires parseurs Java, Matlab) dans un but de comparaison de l'efficacité des différents mécanismes mis en place dans le « framework ETP » ([EXP03], [ARM05] et [VAN05]).

I. Motivations

L'outil TAU Generation 2 ([Telelogic]) permet une modélisation du temps dans les différentes exécutions du système modélisé. Le temps de la simulation n'est pas le temps réel ou absolu, il s'agit d'un temps relatif à l'exécution du modèle qui possède une évolution totalement indépendante de l'horloge. En effet, le temps est totalement simulé et il en revient à l'architecte du système de mettre en place les temporisations adéquates dans son modèle afin de permettre l'écoulement de celui-ci dans la simulation.

D'autre part, l'outil propose également, en plus d'une trace graphique difficilement exploitable de façon logicielle, une trace textuelle des différents évènements qui régissent l'évolution du modèle. Enfin, une fonction de librairie standard permet à l'architecte d'insérer dans son modèle des affichages textuels spécifiques dans la fenêtre de trace.

Etant donné ces deux fonctionnalités du logiciel, il apparaît possible d'effectuer certaines mesures de performances. Néanmoins, le cadre et les outils adéquats doivent être spécifiés.

La réalisation de ces mesures et leur traitement permettrait d'avoir d'une part une nouvelle vision du fonctionnement des mécanismes de par l'approche axée performances. Dans le cas où ces résultats peuvent être corrélés avec ceux observés une fois l'implémentation terminée, une méthodologie de prédiction de performances pourrait être envisagée. D'autre part, certaines erreurs de conception qui ne sont pas directement visibles lors de la simulation du modèle mais qui apparaissent évidents dans l'étude des performances pourront être détectées plus facilement.

Dans le cadre des mécanismes mis en œuvre dans le « framework ETP », ces mesures permettront d'avoir une comparaison des modules entre eux de par leur performance dans différents environnements simulés (taux de pertes variables et délais paramétrables). Dans le cadre de l'adaptation dynamique des mécanismes au cours de la communication, les performances observées ici permettraient de définir, de façon préliminaire à l'implémentation, une euristique de composition de ces modules en fonction des paramètres de qualité de la liaison.

II. Mise en œuvre

A. *Présentation générale*

La procédure qui mène à l'obtention d'une évaluation de performances se décompose selon le schéma de principe suivant :

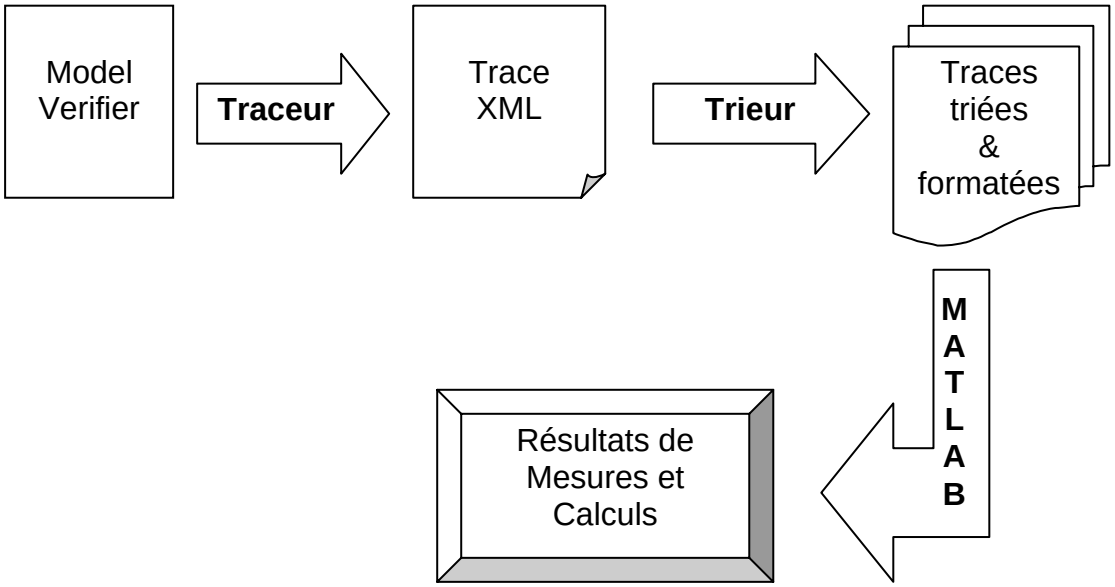

Figure 1 : Schéma de principe

On observe que dans un premier temps, c'est le « Model Verifier » du logiciel TAU qui procède à l'écriture d'une trace d'exécution au format XML ou Texte. Cette trace d'exécution contient toutes les informations que l'architecte du modèle a souhaité y faire figurer.
Vient ensuite le trieur, il s'agit là d'une entité tierce qui a pour but de transformer la trace Texte ou XML en un fichier contenant les données ordonnées selon par exemple le type d'évènement (émission, réception…).

Enfin, l'utilisation des différents fichiers fournis par le trieur dans lesquels les données sont proprement formatées pour matlab est envisagée. Ces données sont chargées dans matlab et une série d'instructions propres au traitement que l'on souhaite réaliser sont effectuées. On obtient en sortie de ceci, une série de courbes et de mesures résultant des observations réalisées.

B. *Le traceur de TAU*

La première étape à la réalisation de la chaîne de traitement correspond à la mise en place en TAU d'un mécanisme permettant à l'architecte de générer facilement des traces d'exécution de son modèle.

Afin de réaliser ceci, le choix de la mise en place d'une classe à méthodes statiques (*eventLogger*) dans le « framework » a été fait. Cette classe propose des méthodes accessibles aux architectes afin de facilement sauvegarder un évènement dans la trace.

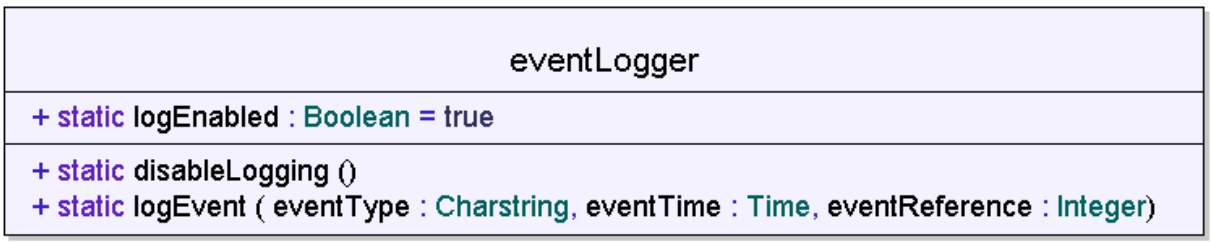

Figure 2 : Architecture de classe du traceur d'évènements

Un point remarquable est la méthode *disableLogging()* qui a pour vocation de permettre à l'utilisateur d'inhiber toute action de traçage d'évènements même si la méthode *logEvent()* est appelée. Ceci permet de ne pas devoir modifier tous les points de traçage présents dans les machines à états lorsque le comportement n'est plus souhaité.

Remarque : La méthode disableLogging() ne fonctionne pas pour l'instant du fait de limitations propres au support par TAU des attributs statiques dans les classes publiques Afin de modifier le comportement de la classe, il faut modifier directement les variables locales de la méthode logEvent().

L'appel à la méthode de trace se fait de la façon présentée ci-dessous :

```
vpdu=AddHeader(vadu);
framework::eventLogger::logEvent("EMISSION", now, vadu.nseq);
```

Figure 3 : Appel de la méthode logEvent() en émission

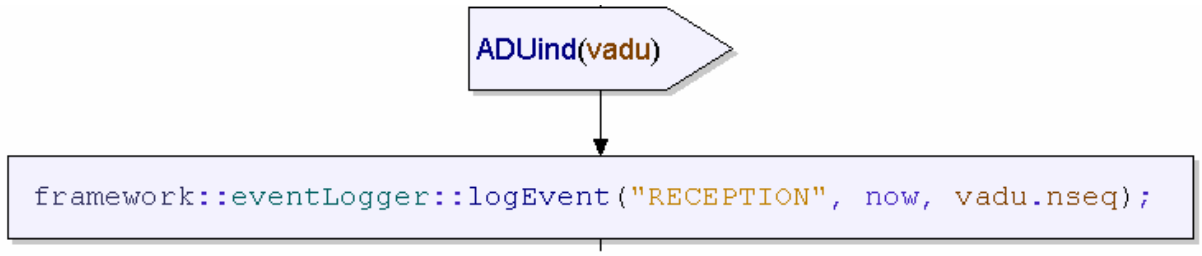

Figure 4 : Appel de la méthode logEvent() en réception

Le scénario présenté sur les figures ci-dessus permet, par exemple, de générer la trace suivante :

```
En textuel :

Type = VEMISSION    Time = 12     Ref = 1
Type = VRECEPTION   Time = 13     Ref = 1

En XML :

<event type=”VEMISSION”>
  <time>12</time>
  <ref>1</ref>
</event>

<event type=”VRECEPTION”>
  <time>13</time>
  <ref>1</ref>
</event>
```

Afin de ne pas noyer ces données dans la trace textuelle réalisée par TAU lui-même sur occurrence des différents évènements qui donnent lieu à une évolution du modèle, il est bon de procéder à la réduction au minimum du volume de ces dernières. Ceci s’effectue par l’utilisation de la commande :

```
set-trace 0
```

Cette commande est à rentrer dans la console du « Model Verifier » avant de démarrer l’exécution du modèle à simuler.

Remarque : Il est important de noter que pour que les données produites par le système de génération de traces présenté ci-dessus pour différents modèles UML simulés successivement puissent être comparées entre elles, un certain nombre d’éléments sont à considérer.

- L’évolution du temps doit se faire de façon comparable et cohérente entre les modèles. En effet, si une construction de paquet est évaluée à une unité de temps pour un modèle et à dix unités de temps dans un autre, il est probable que la comparaison des résultats soit faussée.

- Les évènements tracés doivent être les mêmes. En effet, il semble illogique de comparer le comportement d’un système vis-à-vis de son utilisation des ressources réseau par exemple avec un autre système pour lequel les données disponibles se réfèrent aux évènements d’interaction avec l’application.

C. Le trieur de données

Une fois les traces textuelles produites par l'application de simulation du modèle UML, il n'est pas possible simplement de les envoyer directement à matlab pour analyse. En effet, matlab a pour vocation de permettre un travail simple et efficace sur des matrices de données numériques, il n'est pas optimisé pour le traitement de fichiers de données même en XML.

Afin de simplifier l'importation des mesures réalisées, un trieur de données peut être mis en œuvre. Plusieurs façons d'effectuer les mêmes traitements existent, par la suite, une approche de traitement des traces textuelles par script shell sera présentée.

1. Trieur shell sur données texte

Le traitement de fichiers en mode texte fait l'objet de nombreux outils en ligne de commande du monde unix. La performance de ces outils donne à l'utilisateur averti un grand contrôle sur leur mise en forme, leur séparation et leur tri sans trop d'efforts.

Quelques outils intéressants seront détaillés ci-après sans que ce document ne soit une référence en la matière de l'art du script shell. Le lecteur intéressé saura se référer aux nombreuses documentations disponibles pour chacun d'entre eux. Leur combinaison afin d'effectuer un traitement cohérent est présenté en annexes.

a) grep : séparation des lignes

L'utilisation de l'outil grep ne se limite pas à celle qui est présenté ici bien au contraire mais dans le cadre du traitement que nous voulons faire, il est possible que l'on souhaite distinguer deux ou plusieurs évènements sur base soit de leur type ; soit du numéro de séquence du PDU sur lequel ils portent ou encore de l'heure d'émission.
Pour cela, l'outil grep permet une grande flexibilité. Pour séparer les évènements sur base de leur type, il est possible de faire appel par exemple à la commande suivante :

```
grep <type> <fichier> > <fichier-destination>
```

ou encore, si l'on désire continuer les traitements sur la sortie :

```
grep <type> <fichier> | <traitement suivant>
```

b) awk : ordonner les colonnes

L'outil awk qui n'est autre que l'interpréteur en ligne de commande du langage de traitement de données qu'est AWK permet de réaliser un traitement différentié sur chacune des lignes du fichier. Son exécution peut se faire de plusieurs façons selon que l'on écrit le script à exécuter directement sur la ligne de commande ou dans un fichier séparé.

```
awk '{<commandes>}' <fichier-données>
```

```
awk -f <fichier-commandes> <fichier données>
```

La commande la plus utile dans notre utilisation est la commande `print` qui permet d'afficher sur la sortie standard des informations. Awk découpe les fichiers en colonnes, dont le séparateur par défaut est l'espace, qu'il est possible d'afficher séparément en utilisant les variables $1…$NR. Ainsi, pour afficher uniquement les colonnes 2 et 3 du fichier, la commande à exécuter est : `print $2,$3`

De plus, il est possible de conditionner l'exécution d'une commande en la précédent d'un test booléen. Dans ce cadre, la syntaxe est la suivante :

```
<test-booléen> {<commandes>}
```

Ainsi la commande d'affichage des colonnes 2 et 3 dans le cadre ou la valeur de la troisième colonne est inférieure à 20 est :

```
$3 < 20 {print $2, $3}
```

Le lecteur intéressé par plus de détails saura se référer à l'excellente documentation existante sur le sujet ([AWK92]).

c) sort : trier les fichiers

Le dernier outil présenté dans cette section est l'outil de tri de choix disponible sur tous les systèmes d'exploitation de type unix ou BSD nommé sort. Cet outil peut travailler soit sur un fichier en entrée soit sur l'entrée standard (éventuellement redirigée par pipe).

L'outil sort permet d'ordonner les lignes un fichier en fonction d'une clé sur laquelle il se basera. Par défaut, le tri s'effectue sur toute la ligne et le tri est un tri alphabétique. Néanmoins, il est possible de modifier ce comportement en spécifiant une clé correspondant à une restriction de la ligne et de spécifier le type de données à trier à l'aide d'options spécifiques passées au programme.

Ainsi afin de trier un fichier de données numériques en en fonction des données de sa seconde colonne dans l'ordre croissant, la syntaxe suivante sera adoptée :

```
sort –k 2 –n <fichier>
```

L'option –k spécifie à sort que la clé de tri doit être restreinte à la seconde colonne du fichier, le séparateur par défaut étant l'espacement. L'option –n spécifie que les données doivent être traitées comme des nombres.

Une autre option intéressante est l'option –n qui permet de trier le fichier dans l'ordre décroissant. Pour plus de détails sur sort, la page de manuel de la commande se positionne comme une très bonne source d'informations.

D. Le script matlab

Le script matlab est la dernière étape de l'analyse des données récoltées dans le sens où il permet la production de résultats. Ces résultats peuvent prendre deux formes distinctes qui sont le tracé et l'affichage textuel de valeurs.

Il est bien au-delà de l'optique de ce document de présenter la totalité des fonctionnalités offertes par matlab, d'une part car de nombreux documents existants s'y attellent et d'autre part car il est toujours possible d'innover en matière de production de données.

Néanmoins, les techniques principales d'obtention de résultats peu complexes sont présentées. Les nouveaux utilisateurs du logiciel y trouveront certainement un base pour continuer leur exploration et les utilisateurs confirmés pourront s'interroger sur les aspects d'optimisation que propose l'outil.

1. Fonctions matlab usuelles

Afin de permettre une compréhension rapide et efficace de la suite du document, une rappel des fonctions les plus utilisées ainsi que de certaines notations propres à matlab sont données ci-dessous.

a) Partition

M(:,1) :	sous matrice des éléments de la première colonne de la matrice M
M(1, :) :	sous matrice des éléments de la première ligne de la matrice M
M(a :b ,c :d) :	Sous matrice contenant les lignes a jusqu'à b et les colonnes c jusqu'à d de la matrice M

b) Affichage

disp('Texte à afficher ') :	Affiche le texte dans la fenêtre de commandes
plot(vecteurX, vecteurY) :	Trace la courbe vecteurY = f(vecteurX)
figure :	Ouvre une nouvelle fenêtre graphique

hold on :	Permet de conserver l'affichage précédent lors des prochains plot
xlabel('Label axe X') :	Permet de définir la légende de l'axe X
ylabel('Label axe Y') :	Idem xlabel() pour l'axe Y
title('Titre') :	Place un titre au dessus du graphique
subplot(x,y,z) :	Découpe la dernière fenêtre d'affichage en x lignes, y colones et place le résultat du prochain plot dans la case z (cases numérotées de gauche à droite et de haut en bas).

c) Opérations sur matrices

max(matrice) :	Recherche la valeur maximale de la matrice
min(matrice) :	Recherche la valeur minimale de la matrice
mean(matrice) :	Calcule la valeur moyenne des éléments de la matrice
diff(vecteur) :	Retourne un vecteur dont les éléments sont la différence entre deux valeurs successives de vecteur.
length(vecteur) :	Revois la taille du vecteur
(matrice)' :	Transposée de la matrice
zeros(x,y) :	Crée une matrice de x lignes et y colonnes remplie de zéros.

2. Chargement de valeurs à partir de fichiers

Les fichiers générés par le trieur sont tels qu'ils se présentent sous la forme de matrices contenant potentiellement deux ou plusieurs colonnes. Ce format de fichier peut être directement inclus dans matlab par l'intermédiaire de l'instruction read(*filename*). Le résultat est une matrice d'autant de lignes et de colonnes que celles contenues initialement dans le fichier source.

Les matrices ainsi chargées peuvent être utilisées ensuite comme n'importe quelle matrice sous matlab.

3. Calculs des débits

Grâce au trieur, il est possible d'obtenir pour chaque paquet sa date d'émission ainsi que sa date de réception. Néanmoins, la notion de débit demande quelques modifications à la forme des données. En effet, il faut savoir combien de paquets sont envoyés ou reçus à chaque instant.

Afin de réaliser ceci, pour chaque temps figurant dans la trace, un parcours du vecteur est fait afin de savoir combien de PDUs possèdent le même temps d'émission/réception. Cette valeur sera considérée comme étant la valeur instantanée du débit à l'instant t donnée en nombre de paquets par unité de temps.

Néanmoins, la courbe que l'on obtient ainsi ne présente pas réellement un résultat exploitable en lui-même. En effet, la granularité d'échantillonnage est trop fine pour pouvoir observer une tendance générale sur les données exploitées.

Afin de pallier à cela, une moyenne des débits instantanés dans une fenêtre dont la taille est définie par l'utilisateur peut être proposée. Cette approche permet d'observer la variation du débit instantané au cours du temps et d'obtenir une courbe qui sera plus exploitable tout en conservant une valeur moyenne correcte.

III. Résultats

Les mécanismes décrits dans ce document ont été mis en œuvre afin de comparer entre eux les trois mécanismes présents dans le « framework » ETP au moment de l'écriture de celui-ci. Pour chacune des simulations de tests, les conventions suivantes ont été suivies :

- La construction d'un PDU est évaluée à 1 ut
- Les médiums ont un taux de perte de 50% dans le sens émetteur-récepteur et de 20% dans le sens récepteur-émetteur
- Les médiums sont configurés avec un délai de 5 ut dans les deux sens pour les faibles délais et 150 ut dans les deux sens pour les délais élevés
- Dans le cas ou aucune évaluation du RTT n'est prévue par le mécanisme testé, le délai des timers de retransmission est fixé à 20 ut ce qui correspond à deux fois la valeur moyenne du RTT compte tenu du délai imposé aux médiums.
- On limitera la capture à l'échange effectif de 1000 paquets de données entre les deux hôtes.

L'environnement simulé ici est fortement bruité mais permet néanmoins des délais acceptables, si l'on considère une équivalence entre les ut et les millisecondes. La caractérisation ainsi faite s'apparente au modèle donné pour une liaison sans fil de type Wifi dans un environnement fortement bruité.

Dans les paragraphes qui suivent, une comparaison des résultats obtenus entre eux est envisagée afin de montrer l'efficacité relative de chacun d'entre eux.

A. Stop & Wait

Comme il est possible de s'y attendre, le mécanisme du stop and wait est de loin le moins performant que l'on puisse utiliser sur le type de réseau modélisé.

Les résultats obtenus pour ce mécanisme sont détaillés par la suite. On notera que les analyses menées donnent lieu à deux types de données. D'une part on trouve les résultats textuels qui résultent de l'analyse de l'intégralité des données récoltées et des résultats graphiques qui sont plus adaptés à montrer l'évolution de certain des aspects de la communication dans le temps.

1. Fortes pertes et délai faible

Les résultats ci-dessous sont obtenus dans le cas d'un médium configuré pour un délai faible (5ut) et un taux de pertes élevé (50%).

```
>> analyseTrace('../SAW-50-5-20-5.trace/EMISSION.dat','../SAW-50-5-20-5.trace/RECEPTION.dat', 1000);

-------- MESURES SUR LE DEBIT D'EMISSION ----------
Debit Moyen    : 0.014187 paquets/ut
Debit Maximum  : 0.02997 paquets/ut
Debit Minimum  : 0.001998 paquets/ut

------- MESURES SUR LE DEBIT DE RECEPTION ----------
Debit Moyen    : 0.014186 paquets/ut
Debit Maximum  : 0.028971 paquets/ut
Debit Minimum  : 0.001998 paquets/ut

-------------- MESURES SUR LE DELAI ----------------
Delai Maximum  : 205 ut
Delai Minimum  : 5 ut
Delai Moyen    : 25.2772 ut

-------------- MESURES SUR LE GIGUE ----------------
Gigue Maximale : 200 ut
Gigue Minimale : 0 ut
Gigue Moyenne  : 28.1863 ut
```

Tableau 1 : Résultats Textuels - Stop & Wait

Le Tableau 1 présente les résultats textuels. Il est possible de noter que les débits sont très petits en comparaison avec ce qui sera observé par la suite. Aussi, le délai minimum correspond à l'émission sans encombre d'un PDU.

Les tracés graphiques permettent de visionner d'une part que l'évolution des délais dans le temps présente des valeurs très dispersées. En effet, l'émission d'un PDU n'ayant aucune incidence sur le délai de transmission du PDU suivant, l'allure de la courbe présente une distribution très dispersée des points.

La courbe du débit présentée Figure 6 montre que la variation du débit se fait suivant un patron répétitif d'oscillations autours de la valeur moyenne qui résulte du délai et des pertes introduites par le médium de transmission.

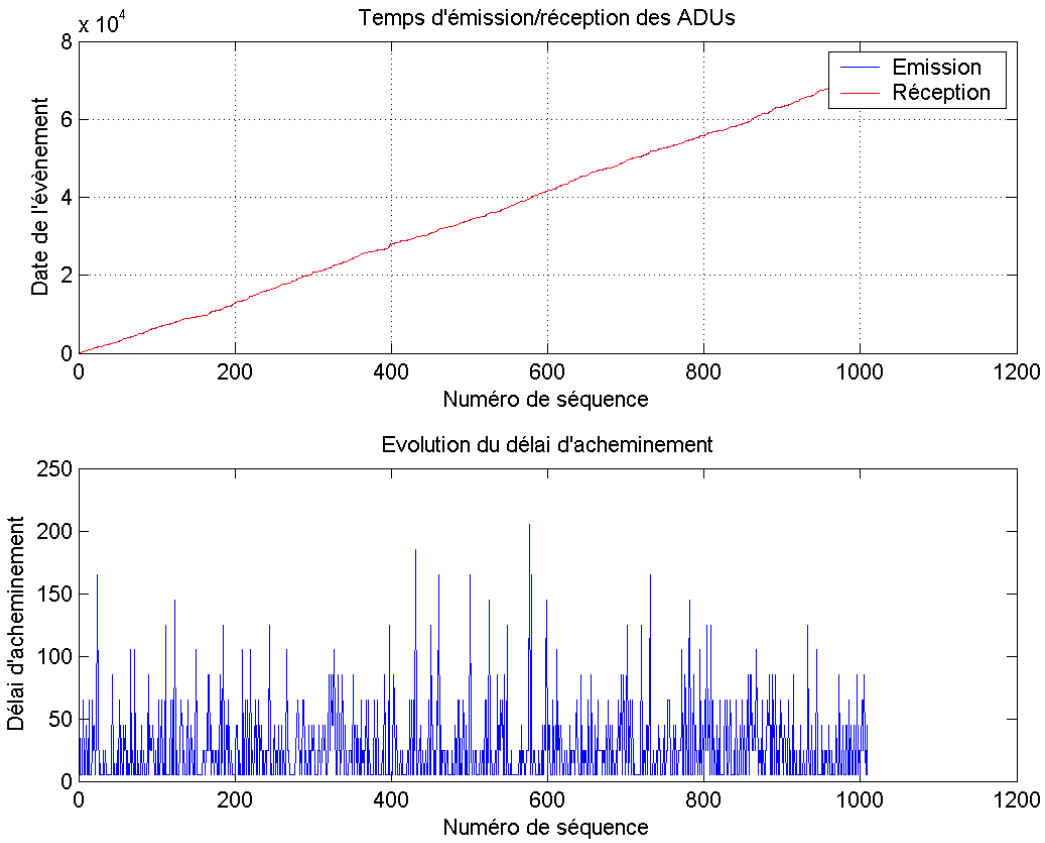

Figure 5 : Date d'évènements - Délais - Stop & Wait

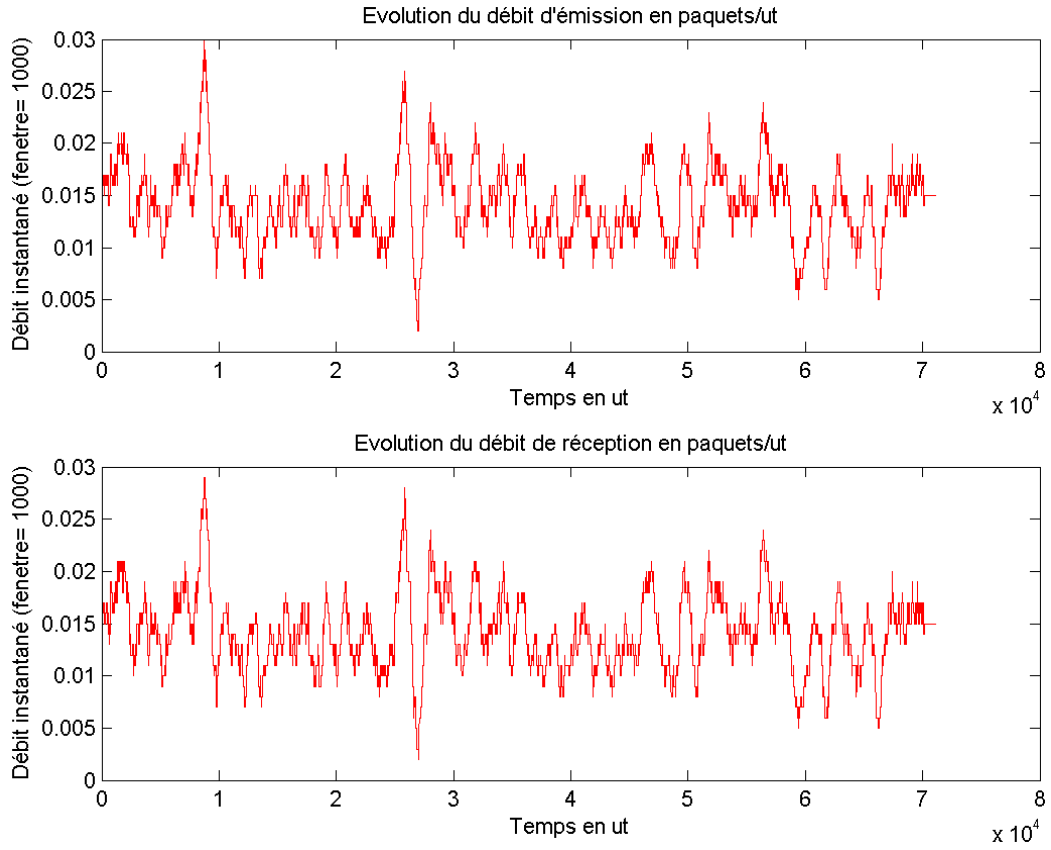

Figure 6 : Evolution des débits - Stop & Wait

2. Délai et pertes élevés

Compte tenu des résultats obtenus pour un délai faible, les mesures concernant une délai de transfert élevé n'ont pas été effectuées car les résultats sont prévisibles et leur comparaison avec les autres mécanismes n'apporte que peu d'éléments nouveaux.

B. Go Back N

Le mécanisme du Go Back N a été implémenté pour le « framework » ETP dans le cadre du travail de Master of Research de M. Armando ([ARM05]). Afin de respecter les conventions établies, seul quelques modifications simples ont du être apportées, notamment en ce qui concerne les différentes durées des timers mis en jeu dans le mécanisme.

Les résultats obtenus avec ce mécanisme permettent de contempler une nette amélioration par rapport au mécanisme simple du stop and wait. Néanmoins, du fait de l'utilisation peu efficace de la bande passante qui résulte de la réémission systématique de l'intégralité de la fenêtre d'émission lorsque survient une perte, le débit moyen ainsi que le débit maximum restent peu élevés (Tableau 2).

1. Délai faible et fortes pertes

Ci-dessous sont donnés les résultats pour un médium configuré pour introduire un délai de 5ut et pour un taux de pertes de 50%.

```
>> analyseTrace('../GBN0.43-50-5-20-5.trace/EMISSION.dat','../GBN0.43-50-5-20-5.trace/RECEPTION.dat', 3000);

-------- MESURES SUR LE DEBIT D'EMISSION ----------
Debit Moyen    : 0.057784 paquets/ut
Debit Maximum  : 0.064978 paquets/ut
Debit Minimum  : 0.046984 paquets/ut

------- MESURES SUR LE DEBIT DE RECEPTION ----------
Debit Moyen    : 0.057483 paquets/ut
Debit Maximum  : 0.064978 paquets/ut
Debit Minimum  : 0.046984 paquets/ut

-------------- MESURES SUR LE DELAI ----------------
Delai Maximum  : 264 ut
Delai Minimum  : 18 ut
Delai Moyen    : 94.6124 ut

-------------- MESURES SUR LE GIGUE ----------------
Gigue Maximale : 157 ut
Gigue Minimale : 0 ut
Gigue Moyenne  : 19.0989 ut
```

Tableau 2 : Resultats Textuels - Go Back N

Les résultats présentés ci-dessus proviennent de l'analyse des données brutes qui concernent l'intégralité des échantillons. En plus de ces résultats textuels, les

courbes présentées Figure 7 et Figure 8 permettent de visualiser l'évolution des performances dans le temps.

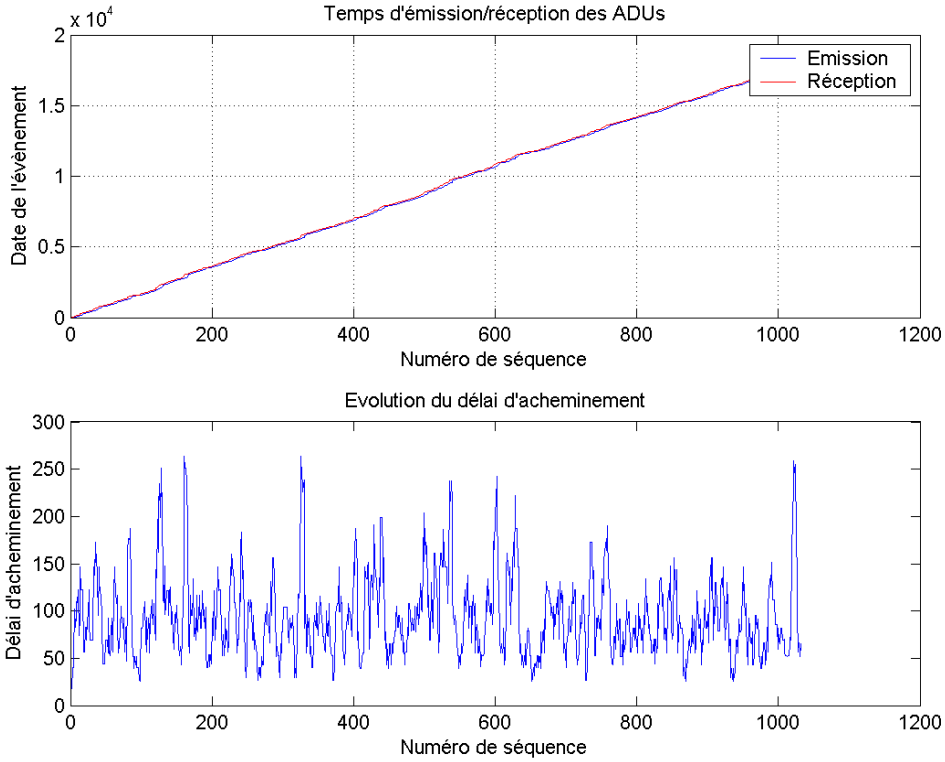

Figure 7 : Dates d'évènements et délai - Go Back N

La Figure 7 présente dans un premier temps un graphe de la répartition temporelle des évènements, ici les émissions et les réceptions. Pour chaque PDU, sa date d'émission ainsi que sa date de réception est tracée, en observant l'éloignement vertical des deux courbes il est possible d'observer le délai de transmission pour le PDU en question. La seconde courbe de cette figure présente l'évolution de ce délai de transmission pour chaque PDU transmis. Les zones de pic représentent les PDUs dont l'acheminement a été le plus long suite aux pertes et retransmissions successives qui ont eu lieu.

La Figure 8 présente l'évolution du débit dans le temps. Le débit est calculé pour chaque échantillon en tenant compte d'une fenêtre de 1000ut sur laquelle est faite la moyenne des émissions. Ceci est fait dans le but de lisser la courbe afin d'en avoir une représentation d'enveloppe.

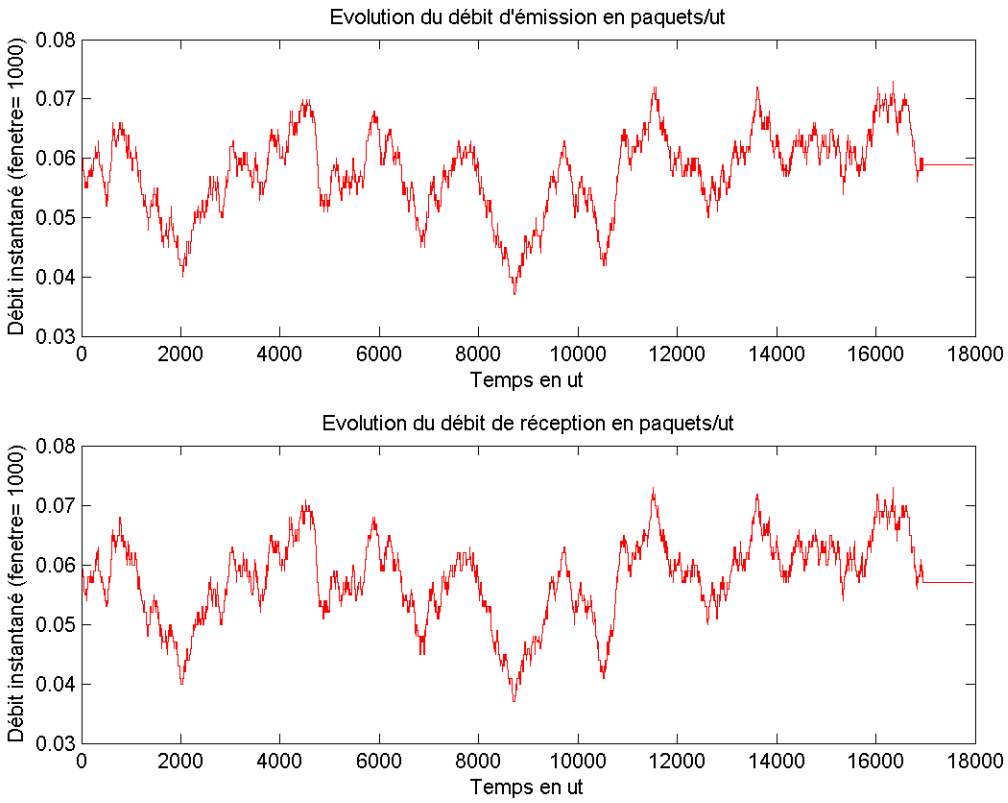

Figure 8 : Débits d'émission et réception - Go Back N

2. Délai et pertes élevés

Une seconde série de mesures a été réalisée avec le médium configuré pour un taux de pertes toujours égal à 50% mais avec cette fois-ci un délai introduit égal à 150ut. Les résultats de cette étude sont présentés ci-dessous :

```
>> analyseTrace('../GBN0.43-50-150-20-150.trace/EMISSION.dat','../GBN0.43-50-150-20-150.trace/RECEPTION.dat', 10000);

-------- MESURES SUR LE DEBIT D'EMISSION ----------
Debit Moyen    : 0.0021808 paquets/ut
Debit Maximum  : 0.0035996 paquets/ut
Debit Minimum  : 0.00089991 paquets/ut

------- MESURES SUR LE DEBIT DE RECEPTION ----------
Debit Moyen    : 0.0021676 paquets/ut
Debit Maximum  : 0.0035996 paquets/ut
Debit Minimum  : 0.00089991 paquets/ut

-------------- MESURES SUR LE DELAI ----------------
Delai Maximum  : 7102 ut
Delai Minimum  : 150 ut
Delai Moyen    : 2543.055 ut

-------------- MESURES SUR LE GIGUE ----------------
Gigue Maximale : 4201 ut
Gigue Minimale : 0 ut
Gigue Moyenne  : 527.3423 ut
```

Tableau 3 : Résultats Textuels - Go Back N - Fort délais

Il est possible de noter sur ces résultats une explosion du délai moyen par rapport aux valeurs obtenues précédemment. De plus, les débits sont nettement inférieurs et la gigue est mal maîtrisée.

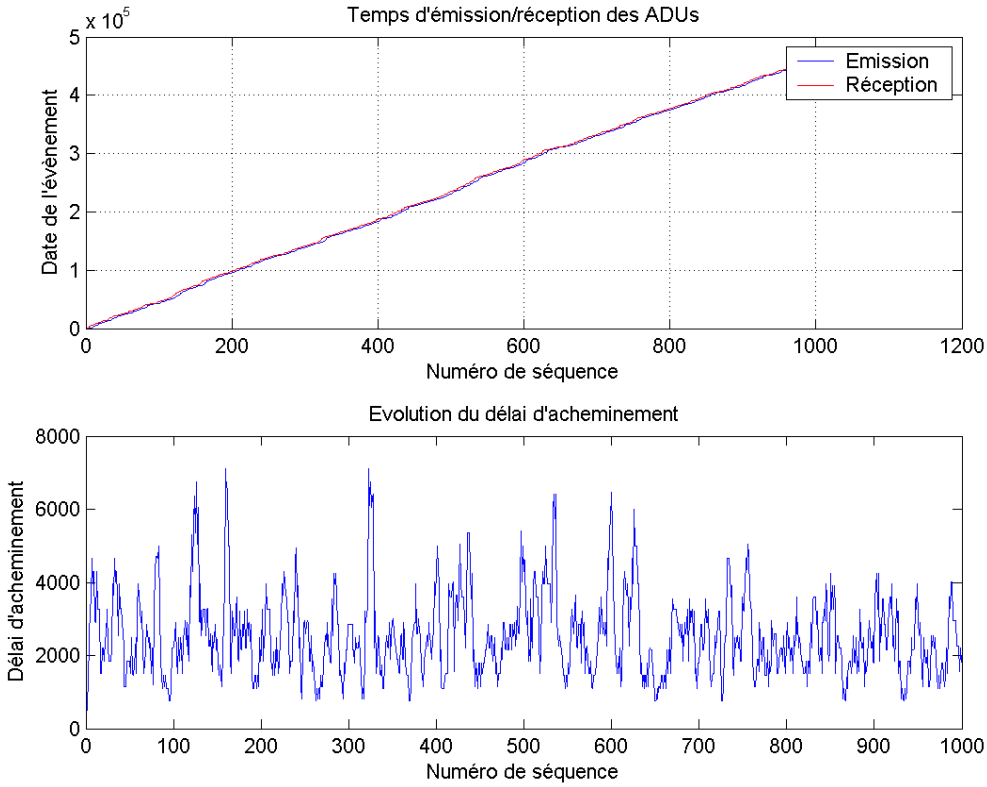

Figure 9 : Date d'évènements et délai - Go Back N - Fort délai

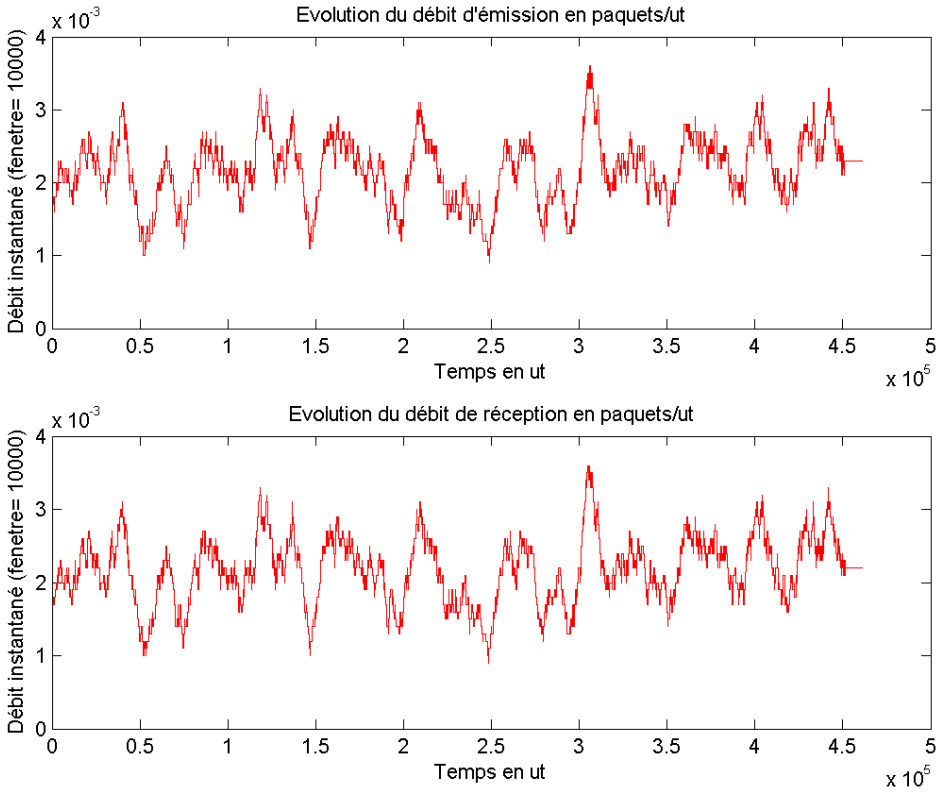

Figure 10 : Evolution des débits - Go Back N - Fort délai

Comme le montre la Figure 9, le délai d'acheminement des paquets

C. Selective Repeat (with SACK)

Le mécanisme du Selective Repeat est celui dont le développement est explicité dans [VAN05], la version avec vecteur d'acquittement (SACK) et dont la retransmission est conditionnée par le résultat du calcul du RTT est utilisé.

1. Délai faible et fortes pertes

Le Tableau 4 présente les résultats textuels obtenus avec ce mécanisme. Il est notable dans un premier temps que les débits obtenus sont plus élevés que ceux qu'il est possible d'obtenir avec les deux mécanismes précédents de par une optimisation de l'utilisation de la bande passante. De plus, bien que la gigue moyenne soit supérieure à celle qui peut être obtenue avec le Go Back N, la gigue maximale est elle inférieure ce qui relève encore une fois de l'efficacité de la sélectivité du mécanisme de retransmission.

```
>> analyseTrace('../SR0.2-50-5-20-5.trace/EMISSION.dat','../SR0.2-50-5-20-5.trace/RECEPTION.dat', 3000);


-------- MESURES SUR LE DEBIT D'EMISSION ----------
Debit Moyen    : 0.06163 paquets/ut
Debit Maximum  : 0.06931 paquets/ut
Debit Minimum  : 0.053316 paquets/ut

------- MESURES SUR LE DEBIT DE RECEPTION ----------
Debit Moyen    : 0.061344 paquets/ut
Debit Maximum  : 0.07031 paquets/ut
Debit Minimum  : 0.051983 paquets/ut

-------------- MESURES SUR LE DELAI ----------------
Delai Maximum  : 336 ut
Delai Minimum  : 24 ut
Delai Moyen    : 98.9132 ut

-------------- MESURES SUR LE GIGUE ----------------
Gigue Maximale : 139 ut
Gigue Minimale : 0 ut
Gigue Moyenne  : 21.2368 ut
```

Tableau 4 : Résultats Textuels - Selective Repeat

Les observations graphiques présentées Figure 11 et Figure 12 viennent conforter l'efficacité notée auparavant. En effet, la courbe présentant l'évolution des délais montre que les délais élevés sont moins fréquents que pour le Go Back N. Ceci est du à la sélectivité des PDUs à retransmettre qui est plus grande et qui permet donc d'éviter d'accroître le délai plus que ce qui est nécessaire.

Au niveau des débits, il est notable que les deux courbes présentent une allure similaire du fait que l'application émettrice applique un mécanisme de contrôle de flux. Néanmoins, une comparaison rapide avec le Go Back N permet de noter que dans le cas présent, la courbe présente moins d'alternances d'une part et que

d'autre part, il est possible de noter que les zones pour lesquelles le débit est élevé tendent à durer plus longtemps. De même, les pentes observées pour les chutes de débit et leur rétablissement sont plus importantes que pour le Go Back N.

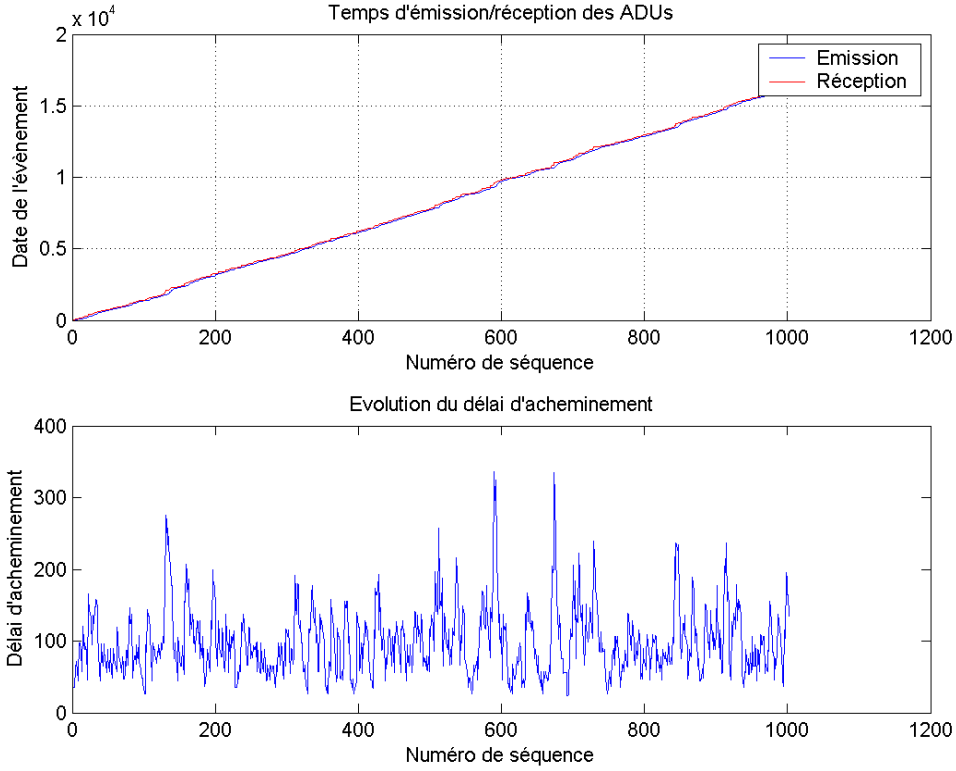

Figure 11 : Date d'évènement et délai - Selective Repeat

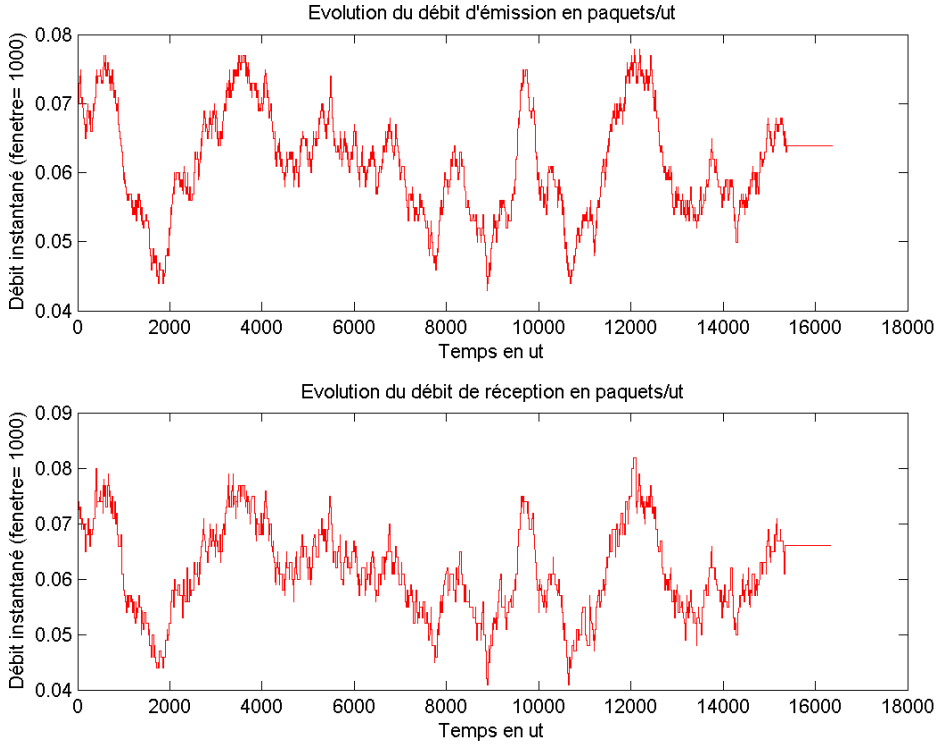

Figure 12 : Evolution des débits - Selective Repeat

2. Délai et pertes élevés

Bibliographie

[EXP03] Ernesto J. Exposito G. – Spécification et mise en oeuvre d'un protocole de transport orienté Qualité de Service pour les applications multimédias. – Rapport LAAS N° 03577

[AWK92] Frédéric LACROIX, Dan BOG - Le Langage AWK – ENSIMAG 1992

[ARM05] F. Armando – Projet de fin d'études – Nouveaux protocoles de transport pour l'adaptabilité au contexte. – INSA-LAAS – Juin 2005

[VAN05] N. Van Wambeke - Mise en place d'un mécanisme de répétition sélective dans le framework ETP – LAAS - Juillet 2005

[Telelogic] Telelogic : http://www.telelogic.com

Table des Illustrations

Figures

Tableaux

ANNEXES

SCRIPT SHELL POUR LE TRIEUR

```
#!/bin/sh

# Ce script shell prépare les traces pour leur présentation a matlab
# Il les mets en forme de façon à avoir deux fichiers, l'un pour
# l'émission, l'autre pour les réceptions, tous deux triés par numéro
# de séquence.

if [ $# -ne 1 ]
  then
      echo "Utilisation : $0 <fichier Trace>"
      exit 1
fi


dirname=`expr "$1" : '\(.*\)\.'`

if [ -d ./$dirname ]
 then
      echo -n "Le repertoire existe deja.. Ecraser[O/n] ? "
      read answer
      if [ $answer = "n" ]
       then
              exit 0
      fi
else
      mkdir ./$dirname
fi

echo Creation du repertoire $dirname

echo "Generation du fichier EMISSION.dat"
grep VEMISSION $1 | awk '{print $6,$9}' > ./$dirname/temp
sort -n -k 2 ./$dirname/temp > ./$dirname/EMISSION.dat

echo "Generation du fichier RECEPTION.dat"
grep VRECEPTION $1 | awk '{print $6,$9}' > ./$dirname/temp
sort -n -k 2 ./$dirname/temp > ./$dirname/RECEPTION.dat

echo "Suppression des fichiers temporaires"
rm ./$dirname/temp

echo "Vos traces sont prêtes dans $dirname"
exit 0
```

SCRIPT MATLAB POUR LE TRAITEMENT

```
function [result] = analyseTrace(e, r, granularite)
% Cette fonction permet de tracer les graphes d'analyse et
% de donner les valeurs caractéristiques défnines pour une trace
% d'exécution TAU du framework.
%
% Les variables e et r sont les noms de fichiers trace
% d'émission et réception générés par le script shell organizeTraces


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Préparation des données
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Chargement des fichiers de trace
em = load(e);
rec = load(r);

% Séparation des données
em_x = em(:,2);
em_y = em(:,1);
rec_x = rec(:,2);
rec_y = rec(:,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% AFFICHAGE EVENEMENTIEL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(2,1,1);
plot(em_x, em_y);
grid on;
hold on;
plot(rec_x,rec_y, 'r');
legend('Emission', 'Réception');
xlabel('Numéro de séquence');
ylabel('Date de l''évènement');
title('Temps d''émission/réception des ADUs');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CALCULS DE DELAI
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

delai = rec_y-em_y(1:length(rec_y));

subplot(2,1,2);
plot(rec_x, delai);
xlabel('Numéro de séquence');
ylabel('Délai d''acheminement');
title('Evolution du délai d''acheminement');

% Calcul des valeurs pour la console
delaiMax = max(delai);
delaiMin = min(delai);
delaiMoy = mean(delai);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCULS DE GIGUE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calcul et affichage de la gigue
```

```
gigue = diff(delai);

figure;
plot(gigue);
xlabel('Intervalle');
ylabel('Gigue en ut');
title('Gigue sur le delai');

gigueMax = max(abs(gigue));
gigueMin = min(abs(gigue));
gigueMoy = mean(abs(gigue));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DEBIT D'EMISSION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

debitMoyE = max(em_x) / max(em_y);

debitI = zeros(max(em_y),1);     % debit Instantané
debitI_x = (1:length(debitI))';

% Pour chaque valeur du temps, on compte le nombre de PDUs envoyés à cet
% instant et on considère que c'est la valeur du débit instantané.
count = 0;
for m = 1:length(debitI)
    count = 0;
    for i = 1:length(em_y)
        if(em_y(i) == m)
            count=count+1;
        end;
    end
    debitI(m) = count;
end

debitE = zeros(length(debitI),1);

% Le débit moyen instantané est calculé en faisant une moyenne sur une
% fenetre de taille: granularite.

for i = 1:length(debitI)-granularite
    debitE(i)=mean(debitI(i:i+granularite));
end

% Traitement des derniers points : quand la fenetre devient trop petite.
debitE(length(debitI)-granularite+1:length(debitI))=
mean(debitI(i:length(debitI)));


subplot(2,1,1);
plot(debitE, 'r+');
xlabel('Temps en ut');
ylabel(['Débit instantané moyen sur fenetre de taille ',
int2str(granularite)]);
title('Evolution du débit d''émission en paquets/ut');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DEBIT DE RECEPTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

debitMoyR = max(rec_x) / max(rec_y);
```

```
debitIr = zeros(max(rec_y),1);     % debit Instantané
debitIr_x = (1:length(debitIr))';

% Pour chaque valeur du temps, on compte le nombre de PDUs reçus à cet
% instant et on considère que c'est la valeur du débit instantané.
count = 0;
for m = 1:length(debitIr)
    count = 0;
    for i = 1:length(rec_y)
        if(rec_y(i) == m)
            count=count+1;
        end;
    end
    debitIr(m) = count;
end

debitR = zeros(length(debitIr),1);

% Le débit moyen instantané est calculé en faisant une moyenne sur une
% fenetre de taille: granularite.

for i = 1:length(debitIr)-granularite
    debitR(i)=mean(debitIr(i:i+granularite));
end

% Traitement des derniers points : quand la fenetre devient trop petite.
debitR(length(debitIr)-granularite+1:length(debitIr))=
mean(debitIr(length(debitIr)-granularite+1:length(debitIr)));


subplot(2,1,2);
plot(debitR, 'r+');
xlabel('Temps en ut');
ylabel(['Débit instantané moyen sur fenetre de taille ',
int2str(granularite)]);
title('Evolution du débit de réception en paquets/ut');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% RESULTATS TEXTUELS POUR LA CONSOLE MATLAB
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp(['-------- MESURES SUR LE DEBIT D''EMISSION ----------']);
disp(['Debit Moyen    : ', num2str(debitMoyE), ' paquets/ut']);
disp(['Debit Maximum  : ', num2str(max(debitE)), ' paquets/ut']);
disp(['Debit Minimum  : ', num2str(min(debitE)), ' paquets/ut']);
disp(' ');
disp(['------- MESURES SUR LE DEBIT DE RECEPTION ----------']);
disp(['Debit Moyen    : ', num2str(debitMoyR), ' paquets/ut']);
disp(['Debit Maximum  : ', num2str(max(debitR)), ' paquets/ut']);
disp(['Debit Minimum  : ', num2str(min(debitR)), ' paquets/ut']);
disp(' ');
disp(['-------------- MESURES SUR LE DELAI ----------------']);
disp(['Delai Maximum  : ', num2str(delaiMax), ' ut']);
disp(['Delai Minimum  : ', num2str(delaiMin), ' ut']);
disp(['Delai Moyen    : ', num2str(delaiMoy), ' ut']);
disp(' ');
disp(['-------------- MESURES SUR LE GIGUE ----------------']);
disp(['Gigue Maximale : ', num2str(gigueMax), ' ut']);
disp(['Gigue Minimale : ', num2str(gigueMin), ' ut']);
disp(['Gigue Moyenne  : ', num2str(gigueMoy), ' ut']);
```