Négociation multi-agents résistante aux pics de charge pour améliorer l'acceptabilité des services d'un fournisseur SaaS ouvert

Amro Najjar, Gauthier Picard, Olivier Boissier

Laboratoire Hubert Curien UMR CNRS 5516, Institut Henri Fayol, Mines Saint-Etienne, Saint-Etienne, France

prenom.nom@emse.fr

RÉSUMÉ. *Le taux d'acceptabilité d'un service et la satisfaction des utilisateurs deviennent des facteurs clés pour éviter le désabonnement des clients et sécuriser le succès de tout fournisseur de logiciel en tant que service (SaaS). Néanmoins, le fournisseur doit également répondre à des charges de travail fluctuantes et minimiser le coût de la location de ressources sur le* cloud. *Pour répondre à ces préoccupations contradictoires, la plupart des travaux existants effectuent unilatéralement la gestion des ressources par le fournisseur. Par conséquent, les préférences de l'utilisateur final et son acceptabilité subjective du service sont pour la plupart ignorées. Afin d'évaluer la satisfaction des utilisateurs et l'acceptabilité du service, des études récentes dans le domaine de la qualité de l'expérience (QoE) recommandent aux fournisseurs d'utiliser des quantiles et des percentiles pour évaluer précisément l'acceptabilité du service utilisateur. Dans cet article, nous proposons un mécanisme de négociation « one-to-many » élastique, résistant à la charge et adaptatif pour améliorer l'acceptabilité du service d'un fournisseur SaaS ouvert. Basé sur l'estimation du quantile du taux d'acceptabilité du service et sur un modèle appris de la stratégie de négociation utilisateur, ce mécanisme ajuste le processus de négociation du fournisseur afin de garantir le taux d'acceptabilité du service souhaité tout en respectant les limites budgétaires du fournisseur. Le mécanisme proposé est mis en œuvre et ses résultats expérimentaux sont examinés et analysés.*

ABSTRACT. *Service acceptability rate and user satisfaction are becoming key factors to avoid client churn and secure the success of any Software as a Service (SaaS) provider. Nevertheless, the provider must also accommodate fluctuating workloads and minimize the cost it pays to rent resources from the cloud. To address these contradicting concerns, most of existing works carry out resource management unilaterally by the provider. Consequently, end-user preferences and her subjective acceptability of the service are mostly ignored. In order to assess user satisfaction and service acceptability recent studies in the domain of Quality of Experience (QoE) recommend providers to use quantiles and percentile to gauge user service acceptability precisely. In this article we propose an elastic, load-spike proof, and adaptive one-to-many negotiation mechanism to improve the service acceptability of an open SaaS provider. Based on quantile estimation of service acceptability rate and a learned model of the user negotiation strategy, this mechanism adjusts the provider negotiation process in order to guarantee the desired service*

Revue d'intelligence artificielle – n° 5-6/2018, 603-625

acceptability rate while meeting the budget limits of the provider and accommodating workload fluctuations. The proposed mechanism is implemented and its results are examined and analyzed.

MOTS-CLÉS : *négociation, adaptation, taux d'acceptabilité, SaaS, cloud computing.*

KEYWORDS: *negotiation, adaptation, acceptability rate, SaaS, cloud computing.*

DOI:10.3166/RIA.32.603-625

1. Introduction

Le taux d'attrition (la proportion de clients perdus sur une période donnée) est un des indicateurs les plus négatifs affectant les fournisseurs de « logiciel en tant que service » (SaaS) (Accenture, 2013). Pourtant, le taux de satisfaction des utilisateurs et le taux d'attrition ne sont pas les seuls défis auxquels doivent faire face les fournisseurs SaaS. Un rapport récent de Cisco prédit que le trafic Internet en heures de pointe augmentera plus rapidement que le trafic Internet moyen entre 2015 et 2020 (Index, 2016). Ainsi, le marché de demain se formera autour d'une demande intensive et fluctuante, avec de fortes attentes client. Afin de palier cette rapide évolution, les fournisseurs de services applicatifs (ASP) migrent de plus en plus vers le *cloud* pour gérer leurs ressources de manière flexible et ainsi minimiser leurs coûts opérationnels. Ainsi, les ASP et les fournisseurs SaaS doivent minimiser le taux d'attrition tout en respectant leurs contraintes budgétaires. Dans le cadre du *cloud computing*, ce problème, objet de nombreuses études, est connu comme la gestion de l'élasticité (Najjar *et al*., 2014 ; Al-Dhuraibi *et al*., 2018). Cependant, la plupart de ces travaux adoptent une approche centralisée où l'ASP prend unilatéralement les décisions d'allocation de ressource. Par conséquent, les préférences utilisateurs sont négligées, et il est souvent présumé que leur seuil d'acceptabilité tolère le meilleur service offert.

Les systèmes multi-agents ont été décrits comme une plate-forme pour distribuer allocation de ressources et coordination dans les écosystèmes de type *cloud* (Talia, 2012). De plus, des travaux récents ont utilisé un système multi-agent pour tenir compte des attentes et de la satisfaction des utilisateurs finaux (Najjar, Serpaggi *et al*., 2016 ; Najjar, Gravier *et al*., 2016). La négociation « one-to-many » multi-agent fournit une plate-forme potentielle pour impliquer les utilisateurs dans le processus de gestion de l'élasticité. Néanmoins, contrairement au cas d'un fournisseur SaaS dont le but est de maximiser le taux d'acceptabilité en concluant autant d'accords que possible, la majorité des travaux existants dans la littérature aborde un scénario où le vendeur cherche à trouver un accord *atomique* conclu avec l'un des acheteurs concurrents. En outre, la plupart de ces travaux supposent l'existence d'un ensemble *fermé* de participants dans lequel les utilisateurs sont supposés être connus à l'avance avant le début du processus de négociation. Cette hypothèse ne tient pas dans l'écosystème en ligne et *cloud* d'aujourd'hui où des milliers d'utilisateurs peuvent entrer dans le processus de négociation chaque minute.

Dans cet article, nous présentons AQUAMan, un nouveau mécanisme de négociation et de coordination multi-enjeu adaptatif [1], élastique et ouvert permettant au fournisseur de cibler le taux d'acceptabilité du service tout en satisfaisant ses contraintes budgétaires. Les utilisateurs peuvent décider d'accepter ou de rejeter le service proposé en fonction de leurs attentes et de l'estimation subjective de la qualité du service (Najjar, Gravier *et al*., 2016). Sur la base de ses mesures de la part des utilisateurs qui trouvent le service inacceptable, le fournisseur ajuste sa stratégie de négociation afin de restaurer le taux d'acceptabilité à ses objectifs prédéfinis. En utilisant le mécanisme proposé, le fournisseur SaaS peut (*i*) assurer un taux d'acceptabilité de service prédéfini précis tout en respectant ses contraintes budgétaires ; (*ii*) s'adapter à la nature dynamique et ouverte de l'écosystème *cloud* où la charge de travail (c.-à-d. le nombre d'utilisateurs entrant dans le système) est variable et subit des pics soudains de charge. Les mécanismes de négociation et de coordination proposés sont développés dans l'architecture EMan (Najjar, 2015 ; Najjar, Serpaggi *et al*., 2016 ; Najjar, Gravier *et al*., 2016).

La suite de cet article est organisée comme suit. La section 2 expose comment la négociation « one-to-many » peut fournir une solution potentielle permettant au fournisseur de satisfaire ses objectifs métier et d'intégrer l'utilisateur dans le processus de décision. La section 3 passe en revue l'architecture EMan, ses agents, ses protocoles de négociation et de coordination. La section 4 détaille AQUAMan, le mécanisme de négociation adaptative, et explique l'approche d'apprentissage et de modélisation de l'adversaire sous-jacent. La section 5 détaille le processus d'évaluation et discute des résultats. La section 6 traite des travaux connexes et, enfin, la section 7 conclut ce document et identifie les perspectives de recherche futures.

2. Motivation

L'élasticité et la gestion des ressources du *cloud* ont été l'objet d'une attention considérable depuis l'émergence du *cloud computing*. Plusieurs travaux de la littérature ont formulé la gestion de l'élasticité comme un problème de satisfaction de contraintes dont le but est de maximiser une fonction d'utilité globale définie par le fournisseur tout en respectant les contraintes budgétaires. Cependant, ces problèmes sont intrinsèquement NP-difficiles. De plus, ils supposent que les demandes de l'utilisateur et la charge de travail appliquée au système sont connues à l'avance. Par conséquent, plusieurs stratégies heuristiques ont été proposées pour surmonter ces limitations (c.f. (Casalicchio, Silvestri, 2013) et les références qui s'y trouvent). Néanmoins, dans la plupart de ces travaux, les préférences des utilisateurs finaux et leur acceptabilité subjective du service sont ignorées ou supposées connues à l'avance par le fournisseur. Par conséquent, le processus de gestion de l'élasticité est fait unilatéralement par le fournisseur.

1. Les termes « AQUAMan » et « mécanisme adaptatif » peuvent être utilisés de manière interchangeable.

Pour comprendre la satisfaction des utilisateurs et leur seuil d'acceptabilité, nous comptons sur la qualité de l'expérience (*Quality of Experience*, ou QoE). Ce dernier est un indicateur paru dans les années 2000 pour évaluer la satisfaction de l'utilisateur final. La QoE est définie comme la qualité de service perçue subjectivement par l'utilisateur et est connue comme étant un facteur déterminant de la décision de l'utilisateur pour accepter ou refuser un service (Hoßfeld *et al*., 2016 ; Möller, Raake, 2014). La QoE vise à fournir une mesure pratique permettant de quantifier la satisfaction des utilisateurs et l'acceptation du service (Möller, Raake, 2014). En particulier, la gestion de la QoE est apparue comme un processus visant à maximiser la QoE tout en optimisant les ressources utilisées (Schatz *et al*., 2014). Cependant, la littérature sur la gestion de la QoE repose largement sur le score d'opinion moyen (*Mean Opinion Score*, ou MOS) pour évaluer la satisfaction et l'acceptabilité du service. Néanmoins, puisqu'il s'agit d'une moyenne des opinions des utilisateurs, le MOS cache des informations importantes sur la diversité des utilisateurs et leurs préférences personnelles (Hobfeld *et al*., 2011). Pour cette raison, d'autres mesures ont été proposées pour permettre au fournisseur de comprendre la satisfaction de l'utilisateur final et d'estimer le taux d'attrition d'un service. Par exemple, des percentiles ont été proposés comme un outil de mesure permettant au fournisseur de s'assurer que, par exemple, 95 % de ses utilisateurs trouvent le service acceptable ou mieux (Hoßfeld *et al*., 2016).

Par définition, un agent est généralement guidé par son propre intérêt et est lié à une perspective individuelle (Wooldridge, 2009). Cela permet aux agents de représenter la subjectivité des opinions des utilisateurs et l'acceptation d'un service donné (Najjar, Serpaggi *et al*., 2016). De plus, les principes de comportement de négociation discutés dans (Pruitt, 2013) fournissent un outil utile pour représenter les attentes des utilisateurs finaux. Ces derniers sont des déterminants clés de la satisfaction des utilisateurs et de l'acceptabilité du service (Sackl, Schatz, 2013 ; Zeithaml *et al*., 1993) (pour plus d'informations sur cette discussion, veuillez vous référer à (Najjar, Gravier *et al*., 2016)).

La négociation « one-to-many » est un sous-type de négociation automatique multi-agent (Lomuscio *et al*., 2003) où un agent (par exemple, un vendeur) négocie simultanément avec plusieurs agents (par exemple, des acheteurs potentiels). Dans la littérature sur la négociation multi-agent, plusieurs approches ont été proposées pour aborder ce type de négociation. Même si elles traitent d'applications différentes, la majorité de ces solutions – comme (Rahwan *et al*., 2002), (An *et al*., 2010) ou (Mansour, Kowalczyk, 2012) – partagent un schéma architectural commun. Pourtant, les stratégies de négociation et de coordination utilisées dans chaque solution dépendent de son but (Najjar, 2015).

La négociation multi-agent « one-to-many » est une approche permettant au fournisseur de négocier simultanément avec plusieurs utilisateurs finaux et d'intégrer leur acceptabilité subjective du service dans le processus de gestion de l'élasticité. Néanmoins, contrairement au cas d'un fournisseur SaaS ou en ligne dont le but est de maximiser le taux d'acceptabilité en concluant autant d'accords que possible, la majorité des travaux existants dans la littérature des négociations « one-to-many » aborde

un scénario où le vendeur cherche à trouver un accord *atomique* conclu avec l'un des acheteurs concurrents. En outre, la plupart de ces travaux supposent un ensemble *fermé* de participants dans lequel les adversaires de l'agent unique sont connus à l'avance avant le début du processus de négociation. De plus, dans la plupart des travaux existants, les offres sont envoyées et reçues de manière synchrone. Ces hypothèses ne tiennent pas dans l'écosystème *cloud* ouvert d'aujourd'hui où des milliers d'utilisateurs peuvent entrer et/ou sortir du système chaque minute et où les sessions de négociation ne sont pas synchronisées. La section 6 fournit d'autres discussions sur les travaux liés aux négociations « one-to-many » et leurs limites, au regard de nos contributions.

3. L'architecture EMan

Dans cet article, nous présentons un nouveau mécanisme de négociation « one-to-many » dénommé AQUAMan (pour Adaptive QUality of experience Aware elasticity Management). AQUAMan est mis en œuvre dans l'architecture EMan (Najjar, Gravier *et al.*, 2016 ; Najjar, 2015). Cette dernière est une architecture multi-agent pour la gestion de l'élasticité SaaS. EMan (figure 1) suit le même schéma architectural discuté dans la section précédente et partagé par la plupart des solutions « one-to-many » existantes dans la littérature. Dans sa version antérieure, les utilisateurs sont représentés par des agents autonomes dont le but est de maximiser la QoE de leurs utilisateurs respectifs. Cependant, ces versions antérieures n'incluent pas le mécanisme adaptatif proposé dans cet article.

Cette section présente les différents types d'agents impliqués dans l'architecture EMan et discute du rôle assumé par le coordinateur.

3.1. Agents

L'architecture EMan représentée dans la figure 1 modélise la négociation qui a lieu entre un fournisseur SaaS et ses utilisateurs finaux ou utilisateurs de services (SU). La négociation entre les fournisseurs SaaS et les fournisseurs de *cloud computing* n'est pas décrite dans cet article. L'architecture EMan contient trois types d'agents : les agents utilisateur de service (notés sa_i), les agents délégués (notés da_i) et un seul coordinateur (noté ca). Ces deux derniers types d'agents représentent le fournisseur.

Agents utilisateurs (SA ou sa_i). Un agent utilisateur de service participe au processus de négociation pour le compte d'un utilisateur de service. Un sa_i a une fonction d'utilité M_{sa_i} qui encode ses préférences. M_{sa_i} est utilisée à chaque cycle t pour évaluer l'utilité des offres $o^t_{da_i}$ provenant du délégué correspondant (voir section 3.2). Si le service comporte J attributs, alors M_{sa_i} est défini comme suit :

$$M_{sa_i}(o^t_{da_i}) = \sum_{j=1}^{j=J} w_{sa_i,j} \cdot \mu_{sa_i,at_j}(o^t_{da_i}[at_j]) \quad (1)$$

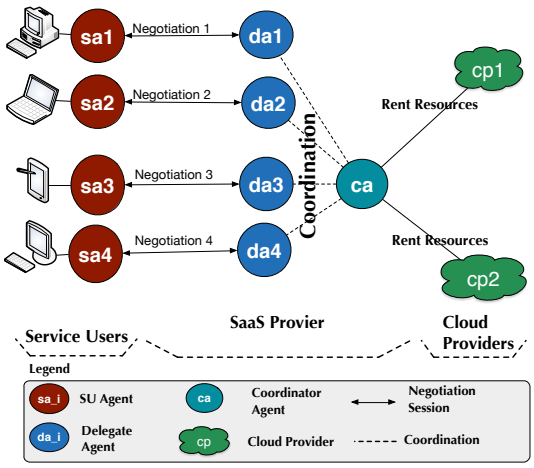


Figure 1. The EMan architecture deployed in the cloud ecosystem

où w_{sa_i,at_j} est le poids associé à l'attribut at_j pour spécifier l'importance que cet utilisateur donne à cet attribut (ces pondérations doivent satisfaire $\sum_{j=1}^{J} w_{sa_i,at_j} = 1$), $o^t_{da_i}[at_j]$ est la valeur dans l'offre $o^t_{da_i}$ du $j^{\text{ème}}$ attribut (i.e. at_j) et μ_{sa_i,at_j} est la fonction d'utilité de cet attribut. Cette fonction est définie comme suit :

$$\mu_{sa_i,at_j}(o^t_{da_i}[at_j]) = \frac{rv_{sa_i,at_j} - o^t_{da_i}[at_j]}{rv_{sa_i,at_j} - pv_{sa_i,at_j}} \tag{2}$$

où $\mu_{sa_i,at_j}(o^t_{da_i}[at_j]) \in [0,1]$ et pv_{sa_i,at_j} et rv_{sa_i,at_j} sont respectivement la valeur préférée (i.e. la meilleure) et la valeur de réserve (i.e. la pire) de sa_i pour cet attribut [2]. Notons que, comme l'ont confirmé des études empiriques, l'utilisateur est capable de choisir les valeurs de $pvat_j$, rv_{sa_i,at_j} et $w_{sa_i,j}$ (Sackl, Schatz, 2013 ; Najjar, Gravier *et al.*, 2016 ; Zeithaml *et al.*, 1993). Par conséquent, sa_i peut les obtenir de l'utilisateur. Ces valeurs ne sont pas divulguées au fournisseur.

Pour prendre sa décision d'acceptation ou de rejet, un sa_i s'appuie sur sa fonction d'utilité et sur son *taux d'aspiration* (AR) courant. $AR^t_{sa_i} \in [0,1]$ indique la quantité d'utilité que sa_i s'attend à obtenir dans ce cycle de négociation t. Pour parvenir à des accords, sa_i fait des concessions en réduisant son AR. Tous les sa_i suivent une stratégie de concession temporelle (ou TBC) (Faratin *et al.*, 1998). Cette hypothèse

2. Comme discuté dans (Najjar, Gravier *et al.*, 2016 ; Najjar, 2015), μ_{sa_i,at_j} peut également être une fonction logarithmique dérivée de la loi Weber-Fechner Law (Thurstone, 1927) et de l'hypothèse logarithmique (Reichl *et al.*, 2010). Cependant, les résultats du mécanisme adaptatif de cet article sont valides pour μ_{sa_i,at_j} linéaire (Equation 2) ou logarithmique.

est assez courante dans la littérature, notamment dans les travaux dont l'objectif est de construire un modèle du comportement de l'adversaire (Baarslag *et al.*, 2015). Par conséquent, $\Delta AR^t_{sa_i}$, la concession faite par sa_i pour le cycle t, dépend du temps restant avant l'échéance. Il est calculé comme suit (Son, Sim, 2012) :

$$\Delta AR^t_{sa_i} = AR^{t-1}_{sa_i} \cdot \left(\frac{t}{T_{sa_i}}\right)^{\lambda_{sa_i}} \tag{3}$$

où T_{sa_i} est la date limite de négociation, et λ_{sa_i} est un paramètre qui contrôle le degré de convexité de la courbe de concession de sa_i. λ_{sa_i} détermine le comportement de sa_i (conciliant, linéaire ou conservateur) (Son, Sim, 2012).

Agents délégués (DA ou da_i). Une fois qu'un nouvel agent utilisateur sa_i entre dans le système, le coordinateur crée un nouvel agent délégué da_i et initie une session de négociation bilatérale avec sa_i. Comme sa_i, da_i a une fonction d'utilité $M_{da_i} \in [0, 1]$. Cependant, l'utilité d'une offre $o^t_{sa_i}$ du point de vue du délégué da_i est déterminée par le coût requis pour servir l'offre. RC est une valeur clé qui détermine le comportement de négociation d'un délégué. Il représente le coût de réserve (maximum) que le délégué dépense sur les utilisateurs. Cette valeur, initialisée par le coordinateur lorsque le da_i est généré, est très importante puisqu'elle permet au fournisseur d'imposer sa contrainte budgétaire, i.e. le coût moyen dépensé pour un utilisateur ne doit pas dépasser RC. Si l'offre n'est pas acceptée par un da_i, il utilise sa stratégie de négociation pour générer une contre-offre. Pour conclure des ententes avec sa_i, le da_i utilise une stratégie de concession fondée sur le temps qui réduit son taux d'aspiration (AR). La concession est calculée comme suit :

$$\Delta AR^t_{da_i} = \frac{1}{T_{da_i}} \tag{4}$$

où T_{da_i} est l'échéance du délégué, initialisée par le ca en fonction de ses connaissances métier. Lorsque da_i atteint T_{da_i}, il ne quitte pas le processus de négociation, il cessera simplement de faire des concessions.

3.2. Négociation bilatérale entre agents utilisateurs (SA) et agents délégués (DA)

Dans l'architecture EMan, les SA et DA n'ont pas accès aux préférences et aux stratégies de négociation des autres agents. Par conséquent, leur comportement de négociation suit la *fonction de décision de négociation* (Faratin *et al.*, 1998) et est déterminé par la fonction d'utilité et la stratégie de négociation. Quant au coordinateur, il est capable de manipuler les stratégies de négociation des délégués si le mode d'adaptation est actif.

Le service négocié est défini par un ensemble de J attributs,$s = \langle at_1, at_2, \cdots, at_J \rangle$. L'objet o^t_i est l'offre ou contre-offre échangée pendant le processus de négociation à l'étape t, entre sa_i et da_i. o^t_i attribue une valeur v^t_k à chacun des attributs décrivant le service négocié. La négociation entre un da_i et un sa_i est basée sur le *protocole*

d'offre alternative. Lorsqu'un sa_i reçoit une offre, il peut (*i*) accepter l'offre, (*ii*) proposer une contre-offre, ou (*iii*) quitter le processus de négociation. Une fois qu'une offre est acceptée par un agent, elle ne peut pas être reniée (Najjar, 2015).

Afin de tenir compte de la nature ouverte et dynamique de l'écosystème *cloud*, les sessions de négociation dans l'architecture EMan sont non synchrones, i.e. certaines sessions seront déjà terminées alors que d'autres sessions seront encore actives ou n'auront pas encore démarré. Pour plus d'informations sur le protocole de négociation et la stratégie d'acceptation, veuillez vous référer à (Najjar, 2015 ; Najjar, Gravier *et al.*, 2016).

Cette section offrait un aperçu de l'architecture EMan. Nous savons que les protocoles de négociation et les stratégies présentés ci-dessus peuvent être exploitables. Toutefois, cette question dépasse le cadre de cet article. Nous considérons que cette question est d'une importance moindre pour la raison suivante :: la négociation dans l'écosystème *cloud* a une forte composante *intégrative* puisque les ressources offertes par le *cloud* semblent être illimitées (Thompson *et al.*, 2010). Par conséquent, au lieu d'exploiter les stratégies de négociation des utilisateurs et de les forcer à faire de lourdes concessions, le fournisseur est plus susceptible de rechercher des règlements gagnant-gagnant permettant de maximiser le taux d'acceptabilité tout en respectant ses contraintes budgétaires.

4. Stratégie de négociation « one-to-many » adaptative

Cette section détaille notre algorithme adaptatif. Afin d'estimer le taux d'acceptation à tout moment t, le fournisseur s'appuie sur un algorithme d'estimation de quantile présenté dans la section 4.1. Si le taux d'acceptation ciblé est violé, le coordinateur active le mode adaptatif. Avec ce mode actif, un délégué da_i doit analyser le comportement de son « adversaire »[3] sa_i, apprendre un modèle de sa stratégie de négociation afin d'estimer son délai de négociation $\tilde{T_{sa_i}}$ et envoyer cette estimation au coordinateur (section 4.2). Ce dernier choisit les sessions de haute priorité (en fonction de leurs délais) et ajuste leurs stratégies de négociation tout en respectant les contraintes budgétaires (section 4.3).

4.1. Déclenchement de l'adaptation

Les quantiles et les percentiles sont des valeurs qui partagent un ensemble fini de valeurs en q sous-ensembles de tailles (presque) égales. La littérature sur la QoE et la satisfaction de l'utilisateur recommande aux fournisseurs de s'appuyer sur les quantiles et les percentiles comme mesures plus précises (par rapport à MOS) pour évaluer l'acceptabilité du service (Hoßfeld *et al.*, 2016).

3. Notons les guillemets autours de la notion d'adversaire, terme utilisé en négociation, bien qu'ici le délégué n'ait pas forcément de relation conflictuelle avec l'utilisateur.

Chaque fois que la session i est terminée, le coordinateur est informé par da_i des résultats de cette session. En utilisant ces données, le coordinateur exécute un algorithme d'estimation de quantile pour détecter le taux d'acceptabilité du service courant.

Soit Q la fonction d'estimation quantile ou percentile. Soit R le jeu de données contenant les résultats des sessions terminées. R peut contenir soit 0, pour les sessions échouées, soit 1 pour les sessions réussies. Si le coordinateur cherche à s'assurer que β pourcents des utilisateurs qui ont fait appel au service à ce jour ont eu une session de négociation réussie (acceptant la qualité de service proposée), le coordinateur doit vérifier que le $(100 - \beta + 1)^{\text{ème}}$ percentile est égal à 1 :

$$Q(R, 100 - \beta + 1) = 1 \tag{5}$$

Tant que cette condition est maintenue, le coordinateur n'a pas besoin d'intervenir dans le processus de négociation. Dès que la condition de l'équation 5 est violée, le coordinateur déclenche le mécanisme d'adaptation en commandant à tous les délégués actifs d'activer leur mode adaptatif. De plus, lorsque le coordinateur engendre de nouveaux délégués, ils entreront également dans ce mode actif.

Notez que le coordinateur continue d'évaluer l'équation 5 même après l'activation du mode adaptatif. La section suivante détaille le rôle des délégués après l'activation du mode adaptatif.

4.2. Apprentissage du comportement de l'adversaire

4.2.1. Estimation de la concession de l'adversaire

Même lorsque le mode adaptatif n'est pas actif, lorsqu'un délégué da_i reçoit $o^t_{sa_i}$ au cycle t du sa_i correspondant, il estime la concession faite par sa_i en comparant $o^t_{sa_i}$ avec $o^{t-1}_{sa_i}$ l'offre précédente faite par sa_i. Puisque da_i n'a pas accès aux préférences de sa_i ou à la fonction M_{sa_i}, il ne peut pas calculer la concession réelle de sa_i. Au lieu de cela, il s'appuie sur sa propre fonction d'utilité pour estimer la concession faite par sa_i en supposant qu'une concession faite par sa_i est synonyme d'un gain d'utilité pour da_i. Ainsi, l'estimation de la concession faite par sa_i au cycle de négociation t, est définie comme suit :

$$c^t_{sa_i} = M_{da_i}(o^t_{sa_i}) - M_{da_i}(o^{t-1}_{sa_i}) \tag{6}$$

Pour apprendre le comportement de concession de sa_i, da_i effectue le suivi des concessions de sa_i pendant la session de négociation. La figure 2 compare $\Delta AR^t_{sa_i}$, les concessions réelles faites par sa_i (en bleu), avec $c^t_{sa_i}$, l'estimation par da_i de la même concession (en rouge). Dans cet exemple, $T_{sa_i} = 80$ cycles et $\lambda_{sa_i} = 3,0$. Comme le montre la figure, bien que la courbe rouge soit bruitée (car elle est estimée

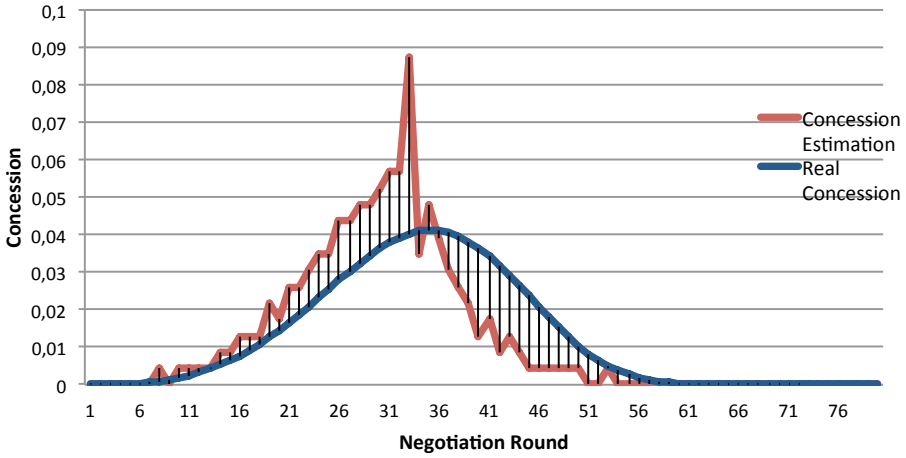


Figure 2. $\Delta AR^t_{sa_i}$, la vraie concession faite par sa_i (en bleu) comparée à $c^t_{sa_i}$, l'estimation faite par da_i (en rouge)

à l'aide de la fonction d'utilité de da_i, puis normalisée), elle semble fournir une estimation adéquate de la concession de sa_i puisque l'objectif principal est d'apprendre T_{sa_i}, la date limite de négociation de sa_i.

4.2.2. Apprentissage du modèle de l'adversaire

Si le mode d'adaptation est actif, les données de concessions recueillies dans le paragraphe précédent devraient être utilisées pour établir un modèle de comportement de concession de sa_i et inférer T_{sa_i}. Ceci peut être réalisé, comme cela a été montré dans la littérature (Baarslag *et al.*, 2015), en utilisant une régression non linéaire supposant que tous les utilisateurs suivent une concession basée sur le temps définie par l'équation 3. Cependant, à la différence des travaux existants prédominants dans la littérature, où apprendre le modèle de négociation de l'adversaire est fait et corrigé à chaque cycle une fois qu'une nouvelle concession est faite, dans les arrangements de négociation « one-to-many » abordés dans cet article, une telle approche n'est pas pratique pour des raisons d'évolutivité. En effet, des centaines ou des milliers d'utilisateurs peuvent négocier avec le fournisseur simultanément. En outre, si le fournisseur exécute les algorithmes d'apprentissage des délégués sur les ressources louées sur le *cloud*, cela peut augmenter considérablement les coûts.

Pour cette raison, dans la solution proposée, un délégué peut exécuter l'algorithme de régression non linéaire une seule fois. Il s'agit donc maintenant de déterminer quand un délégué doit le faire. D'une part, si un délégué lance ce processus trop tôt dans la session de négociation, le processus de régression n'aura que quelques points d'entrée. D'autre part, si le délégué attend trop longtemps, l'utilisateur correspondant risque d'atteindre son échéance et de quitter le processus de négociation.

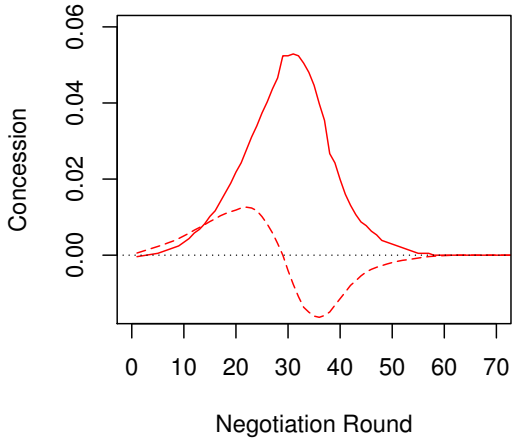


Figure 3. Une version lissée de $c^t_{sa_i}$ (ligne pleine) et sa dérivée (ligne pointillée)

Tous les sa_i suivent une stratégie de concession fondée sur le temps dans laquelle le taux de concessions de sa_i ralentit quand sa_i a déjà fait la plupart des concessions qu'il pouvait faire. Par conséquent, quel que soit le type de sa_i (conservateur ou linéaire) et de son échéance réelle T_{sa_i}, lorsque le taux de changement (ou la dérivée) de l'estimation par da_i de la concession de sa_i diminue significativement en valeurs négatives, cela signifie que sa_i a fait la plupart de ses concessions et que, pour le reste de la session de négociation avant que sa_i atteigne T_{sa_i}, sa_i cessera de faire des concessions considérables. Ainsi, si da_i lance sa régression non linéaire à ce stade, il aura le plus de points de données utiles.

La figure 3 superpose $c^t_{sa_i}$ avec sa première dérivée. Le sa_i de cette figure est exactement le même que celui de la figure 2, mais nous avons utilisé un filtre Savitzky-Golay pour lisser la courbe et filtrer le bruit (Savitzky, Golay, 1964). Pour calculer le taux de variation des concessions, nous calculons la dérivée et nous lissons le résultat en utilisant une variante du même filtre.

L'algorithme de régression non linéaire prend comme variables d'entrée le cycle de négociation (t) et la concession estimée ($c^t_{sa_i}$) et émet des valeurs estimées des paramètres T_{sa_i} et λ_{sa_i}, notée $\tilde{T_{sa_i}}$ et $\tilde{\lambda_{sa_i}}$ respectivement. Alors, d'après $\tilde{T_{sa_i}}$, da_i calcule $\tilde{r_i} = \tilde{T_{sa_i}} - t$, le nombre estimé de cycles [4] restant dans la session i. Tant que la session i n'est pas terminée, da_i met à jour $\tilde{r_i}$ à chaque cycle et l'envoie à plusieurs reprises au coordinateur.

4. Les termes *cycle* et *round* sont utilisés indifféremment ici.

4.3. Adaptation de la négociation

Comme nous l'avons expliqué dans la section précédente, le coordinateur reçoit $\tilde{r_i}$ de chaque session de négociation i dont le délégué da_i considère que sa_i s'approche de la fin de son échéancier et mérite d'être priorisé. Ces estimations $\tilde{r_i}$ sont stockées dans une *liste des priorités* qui est triée en continu par ordre croissant : une session dont la date limite est estimée plus tôt sera en haut de la liste.

Notons que puisque les sessions de négociation sont non synchronisées, le coordinateur doit répéter le tri chaque fois qu'il reçoit une nouvelle estimation d'un délégué da_i. En outre, chaque fois qu'une session est terminée avec succès ou non, le coordinateur supprime son enregistrement de la liste des priorités. Ainsi, dans la liste des priorités, les sessions sont triées de la plus haute à la plus basse des priorités. Maintenant le coordinateur doit décider combien de ces sessions prioritaires doivent être *priorisées*. Pour ce faire, le coordinateur entreprend ce processus comme suit :

1. Estimer le taux d'acceptabilité courant (en utilisant la fonction d'estimation de quantile comme décrit dans la section 4.1). Ensuite, calculer la différence entre le taux d'acceptabilité souhaité et sa valeur courante actuelle.

2. Estimer p le nombre de sessions réussies requises pour rétablir le taux désiré et choisir les premières p sessions de la liste de priorités et les *prioriser* en ajustant leurs stratégies de négociation pour les encourager à accepter le service.

Lorsque la stratégie de négociation d'un délégué da_i est ajustée, da_i utilise une stratégie de concession fondée sur le temps définie dans la section 3.1, affectées des modifications suivantes. Tout d'abord, le délai de négociation de da_i devient $\tilde{r_i}$ au lieu de T_{da_i}. Deuxièmement, le coût de réserve de da_i est augmenté de la valeur notée $RcPrio$. $RcPrio$ est calculé en divisant l'excédent disponible dans *Surplus* parmi toutes les sessions priorisées. *Surplus* est une variable dans laquelle le coordinateur accumule le surplus obtenu lors de sessions de négociation réussies. Plus précisément, lorsqu'une session est déclarée réussie, le coordinateur met à jour sa variable *Surplus* comme suit :

$$\textit{Surplus} \leftarrow \textit{Surplus} + surplus_i \tag{7}$$

où $surplus_i$ est l'excédent obtenu de la session de négociation réussie i. $surplus_i$ est calculé comme suit :

$$surplus_i = RC - Cost(\Pi(\hat{o_i})) \tag{8}$$

Où RC est le coût de réserve des délégués, $\hat{o_i}$ est l'offre acceptée dans la session i et $Cost(\Pi(\hat{o_i}))$ est le coût à payer par le fournisseur pour satisfaire à cette offre. Ainsi, la différence entre le coût réel et le coût (maximum) de la réserve est considérée comme excédentaire. Notez que les délégués ne peuvent pas obtenir plus de

$RcPrio_{max} = RC/2$, considéré comme la valeur maximale pour les sessions priorisées. Troisièmement, le coût préféré de da_i est changé pour le coût de la dernière offre qu'il a faite. $AR^t_{da_i}$ est remis à 1.

Cette section a présenté la modélisation de l'adversaire et les processus d'adaptation de la négociation. La section suivante présente l'évaluation expérimentale de notre approche

5. Évaluation expérimentale

Pour évaluer AQUAMan, notre mécanisme adaptatif, nous l'avons mis en œuvre dans l'architecture EMan (Najjar, Gravier *et al.*, 2016 ; Najjar, 2015). Cette dernière est implémentée en utilisant Repast Simphony (North *et al.*, 2005), un environnement de simulation multi-agent. L'évaluation est organisée en trois expériences. La première expérience (section 5.2) vise à évaluer l'algorithme d'adaptation. La deuxième expérience (section 5.3) examine l'impact de la charge de travail appliquée au fournisseur (c.-à-d. le nombre d'utilisateurs entrant dans le système par minute) sur le taux d'acceptabilité. La troisième expérience (section 5.4) évalue le surcoût des mécanismes de négociation et de coordination. Avant de discuter de ces expériences, la section suivante présente leurs paramètres.

5.1. Paramètres expérimentaux

Le nombre total d'utilisateurs entrant dans le système est noté $|SU|$. Dans les expériences suivantes, $|SU| = 10000$ utilisateurs. Les utilisateurs entrent dans la simulation suivant un processus aléatoire de Poisson dont la valeur moyenne est A par minute. Le service dans le scénario expérimental implique deux attributs: l'un est le délai de livraison du service tandis que l'autre représente la qualité du service. Les profils utilisateur (réserve, valeurs préférées et pondérations pour chaque attribut) sont générés au hasard. Les délais de négociation des sa_i sont générés au hasard : $T_{sa_i} \in [40 : 120]$, et $\lambda_{sa_i} \in [1 : 8]$ (i.e. les utilisateurs peuvent être linéaires ou conservateurs). Le coût des services acceptables par les utilisateurs varie de $0,1$ à $0,9$. RC, le coût de réserve du délégué (le coût maximal alloué à un utilisateur non prioritaire) est fixé à $0,6$. Ce paramètre représente les contraintes budgétaires du fournisseur. Par conséquent, sans le mode d'adaptation, environ un quart des utilisateurs n'accepteront pas le service puisque RC ne peut pas satisfaire leur service le moins acceptable, $RcPrio_{max} = RC/2$. Par conséquent, lorsqu'un utilisateur sa_i est prioritaire, le RC du délégué respectif da_i est au plus augmenté de $RC/2$. $Goal$ est le pourcentage d'utilisateurs que le fournisseur cherche à satisfaire. Son impact est évalué dans la section suivante. Notez que les résultats décrits ci-dessous restent valides même si les valeurs des paramètres ci-dessus (par exemple, RC, $RcPrio_{max}$) sont modifiées.

Veuillez vous référer à notre travail précédent (Najjar, Mualla *et al.*, 2017) pour une évaluation complète et une analyse de l'impact de ces paramètres liés aux coûts.

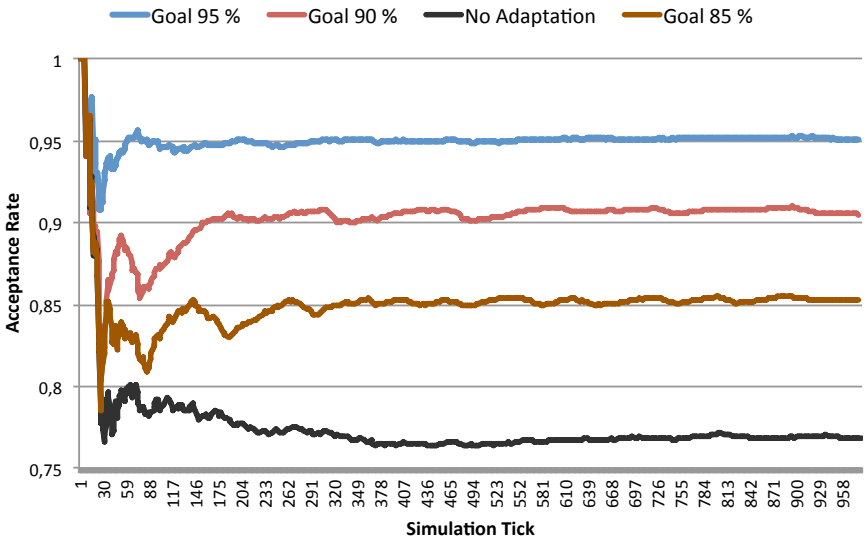


Figure 4. Taux d'acceptation avec $A = 160$.

De plus, nous avons obtenu des résultats similaires lorsque la fonction d'utilité d'attribut d'utilisateur est une fonction logarithmique dérivée de l'hypothèse logarithmique (Reichl *et al.*, 2010 ; Najjar, Gravier *et al.*, 2016).

5.2. Évaluation du mécanisme d'adaptation

La figure 4 montre les résultats de cette expérience. La courbe bleue représente le taux d'acceptation lorsque $Goal = 95\ \%$. Comme le montre la figure, au début de la simulation, la valeur du taux d'acceptation a oscillé avant de se stabiliser sur la valeur d'objectif ($Goal = 95\ \%$). Les courbes rouge et marron, qui tracent le taux d'acceptation lorsque $Goal = 90\ \%$ et $Goal = 85\ \%$, montrent des résultats similaires. Pour comparer les résultats du mécanisme d'adaptation, la courbe noire trace le taux d'acceptation lorsque le mécanisme est désactivé.

Comme on peut le constater à partir de ces résultats, le mécanisme d'adaptation a réussi à rétablir le taux d'acceptation de l'objectif prédéfini $Goal$ d'une manière précise. Ceci est expliqué par le fait que l'algorithme de prédiction discuté dans la section 4.2 a réussi à prédire le délai de négociation utilisateur T_{sa_i} avec une précision acceptable. Le taux d'erreur moyen entre la prédiction $\tilde{T_{sa_i}}$ et la valeur réelle T_{sa_i} est d'environ 6 cycles, ce qui signifie qu'un délégué da_i surestime/sous-estime T_{sa_i} par trois cycles de négociation moyenne.

Intuitivement, le taux d'acceptabilité de service atteint par l'algorithme de prédiction est livré avec plus de coûts investis par utilisateur. Cependant, cette augmentation

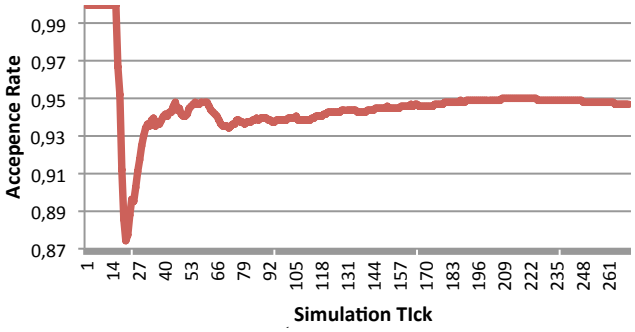


Figure 5. Taux d'acceptation (Goal=95 % et $A = 1280$)

ne viole pas la contrainte budgétaire du fournisseur : le coût moyen par utilisateur était de $0,55$, $0,52$, $0,5$ et $0,44$ pour $Goal = 95$ %, $Goal = 90$ %, $Goal = 85$ % et non-adaptatif, respectivement. Ainsi, la contrainte budgétaire du fournisseur (i.e. le coût moyen par utilisateur ne dépasse pas $RC = 0,6$) a été satisfaite parce que le fournisseur s'appuie sur le *Surplus* pour servir les utilisateurs priorisés.

Notez que l'étude du rapport coût-efficacité du mécanisme adaptatif et de ses limites pour des contraintes de coûts données dépasse le cadre de cet article puisque ces problèmes ont été traités dans notre travail précédent (Najjar, Mualla *et al.*, 2017).

5.3. Impact de la charge A

Cette expérience étudie l'impact de A, la charge de travail appliquée au système, sur le taux d'acceptation afin d'évaluer l'élasticité du mécanisme d'adaptation. Le but de cette expérience est double : d'abord, nous étudions la réponse du système à des charges de travail élevées mais constantes ; et, nous évaluons la résistance à la charge du système par l'application d'un pic de charge soudain.

Tableau 1. Impact de la charge de travail A sur le taux d'acceptation

A	20	40	80	160	320	640	1280	2500
Avec adaptation	95 %	95 %	95 %	95 %	95 %	94,9 %	94,7 %	93,3 %
Sans adaptation	77 %	77 %	77 %	77 %	77 %	77 %	77 %	77 %

Le tableau 1 montre le taux d'acceptation, à la fin de la simulation, lorsque $Goal = 95$ %[5] et A augmentant de 20 à 2 500 utilisateurs par minute. Comme le montre le tableau, le mécanisme s'est avéré très élastique. Lorsque $A \in [20 : 320]$ le taux d'acceptation ne change pas à mesure que A augmente. Toutefois, lorsque A augmente à des

5. Nous avons effectué la même expérience avec différentes valeurs de $Goal \in \{92\ \%, 90\ \%, \ldots\}$ et avons obtenu des résultats similaires.

valeurs plus élevées (640 à 2 500), le taux d'acceptation témoigne d'une légère diminution. Cependant, même avec un très haut A, les résultats obtenus par le mécanisme d'adaptation restent très proches de l'objectif. Cette diminution est expliquée comme suit : l'algorithme d'adaptation est réactif et le coordinateur ne peut évaluer le taux d'acceptation qu'à chaque fois qu'une session se termine avec succès ou sans succès. Ainsi, lorsque A est trop élevé, un nombre significatif de sessions peut échouer dans le même cycle de simulation et l'algorithme d'adaptation exécuté par le coordinateur souffre d'un léger retard et devient un peu court pour répondre à l'objectif prédéfini ($Goal = 95\ \%$), comme on peut le voir dans le tableau. Pourtant, ce résultat est à relativiser pour la raison suivante : $|SU|$, le nombre total d'utilisateurs dans la simulation, influence le seuil d'élasticité du mécanisme d'adaptation. Par exemple, lorsque $|SU| = 10\ 000$ et $A = 2\ 500$, les utilisateurs entrent dans le système en 4 vagues massives représentant environ un quart du nombre total d'utilisateurs. Ainsi, le coordinateur n'a pas assez de temps pour rétablir le taux d'acceptation à ($Goal = 95\ \%$). La figure 5 montre comment le mécanisme d'adaptation réagit à une chute du taux d'acceptation lorsque $A = 1\ 280$. Le coordinateur parvient à restaurer la valeur de l'objectif prédéfini ($Goal = 95\ \%$). Pourtant, une nouvelle chute se produit avec la nouvelle vague d'utilisateurs et la simulation est terminée. Notez que dans un scénario réaliste, lorsque $|SU|$ tend vers l'infini, le coordinateur sera toujours en mesure de restaurer le taux d'acceptation dans sa valeur d'objectif, même avec un A significatif.

5.3.1. Résistance aux pics de charge

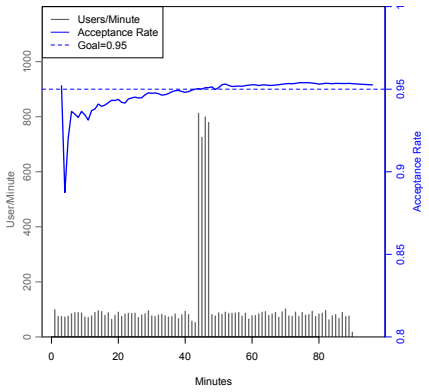
Pour évaluer l'impact des pics de charge, dans cette expérience, A, le taux d'arrivée, subira une augmentation soudaine (c.-à-d. un pic de charge). En moyenne, 80 utilisateurs par minute entreront dans le système. Pourtant, pendant quatre minutes, un pic de charge augmente A jusqu'à 800 utilisateurs par minute. Ainsi, quatre vagues d'utilisateurs entreront dans le système chaque vague contenant 800 utilisateurs en moyenne.

La figure 6a trace le taux d'acceptation atteint par le mécanisme adaptatif avec une charge de travail $A = 80$ utilisateurs par minutes en moyenne et un pic de charge $A = 800$ à la 45^e minute pendant quatre minutes [6].

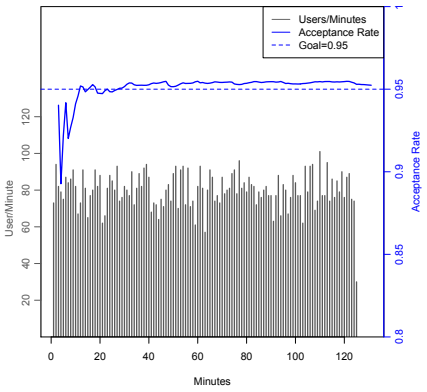
Comme le montre la figure, après une chute du taux d'acceptabilité au début de l'expérience (vers la 10^e minute), le mécanisme adaptatif est activé et parvient à rétablir le taux d'acceptabilité défini par le fournisseur ($Goal = 95\ \%$ tracé par la ligne pointillée bleue sur la figure). Plus important encore, le pic de charge n'a pas d'impact sur le taux d'acceptation atteint. Le $Goal = 95\ \%$ a été atteint avant le pic de charge et il reste satisfait pendant et après le pic de charge.

La figure 6b trace la valeur du taux d'acceptation mais cette fois, une charge de travail uniforme de $A = 80$ utilisateurs par minute en moyenne a été appliquée. La comparaison des deux figures montre que la principale différence est que la simulation

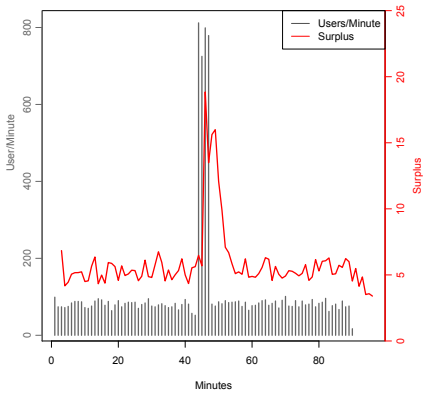
6. Dans cette expérience, *minute* équivaut à 30 ticks de simulation.



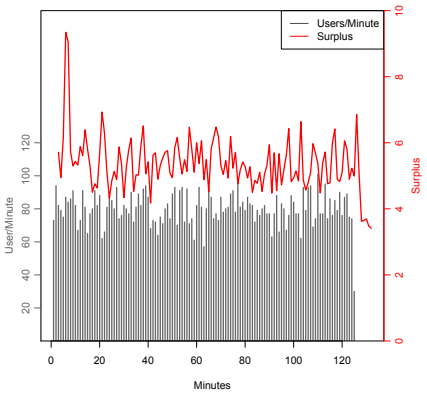
(a) Impact du pic de charge sur le taux d'acceptation (en bleu).



(b) Taux d'acceptation (en bleu) avec une charge constante $A = 80$.



(c) Impact du pic de charge sur le *Surplus* (en rouge).



(d) *Surplus* (en rouge) avec une charge constante $A = 80$.

Figure 6. Les lignes verticales en gris représentent le nombre d'utilisateurs entrant dans le système, par minute. La courbe bleue (resp. rouge) représente le taux d'acceptation (resp. le contenu du Surplus) avec (à gauche) ou sans (à droite) pic de charge

dans 6a s'est terminée plus tôt (autour de la 90^e minute) à cause du pic de charge alors que dans la figure 6b la simulation a duré environ 30 minutes de plus. Malgré cette différence mineure, le pic de charge n'a pas eu d'impact significatif sur le taux d'acceptation atteint par le mécanisme adaptatif et dans les deux cas (avec et sans pointes de charge), le $Goal$ prédéfini a été atteint de manière précise.

Pour comprendre cette résistance aux pics de charge, la figure 6c montre l'impact d'un tel pic de charge sur le surplus recueilli par le mécanisme de redistribution des surplus (voir section 4.3). La courbe rouge trace le contenu du *Surplus* utilisé pour les sessions prioritaires. Comme on peut le voir sur la figure, la quantité disponible dans *Surplus* augmente significativement pendant le pic de charge, puis redescend à ses valeurs moyennes après la fin du pic de charge. Ceci s'explique car le pic de charge augmente le nombre d'utilisateurs entrant dans les systèmes. En moins de quatre minutes, environ 2400 utilisateurs entrent dans le système en quatre vagues massives. Certains des utilisateurs arrivant dans le pic de charge peuvent être satisfaits rapidement et facilement en raison de leurs attentes relativement moins coûteuses. Par conséquent, le nombre de sessions réussies augmentera, ce qui augmentera le montant total des surplus collectés par le coordinateur. Ce surplus sera ensuite utilisé par le mécanisme de redistribution des excédents pour dépenser les utilisateurs intransigeants afin de conserver la valeur $Goal$. C'est pourquoi le contenu du *Surplus* descendra à son niveau d'avant lorsque le pic de charge sera terminé.

Enfin, la figure 6d trace le contenu de *Surplus* quand aucun pic de charge n'est appliqué. Comme on peut le voir sur la figure, après une augmentation initiale du surplus, le mécanisme adaptatif est activé et le *Surplus* est utilisé pour les sessions prioritaires. Cependant, comme aucun pic de charge n'est appliqué, contrairement à la figure 6c, le *Surplus* dans la figure 6d ne voit aucune augmentation significative.

5.3.2. Discussion

Cette expérience a évalué l'impact de la charge de travail sur le taux d'acceptation atteint par le mécanisme adaptatif. Dans la première partie de cette expérience (tableau 1 et figure 5), nous avons montré que le taux d'acceptation atteint par le mécanisme adaptatif reste inchangé avec des charges de travail relativement élevées (jusqu'à 1280 utilisateurs par minute). Cependant, une charge de travail intense mais uniforme n'est pas souvent le cas. Au lieu de cela, dans le marché du *cloud* d'aujourd'hui, les fournisseurs SaaS sont souvent exposés à des pics de charge ou à des charges de travail importantes aux heures de pointe, pendant les week-ends ou pendant les vacances. Pour cette raison, dans la deuxième partie de cette expérience, nous avons appliqué un pic de charge qui a décuplé la charge de travail en moins de cinq minutes. Les résultats de cette expérience (figure 6) prouvent que le mécanisme adaptatif est résistant aux pics de charge. Cela le rend capable de gérer la nature dynamique de l'écosystème *cloud* où des milliers d'utilisateurs peuvent se précipiter dans le portail de service en quelques minutes. De plus, nous avons expliqué comment cette caractéristique de résistance au pic de charge est réalisée par le mécanisme de redistribution de surplus. En particulier, le mécanisme de redistribution des surplus absorbe le sur-

plus obtenu lors des séances de négociation réussies et l'utilise pour servir les utilisateurs intransigeants. Ainsi, le *Surplus* agit comme un tampon permettant de stabiliser le taux d'acceptabilité (voir figure 6a) malgré le pic de charge.

5.4. Surcoût de la coordination et de la négociation

Pour évaluer les surcoûts induits par la coordination et les mécanismes de négociation, nous comptons le nombre de messages échangés pour entreprendre l'intervention du coordinateur et le nombre de messages échangés respectivement lors des sessions de négociation bilatérales.

Sans le mécanisme d'adaptation, le coordinateur est sollicité une seule fois par session pour obtenir le résultat de la session (succès ou échec). Avec le mécanisme adaptatif, le nombre de messages échangés entre les délégués et le coordinateur augmente d'environ $50\ \%$. Nous considérons que cette augmentation est insignifiante pour les raisons suivantes. Tout d'abord, même avec le mécanisme adaptatif, les tâches assumées par le coordinateur sont légères comme cela a été discuté dans les sections 4.1 et 4.3. Deuxièmement, étant donné qu'il est probable que le fournisseur SaaS gère les délégués et le coordinateur sur le même centre de données dans le *cloud*, le coût de communication entre eux est négligeable.

En ce qui concerne le nombre de messages de négociation, les résultats montrent une légère diminution ($\approx 7\ \%$) lorsque le mécanisme adaptatif est actif, car ce dernier aide à conclure des accords plus rapidement.

6. Travaux connexes

Cette section aborde les travaux relatifs à l'adaptation et à l'apprentissage dans la négociation multi-agent ainsi que les travaux connexes dans le domaine de la négociation.

6.1. Apprentissage du modèle de négociation de l'adversaire

Apprendre et modéliser le comportement de négociation de l'adversaire est un sujet à part entière de recherche (Baarslag *et al.*, 2015). Le modèle de l'adversaire est généralement utilisé pour aider un agent à s'adapter au comportement de son adversaire, atteindre des règlements gagnant-gagnant, ou minimiser le coût de la négociation. Diverses techniques d'apprentissage sont utilisées dans la littérature. Certaines approches apprennent la stratégie d'acceptation de l'adversaire ou son profil de préférence. Plus important encore, apprendre la date limite de l'adversaire a reçu une attention considérable. En particulier, certains travaux utilisent des techniques d'apprentissage bayésien (Ji *et al.*, 2014), tandis que d'autres utilisent la régression non linéaire (Hou, 2004) ou utilisent les deux. Néanmoins, tous ces travaux traitent de la négociation bilatérale, un choix qui ne s'applique pas à la gestion de l'élasticité lorsque le fournisseur SaaS cherche à maximiser le taux d'acceptabilité du service.

6.2. Négociation « one-to-many »

En dépit de la littérature relativement riche sur les négociations « one-to-many », l'objectif de la plupart de ces travaux, modélisant des négociations entre un acheteur et de nombreux concurrents vendeurs, est de trouver un accord unique (i.e. *atomique*) qui maximise l'utilité de l'acheteur tandis que d'autres sessions, moins bénéfiques, sont avortées (Mansour, Kowalczyk, 2012 ; Rahwan *et al*., 2002). En outre, les mécanismes de négociation et de coordination existants (par exemple (Rahwan *et al*., 2002 ; Mansour, Kowalczyk, 2012)) sont principalement *fermés* et *synchrones*. Par conséquent, ces solutions ne peuvent pas tenir compte de la nature agile de l'écosystème où des milliers d'utilisateurs peuvent surgir sur le portail de services. Les travaux récents dans le domaine de la composition du service utilisent la négociation « one-to-many » pour atteindre des accords *composite* avec plus d'un vendeur de services atomiques afin de construire un service composite (Richter *et al*., 2012 ; Mansour, Kowalczyk, 2014). Dans (Richter *et al*., 2012), les auteurs proposent un mécanisme de négociation « one-to-many » qui redistribue le surplus obtenu lors des sessions de négociation réussies aux sessions de négociation en cours afin d'augmenter leurs chances de succès. Néanmoins, ces travaux supposent l'ensemble de vendeurs atomiques *fermé* : ils sont tous connus avant le début de la négociation.

La négociation « one-to-many » a également été utilisée dans le domaine du *cloud computing*. Les auteurs de (An *et al*., 2010) développent une approche basée sur la négociation pour gérer l'allocation des ressources dans ce cadre. Cependant, un fournisseur accepte une offre si et seulement s'il peut obtenir un gain immédiat en acceptant l'offre. Pour cette raison, le taux d'acceptabilité des utilisateurs n'est pas pris en compte et aucun mécanisme d'adaptation n'est proposé. (Siebenhaar *et al*., 2012) développent un mécanisme de négociation simultanée de SLA dans les systèmes basés sur le *cloud*. Pourtant, la principale contribution du travail semble être axée sur le protocole. Par conséquent, il ne fournit pas un mécanisme pour représenter l'acceptabilité des utilisateurs ni offre une solution pour ajuster les comportements de négociation des délégués pour tenir compte des objectifs du fournisseur.

Dans nos travaux précédents (Najjar, Boissier, Picard, 2017b ; 2017a), nous avons présenté une version initiale du mécanisme adaptatif. Dans cette version initiale, les aspects de charge de travail et résistance aux pics de charge n'ont pas été étudiés. De plus, ces travaux ne contenaient qu'une évaluation expérimentale partielle.

7. Conclusions & travaux futurs

Cet article a présenté un mécanisme de négociation « one-to-many » adaptatif, conçu pour améliorer le taux d'acceptabilité d'un fournisseur SaaS tout en s'accommodant des pics de charge et en respectant ses contraintes budgétaires. L'approche proposée confère au fournisseur un contrôle précis du taux d'acceptabilité souhaité. En outre, comme illustré par nos résultats expérimentaux, avec la solution proposée, le fournisseur SaaS est capable de faire face à la nature dynamique et ouverte de l'écosystème du *cloud* et de répondre aux pics de charge.

Nos futurs travaux de recherche viseront à donner au fournisseur un contrôle plus précis sur le niveau de satisfaction des utilisateurs qu'il cherche à atteindre. En particulier, le fournisseur devrait être en mesure de s'assurer qu'un pourcentage prédéfini d'utilisateurs considère que le service est *Good* ou *Better* (Hoßfeld *et al.*, 2016). Plusieurs rapports de l'ITU (ITU, s. d.) et de l'ETSI (ETSI, s. d.) recommandent d'aller dans cette direction (Hoßfeld *et al.*, 2016). Cependant, la plupart des travaux existants adoptent une approche centrée sur le fournisseur basée sur le MOS (Hoßfeld *et al.*, 2016). Nos travaux de recherche futurs traiteront de ce problème. Cela nécessite des mises à niveau dans l'algorithme de modélisation de l'utilisateur et le mécanisme d'adaptation qui en résulte. Un autre travail en cours permet aux agents utilisateurs du service sa_i d'utiliser des stratégies de négociation autres que les concessions basées sur le temps (TBC).

Bibliographie

Accenture. (2013). *Accenture 2013 global consumer pulse survey global & u.s. key findings* report. Dublin, Ireland.

Al-Dhuraibi Y., Paraiso F., Djarallah N., Merle P. (2018). Elasticity in cloud computing: state of the art and research challenges. *IEEE Transactions on Services Computing*, vol. 11, n° 2, p. 430–447.

An B., Lesser V., Irwin D., Zink M. (2010). Automated negotiation with decommitment for dynamic resource allocation in cloud computing. In *Proceedings of the 9th international conference on autonomous agents and multiagent systems: volume 1-volume 1*, p. 981–988.

Baarslag T., Hendrikx M. J., Hindriks K. V., Jonker C. M. (2015). Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems*, p. 1–50.

Casalicchio E., Silvestri L. (2013). Mechanisms for sla provisioning in cloud-based service providers. *Computer Networks*, vol. 57, n° 3, p. 795–810.

ETSI. (s. d.). *European telecommunications standards institute.* http://www.etsi.org/.

Faratin P., Sierra C., Jennings N. R. (1998). Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, vol. 24, n° 3, p. 159–182.

Hobfeld T., Schatz R., Egger S. (2011). Sos: The mos is not enough! In *Quality of multimedia experience (qomex), 2011 third international workshop on*, p. 131–136.

Hoßfeld T., Heegaard P. E., Varela M., Möller S. (2016). Qoe beyond the mos: an in-depth look at qoe via better metrics and their relation to mos. *Quality and User Experience*, vol. 1, n° 1, p. 2.

Hou C. (2004). Predicting agents tactics in automated negotiation. In *Intelligent agent technology, 2004.(iat 2004). proceedings. ieee/wic/acm international conference on*, p. 127–133.

Index C. V. N. (2016). *Cisco vni forecast and methodology, 2015-2020. cisco white paper, june 1, 2016.*

ITU. (s. d.). *International telecommunications union.* https://www.itu.int/.

Ji S.-j., Zhang C.-j., Sim K.-M., Leung H.-f. (2014). A one-shot bargaining strategy for dealing with multifarious opponents. *Applied intelligence*, vol. 40, n° 4, p. 557–574.

Lomuscio A. R., Wooldridge M., Jennings N. R. (2003). A classification scheme for negotiation in electronic commerce. *Group Decision and Negotiation*, vol. 12, n° 1, p. 31–56.

Mansour K., Kowalczyk R. (2012). A meta-strategy for coordinating of one-to-many negotiation over multiple issues. In *Foundations of intelligent systems*, p. 343–353. Springer.

Mansour K., Kowalczyk R. (2014). On dynamic negotiation strategy for concurrent negotiation over distinct objects. In *Novel insights in agent-based complex automated negotiation*, p. 109–124. Springer.

Möller S., Raake A. (2014). *Quality of experience*. Springer.

Najjar A. (2015). *Multi-agent negotiation for qoe-aware cloud elasticity management*. Thèse de doctorat non publiée, École nationale supérieure des mines de Saint-Étienne.

Najjar A., Boissier O., Picard G. (2017a). An adaptive one-to-many negotiation to improve the service acceptability of an open saas provider. In *International workshop on agent-based complex automated negotiations (acan).*

Najjar A., Boissier O., Picard G. (2017b). Aquaman: An adaptive qoe-aware negotiation mechanism for saas elasticity management (extended abstract). In *International conference on autonomous agents and multi-agent systems (aamas).*

Najjar A., Gravier C., Serpaggi X., Boissier O. (2016, Oct). Modeling user expectations satisfaction for saas applications using multi-agent negotiation. In *2016 ieee/wic/acm international conference on web intelligence (wi)*, p. 399-406.

Najjar A., Mualla Y., Boissier O., Picard G. (2017, Aug). Aquaman: Qoe-driven cost-aware mechanism for saas acceptability rate adaptation. In *2017 ieee/wic/acm international conference on web intelligence (wi).*

Najjar A., Serpaggi X., Gravier C., Boissier O. (2014). Survey of elasticity management solutions in cloud computing. In *Continued rise of the cloud*, p. 235–263. Springer.

Najjar A., Serpaggi X., Gravier C., Boissier O. (2016). Multi-agent systems for personalized qoe-management. In *Teletraffic congress (itc 28), 2016 28th international*, vol. 3, p. 1–6.

North M. J., Howe T. R., Collier N. T., Vos J. (2005). The repast simphony runtime system. In *Agent 2005 conference on generative social processes, models, and mechanisms. argonne, illinois, usa: Argonne national laboratory.*

Pruitt D. G. (2013). *Negotiation behavior*. Academic Press.

Rahwan I., Kowalczyk R., Pham H. H. (2002). Intelligent agents for automated one-to-many e-commerce negotiation. In *Australian computer science communications*, vol. 24, p. 197–204.

Reichl P., Egger S., Schatz R., D'Alconzo A. (2010). The logarithmic nature of qoe and the role of the weber-fechner law in qoe assessment. In *Communications (icc), 2010 ieee international conference on*, p. 1–5.

Richter J., Baruwal Chhetri M., Kowalczyk R., Bao Vo Q. (2012). Establishing composite slas through concurrent qos negotiation with surplus redistribution. *Concurrency and Computation: Practice and Experience*, vol. 24, n° 9, p. 938–955.

Sackl A., Schatz R. (2013). Evaluating the impact of expectations on end-user quality perception. In *Proceedings of international workshop perceptual quality of systems (pqs)*, p. 122–128.

Savitzky A., Golay M. J. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, vol. 36, n° 8, p. 1627–1639.

Schatz R., Fiedler M., Skorin-Kapov L. (2014). Qoe-based network and application management. In *Quality of experience*, p. 411–426. Springer.

Siebenhaar M., Lampe U., Schuller D., Steinmetz R. *et al*. (2012). Concurrent negotiations in cloud-based systems. In *Economics of grids, clouds, systems, and services*, p. 17–31. Springer.

Son S., Sim K. M. (2012). A price-and-time-slot-negotiation mechanism for cloud service reservations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, n° 3, p. 713–728.

Talia D. (2012). Clouds meet agents: Toward intelligent cloud services. *IEEE Internet Computing*, n° 2, p. 78–81.

Thompson L. L., Wang J., Gunia B. C. (2010). Negotiation. *Annual review of psychology*, vol. 61, p. 491–515.

Thurstone L. L. (1927). Psychophysical analysis. *The American journal of psychology*, vol. 38, n° 3, p. 368–389.

Wooldridge M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Zeithaml V. A., Berry L. L., Parasuraman A. (1993). The nature and determinants of customer expectations of service. *Journal of the academy of Marketing Science*, vol. 21, n° 1, p. 1–12.