Vers une infrastructure participative d'hébergement de services à destination des communautés virtuelles : Orchestration dynamique de micro-services selon les conditions d'utilisation

Bruno Stévant

DOCTORAT BRETAGNE LOIRE / MATHSTIC

INSA RENNES

THÈSE DE DOCTORAT DE

L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Bruno STÉVANT

Vers une infrastructure participative d'hébergement de services à destination des communautés virtuelles.

Orchestration dynamique de micro-services selon les conditions d'utilisation.

Thèse présentée et soutenue à IRISA, Rennes, le 23 mai 2022
Unité de recherche : UMR IRISA
Thèse N° : 22ISAR 09 / D22 - 09

Rapporteurs avant soutenance :

Pierre SENS	Professeur, Sorbonne Université
Stefano SECCI	Professeur, Cnam

Composition du Jury :

Président :	Guillaume PIERRE	Professeur, Université Rennes 1
Examinateurs :	Françoise BAUDE	Professeur, Université Côte d'Azur
	Pierre SENS	Professeur, Sorbonne Université
	Stefano SECCI	Professeur, Cnam
Dir. de thèse :	Jean-Louis PAZAT	Professeur, INSA Rennes
Co-encadrant :	Alberto BLANC	Maître de conférence, IMT Atlantique

À mon grand-père René.

Remerciements

En premier lieu, je tiens à remercier Jean-Louis Pazat et Alberto Blanc d'avoir accepté de diriger et d'encadrer cette thèse. Le chemin suivi a été long et pas toujours facile depuis l'idée d'origine, esquissée sur un coin de tableau blanc jusqu'à la soutenance. Je leur suis reconnaissant de la patience et de la persévérance dont ils ont fait preuve pour m'accompagner pendant ces sept années. Leurs recommandations m'ont aidé à surpasser l'ingénieur qui est en moi par toujours plus de rigueur scientifique et d'exigence dans les résultats.

Je remercie également les membres du jury, en commençant par Stefano Secci et Pierre Sens qui m'ont fait l'honneur d'accepter d'être rapporteurs de ma thèse. J'ai particulièrement apprécié qu'ils aient compris les enjeux qui ont motivé ce travail. Je remercie aussi Françoise Baude pour sa relecture attentive du manuscrit. L'ensemble de leurs remarques ont contribué à l'amélioration de ce document, que j'espère maintenant à la hauteur de leurs attentes.

Je tiens ensuite à remercier mes collègues du département SRCD de IMT Atlantique qui ont soutenu ma démarche de poursuivre une thèse en parallèle de mes activités de chargé d'enseignement et de recherche à l'école. Je leur suis redevable des aménagements réalisés dans le fonctionnement de l'équipe pour me laisser le temps nécessaire à l'aboutissement de ce travail. Leurs conseils m'ont été précieux notamment dans la traversée en solitaire et sans escale qu'est la rédaction du manuscrit. Cette période a été d'autant plus difficile pendant le confinement lié à la pandémie de Covid-19. Les encouragements de toute l'équipe ont été autant de vent dans mes voiles qui m'ont aidé de franchir ce cap.

Enfin je tiens à remercier mes proches, mes amis et ma famille, pour m'avoir soutenu tout au long de cette aventure. Je sais les efforts qu'ils ont aussi consentis dans ce travail, notamment pour supporter mes moments d'absence pour surveiller le déroulement d'une expérimentation ou rédiger un paragraphe d'article. Promis, j'essaierai désormais d'être un ami, un mari et un père plus présent.

TABLE DES MATIÈRES

Chapitre 1

Vers un hébergement participatif des services d'une communauté virtuelle

Très bien. Donc l'univers n'est pas tout à fait comme vous le pensiez. Vous feriez mieux de réarranger vos croyances, alors. Parce que vous ne pouvez certainement pas réarranger l'univers.
– **Isaac Asimov**, *Nightfall*

L'étude réalisée dans cette thèse s'intéresse au déploiement d'applications sur une infrastructure participative. Nous allons définir en introduction la motivation de ce travail. Nous souhaitons en effet répondre au besoin d'une solution d'hébergement des services destinés à une communauté virtuelle. Car s'il existe déjà des solutions d'hébergement gratuites ou commerciales, celles-ci ne répondent pas complètement aux différentes exigences des usages d'une communauté en terme d'utilisabilité, de pérennité et de respect des données personnelles. Après avoir défini ces exigences, nous comparerons les réponses qu'apportent les solutions proposées actuellement. Nous proposerons ensuite une nouvelle approche où les membres d'une communauté peuvent participer eux même à l'hébergement de ces services en partageant les ressources des équipements qu'ils possèdent chez eux. Cette solution d'infrastructure participative soulève certains problèmes que nous décrirons à la fin de cette section et pour lesquels nous proposerons des solutions dans la suite de ce travail.

1.1 Besoin d'hébergement pour la communauté et solutions existantes

Afin de bien comprendre le problème qui nous intéresse ici, nous allons d'abord définir ce que nous appelons une communauté virtuelle et expliciter son besoin d'hébergement de services en

ligne. Nous listerons ensuite les solutions dhébergement qui se proposent à ces communautés et nous évaluerons la réponse qu'elles apportent à leur besoin.

1.1.1 Les communautés virtuelles et leurs services

Émergence des communautés locales et virtuelles

Le phénomène de communauté fait partie intégrante de l'humanité depuis probablement ses origines. Les individus ont en effet très tôt compris l'intérêt se réunir et d'interagir entre personnes partageant un même destin (une tribu), une même origine (une famille), un même métier (une corporation), un même lieu de vie (un voisinage), etc. Nous désignons ici par le terme communauté le concept utilisé en sociologie couvert par le terme anglais *community of interest* [42]. Ce terme permet de qualifier différents groupes d'individus partageant un intérêt commun, comme les parents d'élève d'une même école, les employés d'une même entreprise, les membres d'un club de sport, etc. L'appartenance à une telle communauté apporte à ses membres des moyens de mieux comprendre et interpréter leur condition en partageant leurs différents points de vue et à chercher des solutions aux problèmes communs qu'ils rencontrent.

Avec l'apparition de l'Internet, les interactions entre membres d'une même communauté ont profité pour leurs échanges d'un nouveau support de communication pervasif, robuste et versatile. En plus des communautés existantes, que l'on peut qualifier « de lieu » , de nouvelles se créent, virtuelles cette fois-ci, exploitant les possibilités offertes par l'Internet pour regrouper des membres dispersés sur toute la planète. Ce phénomène a été prédit par Joseph Carl Robnett Licklider et Robert Taylor dans un article daté de 1968 et intitulé *The Computer as a Communication Device* [58]. Dans les années 1990, alors que l'accès à l'Internet se démocratise, Howard Rheingold, dans son livre *Communautés virtuelles* [72], étudie l'émergence de ce phénomène et décrit comment des internautes peuvent se retrouver autour d'un même centre d'intérêt et échanger leurs idées dans des forums de discussions. Dans un second livre, *Foules intelligentes (Smart Mobs)* [71], il étend son étude à l'enrichissement des interactions sociales à travers les nouveaux outils de communication et de partage de contenus en mobilité qu'offre l'accès sans fil à l'Internet.

Le phénomène des communautés virtuelles a suscité depuis un intérêt de la part des sociologues d'autant plus important que les réseaux sociaux, évolution de ces communautés vers des médias de groupes (*many-to-many media*), ont accaparé une grande partie de l'usage de l'Internet [1]. Parmi les définitions possibles de la communauté virtuelle, [70] souligne :

1. 32% selon des études publiées en 2019, mais il difficile de corroborer ce chiffre.

> [Les réseaux de communications] affranchissent l'activite humaine des contraintes de la matiere, de l'espace et du temps, ouvrant sur des possibilites inedites. En ce sens, le virtuel est davantage « plein » que l'actuel, il est « hyperreel », et les technologies du virtuel sont percues comme liberatrices, dans la mesure ou elles ouvriraient une porte sur toute la richesse du reel.

Les communautés « réelles » ont ainsi emprunté le pas des communautés « virtuelles » pour s'en approprier leurs outils et profiter comme elles des facilités offertes par les réseaux de communication, capables de faciliter voire même d'amplifier les interactions entre membres.

Besoins d'une communauté virtuelle en terme de services en ligne

De la même façon qu'une communauté physique a besoin d'un espace où les membres puissent interagir (local dédié, événements, etc.), les communautés virtuelles se réunissent autour d'un ou plusieurs outils qui servent de support à ces échanges. Par exemple, la communauté virtuelle des membres d'un club sportif se dote d'un premier outil pour partager des photos, d'un second pour éditer un calendrier partagé, etc. Ces outils coopératifs permettent de créer des contenus communs à partir des contributions individuelles de chaque membre. Chaque communauté a des besoins différents. Il existe aujourd'hui des outils génériques (wikis, serveurs de calendrier, galeries photos) capables de s'adapter aux besoins spécifiques d'une communauté par simple configuration.

La communauté est en droit d'exprimer un certain nombre d'exigences auxquelles doit répondre un outil afin qu'elle puisse l'adopter : des exigences sur les fonctionnalités offertes qui doivent s'aligner avec les besoins de la communauté, mais aussi des exigences non-fonctionnelles qui précisent les attentes sur l'utilisabilité de l'outil. Une communauté virtuelle étant par essence dispersée et ses membres connectés à l'Internet par divers moyens, il sera attendu de l'outil qu'il soit globalement accessible quelque soit le contexte d'utilisation. Cette accessibilité devra être de plus assurée dans le temps, permettant une disponibilité suffisante de l'outil. Afin d'être accepté par les utilisateurs et pour faciliter les échanges, l'outil aura des performances non-contraignantes, en terme de temps de réponse notamment. L'aspect financier fait aussi parti de ces exigences non-fonctionnelles, les frais liés à la mise à disposition de l'outil, s'ils existent, devront constituer des dépenses acceptables pour la communauté. Enfin, la communauté virtuelle pourra porter une attention particulière aux garanties apportées par l'outil sur la sécurisation (intégrité, non-répudiation) et la confidentialité des échanges, propriétés nécessaires et complémentaires à la confiance que se portent entre eux les membres de la communauté.

1.1.2 Solutions d'hébergement existantes

Ces exigences concernent aussi bien le logiciel utilisé comme base de l'outil que la solution d'hébergement choisie pour cette base logicielle pour constituer un outil utilisable pour la communauté virtuelle. Nous allons étudier 3 solutions d'hébergement : les plateformes de services prêts à l'emploi, l'hébergement sur un serveur dédié loué chez un fournisseur d'infrastructure et l'hébergement domestique sur un équipement localisé chez un membre de la communauté. Ces solutions sont toutes capables d'assurer la mise à disposition des outils à l'ensemble de la communauté. Elles diffèrent cependant selon les autres exigences non-fonctionnelles que peut exprimer la communauté : performance de l'accès aux outils, disponibilité des outils, passage à l'échelle pour suivre l'évolution de la communauté, facilité de maintenance, contrôle sur l'évolution des outils, confidentialité des données et coût financier de la solution.

Les plateformes de services

Depuis 2005 et notamment l'apparition de nouvelles fonctionnalités des navigateurs Web 2.0, plusieurs entreprises proposent des plateformes de services en ligne facilitant la création des communautés virtuelles, avec des succès différents. On peut citer parmi elles : MySpace, Orkut, Flickr, Youtube, Facebook. Ces plateformes offrent à la demande les outils nécessaires aux communautés sous forme de services à la demande (Software-as-a-Service (SaaS)) dont l'accès est généralement gratuit pour leurs utilisateurs, qu'ils soient contributeurs ou consommateurs de leurs contenus. Ces outils sont déployés dans des centres de données aux infrastructures redondées et distribuées géographiquement dans différentes régions du monde afin d'offrir des solutions d'hébergement à la fois multiples et proches de ses utilisateurs. Les logiciels de ces outils sont maintenus et en constante évolution pour s'adapter aux besoins des utilisateurs. Ces plateformes répondent donc en grande partie aux attentes des communautés en terme d'accessibilité, de disponibilité, de performances, de maintenance et de coût. Parce qu'elles sont publiques, ces plateformes regroupent un nombre important d'utilisateurs de différentes communautés. Elles peuvent donc donner l'opportunité à ces communautés de s'ouvrir à de nouveaux membres déjà présents sur ces plateformes.

Cependant la contre-partie de l'utilisation gratuite de ces services est une utilisation souvent abusive des données personnelles par les plateformes. En effet, l'enjeu pour les fournisseurs de ces services est de trouver un moyen de rentabiliser l'infrastructure et les outils offerts aux communautés virtuelles. Comme beaucoup de médias gratuits, elles ont fait le choix de se financer par la publicité, la promotion de contenus sur leur plateforme et la revente d'informations personnelles à des entreprises spécialisées dans le profilage de clientèle [22]. Ce choix implique que

ces plateformes intègrent à leurs services des procédés de suivi d'activité et de collecte de données personnelles, leur permettant de recueillir des informations sur le comportement de leurs utilisateurs et leurs centres d'intérêt. Les utilisateurs sont informés de ces pratiques lorsqu'ils acceptent les conditions d'utilisation de ces outils. Mais peu de ces utilisateurs sont sensibilisés à l'impact potentiel de ces pratiques sur leur vie privée. En juin 2013, plusieurs journaux publient une série de révélations autour de programmes de surveillance des communications sur Internet initiés par les agences fédérales américaines. Ces informations, exfiltrées par Edward Snowden, ancien employé de la NSA, montrent notamment comment à travers le programme PRISM, la NSA et le FBI ont acquis légalement des accès privilégiés sur des plateformes d'échanges [86]. Même si la surveillance par ces agences des réseaux téléphoniques et informatiques était publiquement connue depuis les années 1980 dans le programme ECHELON, ces publications ont eu une résonance d'autant plus forte qu'elles montrent une collusion entre les agences fédérales et les firmes privées responsables des plateformes de services.

Hébergement sur un serveur dédié

Pour s'affranchir des risques de détournement de ses données, la communauté doit disposer elle-même des outils dont elle a besoin et de devenir autonome dans l'hébergement de ses outils. Il lui est nécessaire pour cela d'avoir accès à des applications offrant les fonctionnalités souhaitées et à une infrastructure pour héberger ces outils qui satisfasse les exigences de fonctionnement. Une base logicielle importante existe aujourd'hui couvrant les besoins des communautés virtuelles. Grâce à ces logiciels, disponibles sous licence libre et open-source, il est possible de disposer d'applications, comme un blog, une galerie de photo, un forum de discussion, ouvertes à la communauté virtuelle. Ces logiciels sont maintenus par une communauté importante de développeurs qui en corrigent les défauts et en améliorent les fonctionnalités. Des hébergeurs proposent des serveurs pré-configurés avec ces logiciels. Grâce à ces offres une communauté virtuelle peut s'approprier ces applications sans besoin de compétences en administration d'un serveur. Elle contrôle ainsi le logiciel utilisé, les fonctionnalités activées et donc les usages des données de la communauté.

La communauté virtuelle a ensuite besoin d'une infrastructure lui permettant de rendre disponibles ces applications sous forme de services à destination de ses membres. Cette infrastructure doit répondre aux exigences d'accessibilité, de disponibilité, de performances et de sécurité. Une possibilité est de louer un serveur dédié dans un centre de données. Un tel serveur constitue une solution d'hébergement fiable et performante, avec un niveau de service équivalent à celui d'une plateforme. Les mécanismes de redondances électriques et informatiques, ainsi qu'une

sauvegarde régulière permet d'assurer une disponibilité d'au moins $99,95\%$ [2] (soit moins de 20 minutes d'indisponibilité par mois). Les performances sont, quant à elles, dépendantes des ressources de calcul du profil de serveur choisi pour l'hébergement.

Cependant l'utilisation d'un tel serveur n'est dans la plupart des cas pas gratuite et nécessite une contribution financière sous forme d'un loyer, certes peu onéreux (une dizaine d'euros par mois pour les offres de serveur dédié d'entrée de gamme), mais qui demande à la communauté virtuelle de s'organiser administrativement et financièrement. La location d'un tel serveur se fait au titre d'une personne morale ou physique, avec un engagement contractuel avec le fournisseur. Il est nécessaire pour la communauté virtuelle de se constituer sous la forme d'une association ou alors de désigner un membre chargé de louer le serveur en son nom. La communauté virtuelle doit de plus assurer la régularité des paiements, ce qui implique que les membres contribuent régulièrement au loyer afin d'assurer la continuité des services.

Il doit être mentionné enfin que la solution d'hébergement sur serveur dédié n'apporte pas non plus une garantie complète sur la confidentialité des données : le serveur et le réseau qui le connecte à l'Internet sont contrôlés par une entreprise qui peut s'arroger le droit d'intercepter des échanges au sein de son infrastructure. Cette même entreprise a la charge d'assurer l'intégrité physique de son infrastructure, ainsi que celle des données qu'elle héberge. Cependant, elle n'est pas toujours infaillible dans cette tâche. Ainsi en 2012 par exemple, le groupement Visa a été victime d'un vol de données hébergées chez un hébergeur à cause d'une défaillance de la sécurité de son système de stockage [83].

Auto-hébergement domestique

Une autre solution d'hébergement consiste à déployer ces applications non pas sur un serveur loué chez un prestataire mais sur un équipement localisé chez un des membres de la communauté. L'auto-hébergement domestique qui se développe en réaction à la centralisation de l'Internet autour des plateformes, théorisée par B.Bayart sous le terme *Minitel 2.0* [39], et aux problèmes de confidentialité des données qui en découlent. Des associations comme Framasoft et le Collectif des Hébergeurs Alternatifs, Transparents, Ouverts, Neutres et Solidaires (CHATONS) promeuvent cette pratique en publiant notamment des guides [3]. L'auto-hébergement domestique nécessite concrètement un équipement de type PC fixe ou un appareil intégré comme un Raspberry Pi sur lequel l'utilisateur peut installer les applications qu'il souhaite activer sous forme de service. Cet équipement devra être allumé en permanence et connecté au réseau domestique pour que

2. `https://www.ovh.com/fr/support/documents_legaux/contractPartOVHCloudFr.pdf`
3. `https://framacloud.org/fr/auto-hebergement/`

ce service puisse être disponible pour la communauté. Le coût d'acquisition d'une telle solution est donc relativement modeste (un Raspberry Pi ne coûte que quelques dizaines d'euros) voire nul si l'utilisateur possède déjà un tel équipement. L'abonnement de la connexion à l'Internet étant déjà intégré dans les dépenses du foyer, il ne reste à charge que la consommation électrique supplémentaire.

Comme pour la solution d'hébergement sur un serveur dédié, la communauté utilisera une base logicielle lui assurant un contrôle des outils et des données. Des distributions logicielles existent aujourd'hui pour déployer facilement des services collaboratifs sur un équipement comme un Raspberry Pi ou une machine virtuelle. Des solutions libres comme Yunohost [4], CosyCloud [5] ou FreedomBox [6] incluent le système d'exploitation ainsi qu'une bibliothèque d'applications activables en un clic. De la même manière qu'avec une solution d'hébergement sur un serveur dédié, ces solutions assurent à la communauté la maîtrise des outils et des fonctionnalités activées. Les données de la communauté seront hébergées chez l'un de ces membres. Le respect de leur confidentialité reposera alors sur la probité de ce membre et sur la confiance que lui accorde la communauté.

Cependant cette solution d'auto-hébergement domestique présente des limitations en termes de performances et de passage à l'échelle selon les moyens investis [2]. Un appareil intégré comme un Raspberry Pi a des performances comparables à celles d'un ordinateur fixe standard pour un service délivrant du contenu statique. Lorsqu'il s'agit délivrer du contenu dynamique, nécessitant un traitement pour chaque requête, le temps de réponse d'un service hébergé sur ce type d'équipement peut être 2 à 3 fois plus important [48]. Cette dégradation se fera d'autant plus ressentir que les utilisateurs seront nombreux. Le réseau d'accès résidentiel qui connecte l'équipement à l'Internet et permet l'accès aux outils par les membres de la communauté présente lui aussi des limites. Ce réseau a une bande passante d'au maximum une centaine de MBit/s et parfois asymétrique (débit réduit dans le sens montant). Ces capacités semblent raisonnables pour le trafic généré par une application à destination d'une communauté restreinte d'utilisateurs. Cependant cette ressource est partagée avec les usages de l'Internet des membres du foyer hébergeant l'équipement, ce qui induit un risque de saturation du réseau d'accès résidentiel.

Une autre limitation de l'auto-hébergement domestique se situe sur la disponibilité des services hébergés. L'utilisation d'un seul équipement connecté à un seul réseau d'accès présentent plusieurs vulnérabilités qui peuvent entrainer l'indisponibilité du service. L'équipement peut devenir inopérant suite à une panne matérielle, d'autant plus probable que les équipements domestiques sont conçus pour des usages ponctuels. Une autre cause d'indisponibilité de l'équipement est la

4. `https://yunohost.org/`
5. `https://cozy.io/`
6. `https://freedombox.org/`

coupure de l'alimentation électrique, hautement probable dans une habitation qui n'a pas de réseau électrique redondé. S'ajoutent à ces causes d'indisponibilité celles qui concernent le réseau informatique domestique. Celui-ci s'appuie sur une passerelle d'accès résidentiel tout aussi fragile que l'est l'équipement en charge de l'hébergement du service. L'accès résidentiel à l'Internet est lui tributaire du bon fonctionnement de la ligne locale et des équipements de l'opérateur.

Enfin, une problématique liée à l'hébergement domestique concerne la pérennisation des services offerts. Si un membre prend en charge l'hébergement du service, que se passe-t-il si celui-ci décide de quitter la communauté, ou simplement de ne plus héberger les services ? Il est alors nécessaire de trouver un nouveau membre volontaire pour assurer l'hébergement. Les opérations nécessaires pour le transfert des données entre l'ancien et le nouvel équipement sont potentiellement hasardeuses, avec un risque de perte de données.

Synthèse des solutions d'hébergement existantes

Nous venons d'analyser les différentes solutions d'hébergement actuellement à disposition d'une communauté virtuelle. Le tableau 1.1 présente une évaluation comparative de ces solutions selon différents critères non-fonctionnels liés à l'utilisabilité (Disponibilité, Performance), la pérennité (Facilité de maintenance, Passage à l'échelle, Coût) et le respect des données de la communauté (Contrôle sur les outils, Confidentialité des données), selon les arguments que nous avons développés précédemment.

Critère	Plateforme de services	Hébergement sur serveur dédié	Auto-hébergement domestique
Disponibilité	Positif	Positif	Négatif
Performance	Positif	Positif	Négatif
Facilité de maintenance	Positif	Neutre	Neutre
Passage à l'échelle	Positif	Neutre	Négatif
Coût	Positif	Négatif	Neutre
Contrôle sur les outils	Négatif	Positif	Positif
Confidentialité des données	Négatif	Négatif	Positif

TABLE 1.1 – Apport des solutions d'hébergement aux besoins d'une communauté virtuelle.

Cette évaluation met en évidence qu'une solution de type plateforme de services répond parfaitement aux critères d'utilisabilité et de pérennité mais ne satisfait pas ceux concernant le respect des données de la communauté. L'auto-hébergement sur un serveur dédié offre une utilisabilité comparable à celle d'une plateforme de service mais offre une pérennité de la solution plus mitigée, avec comme inconvénient majeur le coût de la solution. Le respect des données est

par contre en amélioration. La solution d'auto-hébergement domestique est la meilleure dans ce domaine. Mais elle souffre de trop grandes limitations pour satisfaire les critères d'utilisabilité et de passage à l'échelle.

Il apparait donc, à la lumière de cette comparaison, qu'aucune solution n'arrive à satisfaire globalement les critères d'hébergement exprimés par la communauté virtuelle. Il faut donc chercher à améliorer l'une de ces solutions pour pleinement répondre à ces besoins. Les plateformes de services étant entièrement contrôlées par leur fournisseur, il est difficile de s'en servir comme base d'une nouvelle solution. L'utilisation d'un serveur dédié nécessitant toujours une contrepartie financière, ce point restera toujours un inconvénient pour ce type de solution. Nous avons donc choisi d'étudier une solution d'auto-hébergement domestique avec l'idée de s'appuyer sur la communauté virtuelle pour lever les limitations que nous avons constatées.

1.2 Définition d'une solution d'hébergement participative

Nous avons mis en évidence dans la section précédente les inconvénients d'une solution d'auto-hébergement domestique où un seul équipement assure la mise à disposition d'applications pour une communauté virtuelle. Nous allons décrire dans cette section une nouvelle approche participative de l'hébergement impliquant la coopération de plusieurs membres de la communauté. Dans cette solution, ces membres contribuent à une infrastructure d'hébergement en y apportant des équipements connectés chez eux. Ces équipements vont se coordonner pour héberger les services de la communauté. Nous verrons que cette solution est envisageable car il existe déjà dans les réseaux domestiques des ressources de calcul, de stockage et de communication sous-exploitées qui pourraient ainsi être mises à contribution pour la communauté. Nous identifierons ensuite les problématiques ouvertes par cette nouvelle approche.

1.2.1 Principes de fonctionnement

Le concept d'infrastructure participative repose sur l'idée de construire une solution d'hébergement à partir de ressources fournies par les membres de la communauté. À l'image de l'engagement bénévole (temps, responsabilités administratives) parfois demandé par la communauté à ces membres, la contribution de certains membres sera de partager des ressources de calcul ou de stockage présentes sur des équipements qu'ils possèdent chez eux. Cette mise en partage se concrétisera par l'installation d'un outillage logiciel spécifique sur chacun des équipements. Ce logiciel assurera la coordination de l'ensemble des équipements partagés dans la commu-

nauté pour en mutualiser les ressources et ainsi créer un environnement d'exécution pour les applications.

L'infrastructure d'hébergement participative constitue en soi une extension de l'auto-hébergement domestique à l'échelle d'une communauté. Elle en conserve donc les propriétés de respect des données personnelles : les informations de la communauté restent hébergées au sein de ses membres, qui se font confiance mutuellement pour en assurer la non-divulgation. La communauté garde aussi le contrôle sur les outils utilisés et les services qu'elle offre. La solution envisagée devra permettre à la communauté de gérer facilement l'activation et la mise à jour des services, à l'image des solutions d'auto-hébergement déjà mentionnées.

L'infrastructure participative prévoit que le contexte d'exécution des applications soit fourni non pas par un équipement comme pour l'auto-hébergement domestique, mais par plusieurs équipements situés dans différents foyers. Cette approche améliore potentiellement la disponibilité de la solution d'hébergement : si un équipement (ou son réseau d'accès) est défaillant, il sera possible d'utiliser un autre équipement situé dans un foyer différent pour y héberger l'application. Les capacités de calcul et de stockage disponibles dans une infrastructure participative vont s'accroitre avec le nombre d'équipements participants. La solution d'hébergement pourra ainsi passer à l'échelle avec l'augmentation du nombre de membres dans la communauté. Enfin, l'infrastructure participative utilisant des ressources déjà existantes, l'ensemble des coûts de la solution sont déjà intégrés dans le budget des foyers contributeurs.

1.2.2 Ressources disponibles dans les réseaux domestiques

Si l'infrastructure participative répond à la plupart des besoins de la communauté virtuelle, cette solution ne peut pas encore faire de promesse en termes de performances des applications hébergées. Les alternatives concurrentes se basent sur des infrastructures de centre de données, avec des processeurs puissants et des réseaux très haut débit pour fournir une qualité d'expérience toujours à la hauteur. Notre solution pourra essayer d'apporter une réponse satisfaisante à ce besoin seulement si les ressources de calcul et de communication disponibles sont suffisantes.

Équipements domestiques

Alors que les progrès technologiques en termes de puissance de calcul et de stockage ont permis l'émergence des centres de données, les équipements domestiques ont eux aussi profité de ces évolutions pour offrir des capacités suffisantes pour de nombreux usages. Les foyers sont maintenant bien équipés en appareils multimédia connectés. Selon une étude du CNC datée de 2018 [23], les

	Freebox Delta	Nvidia Shield	Microsoft Xbox OneX	Synology DS720+
Fonction	Passerelle Internet	Décodeur TV	Console de jeux	Serveur de stockage
Processeur	ARMv8 2.2GHz	Tegra X1 2GHz	AMD Jaguar 2.3GHz	Intel Celeron 2GHz
Nb. Coeurs	8	8	8	4
Mémoire vive	2Go	2Go	9Go	2Go
Stockage	1To	8Go	1To	>1To
Accélération graphique	Non	Oui	Oui	Non

TABLE 1.2 – Caractéristiques techniques d'équipements résidentiels en 2021.

foyers français était équipés pour 76% d'au moins un ordinateur (41% d'un ordinateur fixe) et 53% d'une console de jeux (49% d'une console de salon). Une étude du CSA datée de 2019 [25] nous indique que 58% des foyers français utilisent un décodeur TV. Nous pouvons ajouter à ces équipements les passerelles Internet domestiques et d'autres équipements comme les serveurs de stockages personnels (NAS).

Le tableau 1.2 montre un exemple des capacités de calcul et de stockage présentes dans différents types d'équipements. La multiplication des usages possibles de ces équipements et l'évolution de l'électronique grand public expliquent une montée en puissance des capacités de calculs et de stockage. Or ces ressources ne sont pas toujours totalement utilisées. La passerelle Internet Freebox Delta propose ainsi d'y installer des machines virtuelles pour profiter des ressources de calculs inutilisées. Le temps d'utilisation moyen d'une console de jeux de salon est d'environ 3 heures par jour [23].

Des ressources de calcul et de stockage sont donc présentes dans les réseaux domestiques et peuvent être sous-utilisées, ce qui ouvre la voie vers de nouveaux usages comme nous le proposons à travers l'infrastructure d'hébergement participative. La présence d'équipements avec des capacités de calcul non-négligeables nous permet d'espérer des performances correctes pour garantir une bonne qualité d'expérience. Mais notre solution devra prendre en compte une certaine hétérogénéité de ces ressources entre les différents équipements ainsi qu'une disponibilité variable dans le temps.

Connectivité des réseaux domestiques

L'utilisation des services hébergés sur l'infrastructure participative va impliquer des échanges entre équipement hébergés dans des réseaux domestiques différents. Ces communications se feront à travers l'Internet auquel chaque foyer est raccordé par un réseau d'accès résidentiel. Chaque équipement sera amené à héberger tout ou partie d'une application utilisée par les membres de la communauté. Les échanges vont donc induire du trafic à la fois descendant (de l'Internet vers le réseau domestique) mais aussi montant (du réseau domestique vers l'Internet). Les réseaux d'accès résidentiel utilisent historiquement le protocole IPv4 qui complique, par son mode de déploiement, ces connexions entrantes. Le protocole IPv6 facilite ce nouvel usage car il permet d'adresser directement l'équipement depuis l'Internet. Aujourd'hui, environ la moitié des accès résidentiels en France utilisent le protocole IPv6 [8].

D'après l'INSEE [47], 77% des foyers français possèdent un accès à l'Internet. Ces accès se font à travers différentes technologies. L'accès à l'Internet haut débit, caractérisé par des débits allant jusqu'à 30 Mbit/s, se fait par la technologie Asymmetric Digital Subscriber Line (ADSL) utilisant le réseau téléphonique. L'accès très haut débit, utilise lui la fibre optique (Fiber To The Home (FTTH)), la terminaison coaxial ou la technologie Very-high-bit-rate Digital Subscriber Line (VDSL) pour des débits supérieurs à 30 Mbit/s et pouvant dépasser les 100 Mbit/s. D'après l'ARCEP, 51% des foyers français connectés le sont en très haut débit, pour 49% en haut débit [9]. La communauté virtuelle pourra donc s'appuyer, pour son infrastructure participative, sur un nombre de membres conséquent connectés par des réseaux très haut débit : la moitié si l'échantillon est représentatif.

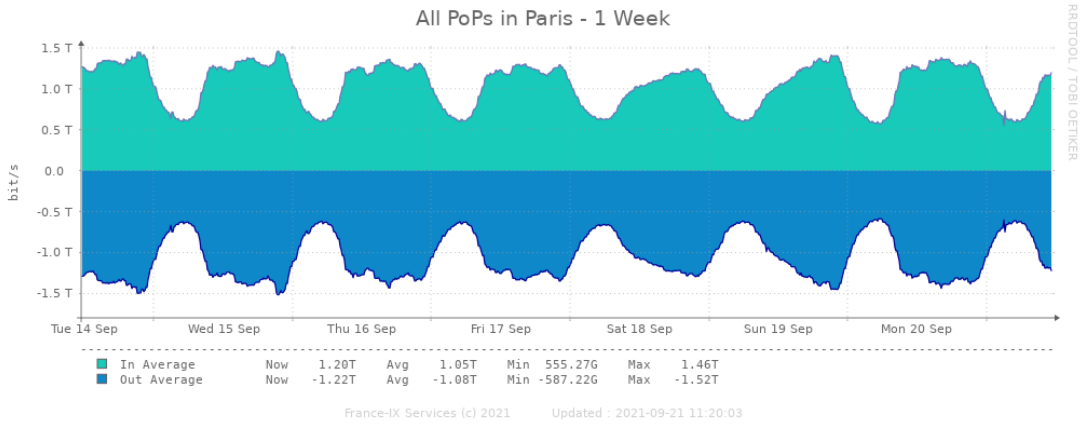

FIGURE 1.1 – Trafic observé au niveau du point d'échange Internet FranceIX pendant une semaine.

Les membres du foyer utilisent leur connexion à l'Internet pour leurs usages personnels, qui génèrent un trafic plus ou moins important sur le réseau d'accès résidentiel. Il est difficile d'estimer cette charge pour en déduire les ressources réseau disponibles dont pourrait profiter la communauté. Il existe dans la littérature des études qui analysent les valeurs des caractéristiques de qualité de service (Quality of Service (QoS)) observables sur des réseaux d'accès résidentiels : bandes passantes montante et descendante, latence, perte de paquet. Nous détaillerons ces analyses dans le chapitre 4. Nous soulignerons ici que ce trafic utilisateur est variable dans le temps, en fonction de l'heure de la journée mais aussi en fonction du jour de la semaine. La Figure 1.1 montre l'évolution du trafic Internet entre différents opérateurs français au niveau du point d'échange FranceIX. Les utilisateurs de ces réseaux étant dans le même fuseau horaire, ce trafic suit un motif régulier, appelé motif *diurne*, avec un plateau aux heures dites ouvrées et un creux pendant la nuit. L'apparence des plateaux est aussi sensiblement différente en fin de semaine que pendant les jours de semaine. Nous pouvons supposer que cette tendance, représentative des usages de l'Internet à grande échelle, est similaire celle pouvant être observée en moyenne sur les réseaux d'accès résidentiels.

Les capacités de communication entre équipements d'une infrastructure participative sont donc diversifiées selon les réseaux d'accès résidentiel raccordant ces équipements. Même si ces réseaux ont une capacité importante aujourd'hui, il est nécessaire de prendre en compte la charge du trafic utilisateur qui, elle, varie dans le temps.

1.2.3 Défis et problématiques ouvertes

La solution proposée d'une infrastructure participative semble pertinente pour répondre aux besoins d'hébergement d'une communauté virtuelle, et envisageable dans le contexte des réseaux domestiques actuels. Mais pour que cette solution soit réalisable, il faudra relever un certain nombre de défis techniques.

Déploiement des applications en fonction des ressources disponibles

Le premier défi concerne la création d'un contexte d'exécution pour chaque application à héberger sur l'infrastructure participative : comment coordonner les équipements et utiliser leur ressources disponibles pour participer à ce contexte ? L'application devra, pour s'exécuter, être installée sur un équipement de l'infrastructure et disposer de l'ensemble des données nécessaires. Notre solution devra fournir un dispositif permettant de déployer automatiquement l'application sur les équipements et d'y transférer ses données.

Ce système devra, lors du déploiement de l'application, prendre en compte les profils multiples des équipements composant cette infrastructure participative. L'équipement devra ainsi avoir une capacité de stockage suffisante pour recevoir les données. Ses capacités de calcul (processeur, mémoire vive) devront aussi être suffisantes pour les besoins de l'application, au risque d'en pénaliser ses performances. Il en est de même avec le réseau d'accès résidentiel concerné, support des échanges entre l'application et ses utilisateurs : ses caractéristiques de QoS réseau doivent être suffisantes pour les besoins de communication de l'application.

Maintien de la qualité d'expérience

Une fois qu'il est possible d'exécuter l'application au sein de l'infrastructure participative avec des performances raisonnables, le deuxième défi est de maintenir une qualité d'expérience acceptable des applications hébergées malgré la disponibilité variable des ressources. Comme nous l'avons décrit précédemment, les ressources des réseaux domestiques sont utilisées de manière concurrente par les usages des membres du foyer. Notre solution ne doit pas pénaliser ces usages prioritaires, au risque d'être rejetée par les membres du foyer. Mais elle doit en même temps offrir à tout instant une bonne qualité d'expérience des applications hébergées pour être acceptée par les membres de la communauté.

De la même manière que nous prévoyons que notre solution puisse réagir à la défaillance d'un équipement, elle devra aussi s'adapter à la disponibilité des ressources au niveau de chaque équipement. Cette adaptation aura des conséquences sur le choix des équipements où sont déployés les applications de la communauté. Il est donc possible que de nouveaux déploiements des applications interviennent au cours du temps. Ceux-ci ne devront pas avoir de conséquence sur l'utilisabilité de la solution.

Sécurité de l'infrastructure participative

Même si notre solution se destine à une communauté d'utilisateurs se faisant confiance mutuellement, il est nécessaire d'en envisager les différents usages malveillants et leurs conséquences. Il est possible d'imaginer, par exemple, un scénario où la communauté est infiltrée par un attaquant cherchant à compromettre l'infrastructure participative, soit en y ajoutant son propre équipement, soit en prenant le contrôle de l'équipement d'un autre membre. Cet équipement va potentiellement être choisi pour exécuter une ou plusieurs applications de la communauté. L'attaquant peut alors profiter de sa situation pour exfiltrer des données de la communauté, ou perturber le bon fonctionnement de l'application. Il est aussi possible d'envisager que l'at-

taquant exploite les ressources de l'infrastructure participative pour d'autres tâches lucratives comme le minage de cryptomonnaie, l'envoi de courriers indésirables ou la participation à des réseaux d'attaques coordonnées (*botnet*).

Notre solution devra donc se prémunir contre ces usages malveillants, tout en restant ouverte à de nouveaux membres et à de nouvelles applications. Un contrôle de l'utilisation des équipements participants devra être possible pour rendre auditable le partage de ressources. Les applications devront être isolées les unes des autres pour éviter tout effet de bord et les données chiffrées, que ce soit au moment où elles sont stockées ou lorsqu'elles sont échangées entre équipements.

Capacité à passer à l'échelle

Notre solution doit pouvoir fonctionner pour des communautés de taille et de répartition différentes. Au début de ce chapitre, nous avons évoqué la communauté des membres d'un club de sport. Ce type de communauté locale rassemble quelques dizaines de membres dans une même zone géographique. L'infrastructure participative que pourra espérer construire cette communauté sera composée elle aussi de quelques dizaines d'équipements relativement proches les uns des autres pour avoir une interconnexion fiable et de bonne qualité. Mais il est possible d'envisager des communautés plus nombreuses avec une répartition géographique plus large. Si l'infrastructure d'hébergement profitera d'un nombre plus important d'équipements, l'interconnexion entre ces équipements sera alors plus hasardeuse et des ruptures de cette interconnexion (*churns*) peuvent survenir. Le passage à l'échelle de notre solution devra prendre en compte ces situations.

Si notre solution devait se généraliser, il nous faut aussi considérer le cas où un même utilisateur contribue à l'hébergement d'applications de plusieurs communautés. Dans ce cas, un même équipement pourrait participer à plusieurs infrastructures. Les applications de ces différentes communautés vont se partager les ressources de calcul et de communication disponibles. L'utilisateur souhaitera peut-être dimensionner les ressources partagées, et choisir leur répartition entre différentes communautés auxquelles il participe. Notre solution pourra aussi s'assurer qu'un même équipement contribue de manière équitable aux différentes communautés.

1.3 Contributions de cette thèse

Cette thèse se propose de définir les contours d'une solution réalisable pour une infrastructure participative d'hébergement d'applications. Nous commencerons dans le chapitre 2 par identi-

fier les solutions existantes dont nous pouvons nous inspirer pour définir notre infrastructure participative d'hébergement. Ce tour d'horizon nous amènera à nous intéresser aux applications composées de micro-services qui présentent de bonnes propriétés pour être déployées dans une telle infrastructure.

Nous étudierons dans le chapitre 3 le placement de ces micro-services sur une infrastructure composée d'équipements avec des caractéristiques différentes en termes de processeur et de réseaux d'accès. L'objectif de l'étude est de trouver le placement aboutissant au meilleur temps de réponse de l'application. Nous définirons un modèle du temps de réponse à partir de celui de chaque micro-service, et nous utiliserons une heuristique pour trouver une solution de placement proche de l'optimal.

Dans le chapitre 4, nous évaluerons les placements choisis par cette heuristique dans des conditions réalistes d'utilisation. Nous définirons d'abord ces conditions en termes de d'intervalles de variations possibles des paramètres QoS réseau. Nous montrerons ensuite comment le temps de réponse de l'application sur des équipements évolue en fonction de ces paramètres pour confirmer que la qualité d'expérience reste acceptable. Cette étude fera apparaitre que dans certains cas, il est nécessaire de reconsidérer ces placements et de les adapter aux nouvelles conditions.

Cette adaptation à l'évolution des conditions sera étudiée ensuite dans le chapitre 5. Nous présenterons l'architecture d'un processus d'adaptation qui se basera sur des mesures des temps de réponses de chaque micro-service individuellement. Sur la base de ces mesures, ce processus prendra les décisions d'adaptation du placement selon une stratégie que nous aurons définie selon notre contexte et nos retours d'expérience. Nous montrerons les résultats de l'évaluation de ce processus d'adaptation sur des équipements en conditions réelles d'utilisation.

La solution proposée dans cette thèse nous apportera ainsi des éléments de réponse pour la distribution des applications sur une infrastructure participative et le maintien de la qualité d'expérience, enjeux que nous avons décrits dans la section précédente. Nous évoquerons au chapitre 6 les perspectives offertes par cette solution, notamment aux enjeux de sécurité et de passage à l'échelle que nous n'aurons pas encore traité.

Chapitre 2

État de l'art des solutions

Le problème quand on a l'esprit ouvert, bien sûr, c'est que les gens insistent pour venir et essayer d'y mettre des choses.
— **Terry Pratchett**, *Le Grand Livre des Gnomes*

Nous allons voir, dans cette section, comment se positionne l'infrastructure participative d'hébergement de services que nous avons décrite au chapitre précédent par rapport aux solutions proposées dans la littérature scientifique. Nous nous intéresserons premièrement aux solutions participatives, qui s'appuient sur le partage bénévole de ressources pour rendre un service à une communauté d'utilisateurs. Notre cas d'usage faisant appel à de l'activation de service à la demande, nous nous pencherons ensuite sur les approches info-nuagiques appliquées sur les équipements de bordure, comme ceux présents dans le réseau domestique. Le parcours de ces domaines nous permettra d'identifier des solutions participatives utilisant une approche info-nuagique afin d'offrir un contexte d'exécution d'applications à la demande. Nous évaluerons ces solutions par rapport aux enjeux que nous avons identifiés dans la section précédente pour identifier les contributions que nous pourrons apporter à l'état de l'art.

2.1 Domaines de solutions existantes

Nous allons tout d'abord parcourir deux domaines de solutions qui recouvrent en partie notre cas d'usage : le partage volontaire de ressources situées sur des équipements terminaux, puis l'exploitation à la demande de ces ressources par une approche info-nuagique.

2.1.1 Solutions de partage de ressources

Au moment de la démocratisation de l'Internet, de la fin du 20ème siecle jusqu'au années 2000, les premiers utilisateurs ont été confrontés à une rareté des ressources mises en lignes : les capacités de calcul disponibles était limités sur les postes terminaux, les contenus présents sur

peu de serveurs et la connectivité restreinte à quelques emplacements. Des techniques sont alors apparues pour améliorer la disponibilité de ces ressources par le partage entre utilisateurs.

Partage de ressource de calcul

Les projets de calcul participatif (ou *Volunteer Computing*) mettent à contribution les ressources processeur de milliers voire millions de machines pour résoudre un problème précis. L'exemple le plus connus de ces projets est SETI@Home [7] dont l'objectif est la recherche de signaux extra-terrestres dans les données du radiotélescope d'Arecibo. Ce projet a démarré en 1999 et s'est poursuivi jusqu'en 2020. D'autres projets lui ont emboité le pas et sont toujours actifs comme Folding@Home [16], démarré en 2000 dont l'objectif est d'explorer les différentes solutions de repliement de protéines, notamment à la recherche de médicament.

Ces projets se basent sur la décomposition du problème à résoudre en de multiples tâches de calculs suffisament simples pour s'exécuter sur le processeur d'un simple ordinateur de bureau en un temps raisonnablement court. La gestion de ces tâches élémentaires est centralisée via un logiciel comme BOINC [6], capable ainsi de créer des grilles de calcul massivement distribuées. L'utilisateur volontaire installe le client sur l'équipement qu'il souhaite partager. Le client reçoit du serveur BOINC du projet une tâche de calcul élémentaire à effectuer et pour laquelle il retournera le résultat.

Le succès de ces projets montre que des utilisateurs sont potentiellement volontaire pour partager leurs ressources de calcul dans l'objectif de participer à un projet d'intérêt général. Cette contribution est bénévole, sans bénéfice personnel. Il a été montré dans [56] qu'en comparant le prix de revient pour une même capacité de calcul, il est parfois plus intéressant d'utiliser une grille massivement distribuée plutôt que des serveurs loués chez un fournisseur d'infrastructure. Des incitations, financières ou sociales (réputation), auprès des utilisateurs peuvent aussi favoriser ce partage de ressources [21].

Partage de contenus et d'espace de stockage

Le partage de fichiers en pair-à-pair s'est développé pour répondre à la rareté des contenus en ligne en permettant à des utilisateurs de partager les fichiers présents sur leur disque dur. Il s'est d'abord orienté sur des contenus musicaux avec des solutions comme Napster. Il s'est ensuite généralisé à n'importe quel type de fichier avec des solutions comme Bittorrent. Le principe est de partager un ensemble de données qui va se propager dans le réseau au fur et à mesure que des utilisateurs le téléchargent. Une fois téléchargé, les données sont automatiquement remises en

partage. Le même contenu est alors partagé par plusieurs sources différentes. Cette duplication des données dans le réseau est mise à profit pour optimiser le débit des transferts. Le projet IPFS [1] exploite ces mécanismes dans l'objectif de fournir un système de fichiers réparti avec de bonnes performances et capable de passer à l'échelle [24].

Les solutions pair-à-pair ont plusieurs propriétés intéressantes [64]. Elles fonctionnent de manière complètement décentralisée, ne nécessitant pour leur fonctionnement que d'un point de rendez-vous (*tracker*) pour la découverte des utilisateurs entre eux. Elles ont de plus une capacité importante de passage à l'échelle : un même partage peut être assuré par un grand nombre d'utilisateurs. Elles sont enfin très robustes et capables de continuer à fonctionner malgré des ruptures de connectivité à grande échelle. L'intérêt de ces solutions se présente dans des réseaux avec un nombre important de participants. Il est moins évident pour une communauté d'utilisateurs plus restreinte, où la duplication des données sera moins importante.

Partage de bande passante réseau

Avant l'essor de l'Internet en mobilité via les réseaux 3G, la connectivité au réseau n'était disponible qu'à certains endroits : bibliothèques, cyber-cafés ou domiciles bénéficiant d'un accès résidentiel. Les réseaux communautaires sans fils (*Wireless Community networks*) se sont développés pour participer à l'extension de cette connectivité à une zone gégraphique plus large. Grâce à la banalisation des technologies de réseaux sans fils au début des années 2000, des possesseurs d'équipements WiFi ont commencé à partager leur connexion Internet avec d'autres utilisateurs, principalement en milieu urbain. Ces initiatives individuelles se sont fédérées en projets communautaires, avec comme objectif de couvrir une ville en réseau et d'offrir un accès ubiquitaire. Citons comme exemple de projets Seattle Wireless [35] aux Etats Unis, un des premiers projets de grande envergure, Ninux en Italie [60] et Guifi.net en Espagne [14] qui sont aujourd'hui des réseaux étendus dans plusieurs villes dans leur pays. Même s'ils sont aujourd'hui concurrencés par les réseaux mobiles 5G, ces réseaux continuent à se développer dans un esprit d'entraide locale et d'indépendance.

Les équipements participants créent un réseau sans-fils maillé capable de fournir certains services. Différentes expérimentations ont ainsi testé la téléphonie [1] ou la diffusion vidéo [79]. L'hébergement d'applications distribuées sur ces équipements est aussi envisagé [33] pour enrichir les services offerts par ces réseaux. Ils peuvent en effet, grâce à un maillage dense d'équipements, fournir des services de proximité dans des scénarios de ville intelligente [59].

1. `https://ipfs.io/`

Ces différentes solutions montrent que le partage volontaire de ressources est pertinent dans différents domaines pour répondre à certains besoins. Leurs utilisateurs sont souvent aussi contributeurs de ces ressources. Mais ces solutions sont limitées à un domaine spécifique, et au partage d'un même type de ressource. Elles imposent des contraintes fortes sur le type d'applications utilisable dans leur contexte. Or dans notre cas, nous souhaitons un partage de ressources de différentes natures, ressources qui seraient utilisables par des applications génériques, comme par exemple les applications collaboratives destinées à la communauté virtuelle.

2.1.2 Solutions info-nuagiques sur équipements terminaux

Les solutions info-nuagiques (ou *Cloud*) offrent des outils pour exploiter, de manière extensible et à la demande, des ressources de calcul et de stockage et les mettre à la disposition de l'utilisateur. Ces solutions utilisent intensivement la virtualisation de ces ressources pour pouvoir ouvrir cette offre à un grand nombre de clients. Les solutions info-nuagiques se catégorisent selon le niveau de service offert à l'utilisateur :

IaaS (pour *Infrastructure as a service*) : la solution offre à l'utilisateur une ou plusieurs machines virtuelles mises en réseau.

PaaS (pour *Platform as a service*) : la solution offre à l'utilisateur un contexte d'exécution distribué pour ses applications.

SaaS (pour *Software as a service*) : la solution offre à l'utilisateur des applications prêtes à l'emploi et le stockage de ses données.

Une solution info-nuagique apporterait à notre cas d'usage une grande facilité de mise à disposition des applications sur une infrastructure participative. Notre scénario prévoit en effet que les utilisateurs de la communauté puissent activer des applications sans besoin de compétence en administration informatique. Le niveau de service *SaaS* offre donc le niveau d'abstraction demandé par les utilisateurs. Pour que ces applications soient le plus générique possible, il est nécessaire qu'elles s'appuient sur un niveau d'abstraction des ressources de l'infrastructure participative. Notre solution devra choisir entre le niveau de service *PaaS*, qui demande une adaptation minimale de l'application au contexte d'éxecution, et *IaaS*, pour lequel aucune adaptation n'est requise car identique au contexte d'éxecution sur un serveur.

L'approche info-nuagique a permis aux fournisseurs de solutions de densifier de manière incrémentale les ressources de calcul et de stockage dans de grands centres de données, afin d'accueillir toujours plus de clients. Ce faisant, cette pratique a créé un abus de langage courant qui identifie le *Cloud* à une solution nécessairement basée sur des centres de données. Or l'apparition des nouveaux usages de l'Internet des Objets, utilisant des capteurs et autres équipements embarqués,

a mis en évidence l'intérêt des ressources de calcul situées en dehors de ces centres de données. Dans la Figure 2.1, ces ressources sont localisées dans une couche intermédiaire nommée *Edge*, située entre les objets connectés et le *Cloud*, désignant ici des grands centres de données. Cette couche *Edge* est constituée d'équipements, dits de bordure, qui peuvent prendre part à une solution info-nuagique avec ses différents niveaux de service. Le concept de *Edge Cloud Computing* vient donc compléter celui du *Cloud* dans les centres de données. L'utilisateur d'une telle solution pourra déployer ses applications sur l'infrastructure constituée de ces équipements de bordure aussi facilement qu'il le ferait sur un centre de données.

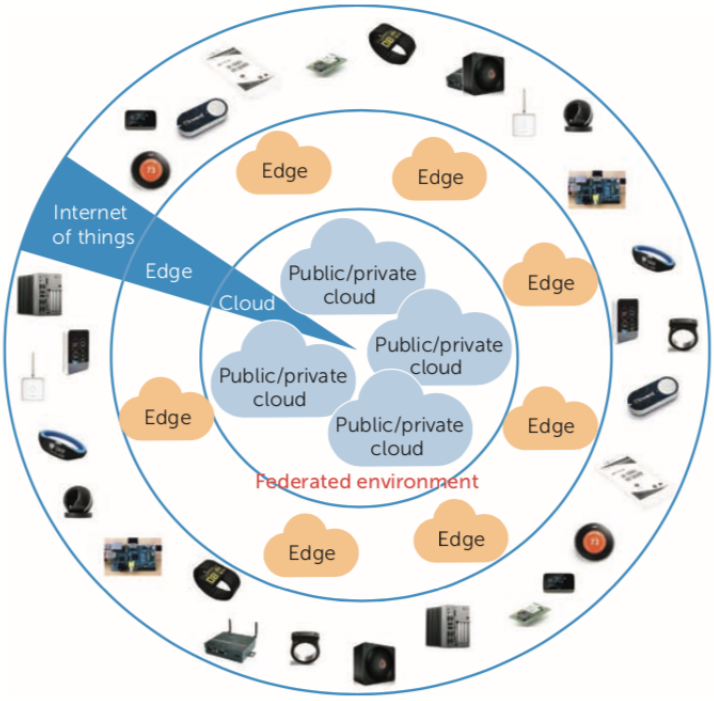

FIGURE 2.1 – Différentes couches info-nuagiques. Illustration tirée de [95]

.

Fog Computing

Le concept de *Fog Computing* [17] s'inscrit dans celui du *Edge Cloud Computing* avec l'idée d'utiliser ces équipements de bordure en appui des centres de données. Il prévoit l'existence de ressources de calcul proches des utilisateurs, distribuées à l'échelle d'une ville ou d'un opérateur réseau. Ces équipements ont une taille variable allant de celle d'un routeur jusqu'à un micro-centre de données composé de quelques serveurs. Les applications déployées sur ces équipements de bordure sont délocalisées depuis les centres de données centraux, avec un objectif de tirer

parti de leur proximité avec l'utilisateur pour en augmenter l'efficacité. L'Internet des Objets est très présent parmi les cas d'usages du *Fog Computing* [94]. Les ressources en bordure sont alors utilisées pour traiter au plus tôt des données générées par des capteurs pour éviter un goulot d'étranglement dans le réseau. D'autres usages du *Fog Computing* [3] ont pour objectif d'améliorer le temps de réponse des services en les localisant au plus près des utilisateurs et évitant ainsi des communications longues distances sur le réseau. Ce concept s'applique difficilement à notre cas car il impose l'existence préalable d'un centre de données sur lequel les services sont instanciés avant d'être délocalisés sur les équipements de bordure. Les équipements participant à une infrastructure de *Fog Computing* ne coopèrent pas directement entre eux. De nouvelles approches sont donc nécessaires [4] pour exploiter le scénario participatif qui nous envisageons.

Osmotic computing

Une autre approche pour l'utilisation des équipements de bordure est celle développée sous le terme d' *Osmotic computing* [95]. Elle propose d'exécuter indifféremment une application dans un centre de données ou sur des équipements de bordure. Cette unification du contexte d'exécution permet d'envisager que les applications puissent migrer entre l'un et l'autre environnement. Des applications de ce concept sont envisagées dans la domotique [75] ou la réalité augmentée [82], avec des scénarios où plusieurs équipements de bordure coopèrent entre eux. Cette coopération nous semble une piste intéressante pour l'environnement d'exécution des applications de l'infrastructure participative.

Cette approche met en évidence que pour s'exécuter dans un centre de données comme sur des équipements de bordure, l'application devra prendre une forme particulière. Les centres de données peuvent facilement exécuter n'importe quelle type d'application grâce à des ressources nombreuses et centralisées. Les équipements de bordure ont eux des ressources plus modestes et surtout éparses. La proposition de l'*Osmotic computing* est de décomposer l'application sous forme de micro-services élémentaires communiquant entre eux. Ces micro-services peuvent alors être déployés indifféremment dans un même centre de données ou dispersés sur plusieurs équipements de bordure. Tant que ces micro-services peuvent coopérer entre eux, l'application restera fonctionnelle.

Les micro-services [67] sont une approche des architectures logicielles orientées service. La conception à base de micro-services d'une application s'oppose à une conception monolithique où toutes les fonctions de l'application sont situées dans le même logiciel. Au contraire, un micro-service intègre une fonction unique de l'application qui peut être déployée indépendamment et réutilisée dans plusieurs applications. Chaque micro-service doit spécifier une interface

décrivant ses fontions disponibles (*API*) bien définie et utilisée par d'autres micro-services. L'application est alors constituée par une composition de plusieurs micro-services indépendants et non spécialisés à l'application.

L'exploitation d'applications basées sur des micro-services nécessite un gestionnaire spécifique, le plus connu d'entre eux étant Kubernetes [2]. Les micro-services utilisent la conteneurisation, un niveau de virtualisation système moins consommateur que la virtualisation d'une machine. Le gestionnaire d'application assure lui l'orchestration de ces conteneurs sur différents serveurs, c'est à dire le choix d'un serveur pour chaque instance de micro-services, puis le déploiement sur le serveur choisi des données nécessaires de chaque conteneur, l'activation du conteneur sur ce serveur et la mise en réseau des différents conteneurs participant à une même application. Cette dernière étape implique notamment la définition d'un plan d'adressage puis de nommage des instances des micro-services pour qu'elles puissent se découvrir puis échanger entre elles. Bien que destinés à une utilisation dans un centre de données, ces outils peuvent aussi être utilisés dans un contexte plus largement distribué sur des équipements de bordures [38][76].

2.2 Analyses de solutions comparables

Nous avons identifié dans la section précédente les différents domaines de solutions participatives où des utilisateurs mettent en commun des ressources de calcul, de stockage et de communication. Nous avons aussi souligné l'intérêt d'une approche info-nuagique pour définir un contexte d'exécution de l'application à partir d'une abstraction des ressources disponibles. Nous allons dans cette section étudier des solutions de ces différents domaines utilisant une approche info-nuagique dans un contexte participatif, afin d'évaluer leurs contributions aux enjeux auxquels nous souhaitons répondre.

2.2.1 Solutions identifiées

Nous avons catégorisé ces solutions en 3 familles : les nuages participatifs, les nuages pair-à-pair et les nuages communautaires.

2. `https://kubernetes.io/`

Solutions de nuages participatifs

Les solution de nuages participatifs (*Voluntary Clouds*) [65][53][62] réutilisent le principe du calcul participatif mais avec une approche info-nuagique. Elles proposent de créer un contexte d'exécution unifié à partir de ressources distribuées fournies de manière volontaire par les utilisateurs. L'exécution des applications est attribuée aux équipements participants par un serveur de coordination. Nous avons identifié trois solutions intéressantes dans notre contexte.

La solution Nebula [74] a pour objectif de créer une infrastructure de calcul à partir de ressources d'équipements partagées bénévolement. Elle se destine aux applications de calcul intensif décomposées sous forme de tâches et nécessitant beaucoup de données, comme Hadoop MapReduce. Elle se différencie ainsi de la solution BOINC qui impose une contrainte sur la quantité de données disponibles pour une tâche de calcul. L'architecture de Nebula distingue les équipements destinés au stockage de ces données, d'autres équipements effectuant le calcul sur ces données. Cette séparation demande donc de considérer à la fois les ressources de calcul et la capacité de communication de chaque équipement. Le coordinateur du système Nebula doit en effet placer chaque tâche de calcul sur un équipement avec un processeur disponible et une bande passante suffisante vers les équipements stockant les données nécessaires à cette tâche. Cette solution a été évaluée dans l'environnement PlanetLab avec une distribution géographique des équipements participants. Les tests effectués ont montré de meilleures performances dans l'exécution des tâches MapReduce par rapport aux implémentations basées sur BOINC. Ces implémentations se basent en effet sur un stockage centralisé des données, ce qui crée un goulot d'étranglement pénalisant les performances. Avec l'approche de Nebula, la distribution des données et le placement adéquat des tâches permet d'éviter ce phénomène. Les tests ont de plus permis de démontrer la robustesse de la solution en cas de défaillance d'un équipement participant, ainsi que sa capacité de passer à l'échelle.

La solution cuCloud [63] propose d'utiliser les ressources disponibles partagées sur des équipements pour y héberger des machines virtuelles. Les ressources sont localisées sur un même site, le cas d'usage est d'utiliser les ordinateurs de bureau inutilisés d'une organisation (*Desktop Cloud*). La solution se base sur la base logicielle Apache CloudStack. Les auteurs ont ajouté un composant pour gérer dynamiquement les équipements participant à l'infrastructure. Ces équipements remontent en permanence la disponibilité de leurs ressources de calcul, permettant au coordinateur d'évaluer leur capacité d'hébergement. Ce coordinateur choisit l'équipement sur lequel déployer chaque machine virtuelle en fonction des ressources disponibles et des besoins exprimés par l'utilisateur en terme de capacité attendue. Les auteurs ont évalué leur solution en comparant l'exécution d'une application d'analyse de données (Hadoop MapReduce) sur deux infrastructures, une fournie par cuCloud et l'autre par le fournisseur Amazon EC2. Il apparait

que la solution proposée est plus performante car bénéficiant d'une meilleure latence réseau. Les auteurs soulignent aussi que les temps de calcul observés pour l'application étaient plus stables sur leur infrastructure locale que sur celle hébergée dans le centre de données d'Amazon. L'expérimentation décrite dans l'article n'incluait cependant pas de scénarios où la variation de la disponibilité des ressources nécessite une migration entre équipements des machines virtuelles.

La solution Cloud@Home [27] [26] a pour objectif de fournir un service équivalent aux fournisseurs d'infrastructure à la demande, non pas à partir de serveurs dans des centres de données, mais d'équipements domestiques partagées bénévolement. L'architecture de cette solution prévoit une gestion des ressources à travers des *brokers* dédiés pour le calcul ou le stockage ayant connaissance des ressources disponibles sur chacun des équipements. Ces ressources sont ensuite exploitées pour créer un système de fichiers global accessible depuis les équipements participants et un orchestrateur planifiant le déploiement de machines virtuelles sur ces équipements. Ce dernier composant choisit l'équipement où déployer chaque machine virtuelle en fonction des ressources disponibles et des contraintes exprimées par l'utilisateur sous forme de Service Level Agreement (SLA). Une première implémentation de cette architecture a été réalisée en 2015 [29] sur la base logicielle Openstack. Les composants de Cloud@Home s'appuient sur les fonctionnalités existantes du sous-système Nova d'Openstack pour la création et la mise en réseau des machines virtuelles. Ce prototype n'inclue cependant pas les fonctionnalités d'enregistrement automatique de nouveaux participants et la supervision des ressources disponibles. Cette implémentation ne permet donc pas de tester des scénarios de variations de la disponibilité des ressources.

Solutions de nuages pair-à-pair

Les solutions de nuages pair-à-pair (*P2P Cloud*)[61] visent à créer une infrastructure d'hébergement à la demande à partir de ressources distribuées sans l'utilisation de serveurs centralisés pour assurer la gestion de ces ressources et le déploiement des applications sur l'infrastructure. L'utilisation des mécanismes pair-à-pair, déjà proposés dans d'autres domaines, apportent à ces solutions des propriétés de robustesse et de capacité à passer à l'échelle.

La solution P2PCS [12] vise à offrir un service d'infrastructure à la demande à l'image des solutions centralisée dans un centre de données. L'infrastructure est ici construite de manière décentralisée à partir des ressources mises à disposition sur différents équipements distribués localement ou géographiquement. Une telle solution se destine à des applications pouvant bénéficier d'être déployées proche de l'utilisateur : diffusion vidéo, jeux en ligne, applications collaboratives. Les auteurs ont développé dans la solution P2PCS des algorithmes pour rendre l'infrastructure

robuste aux partitionnements du réseau pair-à-pair. La gestion des ressources est basée sur le concept de *slice*, sous-ensemble du réseau pair-à-pair créé en fonction du besoin exprimé par un utilisateur. Les équipements participants sont classés selon différentes métriques décrivant notamment les ressources disponibles localement. Le système identifie à partir de ce classement les équipements à inclure dans le *slice* attribué à chaque utilisateur. L'article décrit une première implémentation de cette gestion des slices à la demande, mais ne montre pas d'évaluation d'un système complet permettant le déploiement d'applications.

La solution MycoCloud [30] exploite elle aussi un réseau pair-à-pair de grande envergure mais se distingue par une approche orientée service. Se basant sur des travaux préalables (MycoNet et MycoLoad) définissant une infrastructure pair-à-pair capable d'héberger des services, MycoCloud se concentre sur le problème du placement de ces services sur cette infrastructure. Considérant que chaque équipement a une capacité limitée de traitement des requêtes pour ces services, Mycocloud cherche à déployer ces services sur les équipements de manière à maximiser le temps de traitement d'un ensemble de requêtes. MycoNet crée à partir d'un ensemble d'équipements un réseau pair-à-pair organisé en grappe (*clusters*), construit à la demande, robuste et capable de passer à l'échelle. MycoLoad utilise une classification des services pour créer dans un réseau Myconet des grappes d'équipements spécialisées pour chaque classe de service. MycoCloud s'intéresse lui à dimensionner dynamiquement ces grappes en fonction des requêtes utilisateurs pour chaque service. Nous n'avons pas trouvé d'évaluation de ce type de solution dans un cadre plus réaliste que celui de la simulation.

Solutions de nuages communautaires

Les solutions de nuages communautaires (*community clouds*) s'appuient sur un réseau communautaire (*Wireless Community networks* par exemple) pour offrir un service d'hébergement d'application avec une approche info-nuagique.

La solution Cloudy est un prototype de service d'hébergement basé sur le réseau communautaire Guifi.net. Ce réseau se déploie sur plusieurs villes d'Espagne (Barcelone, Valences, Madrid) et compte plus de 30.000 équipements participants [3]. Parmi plusieurs études académiques s'appuyant sur ce réseau, l'une d'entre elles [52] s'est donné pour objectif d'exploiter les ressources de calcul de ces équipements avec une approche info-nuagique. Une première approche nommée Cloudy se base sur une plateforme matérielle homogène entre les participants, connectée via le réseau Guifi.net. Sur chacun de ces équipements sont déployés différents services sous la forme

3. `http://guifi.net/guifi/menu/stats/nodes`

de conteneurs logiciels. Ces services sont ensuite accessibles de tous les équipements utilisant la solution Cloudy.

Une seconde étude [80] étend la solution Cloudy à des applications distribuées (vidéo streaming, architecture web 3-tiers) et s'intéresse au placement des différents composants de l'application sur les équipements. Elle propose un algorithme pour instancier le graphe des composants de l'application sur le graphe de connectivité réseau entre les équipements. L'objectif est de minimiser le temps de réponse de l'application en optimisant la charge des liens utilisés dans le réseau communautaire. Les résultats montrent une amélioration significative des performances des applications grâce au choix d'un placement des composants adapté aux caractéristiques des liens du réseau. Les auteurs ont évalué cet algorithme dans des conditions réelles d'utilisation du réseau [78] et les résultats se sont montrés satisfaisants.

2.2.2 Analyse des solutions existantes

Critères de comparaison

Nous allons évaluer la réponse apportée par les différentes solutions que nous venons de décrire, aux enjeux que nous avons identifiés dans le chapitre 1 pour notre approche d'infrastructure participative d'hébergement à destination d'une communauté virtuelle d'utilisateurs.

Déploiement des applications en fonction des ressources disponibles La solution recherchée doit être capable d'identifier les profils d'équipements avec des capacités de calcul et de communication compatibles avec les besoins de l'application et les exigences de l'utilisateur.

Maintien de la qualité d'expérience La disponibilité des ressources dans une infrastructure participative étant variable globalement ainsi qu'au niveau de chaque équipement, la solution doit être capable d'adapter le déploiement de l'application en fonction de ces variations pour en maintenir la qualité d'expérience par les utilisateurs.

Sécurité de l'infrastructure participative La solution recherchée doit être robuste aux usages malveillants visant à perturber son fonctionnement, ainsi qu'aux détournements de ressources pour des usages en dehors de la communauté.

Capacité à passer à l'échelle La solution recherchée doit pouvoir être utilisée dans un contexte avec un nombre important d'utilisateurs, ainsi que prévoir des scénarios de participation dans plusieurs infrastructures.

Nous caractériserons de plus chaque solution par le niveau de service proposé pour décider laquelle des approches Infrastructure-as-a-Service (IaaS) ou Platform-as-a-Service (PaaS) est

plus indiquée dans notre contexte. Nous évaluerons enfin le niveau de maturité des solutions, pour identifier les solutions qui auront été testées dans un contexte réel d'utilisation.

Évaluation des solutions

Le résultat de cette analyse est résumé dans le Tableau 2.1. Nous détaillons ensuite les contributions que nous avons relevées dans les solutions retenues.

Les solutions telles que CuCloud, Cloud@Home ou P2PCS, permettent à l'utilisateur d'héberger des machines virtuelles. Elles offrent donc un niveau de service assimilable à celui d'IaaS. Ces solutions définissent une garantie de service sur la base des SLA contractualisés avec l'utilisateur selon ses besoins, notamment l'application qu'il souhaite héberger. L'infrastructure a ensuite l'obligation de satisfaire ces exigences dans le cadre défini par le SLA. Par exemple, une solution IaaS cherchera à assurer auprès de l'utilisateur l'utilisation d'une Virtual Machine (VM) avec un taux de disponibilité de 99.9% avec des caractéristiques de processeur et de réseau spécifiés dans le SLA. Une VM ne pouvant s'exécuter que sur un seul équipement, les solutions IaaS sélectionnent donc cet équipement de telle sorte que ses ressources locales disponibles soient compatibles avec des exigences exprimées par l'utilisateur pour cette VM. Elles incluent donc un système de supervision de ces ressources pour en connaitre la disponibilité sur tous les équipements participant en chaque instant. Si les ressources locales diminuent en dessous des exigences définies pour la VM hébergée, la solution doit normalement migrer celle-ci sur un autre équipement avec des ressources suffisantes. CuCloud ou Cloud@Home s'appuyant sur respectivement sur CloudStack et OpenStack, ce mécanisme doit normalement être disponible. Mais nous n'avons pas trouvé, dans l'évaluation de ces solutions, de scénarios impliquant cette migration.

Il nous semble important ici de noter que le provisionnement de ressources sur la base de SLA ne peut garantir l'expérience utilisateur qu'à deux conditions. Les ressources demandées devront d'une part bien correspondre aux besoins de l'application pour assurer un temps de réponse suffisant. L'expression de ce besoin demande une connaissance à priori du comportement de l'application en fonction des ressources mises à disposition et du nombre de requêtes à traiter. Il est d'autre part nécessaire, pour évaluer pleinement la qualité d'expérience, de prendre en compte la QoS des réseaux entre l'application et les terminaux des utilisateurs. Les solutions IaaS ayant pour objectif d'héberger des services ouverts au plus grand nombre d'utilisateurs, ces réseaux sont hors de portée de leur contrôle et elles ne peuvent donc en garantir les paramètres.

Les solutions d'hébergement de services comme MycoCloud et Cloudy, ou de déploiement de tâches comme Nebula, se situent à un niveau de service plus proche de l'application, et sont

Solution	Service proposé	Adaptation aux ressources par équipement	Maintien de la QoE	Passage à l'échelle	Sécurité	Maturité de la solution
Nuages participatifs						
CuCloud [63]	Mise à disposition de machines virtuelles	Sélection de l'équipement selon le profil de la MV à créer	Disponibilité des ressources supervisées mais pas de migration effectuée en cas de changement	Objectif donné à plusieurs milliers d'équipements, mais pas évalué	Module de sécurité prévu, mais non implémenté	Implémentation sur base de CloudStack, testée avec 3 serveurs
Cloud@Home [29]	Mise à disposition de machines virtuelles	Sélection de l'équipement selon le profil de la MV à créer	Supervision des noeuds basées sur OpenStack	Aucune	Aucune	Implémentation préliminaire sur OpenStack sans évaluation.
Nebula [74]	Traitement de données sous forme de tâches de calcul	Placement des tâches de calcul en fonction de la QoS réseau disponible vers les données	Tolérance aux fautes et au partitionnement du réseau par réplication des données	Solution capable de passer à l'échelle en nombre de participants et en volume de données	Aucune	Évaluation sur PlanetLab sur 30 serveurs répartis géographiquement
Nuages pair-à-pair						
P2PCS [12]	Mise à disposition de machines virtuelles	Classement des équipements pour identifier les candidats compatibles à la demande de l'utilisateur	Tolérance au fautes et partitionnement du réseau	Evaluation en simulation des protocoles sous-jacents sur de très larges réseaux ($> 2^{10}$ nœuds)	Aucune	Prototype pour validation du concept
MycoCloud [30]	Hébergement de services	Sélection des équipements en fonction de la classe de service	Dimensionnement automatique en fonction du volume de requêtes	Solution évaluée sur plus de 1000 nœuds simulés	Aucune	Solution testée sur simulateur, dans un contexte réseau homogène et stable
Nuages communautaires						
Cloudy[52] [78]	Déploiement de micro-services	Placement des services pour maximiser la bande passante restante et éviter les congestions dans le réseau	Redondance par duplication du service	Solution évaluée en simulation sur un réseau de 71 nœuds	Aucune	Évaluations ponctuelles en conditions réelles sur un sous-ensemble du réseau communautaire Guifi

TABLE 2.1 – Comparaison de solutions existantes

donc assimilables à des solutions PaaS. Elles ne définissent pas d'exigences utilisateurs sous forme de SLA mais cherchent à minimiser le temps de réponse de l'application hébergée afin de maximiser l'expérience utilisateur. Le placement des services ou des tâches est choisi dans cet objectif. Chacune de ces solutions a choisi une approche différente dans ce choix de placement : Nebula cherche à placer les tâches de calcul proches des données pour diminuer les temps de communication ; Cloudy place les services sur les équipements en fonction de la bande passante disponible pour éviter les congestions ; MycoCloud augmente le nombre de services pour traiter plus de requêtes simultanément. Cette dernière adapte dynamiquement ce placement en fonction du nombre de requêtes. Cependant elle ne supervise pas les ressources disponibles localement pour adapter ce placement en cas de changement. La solution Nebula ne considère que le cas de pertes d'équipement en réaffectant les tâches non-terminées sur d'autres équipements. La solution Cloudy utilise la disponibilité de chaque équipement en plus de la bande passante dans ces critères de sélection pour le placement des services. Mais l'article ne décrit pas comment adapter ce placement en cas de changement dans ces paramètres.

Les solutions que nous venons d'analyser se destinent à différents contextes d'utilisation avec différents volumes d'équipements participants. Les solutions de nuages pair-à-pair utilisent des mécanismes pouvant fonctionner à une échelle de plusieurs milliers d'équipements. Mais nous n'avons pas trouvé d'évaluation de ces solutions dans de tels scénarios en conditions réelles. Seules les solutions Nebula et Cloudy ont démontré leur capacité à fonctionner expérimentalement sur plusieurs dizaines d'équipements physiques. Enfin la problématique de sécurité n'est pas traitée dans les solutions identifiées. Les auteurs de CuCloud mentionnent pourtant dans leur article l'importance de cette dimension dans une infrastructure participative et prévoient à cet effet un module de sécurité dans leur solution. La réalisation de ce module n'est cependant pas décrite dans leurs travaux.

2.3 Positionnement de notre contribution

Cette étude des solutions existantes nous permet de dégager les axes d'une contribution scientifique aux défis que nous avons identifiés pour la réalisation d'une infrastructure participative d'hébergement.

Les solutions de type IaaS, comme Cloud@Home ou CuCloud, montrent certaines limites car elles nécessitent, au niveau de chaque équipement, des ressources nécessaires pour le support de virtualisation. L'approche par tâches de calcul utilisée par Nebula est intéressante mais demande une adaptation forte des applications collaboratives classiques que nous envisageons. Les applications orientées services, comme utilisées dans MycoCloud ou Cloudy, nous semblent donc

plus adaptées au déploiement sur des équipements domestiques. Elles permettent d'exploiter des ressources de calcul réduites et hétérogènes, à l'image des plateformes utilisées pour Cloudy. Les applications utilisées dans cet exemple précis, sont cependant assez simple (Web 3-tiers). Il nous semble intéressant d'envisager des applications plus complexes en terme d'interactions entre services, comme dans le cas de MycoCloud. Les micro-services, proposés dans l'approche *Osmotic Computing* nous semblent donc une piste à suivre.

Notre solution d'infrastructure participative offrira un niveau de service PaaS et permettra l'hébergement de ces micro-services. Elle inclura donc les mécanismes nécessaires pour leur déploiement sur les équipements participants : mise à disposition sur l'équipement des données nécessaires à l'exécution du micro-services, instanciation de micro-services sur chaque équipement, mise en réseau des services instanciés sur les différents équipements, découverte des instances entre les micro-services. Les clients accèdent à l'application à travers leur réseau domestique et communiquent avec les équipements hébergeant les micro-services de l'application.

Dans le fonctionnement de ce type de solution, l'expérience utilisateur de l'application dépendra de l'adéquation entre les besoins de chaque micro-service et des capacités de calcul et de communication au niveau de chaque équipement. Nous présenterons dans le chapitre 3 un modèle de cette expérience utilisateur pour une application composée de micro-services déployée sur une infrastructure participative. Nous utiliserons pour définir cette expérience la localisation des clients, qui est connue dans notre cas car les clients sont, en tant que membres de la communauté virtuelle, participants de l'infrastructure participative. Pour définir les capacités de communication de chaque équipement, nous n'aurons pas une connaissance totale de la topologie du réseau et de la qualité des liens comme dans la solution Cloudy. Mais la connaissance des caractéristiques des réseaux d'accès résidentiels peut nous aider à trouver pour chaque micro-service un placement adéquat. Nous proposerons, sur la base de notre modèle de l'expérience utilisateur, une heuristique pour trouver un tel placement, en fonction de la capacité de calcul et de communication de chaque équipement, ainsi que des besoins de chaque micro-service, maximisant cette expérience utilisateur.

Notre objectif est de démontrer la faisabilité de l'infrastructure participative dans des conditions réelles d'utilisation. Il nous faudra donc évaluer ces premiers résultats sur une instance réelle d'infrastructure participative. Nous décrirons cette plateforme dans le chapitre 4. En déployant une application fonctionnelle sur cette plateforme, nous pourrons évaluer son comportement et les variations de ses performances, notamment liées aux usages concurrents sur le réseau d'accès domestique. Si cette dégradation est trop importante, il peut alors être intéressant de changer la localisation de certains micro-services. Ils sont en effet plus faciles à migrer qu'une machine

virtuelle. Ce principe est d'ailleurs repris du concept d'*Osmotic Computing*. Il nous sera alors possible déplacer un service d'un équipement vers un autre lorsque les capacités de calcul ou de communication sont réduites par des usages concurrents. Le placement des micro-services est ainsi capable de s'adapter aux variations de la disponibilité des ressources. Nous proposerons, dans le chapitre 5, un mécanisme d'adaptation de ce placement.

Il est aussi possible d'envisager un dimensionnement automatique du nombre d'instances de micro-services pour s'adapter aux volumes de requêtes à traiter, comme décrit dans la solution MycoCloud. Nous ne traiterons pas, dans cette thèse, cette dimension de passage à l'échelle, ni celle de l'augmentation du nombre d'équipements intervenant dans l'infrastructure participative. Les solutions Nebula et Cloudy ont été testées sur quelques dizaines d'équipements, ce qui nous semble un dimensionnement raisonnable dans notre cas d'étude. Les solutions à grande échelle que nous avons identifié n'ont quant à elles démontré leurs résultats que par simulation. Nous évoquerons la problématique du passage à l'échelle dans les perspectives de ce travail dans le chapitre 6, à la lumière des résultats que nous aurons obtenus.

Chapitre 3

Optimisation du placement d'une application sur une infrastructure participative

Une façon de traiter une tâche impossible était de la découper en un certain nombre de tâches simplement très difficiles, et de diviser chacune d'entre elles en un groupe de tâches horriblement difficiles, et chacune d'entre elles en tâches délicates, et chacune d'entre elles. . .
– **Terry Pratchett,** *Le Grand Livre des Gnomes*

Nous allons traiter dans ce chapitre du déploiement d'une application sur une infrastructure d'hébergement participative. Dans cette phase de déploiement, nous nous intéressons particulièrement au placement des micro-services composant l'application sur les équipements partagés par les membres de la communauté. Notre objectif est de trouver un placement permettant d'optimiser les performances de l'application du point de vue de ses utilisateurs. L'indicateur choisi pour caractériser ces performances est le temps de réponse de l'application. Nous décrirons dans ce chapitre un modèle de ce temps de réponse à partir des différents paramètres liés au placement des micro-services et des caractéristiques de l'infrastructure. Nous en déduirons une formalisation du problème d'optimisation du temps de réponse. Nous proposerons deux heuristiques à ce problème et nous évaluerons les solutions proposées par ces heuristiques sur différents cas d'étude.

3.1 Cadre de l'étude

Nous allons dans cette section fixer le périmètre de cette étude. Nous définirons le cas d'usage à partir duquel nous exprimerons les hypothèses de départ de notre étude.

3.1.1 Cas d'usage

Le scénario d'usage de l'infrastructure participative pour notre étude implique une communauté d'individus de taille raisonnable, entre 10 et 50 personnes, comme une association locale ou un club sportif. Les membres de cette communauté souhaitent partager des photos prises dans des événements en lien avec les activités de la communauté. Afin de rendre ces photos disponibles, la communauté souhaite se doter d'une application en ligne permettant de déposer ces photos et de les consulter ensuite sous forme d'albums. Soucieuse du caractère privé de ces données et voulant se prémunir contre une possible divulgation d'informations, la communauté ne souhaite pas utiliser une application de partage en ligne gratuite. Les membres de la communauté ne possèdent pas non plus les compétences techniques suffisantes pour déployer et administrer un serveur pour héberger l'application.

Pour répondre à son besoin de manière adaptée, la communauté choisit de mettre en œuvre une infrastructure d'hébergement participative. Les membres de la communauté activent sur des équipements présents dans leur réseau domestique, une fonction de partage de ressources de calcul et de stockage avec les autres membres. Un gestionnaire d'infrastructure d'hébergement met en commun ces ressources partagées pour créer un environnement d'exécution pour l'application. Ce gestionnaire propose d'activer l'application de partage de photo à la demande en la déployant sur les équipements de l'infrastructure.

Afin d'être distribuée sur les différents équipements de l'infrastructure participative, l'application est architecturée sous forme de micro-services. Chaque service représente une fonction spécifique de l'application. Les services sont interdépendants : ils communiquent entre eux à travers le réseau pour réaliser leurs tâches spécifiques. Lors du déploiement de l'application, le gestionnaire de l'infrastructure active les micro-services concernés sur les équipements et les interconnecte de manière à rendre possible les communications. L'infrastructure choisit pour chaque micro-service un équipement pour l'héberger, en fonction des caractéristiques de l'équipement et des besoins spécifiques du service, avec un objectif défini. Nous allons étudier ici comment décider du placement des services avec pour objectif celui d'optimiser les performances de l'application.

3.1.2 Hypothèses

Hypothèses sur l'infrastructure d'hébergement

Nous nous plaçons dans un contexte initial où les membres de la communauté ont rendu disponibles les ressources de leurs équipements pour l'hébergement des micro-services. Nous supposons

ici l'existence d'un système logiciel présent sur chacun des équipements assurant le partage de ces ressources et d'un gestionnaire pour en assurer la mutualisation. Nous nous appuierons sur ce système pour l'activation d'un micro-service sur un équipement et son interconnexion avec les autres micro-services activés sur les équipements distants.

Nous modélisons l'infrastructure participative d'hébergement comme un ensemble d'équipements avec des capacités de calcul hétérogènes, connectés à des réseaux d'accès de différentes capacités. L'interconnexion se fait à travers un réseau neutre, sans contraintes sur les communications entre équipements. La Figure 3.1 donne une représentation de cette architecture : les cercles représentant les équipements, relié au nuage représentant l'interconnexion par des traits représentant les réseaux d'accès. Les capacités des processeurs des équipements sont ici représentées par des cercles de diamètres différents et celles des réseaux d'accès par des traits pointillés ou continus.

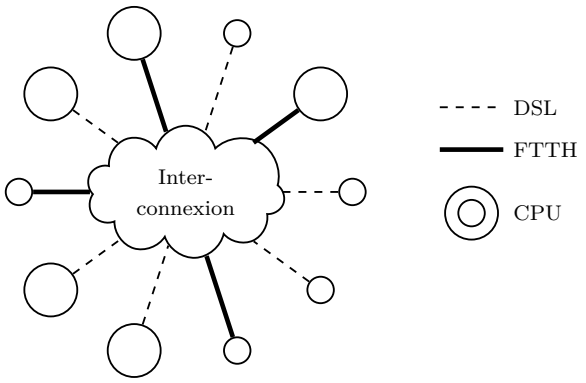

FIGURE 3.1 – Représentation d'une infrastructure participative et des ressources disponibles.

Nous considérons dans notre étude la disponibilité des ressources de calcul et de bande passante comme les principaux facteurs qui influencent les performances d'un micro-service hébergé sur un équipement. Le type de stockage utilisé sur l'équipement peut aussi modifier ces performances, mais nous faisons ici l'hypothèse que cette influence est négligeable. Dans notre cas d'usage, les utilisateurs contribuent à l'infrastructure participative avec des équipements possédant des ressources de calcul hétérogènes. Nous considérons dans notre cas d'étude deux catégories d'équipements : ceux avec d'importantes capacités de calcul et ceux avec de faibles capacités. Dans la première catégorie se retrouvent des équipements comme les ordinateurs personnels ou les consoles de jeux. La seconde catégorie comporte des équipements avec des profils plus réduits comme les set-top-box ou les serveurs de stockage.

De la même façon, les ressources de bande passante partagées par les utilisateurs sont hétérogènes. Les caractéristiques de bande passante varient selon le type de connectivité résidentielle. Nous considérons dans cette étude deux types de connectivité. Un premier type utilisant la technologie d'accès sur le réseau cuivre (Digital Subscriber Line (DSL)) se caractérise par une bande passante asymétrique avec un débit montant faible pour un débit descendant plus important. Le second type de connectivité utilise la technologie fibre optique (FTTH) et se caractérise par un

débit symétrique entre 2 à 10 fois plus important qu'avec la technologie DSL. Nous considérons dans notre étude que les capacités des réseaux d'accès des équipements seront prédominantes dans les performances de l'application, les réseaux intermédiaires entre les équipements étant suffisamment dimensionnés pour ne pas avoir d'influence.

Hypothèses sur l'application

Nous avons choisi pour cette étude d'étudier une application de partage de photos mentionnée dans le cas d'usage. Les utilisateurs font appel à deux fonctionnalités principales de cette application. La première fonctionnalité permet aux utilisateurs d'envoyer des fichiers images depuis leur équipement vers l'application. La seconde fonctionnalité permet à l'utilisateur de consulter depuis son équipement des images déposées sous forme d'albums. Les clients de l'application sont des membres de la communauté et accèdent donc à l'application à partir d'un des équipements participant à l'infrastructure participative.

Cette application est composée de différents micro-services interdépendants. Son architecture suit les principes de conception des micro-services énoncés par M. Fowler et J. Lewis dans l'article [36].

— L'architecture de l'application est dite «orientée service» : elle est composée de plusieurs services indépendants, exposant chacun une interface décrivant les fonctions de ce service qui peuvent être utilisées par d'autres services.
— Le couplage entre les services est faible : un service de l'architecture peut être remplacé par un autre, tant que ce dernier fournit les mêmes fonctions que celles utilisées par les autres services.
— Un micro-service est un composant logiciel simple qui prend en charge un nombre limité de fonctionnalités, généralement liées à un même type de données ou de traitement.
— En règle générale, un micro-service doit traiter chaque requête de manière indépendante et doit donc fonctionner sans conserver d'état. Les micro-services devant maintenir un état doivent rester peu nombreux et sont réservés aux fonctions assurant la persistance des données.

Cette application est une composition de plusieurs micro-services qui réalisent différentes fonctions de l'application. Nous avons identifié 4 fonctions applicatives pour réaliser les 2 fonctionnalités utilisateurs mentionnées plus haut. Chaque fonction est réalisée par un micro-service que nous avons nommé dans l'architecture :

MetaHub (MH) Ce micro-service gère les métadonnées des images de l'application (format de l'image, utilisateur propriétaire, etc.). Il permet de les consulter à travers une requête `GET` et de stocker de nouvelles valeurs par une requête `POST`.

PhotoHub (PH) Ce micro-service permet d'accéder aux données binaires d'une image (bitmap) par une requête `GET`, et de stocker les données d'une nouvelle image par une requête `POST`.

ThumbHub (TH) Ce micro-service permet de récupérer, par une requête `GET` indiquant l'identifiant d'une image, une version de taille réduite (vignette) pouvant être intégrée dans un album.

WebUI (UI) Ce micro-service renvoie à travers une requête `GET` une page HTML contenant les images sous forme d'une galerie. Il permet aussi aux utilisateurs de déposer une image à travers une requête `POST`.

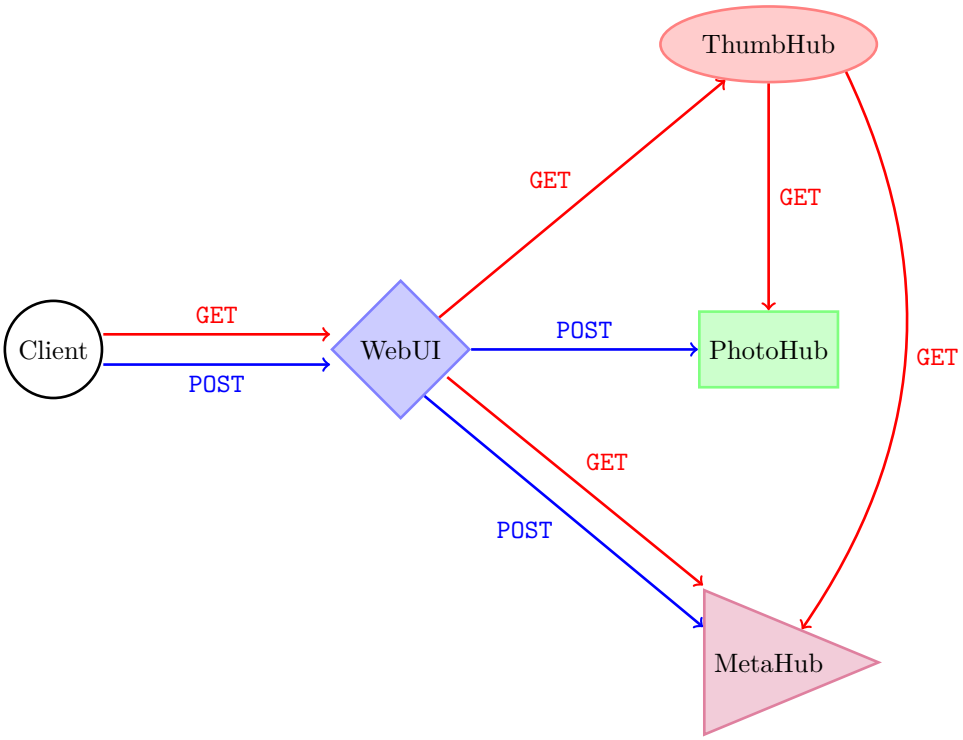

FIGURE 3.2 – Graphe des appels entre services de l'application de partage de photos.

Ces micro-services sont interdépendants : certains micro-services ont besoin pour réaliser leur tâche de faire appel à travers le réseau aux interfaces proposées par d'autres micro-services. Nous avons défini les appels entre les micro-services nécessaires à la réalisation des fonctionnalités de l'application sous la forme d'un graphe de dépendance illustré dans la figure 3.2. Chaque flèche de cette figure représente un appel entre deux micro-services, son origine étant l'initiateur de la

requête. Chaque appel utilise une fonctionnalité de l'interface du service identifiée dans la figure par le mot-clé `GET` (consultation d'une donnée) ou `POST` (dépôt d'une donnée).

Ce graphe reste constant en cours d'exécution de l'application. Chaque appel à une des fonctionnalités va toujours nécessiter les mêmes tâches et donc générer les mêmes requêtes entre micro-services. Par exemple, l'appel à la fonction `GET` de l'application va toujours générer le même nombre de génération de vignettes. Nous considérons de plus que les tailles des photos et des vignettes restent constantes. Chaque appel à une même fonctionnalité va donc engendrer des échanges de données en quantité similaire. Nous assurons ainsi que l'application, selon un même placement et dans des conditions stables d'utilisation des ressources de calcul et du réseau, aura un temps de réponse constant et reproductible pour chaque appel à ses fonctionnalités.

Pour que cette application soit fonctionnelle, il est nécessaire que chaque micro-service soit déployé au moins une fois sur l'infrastructure. Nous limiterons cependant notre étude au cas où chaque micro-service n'est déployé qu'une seule fois dans l'infrastructure. Le gestionnaire assure la mise à disposition du code et des données nécessaires pour chaque micro-service sur l'équipement choisi pour son placement. Une fois activé, chaque micro-service est visible des autres services déployés et peut communiquer avec eux. Il n'existe pas de dépendance entre les services et les équipements : le placement des services est libre et l'application sera entièrement fonctionnelle quelque soit le placement choisi.

3.1.3 Problématique du placement initial

Nous avons décrit dans le cas d'usage la présence d'un système permettant de déployer et d'interconnecter des micro-services sur les équipements afin d'activer une application. Au sein de ce système, une phase préliminaire au déploiement consiste à choisir sur quel équipement chacun des micro-services devra être placé. Comme nous allons le voir, ce choix sera déterminant pour les performances de l'application.

Indicateur de performance retenu

Nous avons choisi pour cette étude de considérer le temps de réponse comme indicateur principal pour caractériser les performances de l'application. Dans notre cas d'usage, l'hébergement participatif de l'application est certes motivé par la protection des données à caractère personnel. Cependant cet argument aura peu de valeur auprès des utilisateurs si l'application offre une expérience dégradée par rapport aux solutions gratuites. La qualité d'expérience (QoE) est une métrique évaluée subjectivement par les utilisateurs finaux, utilisée par exemple dans le domaine

du codage vidéo pour caractériser l'impact de la transmission d'une vidéo sur son visionnage. Cette métrique subjective peut être corrélée à la qualité de service, qui est une métrique objective se définissant par plusieurs paramètres mesurables comme le temps de réponse, la disponibilité ou la fiabilité du service. Parmi ces paramètres, le temps de réponse a une influence importante sur l'expérience des utilisateurs [32].

Le choix du temps de réponse comme indicateur de performance a un double intérêt pour notre étude : il est quantifiable et décomposable. Le temps de réponse de l'application se mesure concrètement comme l'intervalle de temps entre l'émission par un client d'une requête vers l'application et la réception de la réponse à cette requête par ce même client. Il est donc possible de comparer les expériences utilisateurs induites par différentes configurations du système en comparant les temps de réponse mesurés pour l'application. Il est de plus possible de décomposer ce temps de réponse en plusieurs intervalles de temps : le délai d'acheminement de la requête entre le client et l'application, le temps de traitement nécessaire à l'application pour le calcul de la réponse et enfin le délai d'acheminement de cette réponse entre l'application et le client. Dans l'objectif de diminuer le temps de réponse, cette décomposition nous sera utile par la suite pour identifier les intervalles de temps pouvant être optimisés.

Incidence du placement sur le temps de réponse

Le temps de réponse de l'application à une requête utilisateur dépend de plusieurs paramètres. Notre objectif étant de réduire ce temps de réponse, il nous faut identifier les paramètres sur lesquels nous pouvons agir dans notre contexte. Nous avons identifié que ce temps de réponse peut se décomposer en un temps nécessaire pour le traitement de la requête et deuxièmement dans le temps d'acheminement des données (la requête puis sa réponse) entre le client et l'application. Le temps de traitement dépend des besoins de calcul de l'application et des caractéristiques de processeur de l'application. Le temps d'acheminement dépend lui de la quantité de données et des caractérisques du réseau (bande passante, latence) entre l'application et le client. Les besoins de traitement et de transfert de données de l'application sont des données initiales de notre problème, car celle-ci nous est imposée. Les capacités processeur et réseau dans une infrastructure participative sont elles aussi fixées au départ. Il n'est pas possible de les modifier comme dans un centre de données avec une infrastructure réseau complètement maitrisée, et des capacités de calcul dimensionnables.

L'infrastructure participative proposant plusieurs équipements avec des capacités de calcul et de communication différentes, il est par contre possible d'influer sur le temps de réponse en choisissant un placement adéquat de l'application. Dans le cas d'une application monolithique, c'est à

Service	Temps de traitement par requête (ms)	Impact des ressources CPU	Taille d'une requête (octets)	Impact de la BP descendante	Taille d'une réponse (octets)	Impact de la BP montante	Impact de la latence
WebUI (GET)	100	Faible	100	Faible	1M	Important	Faible
WebUI (POST)	50	Faible	5M	Important	50	Faible	Faible
Metahub (GET)	50	Faible	50	Faible	1000	Faible	Important
Metahub (POST)	50	Faible	200	Faible	50	Faible	Important
Photo-Hub (GET)	50	Faible	50	Faible	5M	Important	Faible
Photo-Hub (POST)	50	Faible	5M	Important	50	Faible	Faible
Thumb-Hub (GET)	500	Important	50	Faible	200K	Modéré	Modéré

TABLE 3.1 – Impact des ressources de calcul et des caractéristiques des réseaux sur le temps de réponse de chacun des services.

dire non-distribuée, il s'agira de trouver le meilleur compromis entre les capacités disponibles sur un équipement et les besoins de l'application. Cette stratégie de placement devient cependant plus complexe pour une application distribuée sous forme de micro-services. Il faut ici considérer que les besoins de calcul et de communication sont différents entre chaque micro-service. Le service MH de notre application, par exemple, répond à des requêtes de recherches dans des données stockées dans une base de données. Le traitement nécessaire est moins demandeur en calcul que pour le service TH qui lui applique un algorithme de réduction de taille d'image afin de générer une vignette. De même les données échangées entre le service UI et MH sont de faible volume (données textuelles) comparées à celles échangées entre le service PH et UI (images). Il doit être noté ici que le sens de ces échanges est aussi important à considérer, et dépend de la requête et du service.

L'hétérogénéité des équipements et des réseaux utilisés dans une infrastructure participative implique donc que le temps de réponse d'un micro-service sera différent selon l'emplacement choisi pour son hébergement. Il est possible d'avoir une première estimation de l'influence de la disponibilité des ressources de calcul et de réseau sur les performances de chaque service en s'appuyant sur les caractéristiques des micro-services. Le tableau 3.1 donne ainsi, pour chaque service de notre application, un ordre de grandeur pour le temps de traitement moyen d'une requête ou les quantités de données transportées. Ces caractéristiques sont issues d'un profilage de l'application en conditions réelles. Nous pouvons ainsi identifier les ressources importantes pour les performances de chaque micro-service. Cette estimation n'est cependant pas suffisante pour déterminer un placement optimal de l'application. Il nous faudrait pour cela des mesures précises des performances de chaque services dans les différentes conditions d'utilisation que nous pourrions rencontrer.

Paramètres pour le processus de placement

L'objectif de cette étude est donc d'identifier une méthode permettant de choisir un emplacement pour chacun des micro-services de l'application minimisant le temps de réponse global de l'application pour l'ensemble des clients. Ce processus de placement des micro-services dépendra de plusieurs paramètres que nous avons déjà évoqués et que nous résumons ici en les rapportant aux trois objets du problème étudié.

Objet 1 : l'application et ses micro-services : Le processus de placement a besoin d'une description préalable des différents micro-services qui composent l'application. Un profilage du code de chaque micro-service est nécessaire pour connaitre ses besoins spécifiques en ressources de calcul et en communication sur le réseau. Le graphe des appels entre les micro-services pour chaque requête de l'application permet d'identifier les interfaces des micro-services impliquées dans le traitement, ainsi que l'émetteur et destinataire de chaque appel.

Objet 2 : l'infrastructure d'hébergement : Parmi les différentes caractéristiques des équipements de l'infrastructure participative, nous avons d'abord retenu la puissance du processeur, car elle est déterminante dans le temps de traitement des requêtes par les micro-services. Les paramètres retenus pour caractériser le réseau résidentiel qui connecte chaque équipement au reste de l'infrastructure sont la bande passante et la latence. Étant donné les types de réseau utilisés, nous différencierons bande passante montante (de l'équipement vers l'infrastructure) et descendante (de l'infrastructure vers l'équipement)

Objet 3 : le critère de performance : Le critère de performance retenu est celui du temps de réponse. Il correspond aux temps d'acheminement et de traitement d'une requête d'un client à

l'application. L'application pouvant répondre à différentes requêtes (requêtes `GET` et `POST` dans le cas de l'application de partage de photos), une pondération entre le temps de réponse de chaque requête sera donnée pour calculer le temps de réponse de l'application pour un client. Le temps de réponse est évalué pour un ensemble défini de clients de l'application. Nous définirons comment obtenir un temps de réponse global de l'application à partir des temps mesurés pour chaque client.

3.2 Etat de l'art des solutions de placement

Le placement de service est un sujet très étudié pour la gestion des ressources dans les centres de données (ou *datacenter*) dans un contexte de *cloud*. Dans le modèle d'infrastructure à la demande (IaaS), l'utilisateur souhaite disposer d'un ou plusieurs machines virtuelles avec un niveau de service (temps de réponse, fiabilité) garanti (SLA). L'objectif des techniques de placement est de trouver une association entre les machines virtuelles et les machines physiques du centre de données qui garantisse à la fois le niveau de service demandé par l'utilisateur, et les exigences du gestionnaire du centre de données en termes de rentabilité économique et d'efficacité énergétique [93]. Les nombreux travaux sur le sujet montrent qu'il s'agit d'une problématique complexe pour laquelle sont proposées plusieurs techniques de recherche de solutions de placement : programmation par contraintes, programmation intégrale linéaire ou algorithmes évolutionnaires [97].

Ces études s'appliquant dans le contexte d'un même *datacenter*, nous remarquons que la localisation des équipements dans le réseau a généralement peu d'influence dans les stratégies de placement, les caractéristiques du réseau étant plutôt homogènes dans un même centre. La topologie du réseau entre plus en ligne de compte dans des scénarios impliquant plusieurs *datacenters* géographiquement distribués (*Multi-Cloud* [40] [69] ou *Federated Clouds* [5]). Les techniques de placement cherchent alors un compromis entre l'utilisation de plusieurs *datacenters* (résilience, indépendance vis-à-vis du fournisseur) et les communications entre les différents centres, qui impactent fortement les performances du traitement des applications. La localisation de l'utilisateur devient un paramètre du placement d'une machine machine virtuelle sur le centre de données le plus proche afin de minimiser les temps de communications avec le client [87] [91]. Le placement s'effectue alors avec une précision proche de l'échelle géographique d'un pays ou, dans la topologie réseau, de celle de l'opérateur.

Les travaux s'intéressant au placement au plus proche de l'utilisateur se retrouvent dans le domaine du *Edge Computing* où les équipements se situent dans les mêmes réseaux d'accès que ceux des utilisateurs. Ce contexte présente donc des similarités avec notre domaine d'étude, de

part l'hétérogéneïté des équipements et les capacités des réseaux. Les travaux sur les Micro-Clouds [81] [80] présentent un cas d'usage, très similaire au nôtre, de placement de services dans une infrastructure communautaire. Les solutions proposées ne s'appliquent cependant qu'à une application monolithique. Nous nous sommes donc focalisés sur les travaux concernant le placement d'applications distribuées entre les équipements. Les auteurs de [13] proposent une modélisation du problème de placement d'applications multi-composants dans le contexte du Mobile Edge Cloud (MEC) sous une forme très générique, incluant la mobilité des utilisateurs. Or la topologie du réseau est ici représentée par un espace à deux dimensions, ce qui est assez éloigné de la réalité. Les solutions orientées *Fog Computing*, comme [92], modélise la topologie réseau sous une forme hiérarchique à plusieurs niveaux, alors que dans notre cas, la topologie est un réseau totalement connecté. Les travaux concernant les micro-services, comme [77] et [50] utilisent des modèles d'application et de topologie réseau proches de notre cas, mais on pour objectif l'efficacité énergétique et une réduction du trafic réseau.

Cet état de l'art des solutions de placement dans le contexte du *cloud* nous permet de conclure que d'une part, dans le contexte d'un *datacenter* unique, le coût des communications par le réseau est généralement considéré comme nul. Dans le contexte d'un système avec plusieurs *datacenters*, les solutions de placement n'utilisent, quant à elles, le réseau qu'en dernier ressort. Les solutions pour rapprocher l'application de l'utilisateur n'ont qu'une précision très relative. D'autre part, les solutions orientées *Edge Computing*, qui prennent en compte les réseaux d'accès, utilisent des modèles d'applications ou de topologies réseaux plutôt éloignées de celles qui nous intéressent ici.

3.3 Modélisation du temps de réponse

Le processus de placement utilise les paramètres identifiés dans le paragraphe 3.1.3 pour évaluer différentes solutions de placement et déterminer ainsi celle qui optimise l'indicateur de performance choisi. Dans notre cas, la solution recherchée est celle qui minimise le temps de réponse global de l'application pour l'ensemble des clients. Nous proposons dans cette section un modèle de ce temps de réponse selon un placement donné utilisant ces paramètres.

3.3.1 Temps de réponse de l'application

Notre formulation du temps de réponse de l'application s'appuie sur sa décomposition en différents temps de réponse de chaque micro-service. Le graphe des appels entre les micro-services de l'application nous permet de décomposer une requête d'un client en plusieurs requêtes nécessaires à son traitement. Nous utiliserons ici la représentation de ces appels sous forme d'un diagramme de séquence, souvent utilisée pour présenter les échanges entre composants logiciels. Cette représentation met en évidence l'enchainement des appels entre les différents services. Si l'ordonnancement des appels ne nous est pas utile ici, ce diagramme nous permet par contre d'identifier les différents délais à considérer dans la décomposition du temps de réponse.

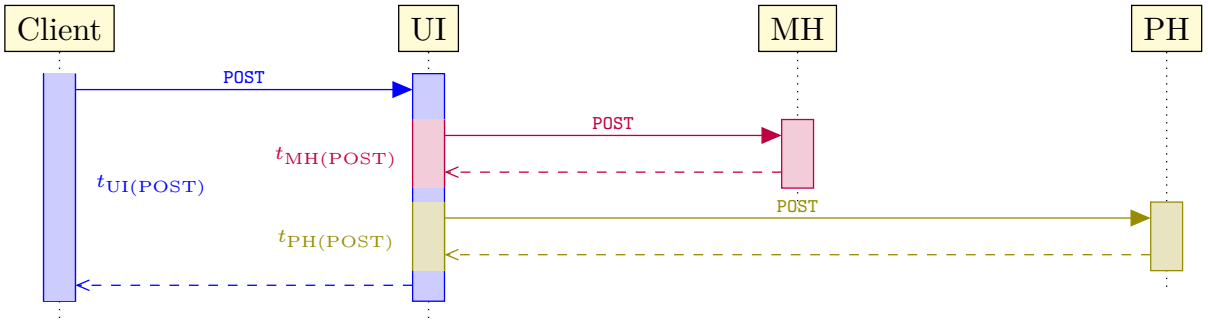

FIGURE 3.3 – Diagramme de séquence pour le traitement d'une requête POST.

La figure 3.3 représente sous forme d'un diagramme de séquence le traitement par l'application d'une requête `POST`, correspondant au téléversement d'une photo dans l'application. Cette requête est émise par le client à destination du micro-service UI. La requête POST contient les données binaires de l'image fournie par le client. Pour le traitement de cette requête, ce micro-service communique avec le micro-services MH pour y enregistrer les méta-données associées à l'image, puis avec le micro-service PH pour stocker les données binaires. Une fois l'image enregistrée, le service UI répond au client que sa requête a été correctement traitée.

Le temps nécessaire pour le traitement d'une requête est représenté sur l'axe de temps de son émetteur par l'intervalle entre la flèche correspondant à l'émission de la requête et celle correspondant à la réception de la réponse. Nous utilisons dans cette figure la notation $t_{s(q)}$ pour représenter le temps de réponse du service s pour la requête q. Ainsi le temps de réponse du micro-service UI pour le traitement de la requête POST est représenté sur l'axe de temps du client par l'intervalle noté $t_{\mathrm{UI(POST)}}$. Nous supposons à partir d'ici que l'application n'est pas chargée : dans le temps nécessaire au traitement de la requête, aucune autre requête d'utilisateur n'est adressée à l'application.

Il est possible de décomposer ce temps de réponse en plusieurs délais : le délai d'acheminement et le délai de traitement. Le délai d'acheminement, que nous noterons d_a, n'est pas représenté sur les axes de temps du diagramme de séquence et correspond au temps nécessaire à la requête `POST` et à sa réponse de traverser le réseau entre le client et le service UI. Le délai de traitement, représenté sur l'axe UI, se décompose en plusieurs délais : le délai de traitement propre au micro-service UI, que nous noterons d_t et les délais induits par les appels aux micro-services MH et PH, respectivement représentés sur la figure par les intervalles $t_{\text{MH(POST)}}$ et $t_{\text{PH(POST)}}$. Nous pouvons donc exprimer le temps de réponse du service UI pour cette requête de la manière suivante :

$$t_{\text{UI(POST)}} = d_a + d_t + t_{\text{MH(POST)}} + t_{\text{PH(POST)}}$$

Les délais d_a et d_t ont une propriété intéressante : ils ne dépendent que des capacités de calcul de l'équipement hébergeant le service UI et de celles du réseau entre cet équipement et le client. Cette propriété nous est importante pour pouvoir modéliser le temps de réponse de l'application en fonction des différentes caractéristiques de l'infrastructure participative et du placement des services. Nous proposons de regrouper ces deux valeurs pour chaque micro-service i sous la notion de temps de réponse spécifique du micro-service que nous noterons r_i. Il correspond au temps nécessaire pour acheminer la requête et sa réponse, auquel s'ajoute le temps de calcul nécessaire pour le traitement interne de la requête par le micro-service.

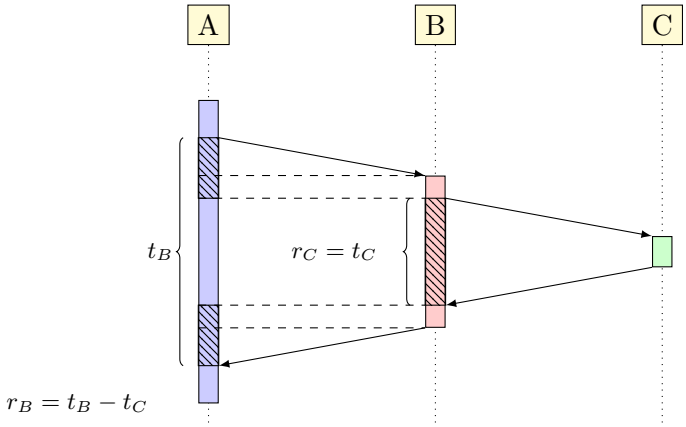

FIGURE 3.4 – Représentation du temps de réponse spécifique.

Dans ce temps de réponse spécifique, nous n'incluons pas les temps induits par les appels du micro-service vers d'autres micro-services. La figure 3.4 donne une représentation de cette valeur dans un chronogramme d'appels entre services. Le service B appelle le service C qui lui n'utilise aucun autre service. Le temps de réponse spécifique du service C r_C correspond alors au temps t_C mesuré depuis B entre l'envoi de la requête et la réception de la réponse. Il est représenté sur

l'axe de temps de B par les parties hachurées. Le service A fait appel au service B. Le temps de réponse spécifique de B r_B est le temps d'acheminement de la requête et de la réponse, ajoutés au temps de traitement interne de la requête par B. Ce temps est représenté sur l'axe de temps de A par les parties hachurées. Il peut se calculer en soustrayant t_C à t_B. Une propriété importante du temps de réponse spécifique est qu'il ne dépend uniquement que des ressources de calcul de l'équipement qui héberge le service et de la capacité du réseau entre cet équipement et le client.

Le temps de réponse du micro-service UI à la requête `POST` $t_{\text{UI(POST)}}$ se décompose comme le temps de réponse spécifique du micro-service UI à cette requête, ajouté aux temps de réponse des services MH et PH :

$$t_{\text{UI(POST)}} = r_{\text{UI(POST)}} + t_{\text{MH(POST)}} + t_{\text{PH(POST)}}$$

Les services MH et PH ne faisant appel à aucun autre micro-service pour cette requête, leurs temps de réponse sont équivalents aux temps de réponse spécifiques :

$$t_{\text{MH(POST)}} = r_{\text{MH(POST)}}$$
$$t_{\text{PH(POST)}} = r_{\text{PH(POST)}}$$
$$t_{\text{UI(POST)}} = r_{\text{UI(POST)}} + r_{\text{MH(POST)}} + r_{\text{PH(POST)}}$$

Nous avons ainsi décomposé le temps de réponse de l'application à la requête `POST` en trois temps de réponse spécifiques de chaque service impliqué. Ces valeurs dépendent chacune du placement relatif de l'emetteur et du destinataire des échanges entre service.

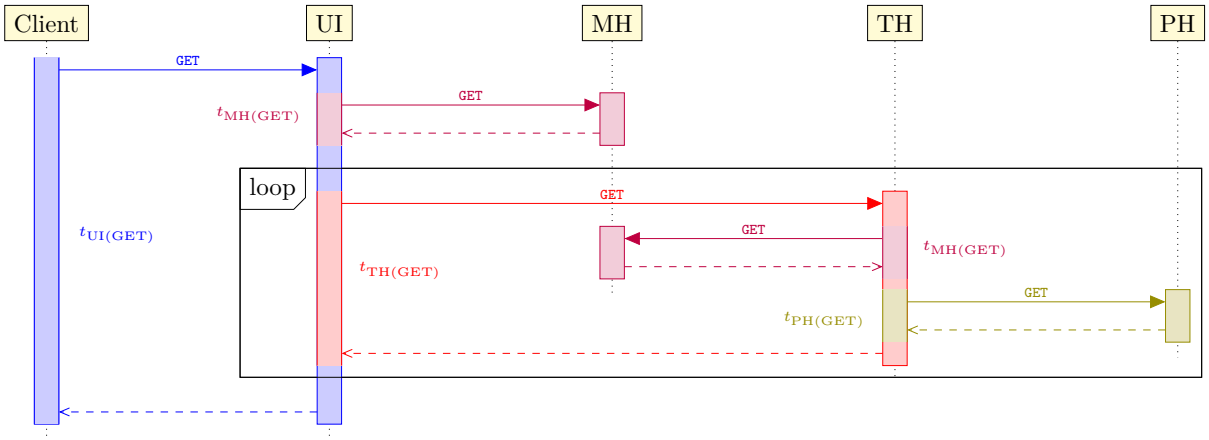

FIGURE 3.5 – Diagramme de séquence pour le traitement d'une requête `GET`.

La figure 3.5 donne le diagramme de séquence pour le traitement de la requête `GET`, correspondant à l'affichage d'une galerie de photos. Ce traitement est plus complexe et fait apparaitre plusieurs appels entre micro-services. Le client envoie la requête au service UI qui s'appuie sur les services

MH et TH pour obtenir les données nécessaires à l'affichage de la page : informations sur les images et vignettes correspondantes. Le traitement de cette requête nécessite autant d'appels au service TH que de vignettes nécessaires, appels représentés dans la figure sous forme d'une boucle. Le nombre de vignettes nécessaires pour une galerie sera fixé par la variable k. A partir de ce schéma d'appel, nous pouvons décomposer le temps de réponse du service UI pour le traitement de cette requête, noté $t_{\text{UI(GET)}}$, comme le temps de réponse spécifique du service UI pour cette requête, auquel s'ajoute le temps de réponse du service MH et k fois le temps de réponse du service TH :

$$t_{\text{UI(GET)}} = r_{\text{UI(GET)}} + t_{\text{MH(GET)}} + k \cdot t_{\text{TH(GET)}}$$

Le service TH faisant appel aux micro-services MH et PH, son temps de réponse se décompose comme son propre temps de réponse spécifique ajouté des temps de réponse de ces micro-services :

$$t_{\text{TH(GET)}} = r_{\text{TH(GET)}} + t_{\text{MH(GET)}} + t_{\text{PH(GET)}}$$

Les micro-services MH et PH ne faisant pas appel à d'autres micro-service pour traiter la requête `GET`, leur temps de réponse est équivalent au temps de réponse spécifique. Le temps de réponse du service UI pour le traitement de cette requête s'écrit donc maintenant :

$$t_{\text{UI(GET)}} = r_{\text{UI(GET)}} + r_{\text{MH(GET)}} + k \cdot (r_{\text{TH(GET)}} + r_{\text{MH(GET)}} + r_{\text{PH(GET)}})$$

Le temps de réponse de l'application se calcule ensuite par une somme pondérée du temps de réponse pour chaque requête. La pondération appliquée à chaque requête correspond à la part de cette requête dans l'usage de l'application. Pour rappel, nous donnons ici une décomposition du temps de réponse pour un client. Il est donc possible ici d'introduire pour chaque client une pondération différente correspondant à un usage spécifique de l'application. À travers cette différenciation, il nous est possible de distinguer des utilisateurs qui utilisent l'application pour de la consultation (requêtes `GET`) de ceux y déposant du contenu (requête `POST`). Le temps de réponse de l'application observé par un client c, noté $t_{c,\text{UI}}$, s'exprime alors par la formule suivante, où $w_{c,q}$ correspond à la pondération pour le client c de la requête $q \in \text{GET}, \text{POST}$.

$$t_{c,\text{UI}} = w_{c,\text{GET}} \cdot t_{\text{UI(GET)}} + w_{c,\text{POST}} \cdot t_{\text{UI(POST)}}$$

Le temps de réponse global de l'application se calcule ensuite à partir de l'ensemble des temps de réponse observés par les clients. Plusieurs valeurs statistiques peuvent caractériser la distribution de cet ensemble : valeur moyenne, valeur médiane, p-ième centile, ... Le choix de la fonction aboutira plus tard à une optimisation différente du placement. L'administrateur du système configurera ici la méthode qu'il jugera la plus adéquate pour son système. Nous représentons

par la fonction f la représentation statistique choisie. Le temps de réponse global de l'application pour l'ensemble des clients C s'exprime donc ainsi :

$$t_{\mathrm{UI}} = f(\{t_{c,\mathrm{UI}} | \forall c \in C\})$$

3.3.2 Temps de réponse par service

Nous avons mis en évidence que le temps de réponse de l'application peut se décomposer en différents temps de réponse spécifiques des micro-services composant l'application. Ces temps de réponses dépendent du placement de l'émetteur de la requête et du service. Nous avons ici besoin d'un modèle pour prédire ce temps de réponse en fonction de ces placements. Un tel modèle se rapporte à une réalité complexe, dû à l'hétérogénéité des services, des équipements et des réseaux. Cette complexité se concrétise par un nombre important de paramètres en entrée du modèle. Pour que ces modèles puissent donner des réponses proches de la réalité, il est nécessaire que chacun de ces paramètres représentent une valeur exacte ou très approchante du problème étudié. Plus le nombre de paramètre est important, plus cette précision est difficile à obtenir.

Cette complexité à définir un modèle de temps de réponse incluant l'ensemble des paramètres influents s'est posée aux auteurs de [68]. Afin de résoudre un problème de placement de service, ils ont proposé une solution utilisant comme donnée initialement connue les temps de réponse d'un même service sur les différents équipements possibles. Cette approche nous semble intéressante pour limiter la complexité de notre problème de placement. Dans notre cas, il est cependant nécessaire de considérer, à la fois l'équipement hébergeant le service, mais aussi l'équipement qui interroge le service. Pour chaque micro-service de l'application s, est donc connue au départ la valeur du temps de réponse spécifique pour une requête q, pour tous les couples d'équipements i, j :

$R_{s(q)}(i,j)$: Temps de réponse mesuré par le client i du service s hébergé sur j pour la requête q

Chaque valeur $R_{s(q)}(i,j)$ dépend des ressources de calcul présentes sur l'équipement j et de la capacité du réseau entre i et j. Les différentes valeurs $R_{s(q)}(i,j)$ nous fournissent donc un modèle nous permettant de prédire le temps de réponse en fonction du placement de ces deux équipements i et j. Les valeurs rendues par cette fonction peuvent être des résultats de mesures préalables ou alors des valeurs synthétiques issues d'un modèle de prédiction du temps de réponse. Nous discuterons par la suite de l'importance de la provenance de ces valeurs.

Il est possible de représenter les valeurs de la fonction $R_s(i,j)$ sous forme d'une matrice carrée ayant pour dimension le nombre d'équipements présents dans l'infrastructure participative. La valeur positionnée dans cette matrice aux coordonnées (i,j) correspondra au temps de réponse du service entre les équipements i et j. Les valeurs de la diagonale de cette matrice seront plus faibles relativement aux autres valeurs car elles correspondent à un temps de réponse du service hébergé sur le même équipement que le client.

$$R_{s(q)} = \begin{bmatrix} 5 & 201 & 250 & 182 & \dots \\ 159 & 6 & 250 & 197 & \dots \\ 354 & 145 & 8 & 165 & \dots \\ 243 & 252 & 127 & 4 & \dots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix}$$

Pour appliquer ce modèle dans la décomposition du temps de réponse de l'application à une requête, il nous faut mettre en évidence dans la formule les équipements qui hébergent chacun des services. Nous nommerons x_s l'équipement hébergeant le service s. La décomposition de $t_{UI(POST)}$ donnée précédement devient alors pour le client c :

$$t_{c,\mathrm{UI(POST)}} = R_{\mathrm{UI(POST)}}(c, x_{\mathrm{UI}}) + R_{\mathrm{MH(POST)}}(x_{\mathrm{UI}}, x_{\mathrm{MH}}) + R_{\mathrm{PH(POST)}}(x_{\mathrm{UI}}, x_{\mathrm{PH}})$$

Il devient dès lors possible d'exprimer la recherche d'un placement minimisant ce temps de réponse sous la forme d'un problème d'optimisation : connaissant les différentes valeurs possibles de $R_{\mathrm{UI(POST)}}$, $R_{\mathrm{MH(POST)}}$ et $R_{\mathrm{PH(POST)}}$, trouver le vecteur $(x_{\mathrm{UI}}, x_{\mathrm{MH}}, x_{\mathrm{PH}})$ minimisant la valeur de $t_{c,\mathrm{UI(POST)}}$ pour l'ensemble des clients c.

3.3.3 Formalisation de l'expression du temps de réponse

Nous allons dans cette section formaliser l'expression du temps de réponse global de l'application dans un cas général d'application afin d'obtenir la fonction objectif du problème d'optimisation.

Soit les variables suivantes :

— N l'ensemble des équipements composant l'infrastructure participative,
— C l'ensemble des équipements clients de l'application ($C \subset N$),
— S l'ensemble des services composant l'application,
— Q l'ensemble des requêtes possibles pour l'application,
— $D_q(i,j)$ la matrice donnant le nombre d'appels du service i au service j pour la requête q,

— $R_{s(q)}(m, n)$ le temps de réponse spécifique du service s pour le traitement de la requête q envoyée par l'équipement m au service hébergé sur l'équipement n,
— $P = (p_s, \forall s \in S)$ le vecteur représentant les placements choisis pour les services s ($p_s \in N$)
— $w_{c,q}$ la pondération de la requête q pour le client c
— f la fonction donnant le temps de réponse global de l'application à partir des temps de réponse de chaque client

Nous commençons par donner l'expression 3.1 du temps de réponse $t_{x,s(q)}(P)$ d'un service s pour une requête q émise depuis un équipement x en fonction du placement P des services. Ce temps de réponse est exprimé sous une forme récursive afin de parcourir le graphe des dépendances entre micro-services, spécifié par la matrice D_q. Le premier terme $R_{s(q)}(x, P(s))$ correspond au temps de réponse spécifique du service s pour la requête q entre les équipements x et $P(s)$. Le second terme est la somme des temps de réponse des services k sollicités depuis l'équipement $P(s)$. Cette récursion est possible si le graphe d'appel des micro-services pour le traitement de chaque requête est acyclique.

$$t_{x,s(q)}(P) = R_{s(q)}(x, P(s)) + \sum\nolimits_{k \in S} D_q(s, k) \cdot t_{P(s),k(q)}(P) \tag{3.1}$$

Nous utilisons cette formule pour exprimer le temps de réponse de l'application pour un client c et une requête q. Nous avons ici besoin d'identifier le service α qui est le point d'entrée de l'application pour les requêtes des clients. Dans le cas de l'application de partage de photo, ce service est UI. Le temps de réponse de l'application observé par le client c, qui sollicite le service α pour la requête q, s'exprime alors ainsi :

$$t_{c,\alpha(q)}(P) = R_{\alpha(q)}(c, P(\alpha)) + \sum\nolimits_{k \in S} D_q(\alpha, k) \cdot t_{P(\alpha),k(q)}(P) \tag{3.2}$$

Le temps de réponse de l'application $t_\alpha(P)$ pour un client c en fonction du placement P s'exprime comme la somme pondérée du temps de réponse des différentes requêtes. Le temps de réponse global de l'application $t_\alpha(P)$ en fonction du placement P se calcule en applicant la fonction f, caractérisant la distribution des temps de réponse pour tous les clients $c \in C$:

$$\begin{aligned} t_{c,\alpha}(P) &= \sum\nolimits_{q \in Q} w_{c,q} \cdot t_{c,\alpha(q)}(P) \\ t_\alpha(P) &= f(\{t_{c,\alpha}(P) | \forall c \in C\}) \end{aligned} \tag{3.3}$$

Dans le cas de notre application de partage de photos, nous pouvons retrouver les formules exprimées dans la section 3.3.1 en définissant les valeurs des variables suivantes. L'ensemble des services composant l'application est défini par :

$$S = \{\alpha, \beta, \gamma, \delta\}$$

où α correspond au service UI, β au service MH, γ au service PH et δ au service TH.

Les appels entre services sont définis pour les deux requêtes `GET` et `POST` par les valeurs données pour D_{GET} et D_{POST} :

$$D_{\text{GET}}(i,j) = \begin{cases} 1, & \text{si } i = \alpha, j = \beta \\ n, & \text{si } i = \alpha, j = \delta \\ 1, & \text{si } i = \delta, j = \beta \\ 1, & \text{si } i = \delta, j = \gamma \\ 0, & \text{sinon} \end{cases} \qquad D_{\text{POST}}(i,j) = \begin{cases} 1, & \text{si } i = \alpha, j = \beta \\ 1, & \text{si } i = \alpha, j = \gamma \\ 0, & \text{sinon} \end{cases}$$

où n est le nombre de vignettes à générer par TH pour une galerie.

Les temps de réponse observés par un client c pour les requêtes `GET` et `POST` s'expriment alors de cette façon :

$$t_{c,\alpha(\text{GET})}(P) = R_{\alpha(\text{GET})}(c, P(\alpha)) + R_{\beta(\text{GET})}(p_\alpha, P(\beta)) + k \cdot [R_{\delta(\text{GET})}(P(\alpha), P(\delta)) + R_{\beta(\text{GET})}(P(\delta), P(\beta)) + R_{\gamma(\text{GET})}(P(\delta), P(\gamma))]$$

$$t_{c,\alpha(\text{POST})}(P) = R_{\alpha(\text{POST})}(c, P(\alpha)) + R_{\beta(\text{POST})}(P(\alpha), P(\beta))) + R_{\gamma(\text{POST})}(P(\alpha), P(\gamma))$$

L'expression du temps de réponse de l'application pour l'ensemble des requêtes et des clients, donnée par la formule 3.3, reste inchangée. Le système devra définir pour chaque client c les poids $w_{c,\text{GET}}$ et $w_{c,\text{POST}}$ afin de préciser pour chacun l'usage des requêtes `GET` et `POST`.

3.4 Optimisation du temps de réponse

Nous venons de définir un modèle pour le temps de réponse de l'application en fonction du placement et du temps de réponse spécifique des micro-services. Nous allons maintenant utiliser ce modèle pour définir le problème d'optimisation et proposer des heuristiques de recherche de solutions.

3.4.1 Définition du problème d'optimisation

Contrainte sur le placement

Le temps de réponse pour un client c est une somme pondérée sur l'ensemble des requêtes $q \in Q$ du temps de réponse $t_{c,\alpha(q)}$ exprimé par la formule 3.2. Nous pouvons distinguer dans cette formule que ce temps de réponse est une somme de deux termes. Le premier correspond au temps de réponse spécifique du service α observé par le client c : $R_{\alpha,q}(c, p_\alpha)$. Le second terme correspond à la somme des temps de réponse pour les appels entre les micro-services de l'application. Un placement optimal représentera une solution qui minimisera aussi bien le premier terme, par un placement optimal du service α par rapport aux clients, et le second terme, par un placement optimal des micro-services de l'application les uns par rapport aux autres.

Le lecteur pourra faire remarquer ici qu'il est possible de réduire considérablement la valeur exprimée par le second terme de l'équation en colocalisant des micro-services de l'application sur un même équipement. Les appels entre ces micro-services ne nécessitant plus de communication sur le réseau, les temps de réponse seraient alors très faibles, correspondant aux valeurs sur la diagonale des matrices $R_{s(q)}$. Or nous savons dans notre cas que la colocalisation peut être problématique si les équipements sont peu fiables. L'efficacité de cette stratégie dépend de plus des ressources disponibles sur l'équipement.

Afin d'assurer une distribution suffisante des services et éviter une co-localisation trop importante, nous contraignons la solution de placement à respecter un nombre maximum m de services hébergés sur un même équipement. Soit S_x l'ensemble des services hébergés sur l'équipement x, alors quelque soit l'équipement $x \in N$, l'ensemble S_x ne comporte alors pas plus de m éléments. Dans le cas où $m = 1$, un seul micro-service pourra être présent sur un équipement. L'ensemble des appels entre micro-services nécessiteront des communications à travers le réseau. Dans les cas où $m > 1$, certaines co-localisations entre services seront autorisées.

Fonction objectif

Nous appellerons $P^*(s)$ le placement des micro-services qui permet d'obtenir un temps de réponse optimal pour l'application. Trouver ce placement consiste à trouver les valeurs $P^*(s) \in N;\ s \in S$ qui minimisent le temps de réponse $t_\alpha(P)$ exprimé dans notre modèle par la formule 3.3. Les contraintes suivantes s'appliquent sur le placement en donnant les valeurs possibles de chaque $P(s)$ ainsi que le nombre maximal m de services hébergés sur chaque équipement x.

$$\begin{aligned} \underset{P}{\text{minimize}} \quad & t_\alpha(P) \\ \text{subject to} \quad & t_\alpha(P) = f(\{t_{c,\alpha}(P) | \forall c \in C\} \\ & P(s) \in N \\ & x \in N, |S_x| \leq m, S_x = \{s \in S | P(s) = x\} \end{aligned}$$

La fonction f donne une valeur représentative (moyenne, valeur médiane, p-ième centile) de la distribution des temps de réponses mesurés par chaque client. Selon le choix de cette fonction, l'optimisation du temps de réponse de l'application aura un résultat différent en terme de qualité d'expérience des utilisateurs. Si la fonction f choisie est la moyenne arithmétique, l'optimisation du temps de réponse cherchera à minimiser la valeur moyenne du temps de réponse des utilisateurs. Si cette fonction est le 95ème centile, l'optimisation va minimiser la valeur maximale du temps de réponse mesurée par 95% des clients. Le choix de cette fonction appartient à l'administrateur du système selon son cas d'usage. Dans notre cas, nous avons choisi d'utiliser la moyenne arithmétique.

3.4.2 Heuristiques de recherche de solutions

Le nombre de placements possibles des services S sur l'ensemble des équipements N est égal à $|N|^{|S|}$. Les contraintes de colocation de services sur un même équipement réduisent quelque peu ce nombre, mais pas de manière significative. Un algorithme de recherche exhaustive d'un placement optimal aura donc une complexité polynomiale selon le nombre d'équipements et exponentielle selon le nombre de services à placer. Il est intéressant de noter que la complexité du problème est constante selon le nombre d'appels entre les services de l'application, ces appels n'étant considérés que pour la définition de la fonction objectif du problème.

Nous proposons deux heuristiques capables de trouver des solutions approchées au problème de placement des micro-services. Nous allons comparer ces deux heuristiques selon l'approximation

qu'elles donnent du résultat optimal, leur fiabilité ainsi que leur temps d'exécution. Ces critères nous permettront de choisir une heuristique pour la suite de nos travaux.

Algorithme glouton

La première heuristique cherche à trouver un placement optimal de chaque service de manière incrémentale. Elle se base sur le constat déjà exposé que le temps de réponse de l'application, défini par la formule 3.2, se décompose en deux termes : un premier terme correspondant au temps de réponse du premier micro-service α avec les clients $c \in C$, le second terme correspondant aux temps de réponse entre les micro-services de l'application. L'idée de l'algorithme glouton est d'abord de trouver un placement du premier micro-service α qui minimise le premier terme, puis sur la base de ce premier placement, trouver un placement pour chacun des autres micro-services qui minimise le second terme. L'algorithme proposé pour trouver un placement de l'application de partage de photo est décrit dans l'algorithme 1 donné à la page 71.

Cet algorithme cherche d'abord à placer le micro-service UI par rapport aux clients en se basant sur ses temps de réponse spécifiques pour les requêtes `GET` et `POST`. Un temps de réponse global de ce service sur l'ensemble des clients est calculé appliquant la fonction f caractérisant leur distribution. Le placement retenu est celui qui minimise ce temps de réponse global. Le placement du service UI est ensuite utilisé comme référence pour placer le service TH. Le placement retenu est celui qui minimise le temps de réponse spécifique du service TH mesuré depuis l'équipement hébergeant UI. La contrainte de co-localisation est vérifiée pour ne placer que m services au maximum sur chaque équipement. De la même façon, les services MH et PH sont placés par rapport au service TH. Le placement de chacun des services donnant lieu à un parcours de la liste des équipements, la complexité de cet algorithme est donc linéaire selon le nombre d'équipements, ainsi que selon le nombre de services à placer.

Cet algorithme s'appuie sur une priorisation des interactions entre micro-services selon leur coût dans le traitement des différentes requêtes de l'application. Les interations les plus couteuses définissent un chemin critique dans le graphe des appels qu'il faut chercher à réduire. Dans notre cas, les interactions prioritaires sont les requêtes `GET` et `POST` entre les clients et le service UI, puis les requêtes `GET` entre UI et TH, entre TH et PH et entre TH et MH. Des appels entre micro-services (par exemple les requêtes `POST` entre UI et PH) n'ont ainsi pas été pris en compte, car considérés comme prenant une part moins importante dans le temps de réponse global de l'application. Dans un cadre générique avec une application quelconque, cet algorithme pourra être efficace à trouver une solution approchée seulement s'il est possible d'identifier dans le graphe des appels entre les services, ces appels les plus coûteux formant un chemin critique.

Méta-heuristique PSO

La second heuristique proposée est fondée sur la méta-heuristique d'optimisation par essaims particulaires (en anglais *Particle Swarm Optimization* ou PSO) [31]. Ces heuristiques appartiennent à la famille des algorithmes évolutionnaires, dont font aussi partie les algorithmes génétiques ou les colonies de fourmis. Les heuristiques basées sur des algorithmes évolutionnaires ont montré des résultats intéressants sur des problèmes d'optimisation de composition de services [49]. La méta-heuristique PSO a notamment montré sa capacité à trouver des solutions proches de l'optimal dans des cas d'usage comportant un nombre très important de combinaisons possibles [57].

La méta-heuristique PSO s'appuie sur le principe d'auto-organisation d'une nuée d'oiseaux lors de son déplacement à la recherche de nourriture. Chaque particule suit les déplacements des autres particules de l'essaim qui convergent vers l'endroit où un minimum local est trouvé. L'essaim initial est composé de particules distribuées aléatoirement dans l'espace des solutions possibles. La position d'une particule représente une solution potentielle. Chaque particule possède un vecteur de vitesse, associée à une certaine inertie et se déplace dans l'espace des solutions à chaque itération de l'algorithme. À la fin d'une itération, l'algorithme évalue les solutions représentées par les positions de chaque particule et modifie les vecteurs de vitesses de manière à faire converger les particules vers les meilleures solutions. L'algorithme s'arrête soit lorsque l'essaim est stable sur un minimum local, soit lorsque le nombre maximal d'itérations est atteint. Dans ce cas, la meilleure solution parmi les positions actuelles des particules est retenue. La recherche de solutions par cet algorithme s'effectue donc au pire en temps constant. Une seconde propriété de cette heuristique est qu'elle n'est pas déterministe. Cet algorithme peut en effet proposer des solutions approchées différentes à chaque exécution sur un même problème, au contraire de la recherche exhaustive et de l'heuristique gloutonne qui fourniront toujours le même résultat.

Il est possible de dériver de cette méta-heuristique une heuristique pour notre problème de placement des micro-services de l'application de partage de photo. Une particule représente une solution de placement P repérée dans un espace à 4 dimensions où chaque position représente un placement possible de chacun des micro-services de l'application. Cet espace est borné sur chaque dimension par le nombre d'équipements. L'heuristique est configurée au départ avec l'ensemble des clients C et les modèles de temps de réponse spécifiques R_s et des paramètres propres à l'heuristique (nombre de particules, facteurs d'inertie). Les particules ne peuvent se positionner que sur des solutions valides au sens des contraintes imposées pour le placement, en l'occurrence le nombre maximal de services pouvant être hébergés sur un même équipement.

Elle evalue ensuite, pour chaque solution trouvée, le temps de réponse selon les formules 3.2 et 3.3 données précédemment.

3.4.3 Validation et comparaison des heuristiques

Nous allons ici évaluer et comparer les deux heuristiques proposées selon différents critères. Le premier critère sera l'approximation par rapport à la solution optimale de la solution approchée fournie par chaque heuristique. Cette approximation sera évaluée sur un échantillon de cas d'étude correspondant à différentes infrastructures participatives. L'évaluation sur ces différents cas doit nous permettre de déterminer la confiance que nous pouvons avoir dans les solutions données par chaque heuristique. Un second critère d'évaluation sera le temps nécessaire à chaque heuristique pour trouver une solution approchée et les variations de ce temps de calcul constatées sur les différents cas d'étude proposés.

Échantillon utilisé pour l'évaluation

Nous avons besoin, pour évaluer le résultat des heuristiques, des cas d'étude correspondant à des infrastructures participatives différentes. Or nous n'avons pas à notre disposition un panel d'infrastructures réelles qui pourraient servir à notre évaluation. Il nous est cependant possible de décrire une infrastructure participative quelconque à partir du nombre d'équipements participants, de la capacité de calcul sur chaque équipement et des paramètres de QoS des réseaux qui les raccordent. Dans notre fonction objectif, ces caractéristiques se retrouvent dans les temps de réponse spécifiques de chaque micro-service $R_{s(q)}$. Chaque temps de réponse $R_{s(q)}(i,j)$ dépend en effet du réseau entre les équipements i et j et des ressources processeur pour traiter la requête q par le service s. Il est donc possible de définir un cas d'étude comme un ensemble de valeurs $R_{s(q)}$ cohérent pour les paramètres définissant une infrastructure participative.

Pour obtenir ces valeurs, nous allons utiliser un modèle génératif des temps de réponse $R_{s(q)}$ en fonction des caractéristiques de l'infrastructure participative. Ce modèle va donner, en fonction des paramètres de capacité de calcul et de QoS réseau pour chaque équipement, une approximation de la valeur de $R_{s(q)}(i,j)$ qui pourrait être mesurée sur une infastructure réelle avec des caractéristiques similaires. Le profilage des micro-services, présenté dans le tableau 3.1, nous renseigne pour chaque requête la quantité d'informations échangée et le temps de traitement demandé. Nous pouvons construire un modèle pour le temps de réponse spécifique $R_{s(q)}(i,j)$ d'un micro-service s pour une requête q entre deux équipements i et j en utilisant ces données, associées aux paramètres de calcul et de QoS réseau des équipements i et j :

$$R_{s(q)}(i,j) = \frac{t_{s,q}}{a_j} + \left(2l_{i,j} + \frac{D_{s,q}}{b_{i,j}}\right) \tag{3.4}$$

avec
- i, j : Équipements hébergeant resp. le client et le service,
- $t_{s,q}$: Temps de traitement de la requête q pour le service s,
- a_j : Facteur d'acceleration du traitement sur l'équipement j,
- $l_{i,j}$: Latence réseau entre i et j,
- $D_{s,q}$: Quantité de données à transférer pour requête q et le service s,
- $b_{i,j}$: Bande passante entre i et j dans le sens du transfert de données.

Le premier terme de ce modèle correspond au temps de traitement par le service s de la requête q sur l'équipement j. Le temps de traitement $t_{s,q}$ est connu grâce aux informations de profilage du micro-service. Le facteur d'accélération a_j est lié aux ressources processeur disponible sur l'équipement j pour ce traitement. Dans notre cas, ce facteur aura une influence principalement pour la requête `GET` du micro-service TH (calcul d'une vignette). Nous ferons donc intervenir cette variable uniquement pour cette requête. Nous distinguerons alors deux types d'équipements, ceux avec de faibles ressources processeurs, pour lesquels $a_j = 1$, et ceux avec des processeurs performants, pour lesquels $a_j = 2$.

Le second terme du modèle correspond au temps nécessaire au transfert entre les équipements i et j de la requête et de sa réponse et dépend donc des caractéristiques du réseau entre ces équipements. D'une part la propagation d'un signal dans le réseau induit un délai quelque soit la quantité de données à transporter. Cette latence $l_{i,j}$ correspond au temps d'aller et retour d'un message vide entre les équipements i et j, aussi appelé Round Trip Time (RTT). Nous intégrons dans notre modèle deux fois ce délai pour prendre en compte la propagation de la requête ainsi que de la réponse. À ce délai s'ajoute un temps de transmission dépendant des données. Ce temps se calcule en divisant la quantité de données à transmettre $D_{s,q}$ par la bande passante $b_{i,j}$ disponible entre les deux équipements i et j. Dans notre cas, nous prendrons en compte ce temps de transmission uniquement pour les messages transportant de l'information utile, donc les requêtes `POST` et les réponses aux requêtes `GET`, pour lesquelles la quantité de donnée à transmettre dans ces messages est connue. Comme certains réseaux de l'infrastructure participative sont asymétriques, le sens de transmission de ces messages nous indique quelle mesure de la bande passante est à prendre en compte. Pour les réponses aux requêtes `GET`, la bande passante sera celle disponible dans le sens j vers i, et pour les requêtes `POST`, dans le sens i vers j.

Entre chaque équipement i et j, nous déterminons les valeurs $l_{i,j}$ et $b_{i,j}$ en fonction du type de réseau d'accès utilisé par ces deux équipements. Comme indiqué précédemment, nous considérons deux technologies d'accès résidentiel FTTH et DSL. Chacune de ces technologies se caractérise par des valeurs de RTT minimale et de bande passante maximale. Cependant il est nécessaire de considérer aussi des valeurs moyennes correspondant aux performances de ces réseaux en cours d'utilisation. Nous avons effectué des mesures sur différents réseaux d'accès de chaque type pour aboutir aux valeurs nominales et moyennes de $l_{i,j}$ et $b_{i,j}$ en fonction de la technologie utilisée. Le dispositif utilisé pour ces mesures est présenté au chapitre 4. Les valeurs obtenues sont présentées dans le tableau 3.2 à la page 72. Nous pouvons ainsi définir pour chaque requête, un temps de réponse minimal et un temps de réponse moyen en fonction du type de réseau utilisé par l'équipement client qui émet la requête, et l'équipement serveur, qui traite la requête. Le temps de réponse minimal est calculé en utilisant la valeur minimale de $l_{i,j}$ et maximale de $b_{i,j}$, et le temps de réponse moyen, en utilisant les valeurs moyennes pour $l_{i,j}$ et $b_{i,j}$. Ces valeurs sont présentées dans les tables 3.3, 3.4, 3.5 et 3.6 visibles à la page 72.

Un élément de notre échantillon représentera une infrastructure participative avec ces propres caractéristiques. Pour générer les valeurs des temps de réponse pour cet élément, nous fixons les 3 caractéristiques suivantes :

— n : le nombre d'équipements composant l'infrastructure participative,
— r_{FTTH} : le ratio des équipements connectés en FTTH,
— r_a : le ratio des équipements avec un taux d'accélération ayant pour valeur 2.

Une fois fixés, ces paramètres permettent de définir un ensemble d'équipements, où chacun est caractérisé par un facteur d'accélération a et une technologie d'accès résidentiel. Les valeurs minimale et moyenne de $R_{s(q)}(i,j)$ sont ensuite obtenues pour chaque couple d'équipement i,j à partir des tables calculées précédemment. Nous choisissons ensuite aléatoirement une valeur pour $R_{s(q)}(i,j)$ dans une distribution de Pareto paramétrée avec ces valeurs minimale et moyenne. Ce choix nous permet de faire varier les valeurs possibles des temps de réponses entre différents couples i,j avec les mêmes technologies d'accès. Dans le cas particulier où l'accès au service est local ($i = j$), la valeur du temps de réponse est égale à celle du temps de traitement.

Nous avons ainsi généré 24 cas d'étude, répartis en deux échantillons, représentant des infrastructures avec des caractéristiques différentes. Un premier échantillon va nous permettre d'évaluer les heuristiques sur des infrastructures avec différents nombres d'équipements, mais avec les mêmes répartitions des capacités de calcul et de communication entre équipements. Ce premier échantillon est donc constitué des valeurs $R_{s(q)}$ obtenues en fixant $r_{\text{FTTH}} = 50\%$ et $r_a = 50\%$ et en faisant varier n entre 16 et 128. Un second échantillon va nous permettre d'évaluer les heuristiques sur des infrastructures avec une répartition différente des capacités de communication.

Il constitué des valeurs $R_{s(q)}$ obtenues en fixant $n = 40$ et $r_a = 50\%$ et en faisant varier r_{FTTH} entre 10% et 90%. Pour chaque cas d'étude ainsi produit, nous choisissons aléatoirement 20% des équipements de l'infrastructure comme clients de l'application.

Résultats des heuristiques

Nous évaluons le placement obtenu pour chaque heuristique par le temps de réponse global de l'application calculé selon les formules 3.2 et 3.3 données précédemment. Les paramètres de ces formules sont fixés de la manière suivante :

— Poids des requêtes : $w_{c,\text{GET}} = w_{c,\text{POST}} = 0.5, \forall c \in C$,
— Moyenne arithmétique des temps de réponse : $f(\{t_c | \forall c \in C\}) = |C|^{-1} \sum_{c \in C} t_c$
— Nombre maximal de services par équipement : $m_i = 1, \forall i \in N$
— Nombre de vignettes calculées par requête `GET` : $k = 5$

En définissant un poids égal des requêtes pour l'ensemble des clients, nous choisissons de ne privilégier aucun usage dans le calcul du temps de réponse, et donc dans les solutions choisies par les heuristiques. La fonction d'agrégation donne la moyenne des temps de réponse pour l'ensemble des clients. C'est donc ce temps moyen que les heuristiques chercheront à minimiser. Enfin nous contraignons les solutions valides à n'héberger qu'un seul service par équipement. De cette manière, nous demandons aux heuristiques de trouver des solutions de placement minimisant les temps de communications sur le réseau.

À partir de ces paramètres, nous calculons le temps de réponse de l'application pour l'ensemble des solutions de placements possibles sur les cas d'étude de l'échantillon. Ce calcul nous donne pour chaque cas d'étude l'étendue des valeurs possibles du temps de réponse de l'application. Parmi ces valeurs, nous identifions la valeur minimale comme la solution de placement optimale. Ces valeurs sont présentées dans la Figure 3.6. La distribution des valeurs possibles pour le temps de réponse est présentée pour chaque cas d'étude sous forme d'un *box plot* (graphique «boîte à moustaches»). Les extrémités basse et haute du graphique représentent respectivement le 5^e et le 95^e centile. Les limites basse et haute de la boîte représentent respectivement le 25^e et le 75^e centile. Enfin la valeur médiane est donnée à l'intérieur de la boîte.

Nous remarquons sur la Figure 3.6 que l'étendue des valeurs possibles du temps de réponse est importante et peut atteindre deux ordres de grandeur pour un même cas d'étude (de quelques secondes à plusieurs centaines de secondes). Cette étendue varie selon la répartition de la connectivité des équipements : plus l'infrastructure inclue d'équipements connectés en FTTH, plus la distribution des valeurs possibles pour le temps de réponse est étroite, ce qui s'explique par une

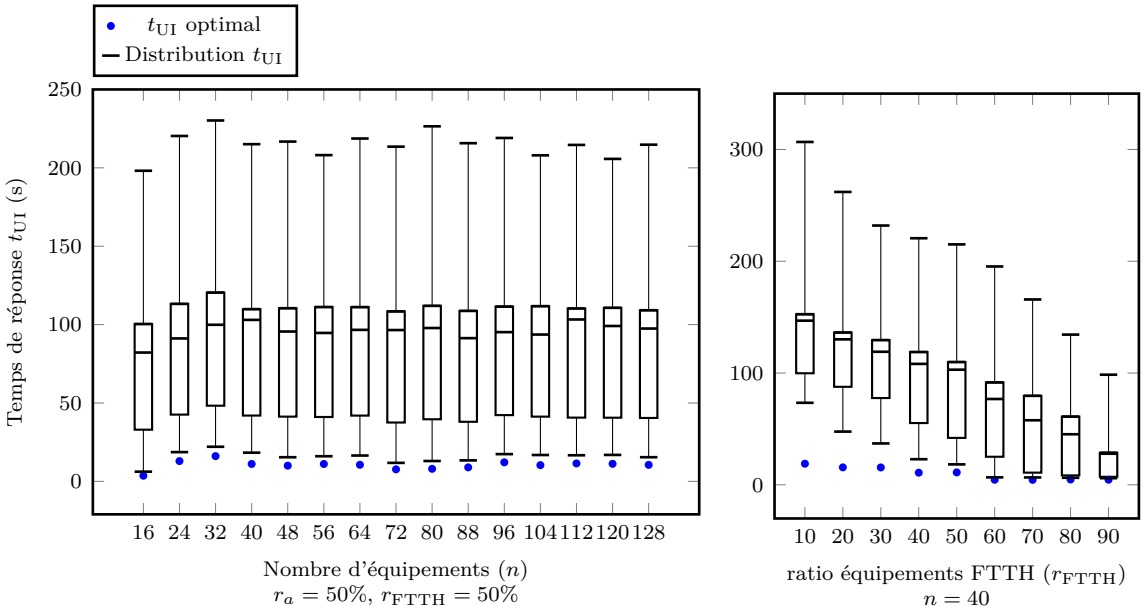

FIGURE 3.6 – Distribution du temps de réponse de l'application t_{UI} et temps de réponse pour la solution optimale, en fonction des caractéristiques des infrastructures.

diminution du temps de communication sur le réseau. Enfin ces résultats nous montrent que les solutions proches de la solution optimale ne sont pas nombreuses. Le 5^e centile (temps de réponse en dessous duquel se trouvent 5% de l'ensemble des solutions) se situe à un écart de 30% à 50% du temps de réponse optimal. Une heuristique efficace devra nous trouver une solution se trouvant parmi ces 5%.

Nous déroulons l'algorithme des deux heuristiques sur les différents cas d'étude des deux échantillons et nous les évaluons par l'écart entre le temps de réponse de la solution de placement proposée et le temps de réponse optimal obtenu précédemment. Nous avons implémenté l'heuristique gloutonne en langage Python. Pour l'heuristique PSO, notre code a utilisé l'implémentation de la librairie Pyswarm [1]. Nous avons paramétré la méta-heuristique avec les valeurs suivantes :

— Nombre d'itération maximale : 200,
— Nombre de particule de l'essaim : 100,
— Inertie des particules : $\omega = 0.5$,
— Facteur d'accélération vers le minimum local : $c_1 = 0.5$,
— Facteur d'accélération vers le minimum global = $c_2 = 0.5$.

Les temps de calcul nécessaires pour chaque heuristique sont très faibles : quelques millisecondes pour l'heuristique gloutonne, une demi-seconde pour l'heuristique PSO, temps mesurés

1. `https://pythonhosted.org/pyswarm/`

en moyenne sur un même processeur 3GHz. Pour référence, le temps de recherche de la solution optimale par la méthode exhaustive va de 1 seconde pour un cas d'étude de 16 équipements, 1 minute pour 40 équipements jusqu'à plus de 3 heures pour 128 équipements sur le même processeur.

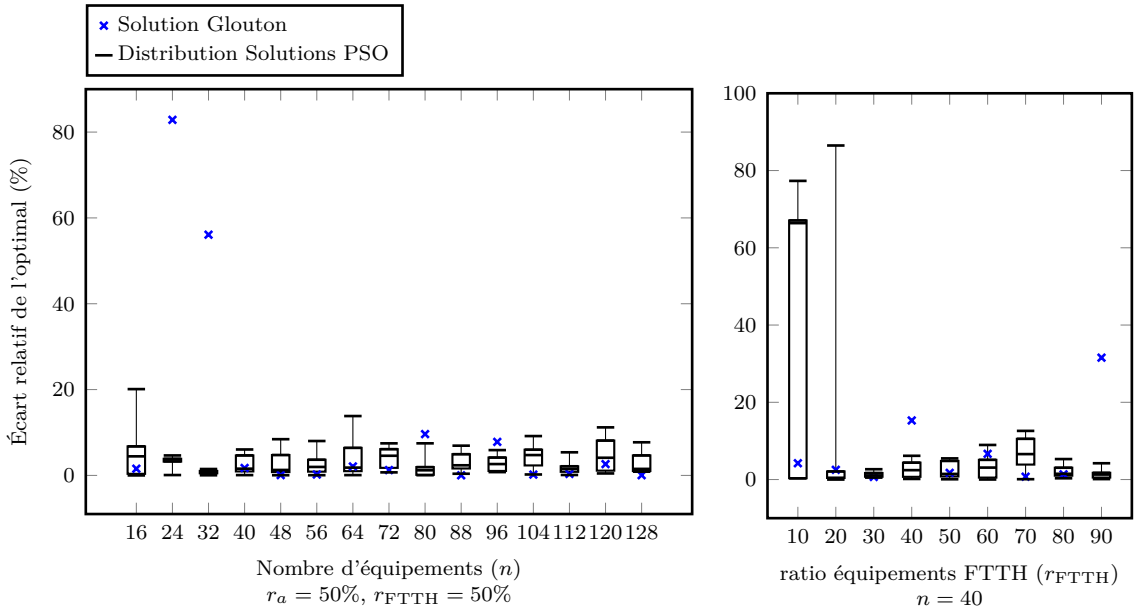

FIGURE 3.7 – Ecart relatif des solutions des heuristiques par rapport à la solution optimale

Le résultat de l'évaluation des heuristiques sur l'échantillon est présenté dans les graphiques de la Figure 3.7. L'heuristique gloutonne étant déterministe, nous indiquons l'écart du temps de réponse de la solution obtenue par un point. L'heuristique PSO est elle non-déterministe. Nous avons donc déroulé 50 fois l'algorithme pour obtenir un ensemble de solutions. La distribution de l'écart de ces solutions avec la solution optimale est présentée dans les graphiques sous la forme d'un *box plot*.

Ces résultats nous montrent que pour 19 des 24 cas d'étude des deux échantillons l'échantillon, les temps de réponse des solutions proposées par les heuristiques gloutonne et PSO sont au maximum 20% supérieurs au temps de réponse optimal. Nous en concluons que ces deux heuristiques sont efficaces selon le critère que nous nous sommes fixé précédemment. Les deux cas d'études où les solutions de PSO n'ont pas donné satisfaction sont ceux avec une répartition respectivement de 10% et 20% d'équipements connectés en FTTH. En examinant la répartition des temps de réponse possibles présentée sur la Figure 3.6, nous constatons que pour le premier cas d'étude, la valeur du 5^e centile présente un écart de plus de 300% avec le temps de réponse optimal. Les solutions proposées par PSO dans ce cas font donc toujours partie des 5% les plus proches de la solution optimale. Pour le second cas d'étude avec 20% d'équipements connectés en FTTH,

seul le 95^e centile est au delà de l'écart fixé. L'heuristique a produit ici quelques solutions moins performantes, cependant toujours présentes dans les 5% les plus proches de la solution optimale.

Nous comptons, dans cette évaluation, 3 cas d'étude pour lesquels l'heuristique gloutonne a proposé une solution avec un temps de réponse supérieur de 30% à la solution optimale. Pour ces 3 cas, la solution obtenue est située dans le deuxième quart des solutions les plus performantes (temps de réponse entre le 25^e centile et la valeur médiane). Ces propositions restent donc acceptables mais nous indiquent que l'heuristique gloutonne ne peut donner un résultat satisfaisant quelque soit le cas d'étude et donc la configuration de l'infrastructure. La confiance que nous pouvons apporter dans le résultat de cette heuristique est donc à relativiser par rapport à ceux obtenus avec l'heuristique PSO.

3.5 Conclusion

Cette étude nous a permis de mettre en évidence plusieurs contributions. Nous avons d'abord identifié les différents paramètres qui interviennent dans le temps de réponse d'une application composée de micro-services déployés sur une infrastructure participative. Pour un tel système distribué sur des équipements et des réseaux hétérogènes, le temps de réponse est une valeur dépendant de nombreux paramètres. Nous avons proposé d'introduire le temps de réponse spécifique pour chaque requête d'un service. Cette valeur nous permet de décomposer le problème en plusieurs termes relatifs à chaque communication impliquée dans le traitement de la requête. Nous en avons ainsi déduit une formalisation du problème d'optimisation du temps de réponse en fonction du placement des services et proposer différentes heuristiques pour identifier des solutions potentielles. Nous avons évalué ces algorithmes et finalement retenu l'heuristique PSO comme celle apportant le plus de garanties dans les solutions proposées.

Une faiblesse de cette étude est qu'elle s'appuie sur une seule application, fabriquée pour notre cas d'usage. Nous n'avons effectivement pas trouvé dans la littérature de modèles de référence pour les applications orientées micro-services sur lesquels nous aurions pu valider nos solutions. Nous considérons que nos travaux pourront s'appliquer à une autre application à partir du moment où il est possible d'en connaître la décomposition des appels entre micro-services sous forme de diagramme de séquence, ainsi que les informations de profilage des micro-services permettant de générer des valeurs pour les modèles de temps de réponse spécifiques, ce qui représente un certain niveau de connaissance de l'application. Des outils de profilage automatique existent [20]. Il peut être envisagé que les développeurs de micro-services rendent systématiquement disponibles ces informations pour faciliter leur réutilisation et optimiser leur déploiement.

Données : $R_{s(q)}$: temps de réponse spécifique pour les services $s \in \text{UI}, \text{MH}, \text{PH}, \text{TH}$ et la requête $q \in \text{GET}, \text{POST}$,
N : Ensemble des équipements de l'infrastructure,
C : Ensemble des clients de l'application,
$w_{c,\text{GET}}, w_{c,\text{POST}}$: Poids des requêtes pour client chaque client c,
f : Fonction caractérisant la distribution des temps de réponse,
m : nombre maximum de services hébergés par équipement
Résultat : $P = \{x_{\text{UI}}, x_{\text{MH}}, x_{\text{PH}}, x_{\text{TH}}\}$: placement optimisé des micro-services

begin
  `/* Trouver le meilleur placement de UI par rapport aux clients */`
  $\text{mint}_{\text{UI}} \leftarrow \infty$
  forall $x \in N$ **do**
    $s_x \leftarrow 0$
    forall $c \in C$ **do**
      $\text{t}_{c,\text{UI}} \leftarrow w_{c,\text{GET}} R_{\text{UI(GET)}}(c, x) + w_{c,\text{POST}} R_{\text{UI(POST)}}(c, x)$
    end
    $\text{t}_{\text{UI}} \leftarrow f(\text{t}_{c,\text{UI}}, c \in C)$
    if $\text{t}_{\text{UI}} < \text{mint}_{\text{UI}}$ **then**
      $\text{mint}_{\text{UI}} \leftarrow \text{t}_{\text{UI}}$
      $x_{\text{UI}} \leftarrow x$; $s_x \mathrel{+}= 1$
    end
  end
  `/* Trouver le placement de TH par rapport au placement de UI */`
  $\text{mint}_{TH} \leftarrow \infty$
  forall $x \in N$ **do**
    $\text{t}_{\text{TH}} \leftarrow R_{\text{TH(GET)}}(x_{\text{UI}}, x)$
    if $\text{t}_{\text{TH}} < \text{mint}_{\text{TH}} \wedge s_x < m$ **then**
      $\text{mint}_{\text{TH}} \leftarrow \text{t}_{\text{TH}}$
      $x_{\text{TH}} \leftarrow x$; $s_x \mathrel{+}= 1$
    end
  end
  `/* Trouver le placement de PH par rapport au placement de TH */`
  $\text{mint}_{PH} \leftarrow \infty$
  forall $x \in N$ **do**
    $\text{r}_{\text{PH}} \leftarrow R_{\text{PH(GET)}}(x_{\text{TH}}, x)$
    if $\text{t}_{\text{PH}} < \text{mint}_{\text{PH}} \wedge s_x < m$ **then**
      $\text{mint}_{\text{PH}} \leftarrow \text{t}_{\text{PH}}$
      $x_{\text{PH}} \leftarrow x$; $s_x \mathrel{+}= 1$
    end
  end
  `/* Trouver le placement de MH par rapport au placement de TH */`
  $\text{mint}_{MH} \leftarrow \infty$
  forall $x \in N$ **do**
    $\text{t}_{\text{MH}} \leftarrow R_{\text{MH(GET)}}(x_{\text{TH}}, x)$
    if $\text{t}_{\text{MH}} < \text{mint}_{\text{MH}} \wedge s_x < m$ **then**
      $\text{mint}_{\text{MH}} \leftarrow \text{t}_{\text{MH}}$
      $x_{\text{MH}} \leftarrow x$; $s_x \mathrel{+}= 1$
    end
  end
end

Algorithme 1 : Algorithme de placement glouton

Type de réseau		Latence $l_{i,j}$ (ms)		Bande passante $b_{i,j}$ (Mb/s)	
i	j	min.	moy.	max.	moy.
DSL	DSL	40	60	2	1
DSL	FTTH	30	45	8	2
FTTH	FTTH	20	25	100	20
FTTH	DSL	30	45	2	1

TABLE 3.2 – Caractéristiques des différents types de réseaux

Service		Type de réseau		Temps de réponse (ms)	
Nom	Requête	i	j	Min	Moyen
UI	GET	DSL	DSL	4180	8220
		DSL	FTTH	1160	4190
		FTTH	FTTH	220	550
		FTTH	DSL	4160	8190
	POST	DSL	DSL	20130	40170
		DSL	FTTH	20110	40140
		FTTH	FTTH	490	2100
		FTTH	DSL	5110	20140

TABLE 3.3 – Temps de réponse spécifique minimal et moyen pour les différentes requêtes du micro-service UI.

Service		Type de réseau		Temps de réponse (ms)	
Nom	Requête	i	j	Min	Moyen
MH	GET	DSL	DSL	134	178
		DSL	FTTH	111	144
		FTTH	FTTH	90	100
		FTTH	DSL	114	148
	POST	DSL	DSL	134	172
		DSL	FTTH	114	142
		FTTH	FTTH	90	100
		FTTH	DSL	110	141

TABLE 3.4 – Temps de réponse spécifique minimal et moyen pour les différentes requêtes du micro-service MH.

Service		Type de réseau		Temps de réponse (ms)	
Nom	Requête	i	j	Min	Moyen
PH	GET	DSL	DSL	20130	40170
		DSL	FTTH	5110	20140
		FTTH	FTTH	490	2110
		FTTH	DSL	20110	40140
	POST	DSL	DSL	20130	40170
		DSL	FTTH	20110	40140
		FTTH	FTTH	490	2100
		FTTH	DSL	5110	20140

TABLE 3.5 – Temps de réponse spécifique minimal et moyen pour les différentes requêtes du micro-service PH.

Service		Type de réseau		Temps de réponse (ms)	
Requête	a_j	i	j	Min	Moyen
TH GET	1	DSL	DSL	1380	2220
		DSL	FTTH	760	1390
		FTTH	FTTH	556	630
		FTTH	DSL	1360	2190
	2	DSL	DSL	1130	1970
		DSL	FTTH	510	1140
		FTTH	FTTH	306	380
		FTTH	DSL	1110	1940

TABLE 3.6 – Temps de réponse spécifique minimal et moyen pour les différentes requêtes du micro-service TH.

Chapitre 4

Évaluation du placement en conditions réalistes

Il avait toujours été convaincu qu'en réfléchissant suffisamment on pouvait venir à bout de tous les problèmes. Le vent, par exemple. Ce phénomène l'avait toujours intrigué, jusqu'à ce qu'il comprenne que le déplacement de l'air était provoqué par l'agitation des arbres.
— **Terry Pratchett,** *Le Grand Livre des Gnomes*

L'heuristique que nous avons présentée dans le chapitre précédent nous permet de déterminer un placement des micro-services d'une application assurant un temps de réponse proche des performances optimales pouvant être atteintes avec les capacités de calcul et de communication de l'infrastructure participative. Cependant, dans un cas réel d'utilisation, ces capacités sont variables car perturbées par plusieurs facteurs extérieurs. Si les équipements de l'infrastructure remplissent d'autres fonctions que celles d'hébergement, comme un serveur de stockage domestique par exemple, les services hébergés vont cohabiter avec d'autres usages et verront leurs performances potentiellement impactées. De la même façon, dès que plusieurs équipements du réseau domestique communiquent vers l'Internet, différents flux réseaux sont en concurrence sur le réseau d'accès résidentiel. Les capacités de communication s'en retrouvent alors réduites pour chacun des flux, induisant des temps de réponse des services plus importants.

Nous souhaitons ici évaluer l'impact de ces usages concurrents sur l'expérience utilisateur. En effet, dans des conditions réalistes d'utilisation, ces perturbations peuvent induire une augmentation du temps de réponse de l'application, susceptible d'effacer les gains de performances obtenus grâce à l'heuristique de placement. Nous nous intéressons donc dans ce chapitre à l'influence des paramètres de QoS réseaux sur le temps de réponse. Nous caractériserons tout d'abord ces conditions réalistes d'utilisation à partir d'une analyse des variations des paramètres de délai, de bande passante et de perte de paquets observées sur des réseaux d'accès résidentiels existants. Nous évaluerons ensuite le temps de réponse de l'application lorsque ces variations sont présentes, tout d'abord dans un réseau émulé permettant d'introduire ces variations de manière contrôlée, puis dans un réseau réel, où les variations sont véritablement induites par du trafic

d'utilisateur. Ces résultats nous permettrons de conclure sur la pertinence de l'optimisation du temps de réponse dans un contexte réaliste d'utilisation.

4.1 Définition des conditions réalistes d'utilisation

Les paramètres de QoS des réseaux d'accès résidentiels subissent des variations que nous allons essayer ici de caractériser pour identifier les intervalles correspondant à des conditions réalistes d'utilisation. Ces variations sont notamment induites par le trafic engendré par les membres du foyer. Ces usages sont concurrents de ceux des services de l'infrastructure participative et utilisent la même ressource qu'est le réseau d'accès. Une première conséquence de ce trafic est une diminution de la bande passante disponible. La latence réseau entre deux équipements de l'infrastructure, mesurable par le RTT, connait elle aussi des variations au niveau des réseaux d'accès résidentiels ainsi que dans les réseaux de collecte et de transit. Des congestions au niveau d'équipements intermédiaires comme les routeurs peuvent aussi perturber l'acheminement du trafic et induire des temps de transit supplémentaires. Nous nous intéresserons aussi à la probabilité de pertes de paquets dans le réseau reliant deux équipements, pertes principalement dues aux congestions sur les équipements intermédiaires. Pour pouvoir évaluer l'influence de ces paramètres sur le temps de réponse de l'application, il nous faut d'abord connaitre les intervalles dans lesquels ils sont susceptibles d'évoluer dans des conditions réalistes d'utilisation.

4.1.1 Données issues de l'état de l'art

Nous avons d'abord recherché dans la littérature des résultats de mesures des paramètres de QoS des réseaux d'accès résidentiels. Nous avons identifié plusieurs études apportant un éclairage sur les variations observables de ces paramètres dans des réseaux réels.

Service SamKnows

SamKnows est une entreprise fournissant des mesures de performances réseaux à des clients comme les opérateurs de réseau ou les autorités gouvernementales de régulation. L'outil de mesure du service SamKnows s'appuie sur un équipement appelé WhiteBox déployé chez des particuliers volontaires. Ce dispositif permet de collecter des mesures de bande passante, de latence, de perte de paquets ainsi que d'autres paramètres correspondant à des usages spécifiques (temps de chargements d'une page Web, temps de résolution DNS). Le boitier est situé en coupure entre le réseau d'accès résidentiel et le réseau domestique, afin de détecter le trafic

utilisateur et de déclencher les mesures de performances du réseau d'accès lorsque ce trafic utilisateur est suffisamment faible pour ne pas perturber les mesures.

La Commission Européenne, dans le cadre du Plan Numérique pour l'Europe en 2012, a commandé à SamKnows une étude sur trois ans de la qualité d'expérience de la connexion résidentielle à l'Internet dans différents états membres de l'Union. L'un des objectifs de cette étude est de comparer la bande passante disponible pour l'utilisateur depuis son réseau d'accès résidentiel, par rapport à la bande passante annoncée par l'opérateur. Entre 2012 et 2014, le service SamKnows a ainsi mesuré différents indicateurs de la performance de l'accès à l'Internet sur un panel de 8582 foyers répartis entre 30 pays. Les différents réseaux d'accès utilisés dans ce panel se répartissent équitablement entre les technologies FTTH, Câble (DOCSIS) et DSL.

Le dernier rapport en date [34], publié en 2015, analyse les résultats des mesures effectuées pendant le mois d'octobre 2014. La comparaison entre la bande passante annoncée par l'opérateur et celle réellement mesurée montre des différences notables. En moyenne sur l'ensemble des panélistes, la bande passante descendante réelle mesurée sur 24 heures représente 83% de celle annoncée pour les réseaux FTTH, 63% pour les réseaux DSL. La bande passante montante mesurée représente elle 92% de celle annoncée pour les réseaux FTTH et 83% pour les réseaux DSL. Ces chiffres mettent en évidence que les performances réelles du réseau d'accès sont différentes des valeurs annoncées par l'opérateur et dépendent d'autres paramètres comme la qualité de la ligne ou les équipements de l'opérateur.

L'analyse des mesures effectuées aux différentes heures de la journée met en évidence des variations quotidiennes de la bande passante disponible présentée sur la Figure 4.1. Ces variations correspondent aux perturbations liées aux usages du foyer de son accès à l'Internet et suivent le cycle diurne de l'activité humaine : la bande passante du réseau d'accès résidentiel est surtout disponible pendant la nuit pour se réduire en journée et notablement le soir. Cette variation s'observe de manière plus notable dans les mesures de latence et de perte de paquet, paramètres qui montrent une augmentation sensible pendant les heures de la soirée. Ces observations sont en corrélation avec une hausse du trafic utilisateur dans l'ensemble du réseau de l'opérateur, qui entraine une charge importante au niveau des équipements du réseau de collecte.

Les mesures effectuées par le service SamKnows mettent donc en évidence deux propriétés des paramètres de QoS des réseaux d'accès résidentiels : premièrement les valeurs maximales attendues pour la bande passante montante et descendante sont plus faibles de 10 à 40% par rapport aux valeurs nominales annoncées par l'opérateur ; deuxièmement cet affaiblissement est variable selon les heures de la journée, selon la charge globale de trafic au sein du réseau de l'opérateur. Les mesures des paramètres de latence et de perte de paquet nous apportent aussi des indications sur les valeurs possibles pouvant être constatées en conditions réelles. Cependant les mesures

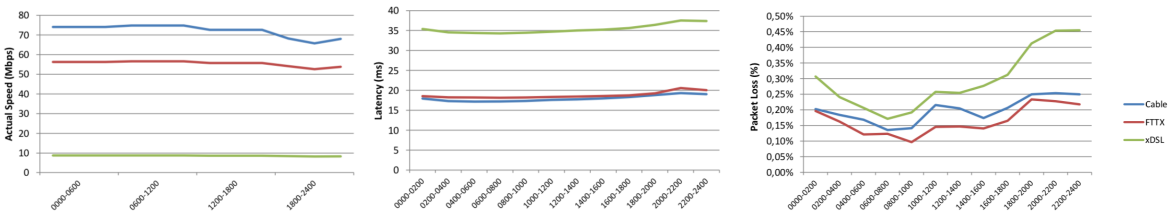

FIGURE 4.1 – Variations quotidiennes des paramètres de QoS des réseaux d'accès résidentiels observées par le service SamKnows (figure extraite de [34]).

effectuées depuis une Whitebox s'effectuent lorsque le réseau d'accès résidentiel est inutilisé. Pour décrire les conditions dans lesquelles sera utilisée l'application déployée sur l'infrastructure participative, il est nécessaire d'avoir en complément des mesures sur un réseau chargé.

Plateforme M-Lab

La plateforme M-Lab (pour *Measurement Lab*) est une initiative académique visant à créer une infrastructure globale de mesure de performances de l'Internet. Cette plateforme se compose de services de mesures, hébergés sur des serveurs répartis en différents points du réseau, ainsi que d'une interface de consultation des résultats de ses mesures. Le principal outil de mesure utilisé est Network Diagnostic Tool (NDT) [1]. Il permet à un utilisateur, depuis une page Web ouverte dans son navigateur, de tester la bande passante disponible sur son réseau d'accès à l'Internet, ainsi que d'autres paramètres comme la latence et le taux de perte de paquet. Les mesures effectuées par cet outil rendent donc compte des performances à un instant donné du réseau entre le navigateur et les serveurs de M-Lab. À la différence du service SamKnows, les mesures par NDT sont potentiellement réalisées en présence d'un trafic utilisateur concurrent. Les mesures collectées à travers l'outil NDT depuis 2009 forment un ensemble de données (*dataset*) de plus de deux milliards d'enregistrements.

Plusieurs publications analysent les résultats obtenus par l'outil NDT afin d'en extraire des caractéristiques de différents réseaux d'accès. L'article [18] étudie des mesures effectuées depuis plusieurs réseaux communautaires d'accès sans fil : Guifi à Barcelone, Ninux en Italie et Athens Wireless Metropolitan Network en Grèce. Les mesures concernent 158 nœuds répartis sur ces 3 réseaux sur une période de 3 mois. À partir des identifiants des équipements participant à ces tests, les auteurs ont extraits du *dataset* NDT les résultats des mesures de bande passante et de latence les concernant. Les résultats montrent une réalité contrastée selon les nœuds considérés : certains ont une latence assez stable, inférieure à 100ms avec une gigue de 20%.

1. `https://speed.measurementlab.net/`

D'autres présentent des variations fortes avec une gigue supérieure à 200%. Pour des nœuds connectés à des réseaux offrant une bande passante descendante supérieure à 100Mbit/s, les mesures montrent que celle-ci peut être réduite de plus de 50%. Les auteurs de l'article constatent ces variations importantes des paramètres sans cependant conclure si elles sont dues au trafic utilisateur ou à des fluctuations sur la connexion sans fil au réseau communautaire.

Les auteurs de [28] ont proposé une analyse quasi-exhaustive des mesures de bande passante disponibles dans les données récoltées par l'outil NDT. Ils mettent en évidence la difficulté d'aboutir à des conclusions à partir d'un grand ensemble de mesures car celles-ci sont effectuées dans des conditions très différentes. Leur analyse a donc porté sur un échantillon de 2 foyers, situés dans le même pays, et présentant des mesures suffisamment représentatives de l'utilisation du réseau à différents instants. Cette analyse met en évidence que les variations de ce paramètre peut présenter différents cycles qui se reproduisent à certains moments de la journée ou de la semaine. Une analyse temporelle de la bande passante par heure de la journée ou jour de la semaine montre ainsi que les variations peuvent être plus importantes que celles observées en analysant la moyenne de ces valeurs sur une période de temps plus longue. Cependant certaines de ces valeurs sont des points atypiques (*outliers*) qu'il convient de retirer de l'étude. Pour un foyer retenu par cette étude, la bande passante mesurée présente ainsi, pendant certaines heures de la journée, une diminution de plus de 75% par rapport à la valeur maximale observée, diminution se reproduisant aux mêmes heures de la journée pendant plusieurs jours.

L'article [44] s'intéresse aux variations de latence observées dans les mesures de l'outil NDT, ainsi que dans les captures réalisées sur un lien d'agrégation de réseaux résidentiels chez un opérateur. Pour chaque mesure du jeu de données NDT est calculé le *RTT span* correspondant à la différence entre les latences maximale et minimale observées lors de la mesure. En assimilant cette valeur à la gigue moyenne, l'analyse de l'ensemble des mesures de l'outil NDT montre une valeur médiane de cette gigue moyenne autour de 180ms soit 200% de la valeur médiane de la latence minimale. L'écart important de ce résultat avec les valeurs observées dans [18] est à relativiser par la diversité beaucoup plus importante des clients concernés par ces mesures. Les mesures effectuées sur le lien d'agrégation d'un opérateur concernent un nombre de clients beaucoup plus restreint (entre 50 et 400 clients). À partir d'une capture du trafic sur ce lien, il est possible de mesurer le temps entre les deux messages d'ouverture d'une session TCP pour en déduire la latence induite par le réseau d'accès. L'analyse de ces mesures met en évidence une relation entre la bande passante utilisée sur le lien montant du réseau d'accès résidentiel et la variation de la latence. Il apparait que, pour un trafic peu important sur le lien, le délai mesuré entre les deux messages TCP est stable. Par contre, si la bande passante est fortement utilisée, ce délai peut subir des variations jusqu'à 100% du délai constaté dans le cas précédent. Ces résultats s'expliquent par un phénomène de congestion au niveau de la passerelle du réseau

domestique : l'accès résidentiel ayant une bande passante relativement faible, les files d'attente au niveau de la passerelle sont fortement sollicitées en cas de trafic important.

4.1.2 Mesures en conditions réelles

Les données disponibles dans l'état de l'art nous fournissent plusieurs indications sur les variations des paramètres de QoS des réseaux d'accès résidentiels. Cependant nous ne pouvons pas encore déterminer les intervalles de validité de ces paramètres tant les cas d'études identifiés sont nombreux et variés, et potentiellement éloignés de notre cas d'usage. Afin d'obtenir une indication précise de ces intervalles dans un contexte représentatif d'une infrastructure participative, nous avons effectué nos propres mesures de ces paramètres à partir de dispositifs installés dans le réseau résidentiel de plusieurs volontaires.

Réseaux résidentiels observés

Notre campagne de mesure s'est réalisée sur un panel de sept réseaux d'accès résidentiels. Les foyers concernés sont tous situés sur la métropole de Rennes, mais sont abonnés à des opérateurs différents, se répartissant équitablement entre les opérateurs grand public Orange et Free. Les utilisateurs de ces réseaux sont des foyers avec ou sans enfants, ayant chacun un usage personnel différent de leur connectivité à l'Internet. Six de ces réseaux utilisent la technologie FTTH avec une bande passante symétrique de 100Mit/s annoncée par leur opérateur. Par soucis de diversité, un réseau utilisant la technologie VDSL a été ajouté à ce panel. Les paramètres de QoS de ces réseaux d'accès ont été observés 24h/24h, dans la limite de la disponibilité des dispositifs de mesures, en incluant les jours ouvrés, les fins de semaine et les congés. Les mesures vont ainsi traduire les différentes périodes d'utilisation de ces réseaux d'accès.

Le tableau 4.1 présente ces différents réseaux avec les durées pendant lesquelles les mesures ont été effectuées. Les durées d'observation pour chaque réseau sont différentes car les dispositifs de mesures ont tous connu des périodes d'indisponibilité, pour des raisons diverses : défaillance du réseau résidentiel, panne électrique, extinction du dispositif par l'utilisateur par soucis de consommation électrique, etc. Il est intéressant de noter aussi que, bien que volontaires au début de l'expérience, certains utilisateurs ont ensuite montré des réticences à garder connecté chez eux un dispositif qu'ils ne maitrisent pas. Motivés par ces aspects, certains volontaires ont ainsi quitté prématurément l'expérimentation.

Nom	Type de réseau	Opérateur	Durée des mesures (jours)
FTTH1	FTTH	Orange	133
FTTH2	FTTH	Orange	101
FTTH3	FTTH	Free	37
FTTH4	FTTH	Orange	92
FTTH5	FTTH	Free	24
FTTH6	FTTH	Orange	175
VDSL1	VDSL	Free	87

TABLE 4.1 – Réseaux résidentiels utilisés pour les mesures des paramètres de QoS

Méthologie des mesures

Les paramètres de QoS sont mesurés entre deux points du réseau : le dispositif de mesure déployé dans chaque réseau résidentiel observé, et un serveur de mesures situé dans notre laboratoire de recherche. Le dispositif de mesure consiste en un boitier de type *Mini-PC* comportant les outils de mesures nécessaires. Ce boitier est raccordé au réseau résidentiel comme n'importe quel autre équipement domestique, avec un débit maximum de 100Mbit/s symétrique. Cette connexion est située si possible au plus proche de la passerelle d'accès résidentiel afin que le trafic interne au réseau domestique ne transitant pas sur le réseau d'accès ne perturbe pas les mesures. Le serveur de mesure est lui connecté au réseau du laboratoire, lui-même relié à Internet grâce au réseau national de la recherche RENATER avec une bande passante symétrique de 200Mbit/s. Les paramètres de QoS sur le réseau du laboratoire et l'accès au réseau RENATER sont supervisés afin de détecter des anomalies éventuelles qui pourraient perturber les mesures. Les paramètres mesurés sont la bande passante montante et descendante disponible, ainsi que le temps d'aller-retour (RTT). Les mesures s'effectuent de manière non-intrusive sur le réseau domestique : les boitiers ne capturent ni n'analysent le trafic existant sur le réseau d'accès résidentiel. Ils évaluent la variation de ces paramètres à partir de flux générés depuis les boitiers. L'ensemble des mesures s'effectuent au dessus du Protocole Internet IPv6, afin de ne pas subir de perturbations éventuelles de la part du mécanisme de translation d'adresse (NAT) présent dans la passerelle d'accès résidentiel.

La bande passante est mesurée à travers l'outil *iperf3*[2] qui mesure la bande passante maximale entre une source et une destination par saturation du lien en générant du trafic sur une

2. `https://github.com/esnet/iperf`

courte période. Lorsque l'outil utilise le protocole de transport TCP, ce qui est le cas dans nos mesures, la quantité de trafic générée est croissante afin d'obtenir des performances optimales (adaptation à l'ouverture de la fenêtre d'émission TCP). L'outil charge ainsi progressivement le lien jusqu'à saturation pendant une durée de 10 secondes, durée suffisante dans la plupart des cas pour obtenir un résultat stable. Nous réalisons deux tests, avec du trafic dans chaque sens de transmission entre l'équipement et le service, pour mesurer la bande passante montante et descendante. Le RTT est simplement mesuré grâce à l'outil *ping* avec la taille standard de message ICMP (64 octets). Un échantillon de 10 échanges ICMP nous permet d'obtenir une valeur instantanée du temps d'aller-retour sur le réseau entre les deux points de mesure. Nous n'avons pas réalisé de mesures de la probabilité de perte de paquet, car celles-ci nécessitent une analyse sur une quantité de trafic importante, ce que nous ne pouvions nous permettre de faire avec des mesures non-intrusives.

Les mesures sont orchestrées depuis le serveur situé dans notre laboratoire. Celui-ci contrôle à distance l'exécution des outils de mesures sur les dispositifs, récupère les résultats obtenus et les centralise dans un fichier spécifique par type de mesure et par réseau observé. Les trois paramètres de QoS des réseaux résidentiels sont mesurés successivement sur l'ensemble des dispositifs disponibles au début d'une session de mesure. Chaque mesure sur un même dispositif est ainsi espacée de plusieurs minutes, le temps pour le serveur d'effectuer les mesures sur les autres dispositifs. Cet intervalle permet de ne pas créer d'effet de bord entre des mesures trop rapprochées l'une de l'autre.

Résultats des mesures

La campagne de mesure s'est déroulée sur une période d'une année. Tous les dispositifs de mesures n'ont pas été disponibles sur l'intégralité de cette période. Même si pour certains réseaux la durée d'observation n'a pas dépassé 24 jours, elle nous a permis de recueillir des données suffisantes pour caractériser les variations des paramètres de QoS sur ces réseaux. Sur une journée complète de disponibilité d'un dispositif, environ 100 mesures de chacun des paramètres de QoS sont ainsi effectuées pour un même réseau. L'ensemble de la campagne nous a ainsi permis de réaliser environ 50000 mesures.

Avant analyse, nous avons filtré les mesures jugées anormales. Tout d'abord, ont été retirées les mesures effectuées lorsque le trafic mesuré sur le réseau d'accès du laboratoire était important, ce qui a représenté quelques heures sur la durée de l'expérimentation. Ensuite, nous avons remarqué certaines périodes pendant lesquelles la bande passante disponible pour l'ensemble réseaux résidentiels d'un même opérateur a chuté brusquement. Ces baisses sont probablement

consécutives d'incidents dans les réseaux de transit entre cet opérateur de réseau résidentiel et l'opérateur RENATER. Une fois ces quelques incidents identifiés, les mesures correspondantes ont été ignorées. Enfin nous avons filtré les résultats des mesures représentant des valeurs extrêmes par rapport à la distribution observée (*outliers*) grâce à la méthode du Z-score[46].

La Figure 4.2 représente les distributions observées des valeurs pour les différents paramètres de QoS réseau sur les différents réseaux domestiques identifiés dans la Table 4.1. Nous constatons que certains réseaux d'accès résidentiels présentent une bande passante descendante disponible relativement stable (FTTH2, FTTH4, FTTH5, FTTH6) alors que pour d'autres réseaux (FTTH1, FTTH3, VDSL1), les mesures ont souvent relevé une bande passante réduite. Par rapport à la valeur maximale mesurée, 50% des mesures depuis ces réseaux montrent une réduction de la bande passante d'au moins 15%, 25% des mesures une réduction d'au moins 25% et enfin 5% des mesures une réduction d'au moins 40%. Ces réseaux d'accès utilisant des technologies ou des opérateurs différents, cette diminution de la bande passante descendante disponible s'explique par le trafic réseau depuis l'Internet vers le réseau domestique (flux vidéo par exemple). Les autres réseaux observés font, quant à eux, un usage modéré de leur bande passante descendante, suffisamment faible pour ne pas perturber la disponibilité de la bande passante.

De la même façon, les réseaux FTTH2, FTTH4, FTTH5, FTTH6 montrent une bande passante montante disponible avec une variation importante, alors que les réseaux FTTH1, FTTH3, VDSL1 ont une bande passante montante disponible stable. Par rapport à la valeur maximale mesurée, 50% des mesures pour les premiers réseaux montrent une réduction de la bande passante montante disponible d'au moins 15%, 25% des mesures une réduction d'au moins 25% et enfin 5% des mesures une réduction d'au moins 50%. De la même façon nous imputerons ces diminutions au trafic lié à l'usage de la connexion à l'Internet depuis le réseau domestique.

De façon intéressante, se dégagent de cette étude de la disponibilité de la bande passante différents profils d'usage. D'une part certains foyers ont une utilisation importante de la bande passante descendante, en accédant par exemple à des services de vidéos à la demande (utilisateurs *streamers*). D'autres part, des foyers utilisent de manière importante leur bande passante montante, probablement à travers des services auto-hébergés ou des outils de partage de fichiers (utilisateurs *seeders*). Ces profils semblent assez exclusifs, nous n'avons pas constaté dans notre panel de foyers présentant à la fois ces deux profils d'usage.

L'analyse des mesures des temps d'aller-retour (RTT) montrent que les valeurs obtenues sont caractéristiques de l'opérateur du réseau d'accès résidentiel. Les réseaux d'accès Orange ont une latence minimale d'au moins 24ms, alors que les réseaux d'accès Free ont une latence minimale d'environ 15ms (17ms avec la technologie VDSL). Les variations observées sur ces mesures de latence sont aussi caractéristiques de l'opérateur. La latence mesurée depuis les réseaux

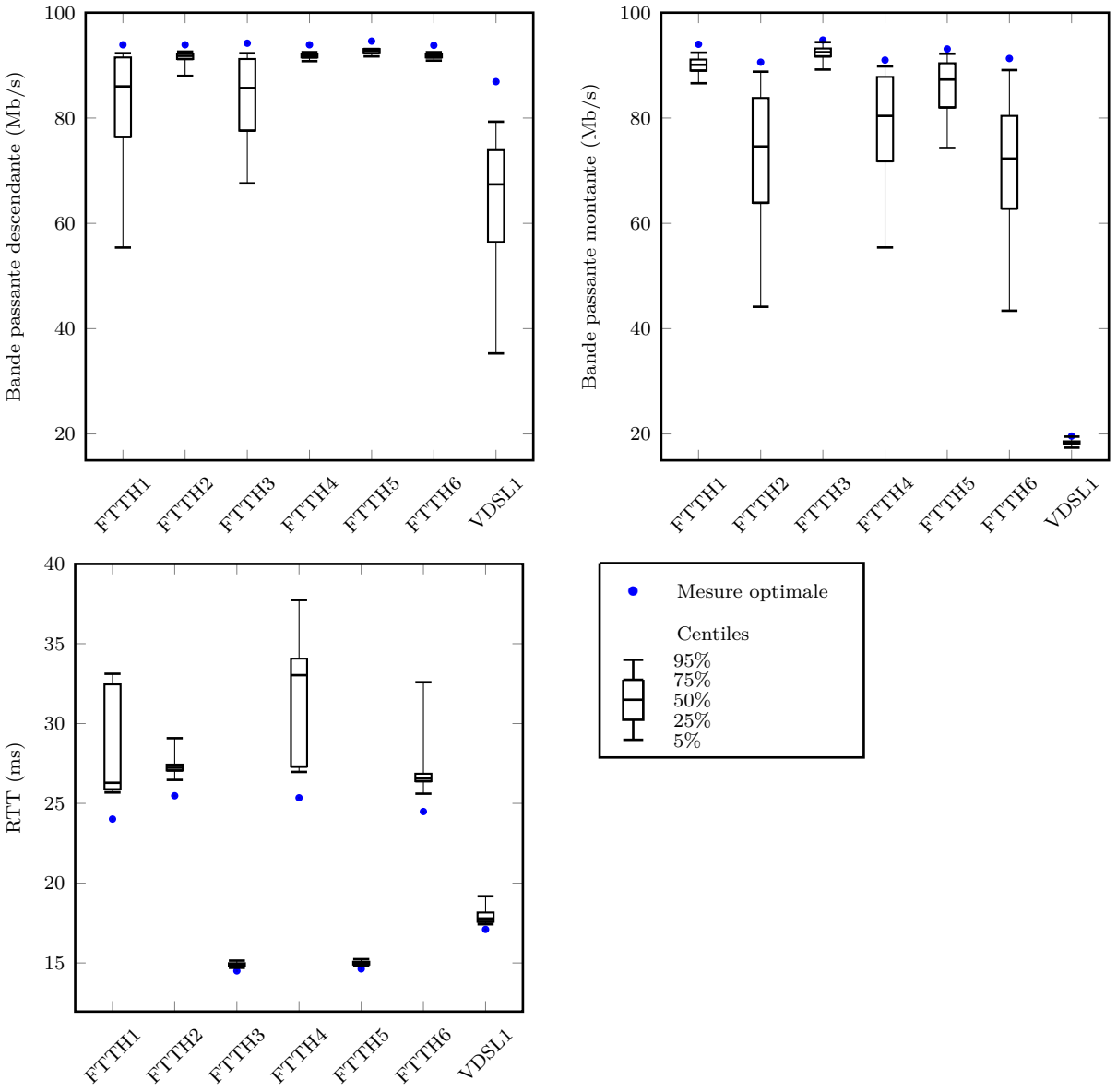

FIGURE 4.2 – Distributions des valeurs des différents paramètres de QoS pour les réseaux d'accès résidentiels observés

d'accès Free est très stable. Celle mesurée depuis les réseaux Orange montrent des variations plus importantes. La gigue sur ces réseaux est, pour 5% des mesures, supérieure à 10ms, soit 40% de la latence minimale. Notre panel contenant des réseaux avec une grande bande passante, les résultats obtenus ne mettent pas en évidence une relation entre l'utilisation de la bande passante et une congestion éventuelle du réseau d'accès résidentiel.

4.1.3 Intervalles réalistes des paramètres de QoS réseau

Les différentes études de la littérature, complétées par les mesures effectuées sur un panel de réseaux d'accès résidentiels, nous permettent de déterminer les intervalles de variation des paramètres de QoS réseau qui caractérisent des conditions réalistes d'utilisation. Nous pourrons ainsi évaluer la variation du temps de réponse lorsque ces paramètres évoluent dans les intervalles de valeurs retenus.

L'étude des réseaux communautaires présents dans le jeu de données de M-Lab que nous avons citée ([18]) montrent des réductions possibles de la bande passante disponible jusqu'à 50%. Cette estimation est en accord avec la seconde analyse citée ([28]), qui a étudié la bande passante de deux réseaux domestiques. En suivant la recommandation de filtrer les valeurs atypiques donnée dans ce dernier article, nous observons que nos mesures corroborent ce chiffre. Notre étude montre de plus que ces variations s'observent aussi bien sur la bande passante montante que descendante. Nous fixons donc l'intervalle de variation de la bande passante disponible entre 50% et 100% de la valeur nominale donnée par le réseau d'accès résidentiel.

Les variations de latence observées dans nos mesures (jusqu'à 50% de la valeur minimale) sont moins importantes que celles présentes dans les mesures de l'outil NDT de M-Labs (entre 100% et 200% de la valeur minimale). Cela s'explique en partie par le fait que les réseaux impliqués dans les mesures de NDT sont plus diversifiés et potentiellement moins stables que les réseaux résidentiels fixes observés dans nos mesures. Afin de couvrir un nombre de situations suffisamment large, nous établirons que dans des conditions réalistes, la gigue prendra une valeur comprise entre 0% et 100% de la valeur minimale.

Concernant la perte de paquet, seule l'étude SamKnows nous donne des valeurs issues de mesures, car elle s'appuie sur une analyse complète du trafic sur le réseau d'accès. Le graphique présentée dans la Figure 4.1 montre une variation de ce paramètre entre 1‰ et 5‰. Cependant ces résultats étant des moyennes, nous estimerons la perte de paquet sur un intervalle plus large entre 0‰ et 9‰.

4.2 Plateforme d'évaluation

Nous avons, dans la section précédente, caractérisé les variations des paramètres de QoS réseau dans les conditions d'un cas réel d'infrastructure participative. Nous cherchons maintenant à évaluer les variations du temps de réponse de l'application en présence de ces variations. La question à laquelle nous voulons aussi répondre est de savoir si le temps de réponse pour la solution de placement choisie reste toujours acceptable lorsque l'application est utilisée dans des conditions réelles. L'ampleur des variations observées nous permettra ainsi de conclure sur la pertinence de l'optimisation effectuée grâce à notre heuristique.

Nous avons défini une plateforme ayant pour objectif d'évaluer le temps de réponse de l'application dans différents contextes d'utilisation selon le nombre d'équipements participant à l'infrastructure, les types de réseaux d'accès résidentiels utilisés et les variations des paramètres de QoS de ces réseaux. Dans un premier temps, nous souhaitons maîtriser l'ensemble de ces paramètres et définir des cas d'étude. Nous utiliserons pour cela notre plateforme d'évaluation au dessus d'une émulation d'une infrastructure participative. Une fois que nous aurons caractérisé le temps de réponse de l'application dans ces différents cas d'étude et identifié l'influence de chaque paramètre de QoS, nous reproduirons cette évaluation avec la même plateforme fonctionnant sur une infrastructure participative réelle et nous comparerons les résultats obtenus.

4.2.1 Architecture fonctionnelle

Notre plateforme doit nous permettre de déployer l'application sur une infrastructure participative quelconque selon notre heuristique de placement, puis d'en évaluer les performances dans différents contextes d'utilisation. Le système que constitue cette plateforme nécessite plusieurs fonctions pour collecter des informations nécessaires au calcul du placement selon notre heuristique, déployer les micro-services de l'application sur l'infrastructure participative, contrôler et mesurer des paramètres de QoS de cette infrastructure lors de l'évaluation et enfin mesurer le temps de réponse de l'application.

Le principe qui dirige l'organisation de ces différentes fonctions est de séparer les fonctions liées au fonctionnement de l'infrastructure participative de celles participant à la décision du placement des micro-services. Un intergiciel permet d'assurer l'interface entre ces deux domaines de fonction. Cette séparation nous permet notamment de pouvoir réutiliser les fonctions de décisions avec des implémentations d'infrastructures participatives différentes, dans notre cas les infrastructures émulées et réelles. La Figure 4.3 présente les différentes fonctions de la plateforme d'évaluation, leur répartition en domaines fonctionnels ainsi que leurs interactions.

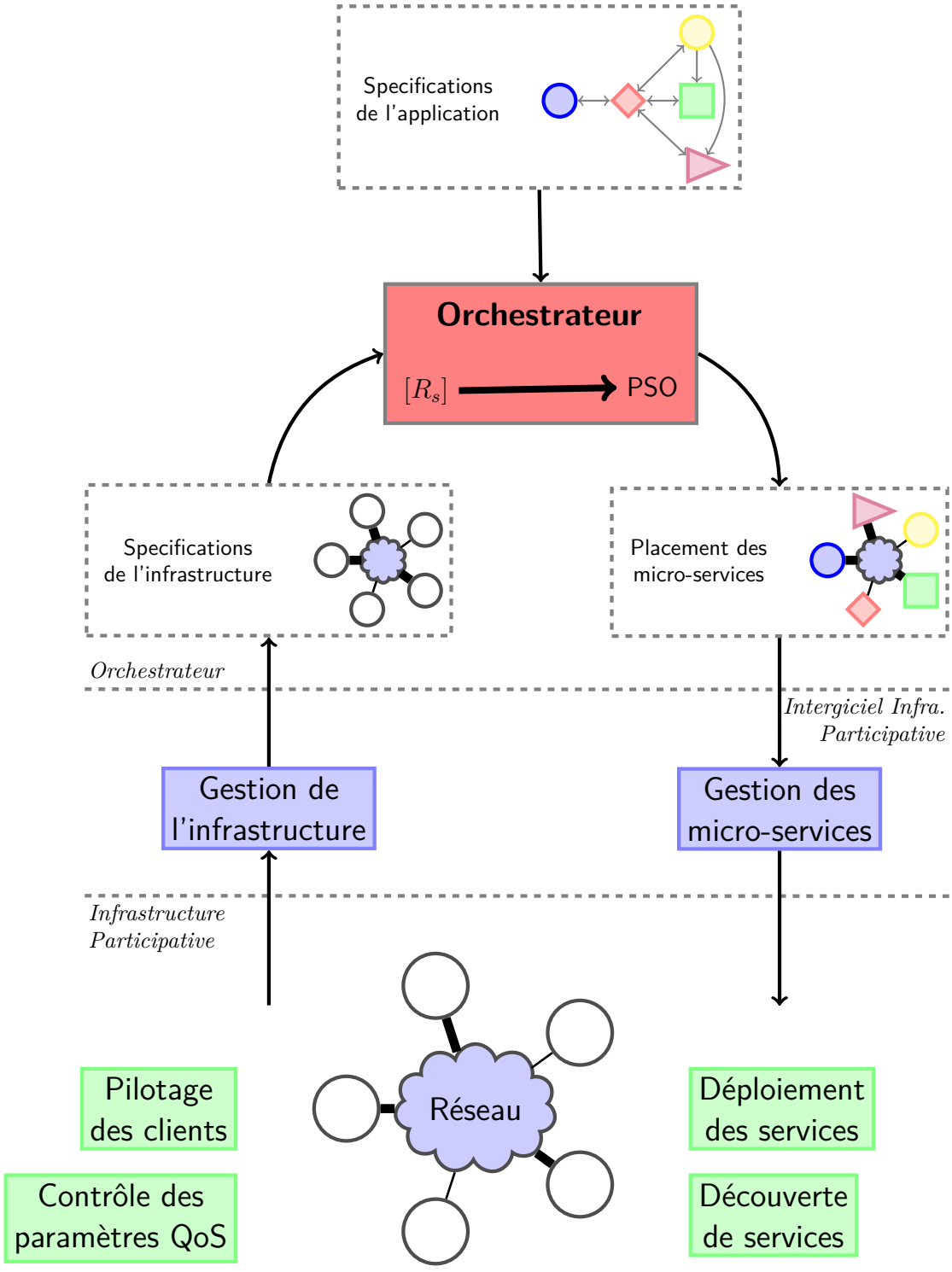

FIGURE 4.3 – Architecture fonctionnelle de la plateforme d'évaluation de l'orchestration.

Les fonctions de décision du placement des micro-services sont intégrées dans un composant dénommé Orchestrateur. Ce composant prend comme entrée les descriptions de l'application et de l'infrastructure participative. Il produit à partir de ces données un placement des micro-services de l'application sur les équipements de l'infrastructure. Ce placement est confié à l'interface avec l'infrastructure participative pour le déploiement effectif des micro-services. Le processus de décision interne à l'orchestrateur est schématisé sur la figure par la représentation des matrices de temps de réponse spécifiques des services $[R_s]$ et de l'heuristique PSO. Nous discuterons par la suite sur la méthode choisie pour la définition des valeurs de ces matrices. L'heuristique PSO calcule à partir de ces valeurs un placement des micro-services, comme décrit dans le chapitre 3.

L'intergiciel fournit une interface permettant à l'Orchestrateur d'accéder aux caractéristiques de l'infrastructure participative nécessaires à la décision du placement de l'application. Cette interface propose une abstraction décrivant les équipements qui composent l'infrastructure, ainsi que les réseaux d'accès résidentiels. L'Orchestrateur sollicite cette même interface afin de transmettre la décision de placement des micro-services pour mise en œuvre sur l'infrastructure participative. Cet intergiciel est spécifique à l'implémentation de l'infrastructure, pour accéder à ses caractéristiques ou y déployer des services, par exemple. Les différentes alternatives d'infrastructures, réelle ou émulée, que nous utiliserons par la suite, s'accompagneront donc d'un intergiciel spécifique pour s'interfacer avec l'Orchestrateur.

La réalisation des tests dans les deux environnements nécessitent d'instrumenter les infrastructures participatives avec certaines fonctions. La première fonction permet de contrôler des clients de l'application. À travers cette fonction, la plateforme peut, de manière coordonnée, initier depuis ces clients des requêtes vers l'application et mesurer leur temps de réponse. Dans le cas de l'infrastructure participative émulée, une interface optionnelle nous permet de contrôler les paramètres de QoS des différents réseaux d'accès résidentiels afin de reproduire différentes conditions d'utilisation. L'infrastructure participative intègre enfin des fonctions de déploiement des micro-services et de découverte de services, spécifiques à l'infrastructure participative considérée et nécessaires au fonctionnement de l'application.

4.2.2 Émulation d'une infrastructure participative

L'utilisation d'une solution émulant une infrastructure participative nous permet d'évaluer le temps de réponse de l'application dans un contexte où nous maîtrisons les conditions d'utilisation. Le premier intérêt est de créer à la demande une émulation d'une infrastructure participative avec des caractéristiques données : nombre d'équipements et technologie de réseaux d'accès résidentiels pour chaque équipement. Le second intérêt est de pouvoir définir de manière contrô-

lée différentes conditions d'utilisation et ainsi vérifier la pertinence et la reproductibilité des résultats obtenus. Cependant, le comportement d'une application déployée sur une infrastructure participative émulée doit se rapprocher le plus possible de celui d'une application déployée dans un cas réel. L'émulation doit donc nous permettre d'exécuter les même micro-services que ceux de l'application évaluée et de les faire communiquer entre eux dans des conditions proches de celles qui peuvent se trouver sur une infrastructure réelle.

Nous avons choisi d'utiliser la solution d'émulation du réseau Mininet [3]. Cet outil offre une émulation réaliste d'un réseau en créant, à partir d'une description donnée, une topologie composée de commutateurs (*switchs*) basés sur l'implémentation OpenVSwitch. Mininet offre la possibilité de modifier en cours d'utilisation différents paramètres de QoS réseau, notamment bande passante, latence et perte de paquet, sur chaque lien du réseau à travers l'outil Netem intégré au système Linux. Nous avons modifié l'implémentation de Mininet pour ajouter à la topologie des liens avec une bande passante asymétrique, afin ainsi d'émuler les réseaux d'accès de type ADSL.

À travers cette émulation de réseau, l'outil Mininet prévoit de connecter entre eux des processus en environnement cloisonné (*chroot*) ou des machines virtuelles, raccordés à la topologie à des emplacements prédéfinis. Dans notre cas, nous souhaitons attacher à ce réseau les micro-services de l'application, livrés sous la forme de conteneurs systèmes. Nous avons donc utilisé la solution Containernet [4], dérivée de Mininet, qui nous permet de déployer les conteneurs des micro-services aux emplacements souhaités dans la topologie émulée du réseau selon le placement décidé par l'orchestrateur.

Dans le cas d'une infrastructure participative avec plusieurs utilisateurs, nous pouvons identifier plusieurs réseaux impliqués :

— les réseaux domestiques sur lesquels sont connectés les équipements hébergeant les services et les clients de l'application,
— les réseaux des opérateurs qui interconnectent plusieurs réseaux domestiques
— le réseau de transit qui interconnecte les réseaux de différents opérateurs.

Nous souhaitons donc construire un modèle générique de topologie permettant de reproduire différents scénarios comportant plusieurs réseaux domestiques et potentiellement plusieurs opérateurs. Mininet nous permet de construire une topologie réseau à base de commutateurs interconnectés par des liens pour lesquels nous pouvons contrôler les paramètres de QoS. Il nous sera intéressant, dans nos scénarios d'évaluation, de contrôler ces paramètres sur des liens repré-

3. `http://mininet.org/`
4. `https://github.com/containernet/containernet`

sentant les réseaux d'accès résidentiels entre des réseaux domestiques et le réseau d'opérateur, ainsi que sur les liens entre les réseau d'opérateurs et le réseau de transit. Nous avons donc défini un modèle pour nos différents cas d'étude à partir d'une topologie en étoile avec 3 niveaux hiérarchiques.

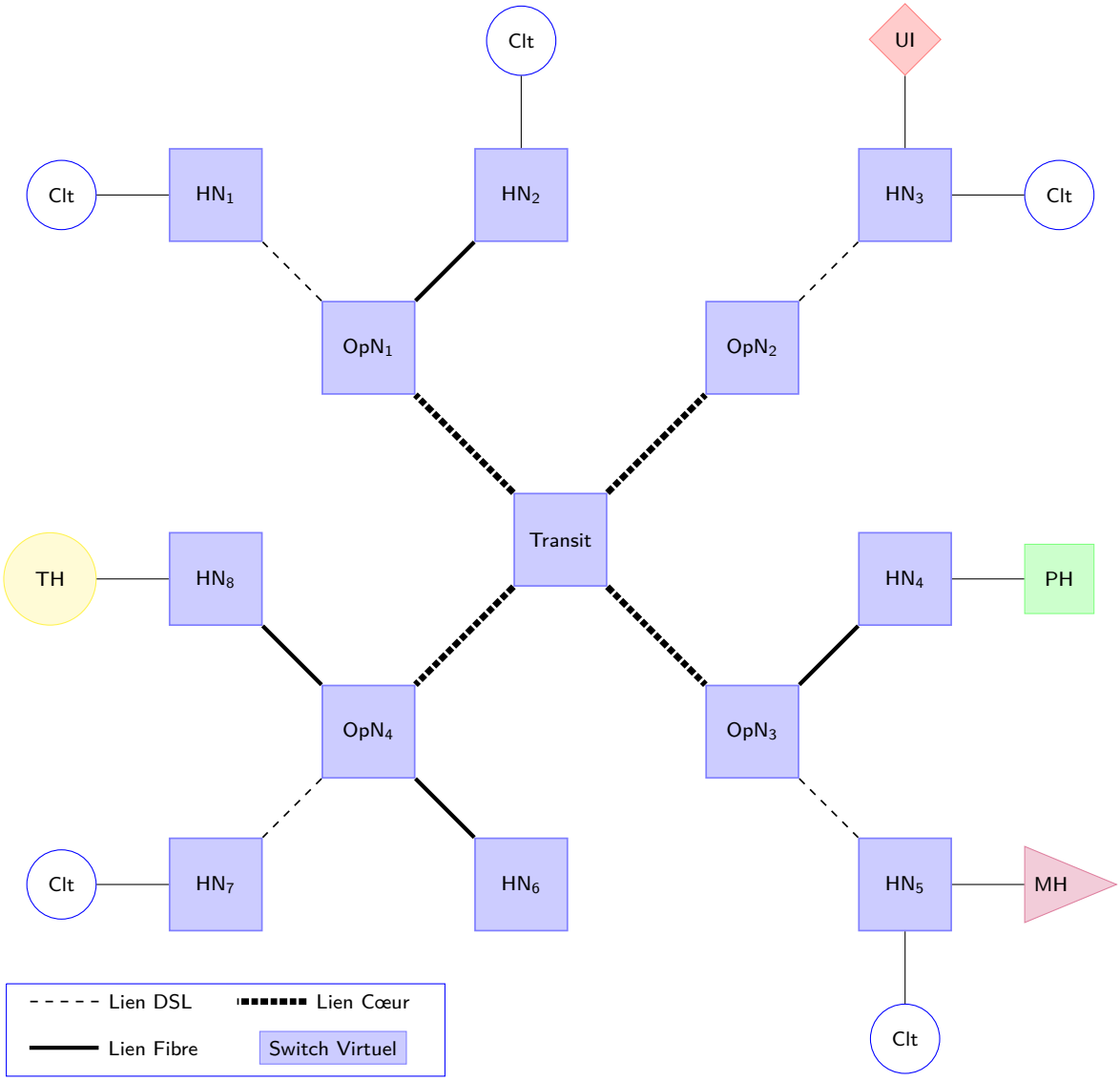

FIGURE 4.4 – Exemple de topologie de réseau émulant une infrastructure participative de 8 réseaux domestiques répartis sur 4 opérateurs, sur laquelle est déployée l'application.

La Figure 4.4 présente, à titre d'exemple, la topologie émulant selon notre modèle un cas d'étude d'une infrastructure participative avec huit réseaux domestiques et quatre réseaux d'opérateur. Cette topologie est construite dans l'outil Containernet à partir de plusieurs *switchs* virtuels.

Chacun de ces commutateurs représente un réseau dans un des niveaux de la hierarchie du modèle : réseau de transit, réseau d'opérateur ou réseau domestique. Nous considérons qu'au sein de ces différents réseaux, il n'y a pas d'influence de la QoS réseau sur les communications. Par contre, nous souhaitons définir des caractéristiques de QoS différentes pour les liens entre ces réseaux selon qu'ils sont des liens DSL, Fibre ou Cœur. L'émulateur nous permet de fixer, pour chacune de ces catégories de liens, des valeurs des paramètres de QoS : bandes passantes montante et descendante, latence, gigue et perte de paquet. Nous pouvons de plus modifier ces paramètres pour chacun des liens de la topologie en cours d'utilisation.

Une fois la topologie construite dans l'émulateur, nous déployons l'application de partage de photos en rattachant ses micro-services aux réseaux domestiques sélectionnés pour ce déploiement. De la même façon, nous attachons un processus, nommé Clt dans la figure, aux réseaux domestiques choisis comme client de l'application pour évaluer son temps de réponse. Lorsqu'un client ou un micro-service de l'application communique avec un autre micro-service, les échanges traversent ainsi la topologie réseau et empruntent les différents liens sur lesquels nous maitrisons les paramètres de QoS réseau. Les temps de réponses observés dépendront donc des variations de ces paramètres, de la même façon que sur une infrastructure réelle. Le traitement des requêtes au niveau de chaque service se faisant sur le même système, nous négligerons les variations potentielles pouvant intervenir sur le temps de traitement.

4.2.3 Cas réel d'infrastructure participative

Nous complèterons notre évaluation par des mesures du temps de réponse de l'application déployée sur une infrastructure participative réelle. Cette infrastructure s'intègre à notre plateforme d'évaluation en substitution de l'infrastructure participative émulée. Elle est constituée d'équipements basés sur une plateforme matérielle commune que nous déployons dans le réseau domestique de participants volontaires. Cette infrastructure va nous permettre d'évaluer les variations du temps de réponse de l'application dans des conditions réelles d'utilisation.

Plateforme matérielle

Les équipements déployés chez les volontaires sont des boitiers APU2, assemblés par l'équipementier PC-Engine [5]. Ces ordinateurs intègrent dans un format très compact (16 cm × 16 cm) une capacité de calcul largement suffisante pour notre application cible (un processeur 4 coeurs 1 GHz et 4 Go de mémoire vive). La Figure 4.5 donne un aperçu de cet équipement. Pour

5. `https://www.pcengines.ch/apu2.htm`

comparaison, un boitier APU2 est cinq fois plus performant que la plateforme Raspberry PI 3. Chaque boitier possède de plus un support de stockage de 32 Go pour les données de l'application. Ce type d'équipement possède plusieurs interfaces réseau Ethernet, mais nous n'en utilisons qu'une seule pour la connexion au réseau domestique. La consommation électrique du boitier est relativement faible, estimée à environ 10 Watts. L'installation d'un tel équipement au domicile des volontaires est donc relativement peu intrusive, ne mobilisant qu'une prise de courant et une connexion derrière la passerelle de l'opérateur de l'accès résidentiel.

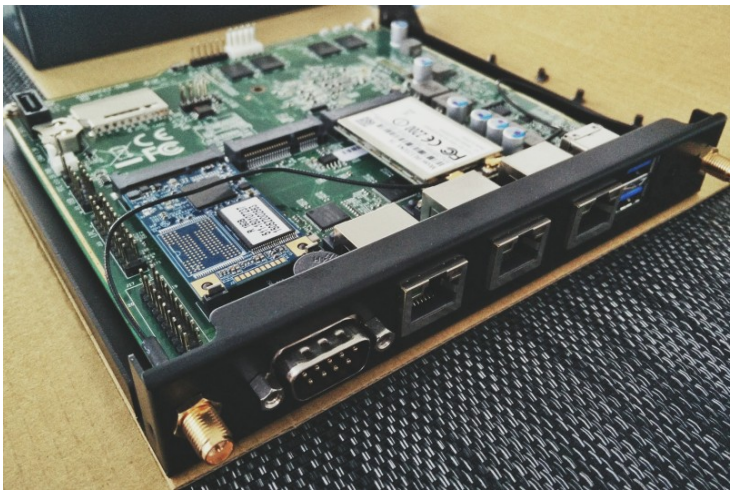

FIGURE 4.5 – Boitier APU2 de l'équipementier PC-Engine.

Chaque boitier embarque un système Linux standard avec les outils de gestion des conteneurs logiciel Docker [6]. Les images des micro-services de l'application de partage de photos sont disponibles depuis un dépôt privé et déployées à la demande sur les différents boitiers. Le déploiement des micro-services est orchestré depuis un serveur capable, à partir du placement des services choisi par l'heuristique, de se connecter directement à chacun des équipements pour y activer les services souhaités. La localisation de chaque service activé est ensuite enregistrée dans le service de découverte Consul [7]. Cet outil permet ensuite aux micro-services de l'application de se découvrir et d'initier leurs échanges.

Afin d'augmenter le nombre d'équipements disponibles, nous avons installé les mêmes composants logiciels sur une machine virtuelle hébergée dans le centre de données d'un fournisseur d'infrastructure à la demande (Tetaneutral, situé à Toulouse). Le Tableau 4.2 présente les configurations des différents équipements utilisés dans notre infrastructure, ainsi qu'une mesure indicative des performances du processeur, évaluées par le système Linux en *Bogomips*.

6. `https://docker.io`
7. `https://consul.io`

Plateforme	Nombre	Processeur	RAM	Bogomips
APU2	7	1x Emb.AMD 1GHz	4 GB	2000
Machine Virtuelle	1	1x Corei7 3GHz	1 GB	6400

TABLE 4.2 – Équipements constituant l'infrastructure réelle

Réseaux utilisés

Chaque boitier a été confié à un volontaire ayant accepté de participer à l'expérimentation, pour être installé dans son réseau domestique. Les réseaux d'accès résidentiels retenus dans ce panel devaient répondre à certains critères techniques. Premièrement ces réseaux doivent offrir un débit montant et descendant suffisant pour le bon fonctionnement de l'application. Ce critère implique que seuls des réseaux FTTH et VDSL ont été retenus. Des tests préliminaires effectués sur des réseaux d'accès ADSL, ont notamment montré que la faible bande passante montante ($<$ 1 Mbit/s) est trop préjudiciable aux performances globales de l'application. Le second critère de sélection des réseaux participant à l'expérimentation est leur capacité à fournir une connectivité IPv6. Nous utilisons en effet l'adressage unicast global IPv6 pour atteindre directement les services activés sur les équipements connectés derrière la passerelle d'accès résidentiel. L'adressage IPv4 nécessite une translation entre l'adressage privé du réseau domestique et l'adressage public de l'Internet. Cette translation restreint les communications directes depuis l'extérieur vers un équipement à l'intérieur du réseau domestique, et donc la capacité d'héberger des services chez les particuliers.

Le Tableau 4.3 donne les paramètres de bande passante et de latence offerte sur les différents réseaux d'accès résidentiel du panel. Ces réseaux correspondent à ceux étudiés dans la section 4.1.2. Il a été demandé aux volontaires de poursuivre un usage habituel de leur connexion à l'Internet. Nous pouvons donc espérer des variations des paramètres de QoS réseau telles qu'observées précédemment. Nous avons ajouté à ce panel de réseaux résidentiels le réseau hébergeant la machine virtuelle (nommé DC dans le tableau) ainsi qu'un réseau situé dans le laboratoire du campus (nommé Lab dans le tableau) pour lequel la bande passante a été réduite à 10 Mbit/s en symétrique. Ce réseau nous permet d'ajouter un nouveau boitier à l'expérimentation.

Nous avons rencontré plusieurs problèmes qui ont rendu complexe l'exploitation de ces différents réseaux pour les mesures que nous souhaitions réaliser. Par exemple, la mise en opération des boitiers sur certains réseaux a nécessité un ajustement manuel des politiques de sécurité au niveau des passerelles d'accès résidentiel, afin d'autoriser les échanges depuis l'Internet vers ces boitiers. Cette étape, rendue nécessaire pour les passerelles de l'opérateur Orange, est tout à fait justifiée pour maintenir la sécurité du réseau domestique. Or il s'avère que cette configuration est

Réseaux résidentiels	BP montante	BP descendante	RTT
FTTH1,2,4,6	100 Mbit/s	100 Mbit/s	25 ms
FTTH3,5	100 Mbit/s	100 Mbit/s	10 ms
VDSL1	20 Mbit/s	80 Mbit/s	12 ms
Lab	10 Mbit/s	10 Mbit/s	10 ms
DC	100 Mbit/s	100 Mbits	20 ms

TABLE 4.3 – Paramètres nominaux de QoS pour les réseaux participant à l'infrastructure réelle.

assez délicate à réaliser et une mauvaise manipulation peut avoir des effets de bords sur l'accès à d'autres services. De plus, des dysfonctionnements au niveau de ces passerelles ont entraîné la perte des configurations des politiques de sécurité, nécessitant de répéter plusieurs fois cette étape au cours de l'expérimentation. Les pertes de connectivité vers les boitiers, qu'elles soient dues à ces problèmes de configuration des passerelles, aux interruptions de l'accès résidentiel ou du réseau électrique évoqués dans la section précédente, ont donc réduit la disponibilité de ces dispositifs pendant nos expérimentations.

4.3 Résultats

Nous allons décrire dans cette section la méthodologie utilisée pour évaluer l'incidence des variations des paramètres de QoS réseau sur le temps de réponse de l'application. Nous comparerons ensuite les résultats obtenus sur cette évaluation sur les infrastructures émulées et réelles.

4.3.1 Mesures sur l'infrastructure participative émulée

Méthodologie d'évaluation

La topologie choisie pour l'évaluation sur une infrastructure émulée comporte 25 réseaux domestiques répartis équitablement entre 5 opérateurs. La moitié des liens émulant l'accès résidentiel sont configurés avec des caractéristiques proches de celles offertes par la technologie FTTH : bande passante montante et descendante de 100Mbit/s symétriques, latence de 10ms. Pour l'autre moitié des liens, émulant une technologie d'accès VDSL, ceux-ci sont configurés avec une bande passante asymétrique de 80Mbit/s sur la voie descendante et 20Mbit/s sur la voie montante, ainsi qu'une latence de 25ms. Les opérateurs sont reliés entre eux à travers le réseau de transit grâce à des liens cœur configurés avec une bande passante de 100Mbit/s symétrique

et une latence de 5ms. Au début de chaque expérimentation, les valeurs définies pour la gigue et la perte de paquet sur chaque lien sont fixées à 0.

Nous choisissons aléatoirement pour l'ensemble des expérimentations un même sous-ensemble de 5 réseaux domestiques qui joueront le rôle de client de l'application. Nous calculerons le temps de réponse global de l'application à partir des mesures recueillies depuis ces clients. Un processus jouant le rôle de client est attaché à chacun de ces réseaux et nous permet de générer de manière contrôlée des requêtes à destination de l'application déployée sur l'infrastructure. La coordination des requêtes des clients nous permet d'assurer que l'application ne traite qu'une requête simultanément, et d'éviter ainsi toute congestion des requêtes au niveau des services. Le temps de réponse de l'application est ensuite calculé à partir des temps mesurés au niveau de l'ensemble des clients pour les réponses à chacune des requêtes selon la formule décrite dans la section 3.3.3. La pondération du temps de réponse par requête q pour chaque client c, $w_{c,q}$, est égale à $0,5$. La fonction pour obtenir le temps de réponse global à partir des temps de réponse pour chaque client est la moyenne arithmétique.

Nous attribuons au début de l'expérimentation les valeurs de bande passante disponible et de latence, décrites ci-dessus, aux réseaux de l'infrastructure émulée. Ces valeurs nous permettent de calculer les temps de réponse spécifiques de chaque micro-service selon le modèle présenté dans la section 3.4.3. Nous initialisons ainsi les différentes matrices $R_{s,q}$ au sein de l'Orchestrateur. L'heuristique PSO détermine ensuite à partir de ces matrices un placement des micro-services selon lequel le temps de réponse de l'application sera proche de l'optimal au début de l'expérimentation, c'est à dire avec des réseaux non perturbés. Une fois l'application déployée selon le placement décidé par l'Orchestrateur, nous commençons l'évaluation de ce temps de réponse tout en modifiant les valeurs des paramètres des QoS réseau.

Nous pouvons modifier pour chacun des liens de l'infrastructure émulée dans ContainerNet les valeurs des différents paramètres de QoS réseaux : les bandes passantes montante et descendante, la gigue moyenne et le taux de perte de paquet. La gigue et la perte de paquet sont des processus aléatoires qui décident pour chaque paquet au moment de sa transmission sur un lien si un délai supplémentaire (gigue) lui est appliqué ou s'il est simplement non-transmis (perte). La valeur choisie de la gigue fixe la valeur maximale du délai supplémentaire pouvant être appliqué. L'émulateur se charge de choisir la gigue pour chaque paquet selon une distribution normale centrée sur la valeur de gigue moyenne donnée. Le taux de perte de paquet fixe lui la probabilité pour un paquet d'être supprimé avant transmission sur le lien. Une fois les liens de l'émulateur configurés avec ces valeurs, il nous suffit de générer un nombre suffisant de requêtes pour évaluer les variations du temps de réponse qu'impliquent la gigue et la perte de paquets. Pour évaluer le temps de réponse de l'application pour un client, le processus génère 4 mesures du temps de

réponse de chaque requête `GET` et `POST`. Cette évaluation est répétée dix fois sur l'ensemble des clients pour avoir un ensemble de données permettant de juger des variations possibles.

Les variations de la bande passante disponibles sont émulées de manière différente. L'émulateur ContainerNet ne peut pas faire varier aléatoirement la bande passante disponible sur un lien entre chaque paquet, comme il peut le faire pour la gigue. Nous avons donc choisi de modifier ces valeurs en reconfigurant les valeurs de bande passante montante et descendante après chaque mesure du temps de réponse de l'application par l'ensemble des clients. Nous définissons pour l'ensemble des liens la bande passante disponible comme un pourcentage de la bande passante nominale du lien. Nous calculons ainsi pour chaque lien une valeur de bande passante moyenne en appliquant ce pourcentage à la valeur nominale de la bande passante. La nouvelle valeur de la bande passante attribuée au lien est ensuite choisie dans une distribution normale centrée sur cette valeur moyenne, avec une variance de 50% de la moyenne, et aura comme valeur maximum celle de la bande passante nominale. L'évaluation du temps de réponse de l'application pour l'ensemble des clients est répétée dix fois. La bande passante de chaque lien prendra ainsi à chaque évaluation des valeurs différentes, tout en respectant la bande passante moyenne attendue selon les conditions d'utilisation émulées.

Résultats

La Figure 4.6 montre l'évolution des temps de réponse de l'application et de chaque requête en fonction de différents intervalles de variations de la bande passante disponible. Ces intervalles varient entre 99% et 40% de la valeur de la bande passante totale sur chaque lien. Les autres paramètres de QoS (latence, taux de perte de paquets) restent fixés pour chaque lien à leur valeur nominale. Les courbes représentent l'évolution de la distribution des temps de réponse mesurés, représentée par les différents n-ème centiles. L'évolution du temps de réponse de l'application montre, comme nous pouvions nous y attendre, qu'il augmente lorsque la bande passante disponible moyenne diminue. Pour une bande passante disponible en moyenne à 50% de la bande passante totale, le temps de réponse mesuré est en moyenne supérieur de 30% à celui mesuré lorsque la bande passante est disponible à 99%. L'allure de la courbe caractérisant le temps de réponse de l'application montre une augmentation linéaire jusqu'à une disponibilité de 99% à 75% de la bande passante, puis ensuite une augmentation exponentielle. Ce changement d'allure des courbes peut s'expliquer en observant les variations des temps de réponse pour les requêtes `GET` et `POST`.

Le temps de réponse moyen de l'application à la requête `GET`, mesuré pour une bande passante disponible en moyenne à 50%, est 20% supérieur à celui mesuré pour une bande passante tota-

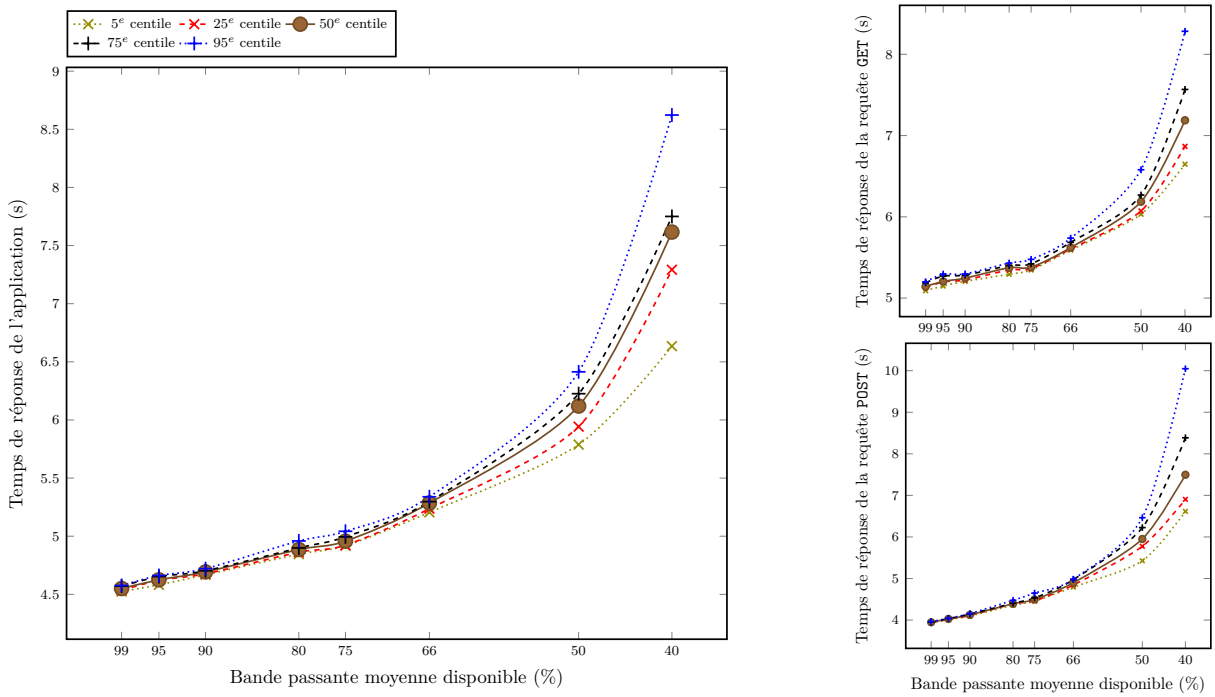

FIGURE 4.6 – Evolution du temps de réponse de l'application et des différentes requêtes en fonction de la bande passante moyenne disponible.

lement disponible. Pour la requête `POST`, la variation du temps de réponse moyen sur ce même intervalle est d'environ 40%. Cette différence peut s'expliquer par le fait que les échanges entre micro-services impliqués dans le traitement de la requête `GET` représentent moins de données que dans les échanges liés au traitement de la requête `POST`. Le temps de réponse de cette dernière requête sera donc plus impacté par la diminution de la bande passante disponible. Ce constat confirme les informations de profilage des micro-services que nous avons présentées lors du chapitre précédent dans la Table 3.1.

La Figure 4.7 montre l'évolution des temps de réponse de l'application et de chaque requête en fonction de la gigue sur les réseaux d'accès résidentiels, dont la valeur varie pour chaque lien entre entre 0% et 100% de sa latence nominale. Les autres paramètres de QoS (bandes passantes montante et descendante, taux de perte de paquets) restent fixés pour chaque lien à leur valeur nominale. Les courbes représentent l'évolution de la distribution des temps de réponse mesurés, représentée par les différents n-ème centiles. Nous observons que le temps de réponse de l'application augmente avec la gigue, les courbes présentant une allure exponentielle. Le temps de réponse moyen mesuré pour une valeur de gigue de 100% de la latence nominale est 5 fois supérieur au temps de réponse mesuré dans des conditions sans gigue. Une augmentation similaire s'observe pour les temps de réponse de la requête `GET`. Les courbes pour la requête `POST`

présentent une rupture marquée au delà d'une valeur de 75% de gigue. Le temps de réponse moyen de cette requête est, pour 75% de gigue, 2 fois supérieur à celui mesuré sans présence de gigue, alors qu'il est 3 fois supérieur pour la requête `GET`. Cette différence peut s'expliquer par le fait que les échanges entre micro-services pour le traitement de cette dernière requête implique des messages de petites tailles, et donc avec des temps de réponse plus sensibles à la variation de la latence.

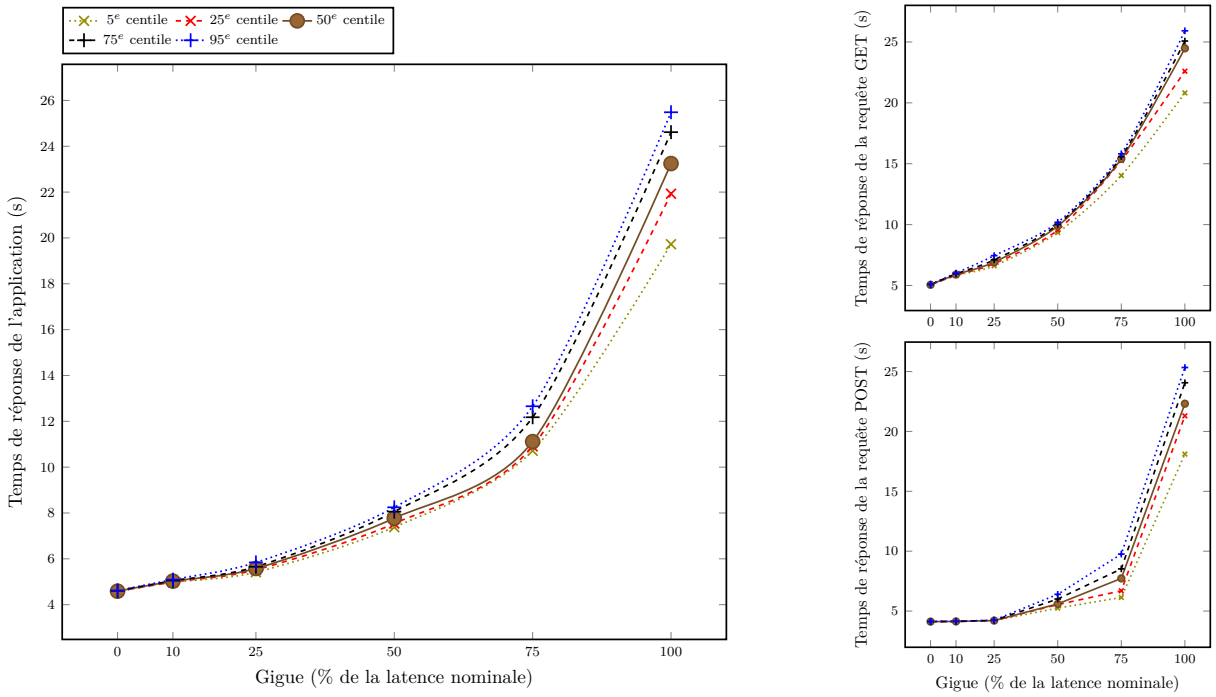

FIGURE 4.7 – Evolution du temps de réponse de l'application et des différentes requêtes en fonction de la gigue sur une infrastructure participative émulée.

La Figure 4.8 montre l'évolution du temps de réponse de l'application et de chaque requête en fonction du taux de perte de paquets sur le réseau. Ce taux correspond à la probabilité qu'un paquet ne soit pas retransmis au niveau de chaque réseau d'accès résidentiel, probabilité variant entre 0% et 1%. Les courbes représentent l'évolution de la distribution des temps de réponse mesurés, représentée par les différents n-ème centiles. Nous observons que le temps de réponse de l'application est stable après une légère augmentation de 20% pour un taux de perte de 0% à 0.2%. Nous en concluons que la perte de paquets, pour des taux représentant des conditions réalistes d'utilisation, n'a que peu d'influence sur le temps de réponse de l'application. Les échanges entre micro-services étant transporté par le protocole TCP, nous expliquons ces résultats par le faible délai supplémentaire qu'impliquent les retransmissions par rapport au temps total nécessaire pour ces échanges.

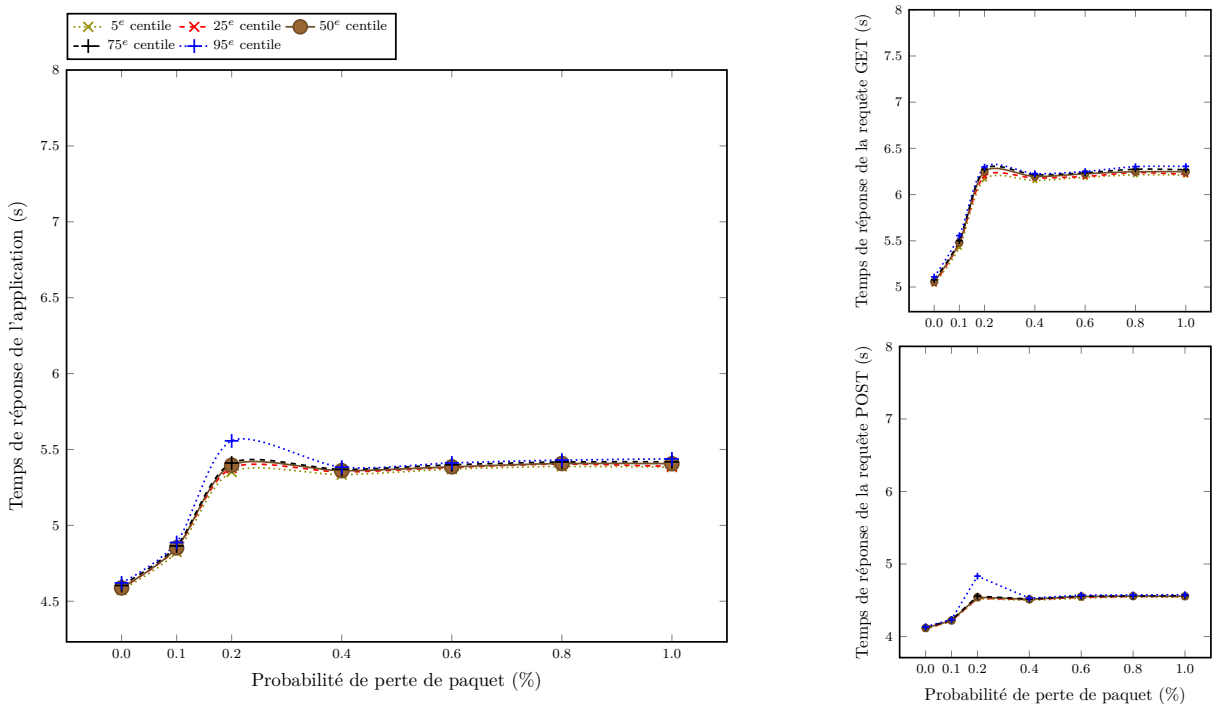

FIGURE 4.8 – Evolution du temps de réponse de l'application et des différentes requêtes en fonction du taux de perte de paquets sur une infrastructure participative émulée.

4.3.2 Mesures sur l'infrastructure participative réelle

Méthodologie d'évaluation

Plusieurs expérimentations ont été nécessaires pour réaliser l'évaluation du temps de réponse de l'application sur l'infrastructure participative réelle. En effet, pour les raisons évoquées précédemment, les équipements de cette infrastructure n'étaient pas tous disponibles en permanence. Il nous est possible de démarrer une expérimentation dès qu'un nombre suffisant d'équipements sont disponibles. Ce nombre est fixé à 5 équipements disponibles sur les 8 possibles. L'expérimentation cesse dès qu'un des équipements n'est plus disponible. Nous avons ainsi réalisé 16 expérimentations représentant une évaluation du temps de réponse de l'application sur plus de 200 heures d'utilisation. Ces expérimentations étant réalisées à différents moments du jour ou de la semaine, et avec des équipements différents, nous avons capturé les variations du temps de réponse de l'application dans différentes conditions d'utilisation.

Les paramètres de QoS des réseaux d'accès (bandes passantes montante et descendante disponibles, latence) sont mesurés en permanence au cours d'une expérimentation selon le dispositif

décrit dans la section 4.1.2. Nous supposons que ces mesures, réalisées en même temps que les échanges entre micro-services, n'ont pas d'influence sur les mesures de temps de réponse de l'application. Les valeurs de bande passante disponible et de latence, mesurées en début d'expérimentation, nous permettent de calculer des valeurs initiales pour les matrices de temps de réponse $R_{s(q)}$ selon le modèle présenté dans la section 3.4.3. L'heuristique PSO calcule ensuite à partir de ces matrices un placement des micro-services permettant d'avoir un temps de réponse de l'application proche de l'optimal dans les conditions de départ de l'expérimentation. Une fois que l'application est déployée selon le placement décidé par l'Orchestrateur, nous commençons l'évaluation du temps de réponse tout en mesurant les valeurs des paramètres des QoS réseau.

Ne disposant que d'un nombre limité d'équipements pour ces expérimentations, nous avons choisi d'évaluer le temps de réponse de l'application depuis l'ensemble de ces équipements qui ont donc tous joué le rôle de client de l'application. De la même manière que lors de l'évaluation sur infrastructure émulée, un client est constitué d'un processus générateur de requête contrôlé à distance. Ces clients sont déployés sur chaque équipement au début de chaque expérimentation et nous contrôlons depuis un serveur distant l'envoi des requêtes vers l'application. Cette coordination des requêtes nous assure que l'application ne traite qu'une seule requête simultanément au moment de l'évaluation du temps de réponse. Pour chaque client, un temps de réponse moyen est calculé à partir de quatre mesures successives du temps de réponse de chaque requête `GET` et `POST` et avec la même pondération $w_{c,q}$ égale à 1. Nous déduisons ensuite le temps de réponse global en faisant la moyenne arithmétique des temps de réponse mesurés pour chaque client.

Résultats

Chaque expérimentation s'est déroulée avec des équipements différents dans des conditions d'utilisation différentes. Pour pouvoir comparer les résultats des mesures réalisées, nous avons cherché à caractériser l'évolution des conditions d'utilisation au cours d'une expérimentation en échantillonnant les paramètres de QoS réseau et le temps de réponse mesuré sur une période de 2 heures. La disponibilité de la bande passante et la gigue pour chaque échantillon sont calculées en prenant comme référence la bande passante maximale et la latence minimale observées sur l'ensemble de l'expérimentation. La moyenne de l'écart des valeurs mesurées sur l'échantillon par rapport à cette référence nous donne une valeur pour chaque paramètre caractérisant les conditions d'utilisation de l'application.

La Figure 4.9 présente les valeurs obtenues pour les échantillons issues de nos expérimentations. Chaque point représente un échantillon dont les coordonnées rendent donc compte des différentes conditions rencontrées pendant la période concernée. Nous observons qu'une majorité des points

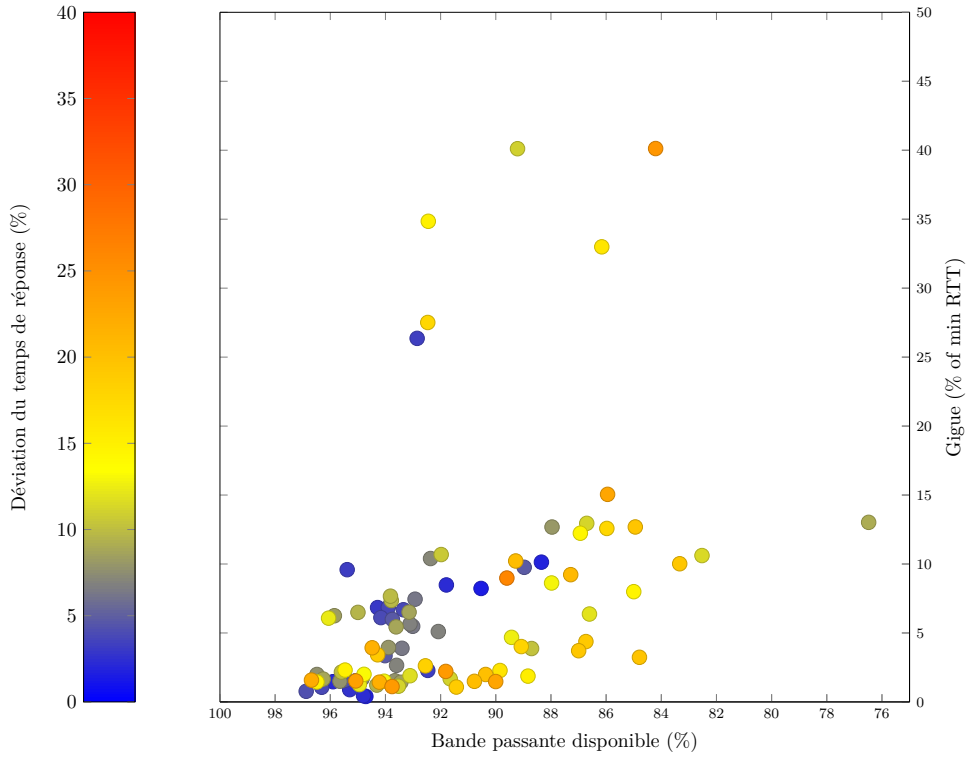

FIGURE 4.9 – Déviations du temps de réponse de l'application observées sur l'infrastructure participative réelle dans différentes conditions d'utilisation.

se concentrent dans un domaine où la bande passante disponible est importante ($> 84\%$) et de gigue faible ($< 15\%$). Les points les plus extrêmes sont eux représentatifs des conditions les moins favorables avec une disponibilité minimale de la bande passante de 76% ou une gigue maximale de 40%. Cette répartition nous montre que les expérimentations réalisées sur l'infrastructure participative réelle se sont déroulées en moyenne dans des conditions relativement bonnes en ce qui concernent les paramètres de QoS réseau. Nous n'avons pas rencontré lors de ces mesures de conditions où la bande passante d'une majorité d'équipements ait été fortement réduite, ou que la gigue ait été au delà des conditions réalistes d'utilisation définies précédemment.

Sur la période de 2 heures correspondant à un échantillon, notre dispositif nous a permis d'obtenir au minimum 5 mesures du temps de réponse global de l'application. Nous calculons la déviation de la moyenne de ces temps de réponse par rapport au temps de réponse minimal mesuré pendant l'expérimentation dont est issu l'échantillon. Cette déviation est représentée par la couleur de chaque point de la Figure 4.9. Nous constatons qu'une grande majorité des déviations du temps de réponse observées sont relativement faibles : 75% des échantillons présentent une déviation inférieure à 30%. Nous remarquons de plus que pour une majorité des échantillons correspondant

à des conditions favorables de QoS réseau (bande passante disponible à plus de 94% et gigue inférieure à 10%), le temps de réponse de l'application présente une déviation faible inférieure à 10%. Ces résultats confirment la stabilité du temps de réponse de la solution de placement choisie par l'heuristique dans des conditions réalistes d'utilisation.

Lorsque ces conditions de QoS réseau se dégradent, nous observons une augmentation de la déviation du temps de réponse, notamment lorsque la disponibilité de la bande passante diminue. Cette observation confirme les résultats obtenus précédemment avec l'émulateur. L'augmentation de la déviation du temps de réponse avec celle de la gigue est par contre moins flagrante dans nos mesures. Nous pouvons apporter deux explications à ce constat. D'une part, peu d'échantillons montrent une gigue supérieure à 20%, les résultats obtenus ne sont donc peut-être pas significatifs de ceux attendus dans ces conditions. D'autre part notre dispositif de mesure de la gigue n'est peut-être pas suffisamment fiable et donne dans certains cas des valeurs de gigue surévaluées.

Nous observons également que certains échantillons présentent une déviation du temps de réponse significativement différentes à d'autres mesures effectuées dans des conditions similaires. Des déviations inférieures à 10% sont ainsi observées pour quelques échantillons présentant des conditions de QoS pourtant défavorables (par exemple pour une bande passante disponible entre 88% et 92%, et une gigue d'environ 10%). Comme précédemment, un manque de précision dans les mesures des conditions de QoS réseau peut expliquer un mauvais positionnement relatif de ces points par rapport aux autres échantillons. D'autres échantillons montrent par contre une déviation plus importante que celle attendue. Ainsi, dans des conditions de QoS réseau plutôt favorables (bande passante disponible entre 94% et 97%, et une gigue inférieure à 5%), nous observons pour certains échantillons une déviation du temps de réponse jusqu'à 25%.

Lors de l'étude de ces résultats, avons constaté que plusieurs de ces échantillons montrant un temps de réponse plus élevé proviennent d'une même expérimentation. La Figure 4.10 montre les paramètres de QoS réseau ainsi que le temps de réponse de l'application pour cette expérimentation. Nous constatons que lors de cette expérimentation, le temps de réponse de l'application augmente sensiblement d'une valeur située autour de 18 secondes à une valeur supérieure à 24 secondes, soit une déviation de plus de 30%. Nous remarquons que les paramètres de QoS réseau pour les différents équipements impliqués dans cette expérimentation sont relativement stables, sauf pour l'équipement VDSL1 qui présente une augmentation importante de la latence (+40%) et une diminution de la bande passante montante disponible (−50%). Comme ces variations ne se présentent que sur un équipement alors que les paramètres pour les autres équipements sont stables, les conditions d'utilisation, évaluées par la moyenne des déviations de ces paramètres, seront considérées comme favorables. Or il s'avère qu'au moment de cette expérimentation, cet équipement hébergeait le service ThumbHub et était donc chargé de générer les vignettes des

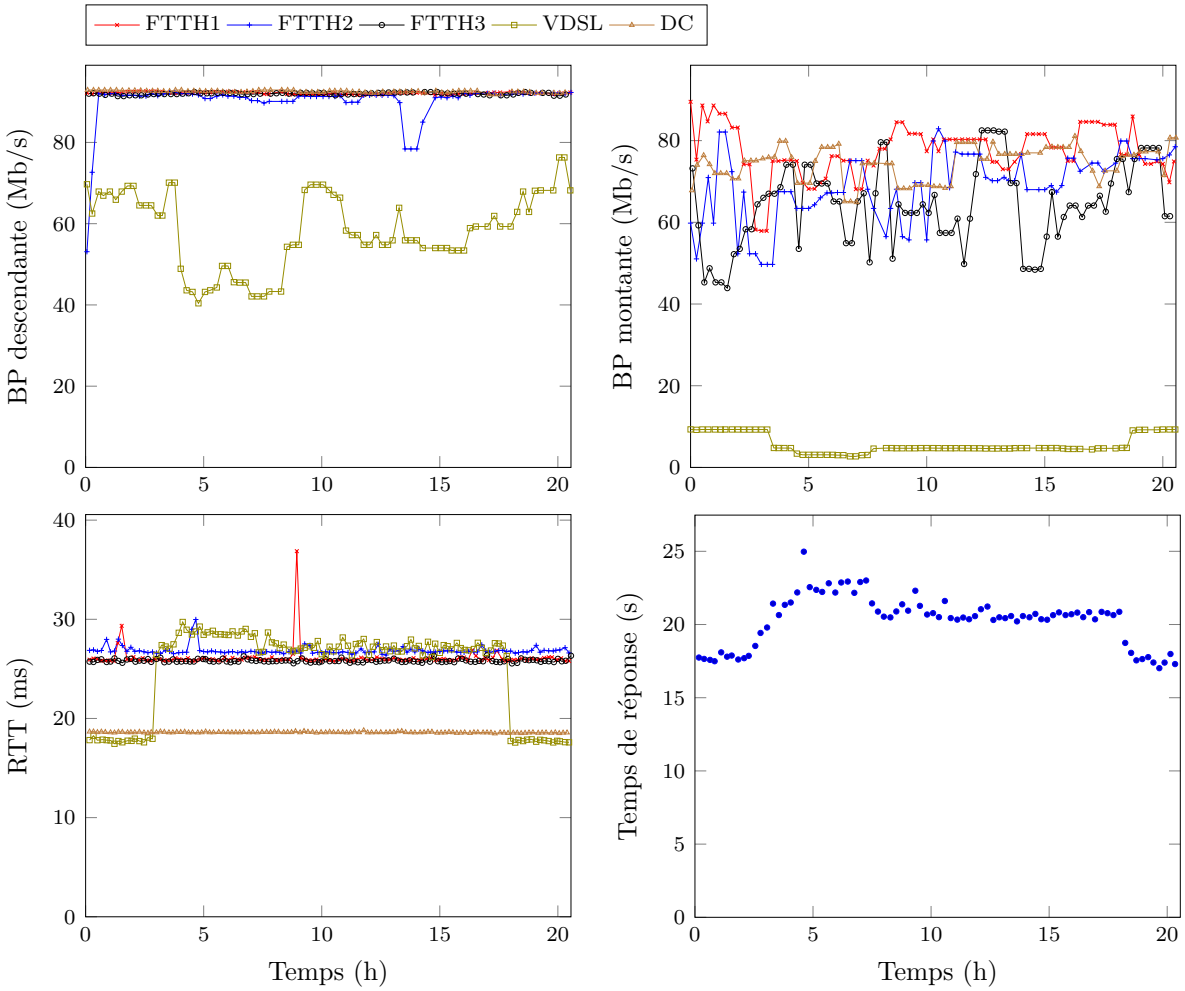

FIGURE 4.10 – Variations des paramètres de QoS réseau et du temps de réponse lors d'une expérimentation sur une infrastructure participative réelle.

photos. La dégradation des paramètres de QoS pour cet équipement en particulier a donc eu un impact plus important sur le temps de réponse de l'application que si l'équipement n'hébergeait aucun service.

4.4 Conclusions

Nous avons illustré dans ce chapitre l'influence de la variation des paramètres de QoS sur le temps de réponse de l'application déployée sur une infrastructure participative. Après avoir caractérisé les intervalles de variations de ces paramètres pour des conditions réalistes d'utilisation, nous

avons mesuré le temps de réponse de l'application déployée sur une infrastructure émulée où nous contrôlons les valeurs de ces paramètres. Pour l'application de partage de photos, nous avons observé que l'augmentation de la gigue s'accompagne d'une forte augmentation du temps de réponse, alors que la même variation due à une diminution de la bande passante disponible est moins importante. La perte de paquet, dans les conditions réalistes d'utilisation, n'a que peu d'impact sur le temps de réponse, grâce à la résilience des protocoles utilisés pour le transport des messages entre micro-services.

Nous avons reproduit cette étude sur une infrastructure participative réelle, où les variations des paramètres de QoS des réseaux sont induites par un véritable trafic utilisateur. Nous observons de la même manière qu'une dégradation des conditions d'utilisation implique une augmentation du temps de réponse de l'application. Les mesures effectuées permettent de mettre en évidence une corrélation entre la diminution de la bande passante disponible et une déviation du temps de réponse. La gigue est un phénomène plus difficile à détecter dans notre dispositif qui ne nous a donc pas permis de mettre en évidence son influence sur les performances de l'application.

Plusieurs échantillons de mesures ont attiré notre attention car ils présentent des déviations du temps de réponse relativement importants dans des conditions d'utilisation apparaissant comme peu perturbées. Nos méthodes de mesures des paramètres de QoS ne nous permettent pas de capturer complètement les conditions de communication entre tous les équipements de l'infrastructure. Les conditions mesurées pour certains de ces résultats atypiques peuvent donc être remises en question. Cependant, nous avons observé parmi ces résultats des déviations importantes du temps de réponse s'observant lorsque des dégradations des paramètres de QoS sont localisées sur un équipement qui héberge un des micro-services de l'application. Il apparait donc que pour pouvoir expliquer des variations du temps de réponse, nous devons étudier les paramètres de QoS non seulement de manière globale, mais aussi particulièrement sur chacun des équipements impliqués dans l'hébergement de l'application, en fonction du service hébergé.

Chapitre 5

Adaptation dynamique du placement des micro-services

Il est bien connu qu'un ingrédient essentiel du succès est de ne pas savoir que ce que l'on tente ne peut pas être fait.
— **Terry Pratchett**, *La Huitième Fille*

Nous avons identifié dans le chapitre précédent que des variations de certains paramètres de QoS réseau influencent significativement le temps de réponse de l'application. Dans ce chapitre nous allons étudier comment, dans ces circonstances, nous pouvons adapter dynamiquement le placement des micro-services pour maintenir les performances de l'application. Ces variations, observées sur un équipement et induites par les usages concurrents sur le réseaux d'accès résidentiel, ont une incidence plus ou moins importante sur le temps de réponse global de l'application, selon que cet équipement héberge ou non un des micro-services de l'application. Dans ce dernier cas, les variations des paramètres de QoS se traduiront par une dégradation du temps de réponse de l'application pour l'ensemble des clients de l'application.

Notre proposition pour réduire l'impact de ces perturbations est d'adapter le placement des micro-services aux nouvelles conditions d'utilisation. Le placement utilisé pour le déploiement initial de l'application est choisi selon des valeurs de temps de réponse des micro-services caractéristiques de certaines conditions d'utilisation. Lorsque ces conditions changent, ces temps de réponse ne sont plus les mêmes et les performances de l'application suivant ce placement peuvent s'en retrouver dégradées. En utilisant un placement alternatif qui n'utiliserait pas les équipements sollicités par un usage concurrent, il serait alors possible d'éviter une dégradation du temps de réponse, et peut-être même de l'améliorer. Nous supposons ici qu'il est possible de modifier les placements des micro-services en cours d'éxécution de l'application, sans impact pour l'utilisateur. Cela suppose notamment que les données de l'application restent disponibles pour les micro-services concernés, quelque soit leur placement.

Pour que cette adaptation soit pertinente, il nous faut d'abord spécifier les conditions dans lesquelles il est nécessaire de modifier le placement. Nous allons utiliser, pour représenter ces

conditions, des indicateurs qu'il nous faudra surveiller pour suivre leur évolution. L'analyse de ces indicateurs va orienter la décision de l'adaptation du placement. Il nous faudra ensuite, si les conditions pour une adaptation sont réunies, choisir un nouveau placement des micro-services, adapté aux nouvelles conditions d'utilisation. Les performances de l'application déployée selon ce nouveau placement seront ensuite évaluées pour valider la pertinence de l'adaptation. L'ensemble de ces actions forment un processus d'adaptation dynamique que nous allons définir dans cette section après avoir parcouru les solutions déjà proposées dans la littérature. Nous définirons les indicateurs que nous avons choisis comme caractéristiques des conditions d'utilisation et les stratégies qui se basent sur ces indicateurs pour décider ou non de l'adaptation du placement, ainsi que du nouveau placement à déployer. Dans une dernière section, nous évaluerons ces stratégies d'adaptation dans des conditions d'utilisation réelles sur notre infrastructure participative de test.

5.1 Travaux existants

Les systèmes informatiques adaptatifs sont l'objet d'études depuis le début des années 2000 [45]. Ces systèmes possèdent des propriétés leur permettant de continuer à assurer leurs fonctions malgré des changements dans les conditions d'utilisation. Pour cela, ces systèmes s'adaptent dynamiquement à ces changements en modifiant eux-même leur fonctionnement, sans intervention directe d'un opérateur. Ces systèmes autonomes sont souvent référencés sous le terme *self-*$*$, car ils assurent différentes propriétés d'autorégulation : auto-configurant (*self-configuring*), auto-réparant (*self-healing*), auto-optimisant (*self-optimizing*), auto-protecteur (*self-protecting*), etc.

Des systèmes montrant de telles propriétés ont trouvé plusieurs domaines d'applications, dont le plus intéressant dans notre cas est la gestion automatique d'applications dans un contexte info-nuagique. Plusieurs études citées dans [84] ont ainsi pour objectif d'assurer dans ce contexte la qualité de service d'applications à travers des mécanismes d'autorégulation. Nous avons analysé dans les publications existantes les solutions offrant un mécanisme d'adaptation autonome et dynamique en fonction de paramètres de qualité de service réseau ou applicative. Nos recherches de solutions se sont concentrées sur les domaines proches de notre problématique que sont les compositions de service, le calcul en bordure (*Edge/Fog Computing*) et les systèmes info-nuagiques sensibles à la qualité de service (*QoS-aware Cloud*). Bien qu'intervenant sur des applications de formes différentes (monolithique, multi-tiers ou composition), ces solutions cherchent à adapter le placement de ces applications sur des équipements distribués pour maintenir une certaine qualité de service.

	Contexte	Adaptation				Evaluation	
	Type d'application	Paramètre(s) considéré(s)	Événements déclencheurs	Stratégie	Actions	Méthodologie	Environnement
Compositions de services							
MOSES [19]	Composition de services	Temps de réponse / Fiabilité	Violation SLA	Réactive	Nouvelle sélection services	-	-
ProAdapt [11]	Composition de services	Temps de réponse	Violation SLA	Proactive	Nouvelle sélection services	Emulation	Réseau Local
Adapt-Broker [73]	Composition de services	*User defined*	Violation SLA	Réactive	Nouvelle sélection services	Emulation	-
Edge/Fog Computing							
FogFrame [85]	Ensemble de services	Disponibilité	Départ / Arrivée de noeuds	Réactive	Redéploiement	Emulation	Datacenter
SEGUE [98]	Monolithique	Temps de réponse	Violation SLA	Proactive	Migration de service	Simulation	-
FogPlan [96]	Monolithique	Latence / Coût	Violation SLA	Proactive	Nouveau déploiement	Simulation	Traces de trafic réel
Qos-aware Cloud							
BASMATI [10]	Monolitique (VM)	Latence	Violation SLA	Réactive	Redéploiement	Simulation	-
FAHP [55]	Multi-tiers	Temps de réponse / Bande passante	SLA et états système	Proactive	Dimensionnement	Simulation	-

TABLE 5.1 – Comparaison de solutions d'adaptation identifiées dans la littérature

Nous avons étudié chacune de ces propositions selon différents critères caractérisant l'adaptation mise en œuvre et l'évaluation de la solution. Le résultat de cette analyse est présenté dans le tableau 5.1. Chacune des solutions définit un ou plusieurs paramètres de qualité de service de l'application que l'adaptation cherchera à maintenir ou optimiser. Nous remarquons que le temps de réponse de l'application, paramètre que nous considérons aussi dans notre étude, est celui le plus souvent mentionné. Certaines solutions d'adaptation visent à maintenir des paramètres supplémentaires comme la fiabilité, le coût ou la bande passante. Ces solutions proposent d'observer l'évolution de ces paramètres pour décider du déclenchement de l'adaptation du placement de l'application. Les solutions cherchant à garantir un temps de réponse utilisent, comme évènement déclencheur de cette décision, le dépassement d'une valeur seuil (ou violation de SLA). Dans notre cas, il est difficile de définir une valeur seuil absolue pour le temps de réponse de l'application attendu sur une infrastructure participative, qui n'apporte aucune garantie en terme de disponibilité ou de performances. Nous nous intéressons plus aux variations importantes du temps de réponse liées aux changements de conditions d'utilisation. Dans cet objectif, nous définirons nos seuils non pas par une valeur fixe correspondant à un niveau de service garanti, mais sous forme d'une déviation maximale acceptable du temps de réponse par rapport aux valeurs mesurées précédemment.

Les solutions d'adaptation que nous avons analysées se caractérisent par la manière dont est évalué ce critère de décision : certaines solutions évaluent ce critère à partir de l'état actuel sur le système, alors que d'autres s'appuient sur des prévisions de l'évolution du critère. Nous distinguons ainsi deux catégories de stratégies d'adaptation : réactives et proactives. Une solution utilisant une stratégie d'adaptation réactive réalise des mesures des paramètres qu'elle a choisi d'observer sur le système en cours d'utilisation et évalue les valeurs obtenues selon le critère de décision qui lui est propre. Un tel système décide donc d'une adaptation une fois le critère d'adaptation rempli, comme par exemple lorsque des mesures dépassent un certain seuil. Dans notre cas, ce type de stratégie est envisageable en ajoutant à notre dispositif la remontée de mesures de paramètres pertinents pour décider de l'adaptation du placement. Une solution utilisant une stratégie proactive analyse l'évolution des paramètres pour anticiper le moment où le critère d'adaptation pourrait être rempli. Un tel système peut ainsi déclencher une adaptation avant que les paramètres observés dépassent le seuil. Ce type de solution est envisageable lorsque l'évolution des paramètres est prédictible. Dans notre cas cependant, nous n'avons pas d'indice probant d'une prédictibilité de l'évolution du temps de réponse. Les auteurs de [96] suggèrent qu'une adaptation à intervalles de temps réguliers peut être une stratégie proactive suffisante dans certains cas. Nous nous inspirerons de cette solution pour définir une première stratégie proactive. Ces deux approches proactives et réactives sont potentiellement utilisables dans notre contexte. Nous tenterons d'en évaluer la pertinence dans cette étude.

Nous soulignons que les contributions que nous avons identifiées appuient l'évaluation de leur solution sur des simulations ou dans des environnements émulés, avec des variations des paramètres de QoS réseau contrôlées ou générées à partir de traces de trafic. Ces dispositifs permettent de valider le comportement de la solution sur des cas d'usage bien identifiés. Les mesures réalisées sur l'infrastructure participative réelle dans le chapitre 4 ne permettent pas dans notre cas de distinguer un ensemble de situations favorables ou défavorables dans lesquelles tester notre système d'adaptation. Cependant nous pourrons profiter de la disponibilité de cette infrastructure pour tester notre solution dans des conditions réelles d'utilisation. Les auteurs de [66] soulignent d'ailleurs l'importance de la validation de la solution proposée par le système d'adaptation. Même si cette solution semble idéale vis à vis d'un modèle pour améliorer la situation, il est important d'en vérifier sa pertinence d'un point de vue utilisateur. Les études identifiées dans cet article montrent que cette validation peut permettre des améliorations dans la solution d'adaptation et constituer un retour pouvant participer à l'entrainement d'algorithmes basés sur l'apprentissage [54]. Dans notre cas, la solution d'adaptation sera directement déployée dans des conditions réelles d'utilisation et nous pourrons mesurer la pertinence de cette solution à partir de mesures du temps de réponse.

5.2 Processus d'adaptation du placement

Nous avons besoin dans notre cas d'un processus d'adaptation permettant de corriger automatiquement le placement des micro-services selon la variation des conditions d'utilisation. Ce processus se doit d'abord d'agir de manière pertinente sur notre système. L'adaptation du placement doit être réalisée à des moments opportuns, ce qui implique la supervision de l'évolution des performances du système selon des conditions d'utilisation. En cas de dégradation de ces performances, l'adaptation du placement doit ensuite être suffisamment rapide pour que l'utilisateur ne la perçoive pas ou très peu. Le processus doit être capable de prendre une décision d'adaptation rapidement selon l'évolution des conditions, ce qui implique une supervision fréquente et des algorithmes de décision à faible temps d'exécution. Enfin nous attendons du processus qu'il agisse de manière autonome. Il doit pouvoir prendre de lui-même les bonnes décisions, ou à défaut, corriger ses erreurs, ce qui implique une évaluation automatique de la solution d'adaptation choisie. Nous chercherons de plus à définir un processus d'adaptation qui puisse facilement s'intégrer au système d'orchestration des micro-services sur une infrastructure participative réelle, présenté dans le chapitre 4. Nous pourrons ainsi tester évaluer la pertinence du processus d'adaptation dans des conditions réelles.

5.2.1 Chaine de décision

La plupart des solutions d'adaptation que nous avons identifiées dans la littérature se basent sur une boucle de rétroaction entre l'observation, la décision et la mise en action de l'adaptation. Une architecture générique pour un tel processus d'adaptation a été formalisée par les travaux d'IBM [51] sous le nom de MAPE-K, acronyme pour *Monitor, Analyse, Plan, Execute, Knowledge*. Elle définit une chaine de décision qui à partir de l'analyse des données de mesures et de supervision d'un système, planifie et exécute des modifications sur le système en vue d'adapter sa configuration aux changements constatés. Ce modèle nous est apparu utile pour décomposer le processus en différentes fonctions et définir leur enchainement aboutissant à une décision d'adaptation ou de conservation du placement actuel des micro-services. Notre objectif est d'intégrer ce processus d'adaptation du placement des micro-services à notre orchestrateur. Le modèle MAPE-K nous est apparu dans cet objectif intéressant car facile à intégrer dans l'architecture d'évaluation que nous avons développée.

La Figure 5.1 illustre la décomposition du processus d'adaptation sous la forme d'une chaine de décision composée de plusieurs fonctions, ainsi que les données utilisées par chaque fonction. Cette chaine reprend les fonctions du modèle MAPE-K : Supervision (*Monitor*), Analyse, Planification, Execution. Nous avons mis en évidence dans cette figure l'étape de décision de

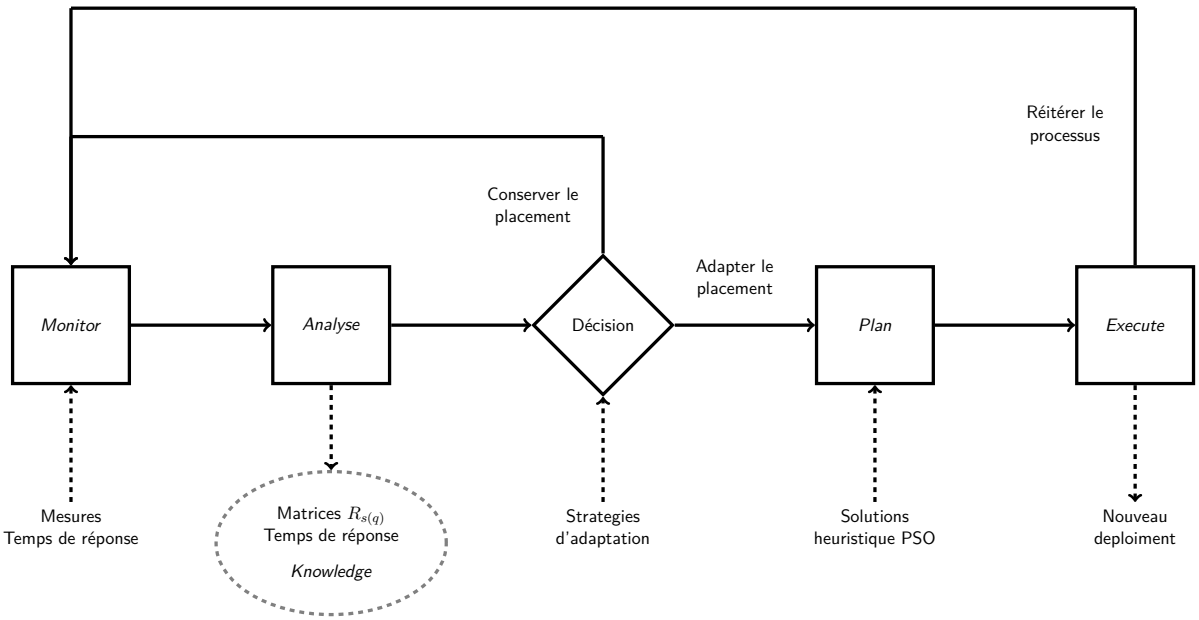

FIGURE 5.1 – Chaine de décision du processus d'adaptation.

l'adaptation du placement, intervenant après la fonction d'Analyse, qui va nous intéresser par la suite. Suite à une décision d'adaptation ou de conservation du placement, le processus d'adaptation retourne à l'étape initiale. Ce processus est donc itératif et suit ainsi l'évolution des conditions d'utilisation de l'application pour prendre de nouvelles décisions si nécessaire.

Une première partie du processus est constituée des étapes de collecte et d'analyse des mesures des paramètres du système. Ces étapes nous permettent de construire une représentation des conditions d'utilisation de l'application sur laquelle s'établira la décision de l'adaptation du placement des micro-services.

La fonction de supervision (*Monitor*) se charge de rassembler un ensemble de données représentant l'état actuel de fonctionnement du système. Dans notre cas, il s'agira de recueillir les mesures caractérisant les performances de l'application selon la disponibilité des ressources de l'infrastructure participative. Nous avons déjà identifié certaines de ces valeurs, comme le temps de réponse de l'application pour chaque client ou les différents paramètres de QoS des réseaux d'accès résidentiels. Nous discuterons dans une section suivante des paramètres à superviser dans le cadre de l'adaptation.

La fonction d'analyse (*Analyse*) exploite les valeurs des paramètres supervisés pour en déduire les indicateurs sur lesquels va se baser la décision d'adaptation. Dans le chapitre 3, nous avons utilisé le temps de réponse global de l'application comme indicateur permettant de comparer les performances de l'application selon différents placements. Nous l'utiliserons ici pour évaluer ces

mêmes performances selon l'évolution des conditions d'utilisation. Il nous sera possible à cette étape de définir d'autres indicateurs si ceux-ci nous semblent pertinents pour la prise de décision. Les valeurs de ces indicateurs sont conservées pour garder un historique du fonctionnement du système et constituer la base de connaissances (*Knowledge*) du processus d'adaptation.

La décision d'adaptation est prise à la fin de cette phase de recueil et d'analyse des données caractérisant le fonctionnement du système. Les indicateurs, ainsi que l'historique de leurs variations, vont nous permettre à cette étape de décider si une adaptation du placement est nécessaire. Cette décision s'établit selon des règles qui définissent une stratégie d'adaptation. Ces règles évaluent les différents indicateurs selon les conditions à remplir pour qu'une adaptation soit requise. Une règle peut par exemple décider d'une adaptation lorsqu'un temps de réponse dépasse un certain seuil. Nous allons évaluer dans ce chapitre différentes stratégies d'adaptation, dont les règles seront définies dans une prochaine section.

La seconde partie du processus intervient une fois prise la décision d'adapter le placement des micro-services pour réaliser cette adaptation. La fonction de planification (*Plan*) s'attache alors à définir le nouveau placement à déployer. Nous allons ici réutiliser l'heuristique de placement Particle Swarm Optimization (PSO) définie dans le chapitre 3 pour déterminer le placement initial. L'heuristique devra alors choisir un nouveau placement en prenant compte des conditions d'utilisation actuelle de l'application pour que ce placement corrige les dégradations du temps de réponse global de l'application. Cette heuristique est capable de nous donner une solution en un temps constant relativement court (inférieur à une seconde). Il est donc possible à cette étape d'effectuer plusieurs fois le calcul d'un placement proche de l'optimal pour aboutir à un ensemble de solutions possibles. Des critères de sélection de placement peuvent alors être appliqués, comme par exemple le nombre de redéploiement de micro-services par rapport au placement actuel, afin de n'en retenir qu'une seule solution.

La dernière fonction du processus consiste à exécuter le déploiement de l'application selon la solution retenue par la fonction de planification. Nous reprenons ici la capacité de déploiement déjà existante dans notre composant Orchestrateur qui s'appuie sur l'intergiciel spécifique de l'infrastructure participative. En sortie de cette fonction, l'application est déployée selon le nouveau placement choisi des micro-services. Le processus d'adaptation est alors en capacité d'évaluer à nouveau le temps de réponse de l'application dans cette nouvelle configuration à travers la fonction de supervision.

5.2.2 Intégration à l'orchestrateur

La Figure 5.2 illustre la nouvelle architecture d'orchestration des micro-services, reprenant l'architecture présentée dans le chapitre 3 et intégrant le processus d'adaptation. Les fonctions de ce processus que nous avons précédemment décrites sont incluses dans le composant Orchestrateur. Ce composant est déjà en charge du calcul du placement à partir des caractéristiques de l'infrastructure et de l'application en utilisant l'heuristique PSO, ainsi que de la coordination du déploiement des micro-services selon ce placement dans l'infrastructure. Les fonctions *Plan* et *Execute* du processus sont donc déjà en partie disponibles. Il nous reste à ajouter à ce composant les fonctions de supervision et d'analyse du comportement du système

Ces fonctions nécessitent de remonter vers l'Orchestrateur des mesures de valeurs caractéristiques du comportement du système et des conditions d'utilisation. Nous avons donc ajouté ces fonctionnalités dans l'intergiciel qui permet à l'Orchestrateur d'interagir avec l'infrastructure participative. Des sondes de mesure sont ajoutées sur chaque équipement de l'infrastructure pour remonter ces valeurs au niveau de l'intergiciel. Nous détaillerons dans la section suivante l'intégration de ces sondes et les valeurs mesurées.

Il est à noter que cette architecture implique de centraliser certaines informations et décisions nécessaires au processus d'adaptation. Les mesures liées à la supervision sont remontées vers un même serveur pour obtenir une vue globale du système. De la même façon, la nouvelle distribution des micro-services est calculée en un même emplacement qui ensuite orchestre le déploiement. Cette centralisation implique qu'un équipement soit dédié à cette tâche. Or dans notre contexte d'une infrastructure participative, il n'est pas forcément évident d'identifier un équipement auquel attribuer ces tâches : celui-ci devra avoir les capacités de calcul nécessaires ainsi qu'être disponible et joignable en permanence. Dans nos expérimentations, nous utiliserons un serveur dédié à ces tâches, externe à l'infrastructure participative. La transposition de cette solution dans un cas d'utilisation réel nécessitera une solution pour remédier à ces problèmes de disponibilité.

5.3 Supervision du système

5.3.1 Choix des paramètres à superviser

Nous avons expliqué dans les chapitres précédents que le temps de réponse global de l'application dépend de plusieurs paramètres caractérisant l'application, l'infrastructure participative et les

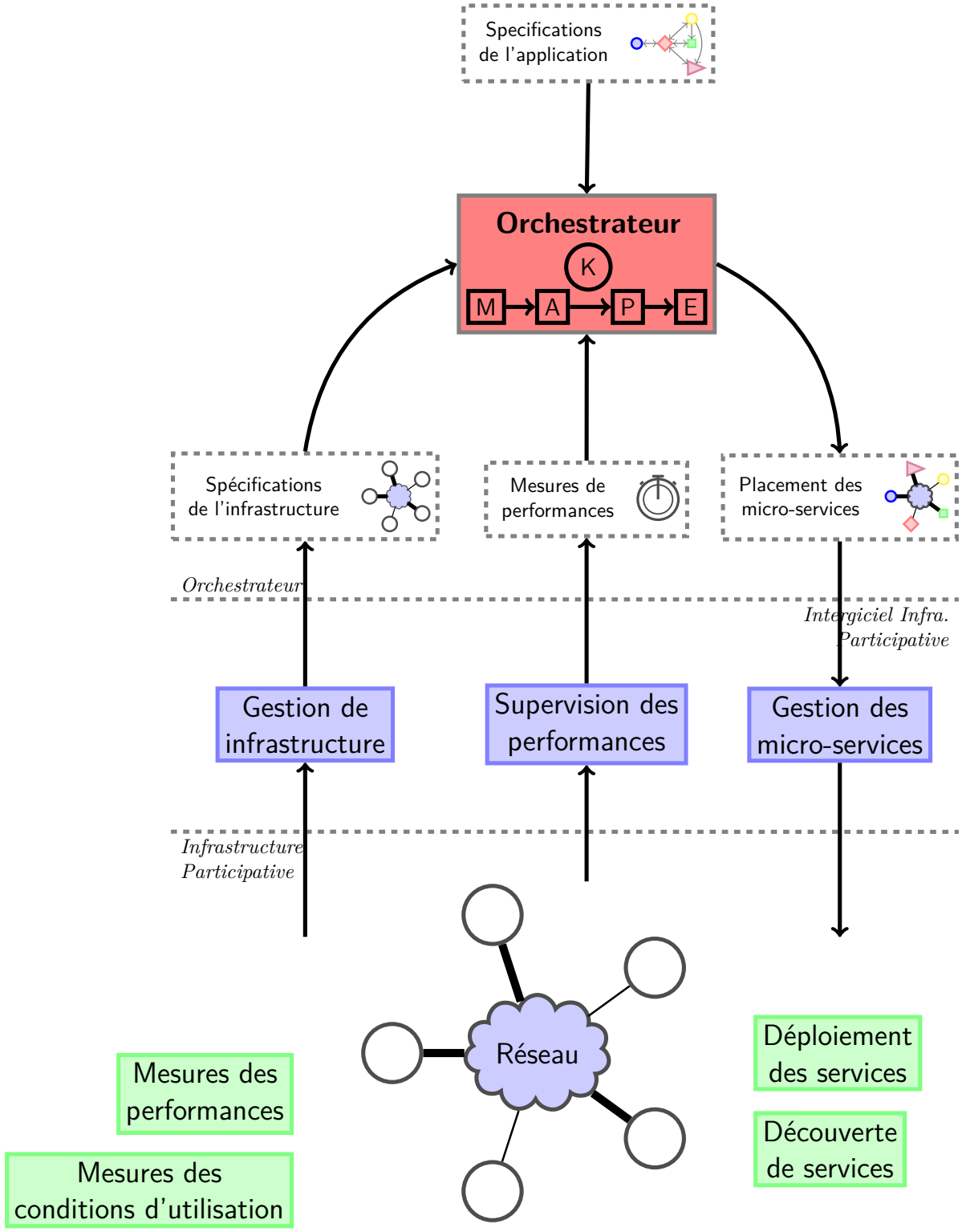

FIGURE 5.2 – Architecture d'orchestration des micro-services sur une infrastructure participative intégrant le processus d'adaptation du placement.

conditions d'utilisation. Il nous faut ici sélectionner parmi ces paramètres lesquels superviser dans le cadre du processus d'adaptation. Cette sélection vise à limiter le nombre de paramètres à mesurer dans l'infrastructure, faciliter l'intégration de sondes de mesures sur les équipements et la remontée des valeurs mesurées et ainsi minimiser l'impact de ces mesures sur le fonctionnement du système.

Pour l'évaluation du placement des micro-services dans des conditions réalistes présentée dans le chapitre 4, nous avons intégré à notre architecture un dispositif de mesure du temps de réponse global de l'application. Cette valeur est notre indicateur clé pour estimer l'expérience utilisateur. La supervision de ses variations nous sera utile pour la prise de décision de l'adaptation du placement. Notre heuristique de placement utilise pour le calcul de solutions de placement des paramètres liés à l'application et à l'infrastructure. Les paramètres de l'application, comme les dépendances entre micro-services, sont fixés comme données initiales du système. Ils n'évoluent donc pas dans le temps. Nous considérons de plus dans cette étude qu'aucune modification n'est apportée à l'infrastructure participative, que ce soit en terme d'équipements participants ou de clients de l'application. Il n'est donc pas utile de superviser ces paramètres.

Les conditions d'utilisation de l'infrastructure sont caractérisées dans notre cas par les paramètres de QoS des réseaux d'accès résidentiels de l'infrastructure. Il semble donc pertinent de superviser ces paramètres pour les réseaux qui hébergent un client ou un micro-service de l'application pour identifier des variations de l'expérience utilisateur. Dans le chapitre 3, nous avons choisi de considérer le temps de réponse spécifique d'un micro-service comme valeur permettant de modéliser les différents paramètres liés aux conditions de son utilisation. Nous avons donc choisi de superviser cette valeur comme indicateur des variations de ces paramètres. Les valeurs mesurées pour ces temps de réponse spécifiques permettent de plus de renseigner les matrices R_s utilisées par l'heuristique PSO avec des valeurs réelles. Le calcul du nouveau placement va ainsi prendre en compte l'évolution des conditions actuelles d'utilisation de l'infrastructure.

5.3.2 Architecture de supervision

Pour superviser les paramètres que nous avons sélectionnés, il nous faut intégrer différents outils de mesures dans notre architecture de gestion des micro-services. Lors de l'évaluation des placements de l'heuristique décrite dans le chapitre 4, nous avons mis en place un dispositif de mesure du temps de réponse de l'application au niveau de chaque client à travers un processus de génération contrôlée de requêtes. Les mesures réalisées au niveau de chacun de ces clients sont maintenant rapportées automatiquement au composant de supervision de l'orchestrateur qui les agrègent pour obtenir la valeur du temps de réponse global de l'application.

Un dispositif supplémentaire est nécessaire pour mesurer individuellement le temps de réponse de chaque micro-service de l'application. Deux approches sont alors possibles. La première approche, que nous qualifierons de supervision passive, consiste à mesurer le temps de réponse de chaque micro-service lors du traitement des requêtes générées par les clients. Une seconde approche, que nous qualifierons de supervision active, mesure ces temps de réponse à partir de requêtes fictive entre instances de chaque micro-service. Nous avons choisi l'approche de supervision passive, car elle nous assure que lors de nos mesures, les micro-services de l'application ne sont sollicités que pour le traitement de la requête d'un seul client. Pour que notre solution reste la moins intrusive et la plus indépendante de l'application, nous avons écarté la possibilité d'instrumenter le code des micro-services avec des instructions de mesure.

La solution retenue est d'intégrer la mesure du temps de réponse dans un processus indépendant, transparent pour l'application mais s'intégrant dans l'acheminement des requêtes. Ce processus est déployé conjointement avec chaque micro-service et contient un serveur mandataire, ou proxy HTTP, protocole utilisé pour le transport des requêtes entre les micro-services de notre application. Ce proxy va intercepter en local les requêtes envoyées depuis le service auprès duquel il est déployé, pour les retransmettre au micro-service destinataire. Il permet ainsi de mesurer le temps nécessaire pour obtenir une réponse à la requête du point de vue de son émetteur. Cette approche est utilisée dans les plateformes de micro-services en production sous le nom de *Distributed tracing* [37]. La figure 5.3 décrit les composants nécessaires à la mesure du temps de réponse et leurs interactions.

Nous avons choisi d'intégrer au proxy l'appel à la fonction de découverte des services pour deux raisons. D'une part, nous déchargeons ainsi l'émetteur de la requête de trouver la localisation du destinataire dans l'infrastructure participative. Cela facilite l'intégration des micro-services qui n'ont plus à s'interfacer avec la fonction de découverte des services de l'infrastructure. D'autre part, nous nous assurons ainsi que les mesures correspondent bien au seul temps de réponse des services, qui sont les valeurs sur lesquelles nous basons notre adaptation, sans inclure le temps de localisation du service. Cette localisation est réalisée en interrogeant le service Consul. Ce service est distribué sur différents équipements de l'infrastructure et ne peut donc garantir un temps de réponse identique selon la requête de localisation.

Les différentes interactions entre composants lors d'une requête à travers le proxy sont les suivantes :

1. Le micro-service à l'origine de la requête est configuré lors de son déploiement pour envoyer sa requête vers le proxy local.
2. À réception de la requête, le proxy localise l'équipement destinataire à partir du nom du service en interrogeant le composant de découverte de service.

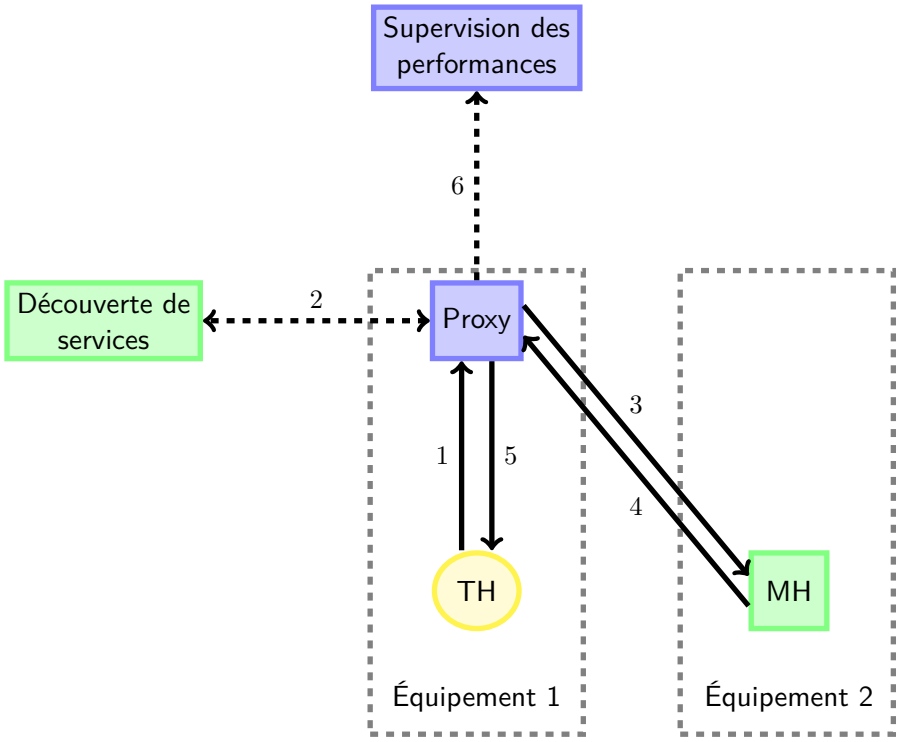

FIGURE 5.3 – Composants nécessaires à la mesure du temps de réponse d'une requête entre deux micro-services.

3. Le proxy transmet la requête vers l'équipement destinataire et commence la mesure du temps de réponse.
4. Le service sollicité répond à la requête. Le proxy termine la mesure du temps de réponse.
5. La réponse est transmise au service à l'origine de la requête.
6. Le temps de réponse mesuré est envoyé au composant de supervision.

Chaque proxy va donc remonter une mesure de temps de réponse à chaque appel entre micro-services. Le traitement d'une seule requête utilisateur par l'application va générer autant de remontée de mesures qu'il y a d'appels entre micro-services. Une solution pour identifier les mesures correspondant à une même requête est d'inclure comme paramètre de cette requête un identifiant unique au moment de sa création, donc au niveau du client. Cet identifiant est recopié au niveau des requêtes subséquentes vers d'autres micro-services. Les proxys interprétant les paramètres de ces requêtes, ils pourront remonter cet identifiant avec les mesures effectuées au composant de supervision. Dans notre implémentation, le micro-service doit cependant recopier lui-même l'identifiant reçu dans les requêtes subséquentes vers d'autres micro-services. Il s'agit là de l'unique instrumentation nécessaire à la supervision dans le code du micro-service.

Au moment de transmettre une mesure de temps de réponse au composant de supervision, le proxy regroupe l'ensemble des informations permettant d'identifier la requête et les parties prenantes dans cet échange, informations qui sont ensuite envoyées au processus d'adaptation :

— Micro-service à l'origine de la requête : information statique configurée au moment du déploiement du micro-service local,
— Micro-service destinataire de la requête : information extraite de la requête à retransmettre,
— Équipement à l'origine de la requête : information statique configurée au moment du déploiement du proxy,
— Équipement destinataire de la requête : information obtenue après localisation du service destinataire,
— Type de requête (`GET` ou `POST`) : information extraite de la requête à retransmettre,
— Identifiant de la requête : information extraite de la requête à retransmettre,
— Temps de réponse du micro-service : mesure effectuée.

5.3.3 Analyse des temps de réponse mesurés

Le temps mesuré par le proxy mesure le temps de réponse du micro-service à une requête. Ce temps inclue le temps transmission de la requête, de son traitement et de la transmission de la réponse, mais aussi le temps de réponse d'autres micro-services nécessaires au traitement de la requête. Ici nous nous intéressons à extraire de cette mesure le temps de réponse spécifique du micro-service pour cette requête. Dans la section 3.3, nous avons exprimé le temps de réponse de l'application en fonction du temps de réponse spécifique de chaque micro-service. Il nous faut maintenant pouvoir exprimer le temps de réponse spécifique en fonction d'un temps de réponse mesuré. La formule 3.2 donnée dans cette section a une expression récursive que nous pouvons réutiliser ici, en fixant le début de cette récursion non pas sur un client mais sur l'équipement hébergeant le micro-service i où est mesuré le temps de réponse du service j. Soit le placement P de l'application, $P(i)$ et $P(j)$ les placements des services i et j. Le temps de réponse d'un service j mesuré pour la requête q depuis l'équipement $P(i)$ s'exprime alors de la manière suivante :

$$t_{P(i),j(q)} = R_{j(q)}(P(i), P(j)) + \sum_{k \in S} D_q(j,k) \cdot t_{P(j),k(q)}$$

Cette expression fait apparaître dans le premier terme le temps de réponse spécifique $R_{j(q)}$ du micro-service j à la requête q émise depuis l'équipement $P(i)$ en fonction de son placement $P(j)$. Le second terme exprime les temps de réponse induits par les appels vers les autres micro-services dont dépend le micro-service j. Pour trouver le temps de réponse spécifique $R_{j(q)}$, il est donc nécessaire de retrancher au temps de réponse mesuré la somme de ces temps de réponse $t_{P(j),k(q)}$ pour les services k dont dépend j. Après le traitement complet d'une requête q par

l'application, l'identifiant de requête nous permet de retrouver parmi l'ensemble des mesures remontée les temps de réponse $t_{P(j),k(q)}$.

Les dépendances de chaque service $D_q(j,k)$ sont décrites dans le graphe des appels de l'application. Un algorithme d'extraction des temps de réponse spécifiques conçu à partir de ce graphe peut donc facilement identifier les temps de réponse des dépendances nécessaires au calcul, mais il sera alors fortement spécialisé à notre application. Une description formelle du graphe des appels (en langage WS-BPEL par exemple) pourrait être utilisé dans un algorithme générique. Dans notre cas, nous savons que le graphe d'appel est acyclique. Dans un tel graphe, l'ensemble des appels provenant d'un même micro-service sont les dépendances dont il faudra soustraire les temps de réponse pour obtenir le temps de réponse spécifique. Dans un graphe cyclique, il serait nécessaire de prendre en compte les itérations possibles dans une boucle par exemple.

Cette propriété nous permet donc de formuler un algorithme simple pour identifier les temps de réponse à soustraire afin d'obtenir le temps de réponse spécifique d'un service. Pour chaque mesure de temps de réponse est précisé le service à l'origine de la requête. Il nous suffit donc d'identifier les mesures pour des requêtes provenant du service concerné et de les soustraire au temps de réponse mesuré. Soit T_q l'ensemble des temps de réponse mesurés pour la même requête q. Nous définissons $T_q(s_j)$ comme le sous-ensemble des temps de réponse mesurés pour appels provenant du service s_j à destination d'autres services pour le traitement de cette requête q.

$$T_q(s_j) = \{t_q(P(s), s_k) \in T_q | s = s_j\}$$

Le temps de réponse spécifique du service s_j appelé depuis l'équipement $P(s_i)$ pour la requête q s'exprime alors par la formule :

$$R_{s_j,q}(P(s_i), P(s_j)) = t_q(P(s_i), s_j) - \sum_{t \in T_q(s_j)} t$$

Pour illustrer ce calcul, prenons l'exemple de la requête permettant à un client d'obtenir une vignette d'une photo. Le graphe des appels entre micro-services pour traiter cette requête est donné dans la figure 5.4. Pour chaque appel numéroté dans cette figure, un temps de réponse sera mesuré et remonté au composant de supervision de l'orchestrateur. Le calcul des temps de réponse spécifiques de chaque micro-service impliqué s'effectue une fois que l'application aura complètement traité la requête. Les mesures de cette requête sont alors extraites de l'ensemble des mesures grâce à l'identifiant de requête. Le tableau 5.2 donne un exemple de mesures remontées pour une telle requête.

L'algorithme calcule le temps de réponse spécifique de chaque micro-service s en déterminant le sous-ensemble $T_q(s)$ à partir des mesures remontées pour la requête q. Dans notre exemple, le calcul sera le suivant :

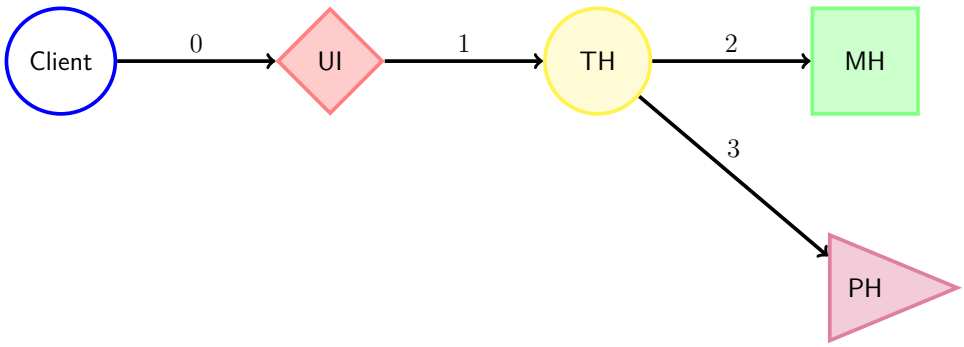

FIGURE 5.4 – Appels entre micro-services pour traiter la génération d'une vignette.

i	Source	Destination	t_i (ms)
0	Client	UI	850
1	UI	TH	650
2	TH	MH	100
3	TH	PH	400

TABLE 5.2 – Temps de réponse mesurés pour chaque service lors du traitement d'une requête.

$$T_{\text{GET}}(\text{UI}) = \{t_1\} \Rightarrow R_{\text{UI(GET)}}(c, P(\text{UI})) = t_0 - t_1 = 200ms$$
$$T_{\text{GET}}(\text{TH}) = \{t_2, t_3\} \Rightarrow R_{\text{TH(GET)}}(P(\text{UI}), P(\text{TH})) = t_1 - (t_2 + t_3) = 150\text{ms}$$
$$T_{\text{GET}}(\text{MH}) = \oslash \Rightarrow R_{\text{MH(GET)}}(P(\text{TH}), P(\text{MH})) = t_2 = 100\text{ms}$$
$$T_{\text{GET}}(\text{PH}) = \oslash \Rightarrow R_{\text{PH(GET)}}(P(\text{TH}), P(\text{PH})) = t_3 = 400\text{ms}$$

Les temps de réponse spécifiques ainsi extraits des mesures sont ensuite enregistrés dans les matrices $R_{s(q)}$ correspondantes. Ces valeurs sont mises à disposition de la fonction de décision pour évaluer l'évolution des temps de réponse des différents services et déterminer si une adaptation est nécessaire, selon la stratégie d'adaptation choisie. Nous conservons un historique des valeurs mesurées pour chaque temps de réponse spécifique mesuré pour un service entre deux équipements $R_{s(q)}(i, j)$. La fonction de décision pourra ainsi s'appuyer sur un historique de l'évolution de ces valeurs. Nous pouvons aussi représenter l'évolution des temps de réponse spécifiques de chaque service sous forme d'un graphique comme celui de la Figure 5.5. Cette représentation présente le temps de réponse de l'application comme la somme des temps de réponse spécifiques des différents services. Elle nous sera utile dans l'analyse du fonctionnement des stratégies d'adaptation. Les matrices $R_{s(q)}$ seront enfin utilisées dans la phase de planification pour évaluer le temps de réponse de l'application pour les nouveaux placements candidats à être déployés.

Nous devons souligner qu'il existe une probabilité que certaines mesures ne soient pas remontées par l'outil de supervision. Plusieurs causes existent comme une erreur au niveau du composant

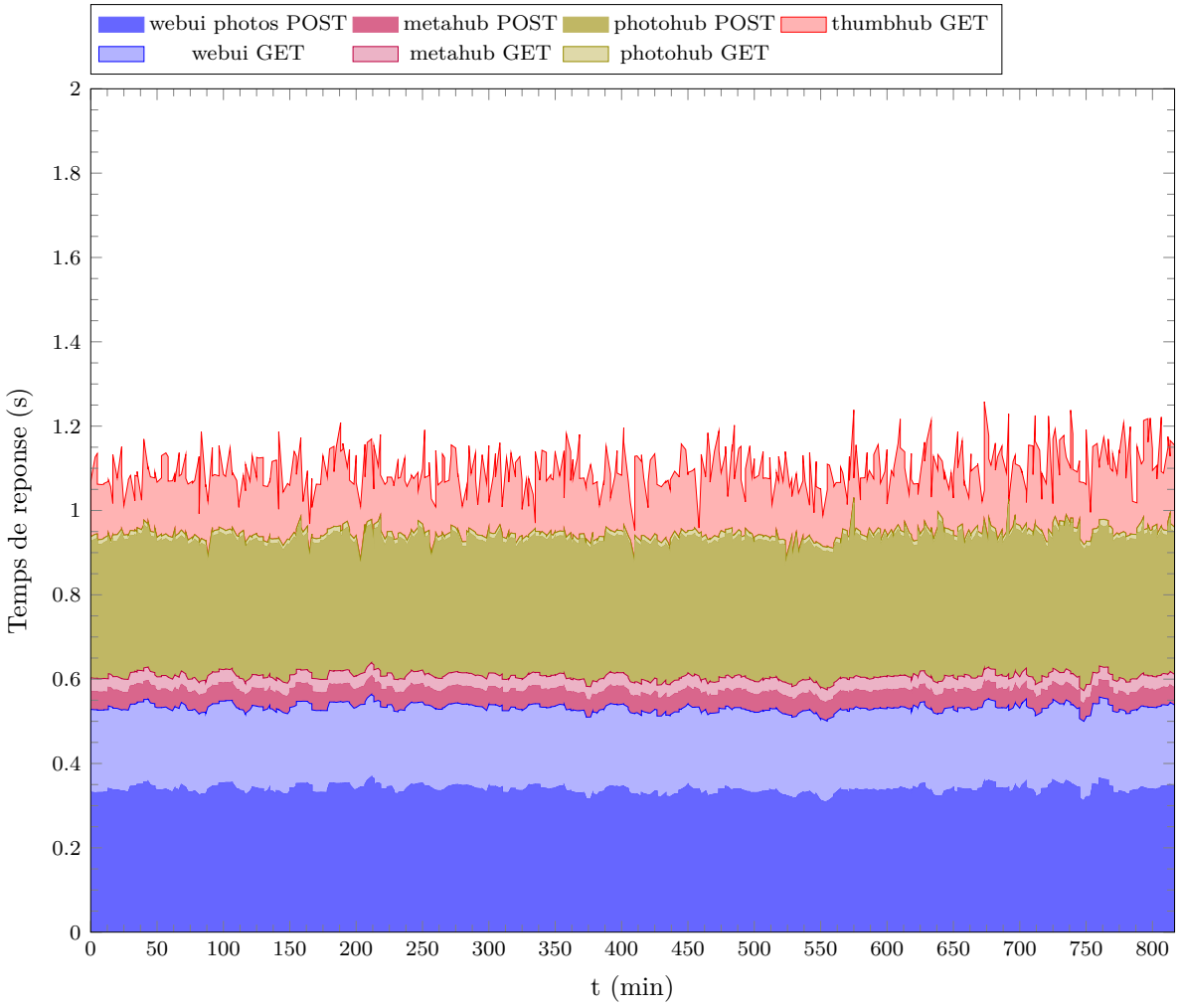

FIGURE 5.5 – Variation des temps de réponse spécifiques des micro-services de l'application.

local de mesure en cas de non-terminaison de la requête par exemple, ou la perte de la mesure lors de sa remontée vers le composant de supervision. Certaines mesures étant alors absentes, l'extraction des temps de réponse spécifiques aboutit à des valeurs inexactes. Nous pouvons heureusement identifier ces mesures incomplètes, car le nombre de mesures pour un même type de requête est constant. Les mesures pour les requêtes qui n'auront par été totalement tracées sont alors rejetées. Les expérimentations que nous avons menées par la suite sur l'infrastructure participative réelle ont montré que la probabilité de présence de mesures incomplètes est inférieure à 1% dans notre cas.

5.4 Décision et adaptation du placement

Le résultat de la phase de supervision décrite dans la section précédente fournit au processus d'adaptation une vue globale sur le temps de réponse de l'application et une vue détaillée des temps de réponse spécifiques mesurées pour chacun des micro-services. Le processus doit maintenant analyser ces données et décider si une adaptation du placement des micro-services est nécessaire. Cette décision est prise en application d'une stratégie d'adaptation qui détermine les données et définit les conditions à remplir pour déclencher un nouveau déploiement. Un nouveau placement sera alors choisi en utilisant l'heuristique PSO qui utilise aussi les données issues de la supervision.

5.4.1 Stratégies d'adaptation

Les travaux existants recensés précédemment nous indiquent qu'il existe deux grandes familles de stratégies d'adaptation : proactives et réactives. Les études mettent en œuvre ces stratégies dans leur propre système et selon leur contexte d'usage. Il est donc difficile de reprendre une stratégie existante pour l'appliquer dans notre contexte. Nous allons voir ici comment exprimer deux stratégies d'adaptation pour notre système, l'une proactive, l'autre réactive.

Définition d'une stratégie proactive

L'article [43] décrit une stratégie qui adapte une composition de service périodiquement à intervalle de temps régulier. Cette stratégie peut être qualifiée de proactive car elle vise à améliorer les performances du système avant que celles-ci ne se dégradent. L'étude SamKnows [34] nous indique que les paramètres de QoS des réseaux d'accès résidentiels suivent une variation journalière. Certains réseaux sont plus sollicités aux heures du soir qu'ils ne le sont en journée. A travers un redéploiement régulier des services, il pourrait être ainsi possible d'exploiter des équipements dont les réseaux sont inutilisés pendant des périodes dites creuses, et ensuite d'adapter le placement pour exploiter d'autres équipements lors des périodes plus chargées.

Dans notre cas, il peut s'avérer pertinent de demander au processus d'adaptation de modifier régulièrement le placement des micro-services. Il s'agira de déterminer la temporalité des adaptations décidées par cette stratégie. Une adaptation en anticipation des variations journalières serait à réaliser à certaines heures de la journée. Nous avons choisi d'effectuer dans nos expérimentations des adaptations sur des intervalles de temps plus restreints d'une à deux heures. Une

telle stratégie va ainsi permettre de tester plusieurs adaptations et d'évaluer plusieurs placements dans différentes conditions d'utilisation.

Définition d'une stratégie réactive

Cette stratégie s'attache à adapter le placement lorsqu'une dégradation du temps de réponse de l'application est constatée. Elle s'appuie sur le temps de réponse global calculé à partir des mesures effectuées par les clients. Il est possible d'estimer cette variation en comparant la dernière valeur mesurée à l'historique des mesures précédentes. Nous avons décrit précédemment que ce temps de réponse peut subir une variation importante causée par une dégradation des paramètres de QoS généralisée ou bien localisée sur un équipement, comme dans le cas présenté dans la section 4.3.2. Si la variation constatée du temps de réponse dépasse un certain seuil, il est alors probable que le système subisse une dégradation des conditions d'utilisation et qu'une adaptation soit alors nécessaire.

Il conviendra de définir avec attention l'évaluation du seuil déterminant une variation significative du temps de réponse. En effet, les mesures de temps de réponse effectuées dans un cas non perturbé, comme celles présentés dans la Figure 5.5, montrent de légères variations dues à l'incertitude des mesures. En comparant la dernière mesure du temps de réponse à la moyenne arithmétique des valeurs mesurées durant la dernière heure par exemple, nous pouvons ainsi identifier une variation suffisamment marquée pour correspondre à un changement significatif dans les conditions d'utilisation par rapport à la période écoulée. Nous avons estimé, à partir des résultats des expérimentations menées dans des conditions réelles, qu'une variation supérieure à un seuil de 15% entre cette moyenne et la dernière valeur mesurée est significative d'un changement dans les conditions d'utilisation. Ce choix pourra être ensuite ajusté en fonction des résultats obtenus en conditions réelles.

5.4.2 Choix du nouveau placement

L'objectif est ici de trouver un placement qui minimise le temps de réponse de l'application alors qu'a été détecté une dégradation des performances liée à un changement des conditions d'utilisation. L'heuristique PSO nous permet de trouver une solution de placement quasi-optimale en évaluant des solutions potentielles à partir des matrices $R_{s(q)}$. Ces matrices sont mises à jour avec les valeurs des temps de réponse spécifiques extraites des mesures de la supervision. Ces valeurs étant représentatives des conditions d'utilisation actuelles et de leurs evolutions, elles pourront guider le choix du placement par l'heuristique.

Dans le cas d'une adaptation proactive récurrente, le placement supervisé aura été utilisé depuis un temps relativement long. Dans nos expérimentations, le placement aura ainsi été évalué pour au moins une heure avec des requêtes régulières. Les matrices $R_{s(q)}$ contiendront donc les temps de réponse spécifiques mesurés pour le placement actuel, ainsi que l'historique des mesures effectuées pour les placements précédents. L'heuristique PSO s'appuie sur cet historique pour évaluer les solutions de placement potentielles. La valeur de chaque temps de réponse spécifique $R_{s(q)}(i,j)$ est alors calculé comme la moyenne arithmétique des 10 dernières mesures présentes dans l'historique. Dans le cadre d'un usage par des utilisateurs réels, les requêtes sont plus espacées dans le temps. La durée entre deux adaptations pourra être plus longue pour obtenir un nombre suffisant de mesures. Il sera peut être aussi nécessaire de réajuster le nombre de valeurs prises en compte dans l'historique.

Dans le cas d'une adaptation réactive, la décision de l'adaptation intervient lorsqu'est constatée une dégradation du temps de réponse global de l'application. Cette dégradation peut être généralisée ou localisée sur quelques équipements. Nous devons nous assurer que le nouveau placement ne réutilise pas les équipements présentant ces perturbations. Si la dégradation est localisée, l'augmentation du temps de réponse global se manifeste par une augmentation du temps de réponse spécifique d'un micro-service, qui aura été mesurée par la supervision et intégrée à la matrice $R_{s(q)}$ correspondante. Lors d'une telle adaptation réactive, le choix du nouveau placement par l'heuristique PSO doit donner plus de poids aux dernières mesures obtenues pour chaque temps de réponse spécifique. Nous calculerons donc la valeur de chaque $R_{s(q)}(i,j)$ par une moyenne exponentielle des 10 dernières valeurs de l'historique des mesures.

En effectuant plusieurs exécutions successives de l'heuristique, il est possible d'obtenir autant de placements candidats pour minimiser le temps de réponse. Le placement qui sera finalement déployé sur l'infrastructure peut alors être choisi selon plusieurs critères. Notre objectif étant de minimiser le temps de réponse, nous choisirons le placement pour lequel l'heuristique aura donné le plus faible temps de réponse estimé par l'heuristique. D'autres critères peuvent ici rentrer en compte, comme par exemple le nombre de micro-services devant être déplacés par rapport au placement actuel. Ce critère est important si le déploiement de micro-services nécessite de transférer entre les équipements des informations comme du code ou des données applicatives. Dans notre cas, nous considérons que le code des micro-services est déjà installé sur tous les équipements, et que les micro-services ne sauvegardent pas de données nécessaires à l'application. Nous ne prendrons donc pas en compte ce critère dans le choix du placement.

5.5 Évaluation du processus d'adaptation

Nous allons décrire dans cette section l'évaluation du processus d'adaptation que nous venons de définir, en conditions réelles sur notre infrastructure participative. Cette évaluation a pour objectif d'affiner une stratégie d'adaptation en partant d'une politique simple d'adaptation à intervalle de temps régulier, puis en lui rajoutant un comportement plus réactif.

5.5.1 Méthodologie d'évaluation

Nous avons mené plusieurs expérimentations sur notre infrastructure participative afin d'évaluer différentes stratégies d'adaptation dans des conditions d'utilisation réelles. Compte tenu de la nature dynamique de notre plateforme de test, ces expérimentations n'ont pas toutes été réalisées avec les mêmes équipements, mais ont permis l'évaluation du processus d'adaptation sur une durée suffisamment longue. Les matrices $R_{s(q)}$, contenant les temps de réponse spécifiques mesurés, sont reprises entre chaque expérimentation afin que le processus se base sur une connaissance préalable de l'historique des variations de ces valeurs. Ces matrices sont utilisées par l'heuristique PSO pour choisir le placement initial des micro-services de l'application. L'heuristique est configurée selon les mêmes paramètres que ceux utilisés pour l'évaluation présentée dans le chapitre 4.

Pour cette évaluation, nous utilisons l'ensemble des équipements disponibles comme clients de l'application. Ces équipements vont solliciter tour à tour l'application via un processus de génération de requêtes que nous contrôlons pour assurer que l'application ne traite qu'une seule requête simultanément. De la même façon que pour l'évaluation présentée dans le chapitre 4, ces processus mesurent le temps de réponse de l'application pour chaque requête du point de vue d'un client. Pour chaque client, un temps de réponse moyen est calculé comme la moyenne de quatre mesures successives du temps de réponse de chaque requête `GET` et `POST` et avec la même pondération $w_{c,q}$ égale à 1. Le temps de réponse moyen pour ce client est ensuite transmis à la fonction de supervision du processus d'adaptation.

Une fois réalisée une itération de mesures de temps de réponse pour l'ensemble des clients, les valeurs remontées sont analysées pour calculer le temps de réponse global de l'application ainsi que les temps de réponses spécifiques de chaque micro-services. Pour chaque requête réalisée, nous extrayons les différents temps de réponse spécifiques des micro-services impliqués dans son traitement selon la méthode présentée à la section 5.3.3. Les différentes requêtes des clients ont produit autant d'appels entre micro-services, donc nous obtenons autant de valeurs de temps de réponse spécifique pour chaque micro-service. Nous considérons que ces valeurs sont repré-

sentative des mêmes conditions d'utilisation. Nous calculons donc une valeur moyenne de ces temps de réponse spécifiques obtenus pour ces requêtes. Cette moyenne est ensuite placée dans l'historique des valeurs de $R_{s(q)}(i,j)$ correspondant.

Une fois ces valeurs calculées, le processus d'adaptation entre dans la fonction de décision. Selon la stratégie configurée au démarrage du processus, celui-ci reviendra à l'étape de supervision pour initier une nouvelle itération de mesures, ou déclenchera une adaptation du placement. nous exécutons l'heuristique PSO afin d'obtenir une solution de placement candidate au déploiement. Cette solution est choisie sur la base de l'historique des valeurs de $R_{s(q)}(i,j)$ mesurées lors des itérations de mesures précédentes. L'heuristique PSO est non-déterministe, elle selectionne des solutions potentiellement différentes à chaque exécution. Nous avons choisi d'exécuter 10 fois l'heuristique pour obtenir un ensemble de placements candidats au déploiement. Nous écartons dans ces candidats les placements identiques à celui utilisé avant l'adaptation. Nous nous assurons ainsi que chaque demande d'adaptation entraine un nouveau déploiement des micro-services, ce qui nous permettra d'observer l'impact de chaque adaptation sur l'expérience utilisateur. Parmi les solutions restantes, nous choisissons le placement pour lequel l'heuristique a estimé un temps de réponse de l'application le plus faible.

Une fois l'application redéployée selon le placement choisi, nous mesurons le temps de réponse global de l'application pour l'ensemble des clients et nous le comparons avec ce même temps observé avant l'adaptation. Nous évaluons ainsi la pertinence de cette adaptation par la différence entre ces temps de réponse. Si le temps de réponse après adaptation est inférieur de 10% de celui avant adaptation, nous considérons que le nouveau placement a apporté une amélioration dans l'expérience utilisateur. L'adaptation du placement était donc pertinente dans ce cas. Inversement, si le temps de réponse après adaptation est supérieur de 10% de celui avant adaptation, nous considérons alors que l'adaptation a dégradé cette expérience. Entre ces deux seuils, nous considérons enfin que l'adaptation n'a apporté aucun changement.

5.5.2 Évaluation de l'adaptation proactive

Nous avons mené plusieurs expérimentations au cours desquelles notre Orchestrateur cherche à adapter le placement des micro-services selon une stratégie proactive. Cette stratégie demande régulièrement un déploiement selon un nouveau placement avec une période de temps définie. Nous avons fixé cet intervalle à 100 minutes, afin d'effectuer un nombre suffisant de mesures des temps de réponse des micro-services pour le choix du nouveau placement.

Au cours de chaque expérimentation, nous récupérons ces mesures afin d'évaluer l'évolution du temps de réponse de l'application. La Figure 5.6 présente cette évolution au cours d'une

expérimentation sur une durée d'environ 30 heures. Les résultats montrés ici sont représentatifs des évolutions constatées sur l'ensemble des expérimentations et illustrent bien le comportement du processus d'adaptation. Sur ce graphique, chaque adaptation est représentée par une ligne verticale bleue positionnée à l'abscisse du temps où cette adaptation est réalisée. La courbe représente l'évolution du temps de réponse global mesuré pour l'application, ainsi que des temps de réponse spécifiques de chaque requête `GET` et `POST` vers les différents micro-services. Le second graphique permet de visualiser les différents placements choisis à chaque adaptation. Chaque axe horizontal correspond à un équipement et montre les différents services hébergés sur cet équipement en fonction du temps. Chacun des placements est identifié par une lettre, reprise sur le graphique des temps de réponse.

Au cours de cette expérimentation ont été réalisées 17 adaptations ayant abouti à 11 placements différents des micro-services, identifiés par les lettres de A à K. Nous constatons grâce à ces graphiques que ces différentes adaptations ont entrainé des variations du temps de réponse de l'application. Les performances mesurées varient de manière plus ou moins importante selon le placement des micro-services. Certaines adaptations ont entrainé une dégradation des performances. Par exemple, l'adaptation du placement C au placement E, qui intervient à la minute 700 de cette expérimentation a entrainé une augmentation de plus de 30% du temps de réponse de l'application. Inversement, au moment de l'adaptation du placement H au placement I après la minute 1100, nous constatons une diminution du temps de réponse de presque 40%.

Nous observons aussi dans cette expérimentation, que des choix de placements différents n'ont pas produit de variation significative du temps de réponse. Ainsi les placements B, C, F, J et K ont donné, dans les conditions de l'expérimentation, des temps de réponse de l'application très similaires, alors que les temps de réponse de chaque micro-service sont différents selon le placement. En comparant les temps de réponse obtenus pour les placements B et C par exemple, nous remarquons que le micro-service *ThumbHub* montre un temps de réponse plus faible pour le placement C, mais dans ce même placement, les temps de réponse des micro-services *WebUI* et *PhotoHub* sont plus importants.

Les placements D, E, G et H sont les quatre placements qui ont produit lors de l'expérimentation les temps de réponse de l'application les plus importants. L'examen des temps de réponse mesurés de chaque micro-service pour ces placements montre que certaines requêtes ont été sous-performantes. Pour le placement H par exemple, la requête POST vers le micro-service *PhotoHub* présente un temps de réponse important par rapport aux autres placements. Le temps de réponse doit être considéré selon le placement du micro-service mais aussi selon celui de l'émetteur de la requête. Dans le placement H, micro-service *PhotoHub* est situé sur l'équipement DC et est sollicité par le service *WebUI*, situé sur l'équipement FTTH1. La requête `POST` implique

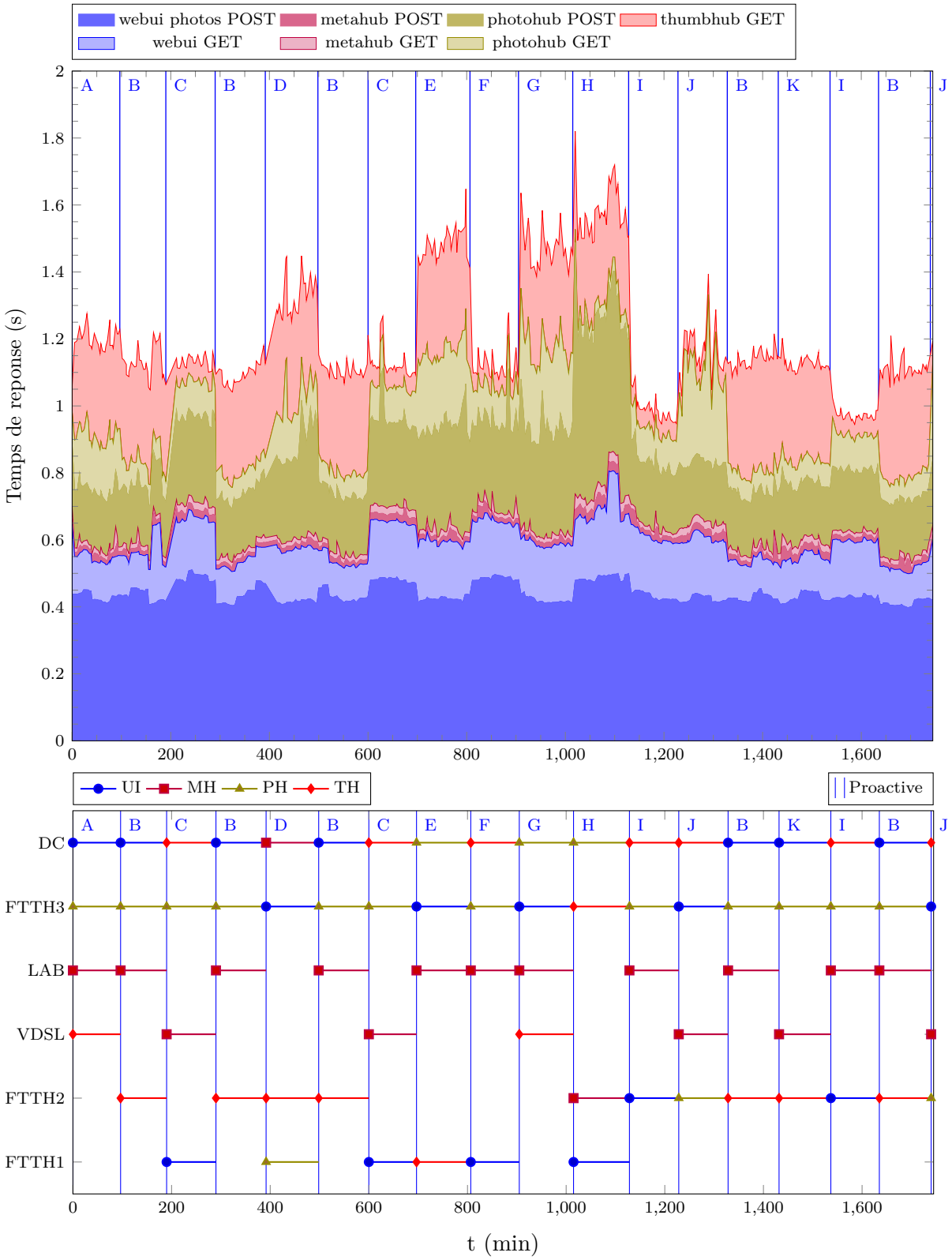

FIGURE 5.6 – Variation du temps de réponse de l'application et de ses micro-services au cours de plusieurs adaptations proactives.

un transfert de données relativement important de l'émetteur vers le service destinataire. Les conditions de QoS réseau durant l'expérimentation n'étaient donc probablement pas favorables à ce trafic.

Nous pouvons nous interroger légitimement sur le fait que ce placement ait été retenu par l'heuristique de placement lors de la phase de planification. Pour rappel, ce placement a été choisi sur la base d'une estimation du temps de réponse de l'application à partir des valeurs des temps de réponse spécifiques présents dans les matrices $R_{s(q)}$. Il est apparu, après investigation, qu'au moment de la sélection du placement par l'heuristique, la matrice correspondant à la requête POST du service *PhotoHub* contenait pour le couple émetteur FTTH1 et récepteur DC une valeur basée sur un historique de mesures avec des valeurs de temps de réponse inférieures à celles mesurées par la suite pour le placement H. Il était donc possible que l'heuristique choisisse cette solution de placement. Cependant, nous pouvons remarquer que cette combinaison d'emplacements pour les services *PhotoHub* et *WebUI* n'est plus retenue dans les placements suivants. La mise à jour de la valeur dans la matrice avec les valeurs mesurées a ainsi guidé le choix du placement pour éviter une combinaison défavorable aux performances de l'application.

5.5.3 Évaluation de l'adaptation corrective

L'évaluation de la stratégie proactive a soulevé le problème que certains placements peuvent être choisis par l'heuristique mais s'avèrent produire des performances peu satisfaisantes une fois déployés. L'heuristique a retenu ces solutions car les valeurs des temps de réponse dans les matrices ne sont pas toujours en cohérence avec la réalité des conditions d'utilisation en vigueur au moment du choix du placement. Si les temps de réponse spécifiques correspondant au dernier placement utilisé ont été mises à jour juste avant l'adaptation, les autres valeurs des matrices ont elles été mesurées dans des conditions probablement différentes. Il est donc possible que ces valeurs conduisent à un mauvais choix de placement. L'expérimentation précédente en est une illustration.

Une solution que nous proposons pour réduire la portée de ce problème est de corriger le placement après le déploiement d'une telle solution sous-performante. Une première étape pour réaliser cette correction sera d'évaluer la différence entre les temps de réponse mesurés avant et après l'adaptation. Si cette différence montre une dégradation trop importante, nous demanderons alors une nouvelle adaptation du placement. Les valeurs mesurées pendant l'évaluation du

placement seront alors intégrées aux matrices et pourront ainsi orienter le choix vers un nouveau placement. Le placement corrigé ne sera peut être lui non plus performant, si il a été choisi à partir de valeurs des matrices non cohérence avec la réalité. Il sera alors nécessaire de corriger de nouveau le placement. Nous allons vérifier si cette stratégie d'adaptations successives permet de converger vers des solutions performantes.

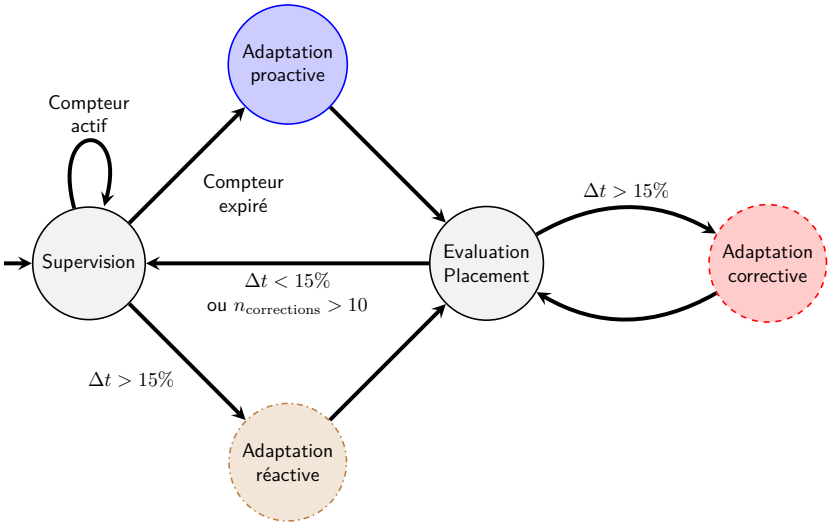

FIGURE 5.7 – Etapes de la stratégie d'adaptation

Notre stratégie d'adaptation est maintenant composée de plusieurs étapes dont les transitions sont représentées dans la Figure 5.7. La stratégie évalue la transition à effectuer à chaque fois que le processus d'adaptation appelle la fonction de décision. Au moment de l'entrée dans l'étape de supervision, un compteur est activé pour déclencher une adaptation proactive à son expiration. Une fois cette adaptation réalisée, le temps de réponse de l'application selon le nouveau placement est mesuré lors de l'étape d'évaluation. Si la différence Δt entre ce temps de réponse et le dernier temps de réponse avant adaptation est inférieure à un certain seuil (ici 15%), le placement est validé et la stratégie retourne dans l'étape de supervision. Si la différence est supérieure au seuil, une adaptation corrective est déclenchée et le nouveau placement est évalué, lui aussi par rapport au dernier temps de réponse mesuré avant l'adaptation proactive. Si le placement corrigé n'apporte pas d'amélioration suffisante, une nouvelle correction est demandée. Nous limitons le nombre d'adaptations correctives successives pour éviter à notre système de boucler entre ces deux transitions en cherchant une correction efficace. Cette limite a été fixée à 10 adaptations correctives successives. Nous ajoutons à notre stratégie la possibilité d'adapter le placement après détection à l'étape de supervision d'une dégradation supérieure au seuil de 15% entre deux temps de réponse de l'application mesurés pour le même placement. Le processus réalise alors une adaptation réactive, suite à un probable changement dans les conditions

d'utilisation de l'application. Le nouveau placement sera alors évalué et potentiellement corrigé, comme pour l'adaptation proactive.

Nous avons testé cette nouvelle stratégie selon la même méthodologie que celle utilisée pour la stratégie proactive. La Figure 5.8 représente les résultats obtenus pendant une période de 10 heures au cours d'une de ces expérimentations. Les adaptations réalisées durant cette période montrent bien le fonctionnement de la stratégie tel que nous l'avons spécifié. Dans cette expérimentation, l'intervalle de temps fixé entre deux adaptations proactives est de 160 minutes. Suite à chaque adaptation, nous demandons à l'ensemble des clients de générer chacun son tour des requêtes vers l'application afin de mesurer le temps de réponse global de l'application. Chaque client générant 4 requêtes à destination de l'application, une itération de mesure complète dure entre 15 et 20 minutes selon les performances de l'application. La période de temps représentée commence au moment où, suite à une adaptation proactive, l'application a été redéployée selon le placement A.

Nous nous intéressons d'abord à la seconde adaptation proactive, qui intervient donc 160 minutes après la première adaptation. Le temps de réponse global mesuré pour l'application selon le placement A avant l'adaptation est d'environ 1 seconde. Une adaptation proactive est déclenchée suite à l'expiration du compteur de 160 minutes et entraine un redéploiement de l'application selon le placement B. Ce placement est ensuite évalué et le temps de réponse global de l'application mesuré selon ce nouveau placement est représenté sur le graphique à environ 20 minutes après l'adaptation. Celui-ci est alors d'environ $1,2$ secondes, soit 20% supérieur au temps de réponse avant adaptation. Selon la stratégie que nous avons définie, une adaptation corrective est alors demandée. Un nouveau placement C est choisi par l'heuristique et déployé à l'instant représenté sur le graphique par une ligne verticale rouge discontinue. Nous mesurons ensuite avec ce nouveau placement un temps de réponse de l'application d'environ $1,3$ secondes. Ce temps étant encore supérieur de plus de 15% par rapport au temps de réponse de l'application avant adaptation avec le placement A, une nouvelle adaptation corrective est donc de nouveau déclenchée. Il en va ainsi de même pour les deux autres placements D et A ensuite choisis, jusqu'à aboutir à une adaptation corrective vers le placement E qui lui permet d'obtenir un temps de réponse proche de celui du placement A avant adaptation, et qui s'avèrera même meilleur par la suite. Notons que ce placement A a été choisi lors d'une adaptation corrective mais a montré des performances moins satisfaisantes qu'auparavant et n'a donc pas été retenu.

Après cette dernière adaptation corrective, le temps de réponse de l'application déployée selon le placement E va rester suffisamment stable pour qu'aucune adaptation ne soit déclenchée avant l'adaptation proactive autour de la minute 400 de la période ici représentée. Suite à cette adaptation, l'application est de nouveau déployée avec le placement A qui s'avère de

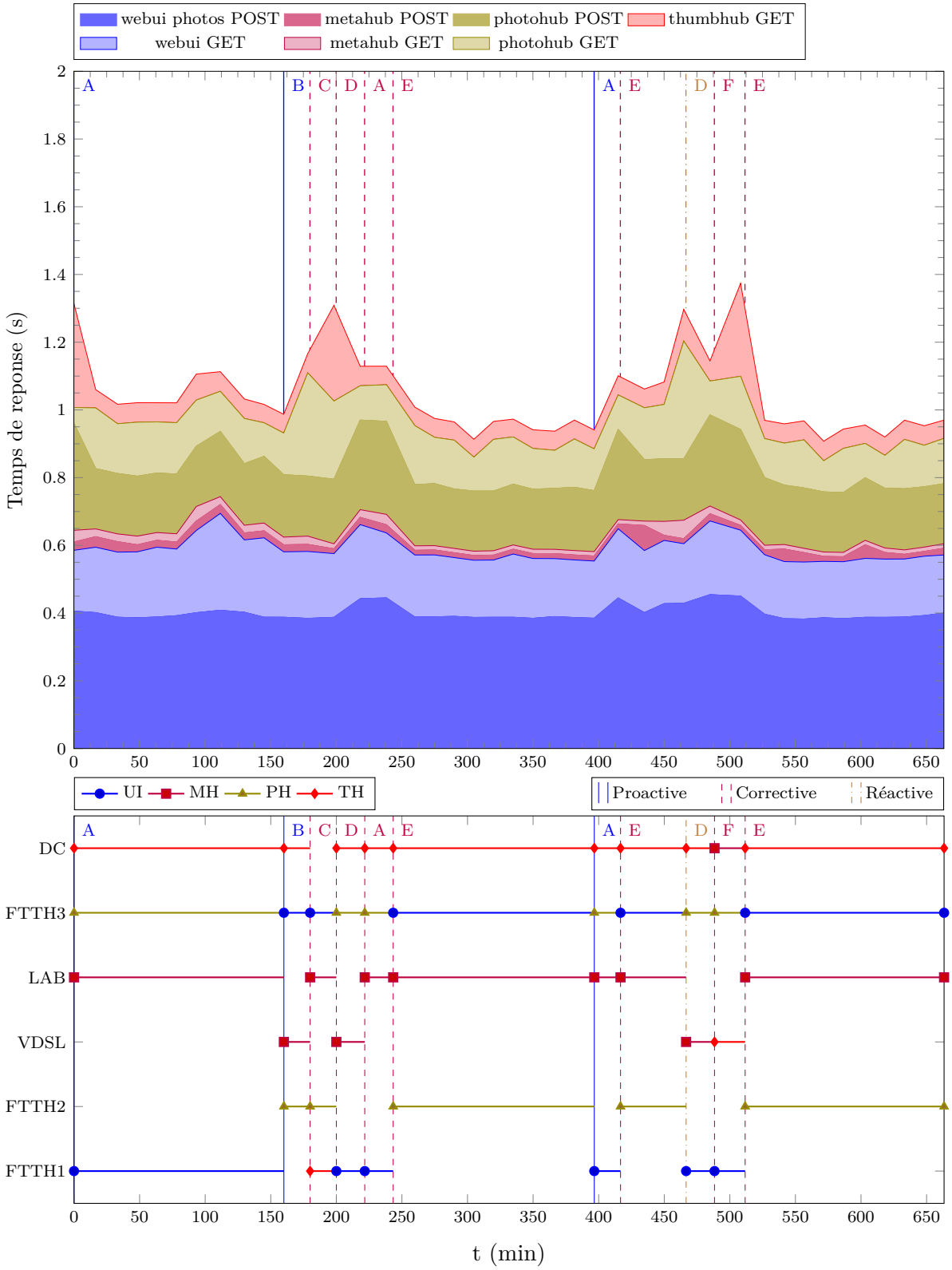

FIGURE 5.8 – Variation du temps de réponse de l'application et de ses micro-services au cours de plusieurs adaptations réactive.

nouveau aboutir dans des performances insatisfaisantes. Le temps de réponse mesuré suite à cette adaptation étant supérieur de plus de 15% au dernier mesuré pour le placement E, une adaptation corrective est déclenchée. L'application est alors de nouveau déployée selon le placement E. Nous constatons alors que le temps de réponse est plus important que celui observé avant l'adaptation proactive, augmentation probablement due à un changement dans les conditions d'utilisation mais pas assez importante pour déclencher une nouvelle adaptation. Ce temps de réponse va rester stable jusqu'à la troisième mesure où il va augmenter au delà du seuil des 15% par rapport à la dernière mesure. Une adaptation réactive est alors déclenchée, à l'instant représenté sur le graphique par une ligne verticale marron pointillée alternée. Lors de cette adaptation est retenu le placement D, qui lui non plus n'apporte pas de résultats satisfaisants. S'en suivent de nouveau deux adaptations correctives, la première vers le placement F puis la suivante vers le placement E. Ce dernier permet finalement de revenir aux valeurs de temps réponse observées précédemment et restera donc en vigueur jusqu'à la prochaine adaptation proactive.

5.5.4 Bilan de l'évaluation des stratégies

Nous avons mené, au gré de la disponibilité des équipements de notre infrastructure participative, plusieurs expérimentations avec chacune des deux stratégies que nous avons décrites précédemment : la stratégie P utilisant seulement les adaptations proactives et la stratégie $P + R + C$ qui permet à la fois des adaptations proactives, réactives et correctives. Le Tableau 5.3 résume les résultats obtenus lors de ces expérimentations. Sont comptées dans ces résultats les adaptations proactives et réactives. Chacune de ces adaptations est évaluée en comparant les temps de réponse moyens observés avant et après l'adaptation. Dans le cas de la stratégie $P + R + C$, si une ou plusieurs adaptations correctives surviennent après une adaptation proactive ou réactive, nous considérons comme temps de réponse après adaptation celui mesuré après la dernière adaptation corrective. Une adaptation sera évaluée comme apportant une amélioration au temps de réponse de l'application si ce temps mesuré après adaptation est inférieur de plus de 10% du temps de réponse avant adaptation. Nous considérons de la même manière qu'une dégradation est constatée lorsque le temps de réponse après adaptation est supérieur de plus de 10% du temps de réponse avant adaptation. Entre ces deux seuils, nous considérons que l'adaptation n'a pas eu d'impact sur le temps de réponse.

Nous constatons que la stratégie P aboutit à un nombre comparable d'adaptations apportant soit une amélioration, une dégradation ou n'ayant que peu d'impact sur le temps de réponse. La stratégie $P+R+C$ aboutit elle majoritairement à des déploiements sans impact significatif sur le temps de réponse. Sur un nombre d'adaptations comparable, les adaptations aboutissant à une dégradation sont beaucoup moins fréquentes avec la stratégie $P + R + C$, leur nombre passant

Stratégies	P	$P+R+C$
Temps total (heures)	182	233
Nombre d'adaptations (P ou R)	81	77
Amélioration du temps de réponse	30	19
Dégradation du temps de réponse	27	16
Peu d'impact sur le temps de réponse	24	42
Déviation standard du temps de réponse	11.4%	10.6%

TABLE 5.3 – Évaluations des adaptations obtenues pour les deux stratégies d'adaptation

de 27 à 16. Nous interprétons cette diminution comme l'apport de la stratégie corrective qui a permis de rectifier des choix de placements contre-productifs. Le nombre d'adaptation apportant une amélioration du temps de réponse avec la stratégie $P+R+C$ est lui aussi inférieur à celui pour la stratégie P. Il faut ici remarquer que ces améliorations interviennent souvent après une adaptation contre-productive. Ces dégradations se manifestant moins souvent avec la stratégie $P+R+C$, il est donc normal que les améliorations soient aussi moins fréquentes.

Nous remarquons cependant que cette stabilité ne se retrouve pas dans la distribution des temps de réponse mesurés sur l'ensemble des expérimentations. Pour chaque expérimentation, nous avons calculé la déviation standard des temps de réponse observée, rapportée à la valeur moyenne du temps de réponse sur l'ensemble de l'expérimentation. Avec la stratégie P, nous observons une déviation standard d'environ 10% par rapport au temps de réponse moyen. La déviation standard moyenne observée avec la stratégie $P+R+C$ est assez similaire. Une explication est que malgré les corrections apportées avec cette stratégie, certaines adaptations ont tout de même entrainé une dégradation significative du temps de réponse. Ces adaptations sont moins fréquentes, mais peuvent impacter temporairement l'expérience utilisateur. Certaines de ces adaptations sont probablement inévitables, dans le cas d'une dégradation globale des conditions d'utilisation par exemple. Nous avons cependant remarqué au cours des expérimentations que certaines adaptations correctives n'ont pas réussi à éviter des dégradations induites par un mauvais choix du placement lors de l'adaptation.

Nous avons analysé pour la stratégie P qu'une des raisons du choix de ces placements sous-performants par l'heuristique PSO est que certaines valeurs dans les matrices $R_{s(q)}$ ne sont plus en cohérence avec les temps de réponse observables dans la réalité. Les mesures pour obtenir ces valeurs ont été faites dans des conditions d'utilisation qui ont depuis évolué. Dans notre solution, les valeurs des matrices $R_{s(q)}$ mises à jour sont celles correspondant au déploiement en cours. Il est donc nécessaire pour mettre à jour d'autres valeurs des matrices, de déployer plusieurs fois l'application selon des placements impliquant des équipements différents pour chaque service. Avec la stratégie P, ces déploiements sont espacés de plusieurs dizaines de minutes, les différentes

valeurs des matrices $R_{s(q)}$ sont donc mises à jour assez peu fréquemment et ne sont donc plus en cohérence avec la réalité. Avec la stratégie $P+R+C$, les adaptations correctives testent successivement plusieurs placements. Les valeurs des matrices sont ainsi mises à jour plus rapidement. Nous avons observé lors de nos expérimentations que sur une durée de 140 heures, la stratégie $P+R+C$ a utilisé 42 placements différents. Sur cette durée, 63% des valeurs des matrices ont été mises à jour au moins une fois. La probabilité d'obtenir un placement sous-performant est alors réduite, sans toutefois être nulle.

5.6 Conclusion sur l'adaptation

Nous avons présenté dans ce chapitre une solution d'adaptation du placement des micro-services en fonction des variations du temps de réponse observés de l'application. Cette adaptation vise à corriger les dégradations observées de ce temps de réponse lors de changement dans les conditions d'utilisation. Nous avons défini un processus d'adaptation basé sur le modèle MAPE-K qui s'intègre à l'Orchestrateur de déploiement des micro-services utilisé dans le chapitre 4. Ce processus s'appuie sur une infrastructure de supervision des temps de réponse des micro-services déployés qui nous permet d'extraire de ces mesures, les différents temps de réponse spécifiques. Ces valeurs sont utilisées ensuite par l'heuristique PSO pour déterminer un nouveau placement, prenant ainsi en compte l'évolution de ces temps de réponse.

Nous avons évalué ce processus d'adaptation sur notre infrastructure participative de test pour comparer deux stratégies d'adaptation dans des conditions réelles d'utilisation. La première stratégie proactive réalise une adaptation régulière du placement en choisissant un nouveau placement à partir des mesures de temps de réponse des micro-services. Les résultats obtenus ont montré que cette stratégie produit majoritairement des adaptations de placements qui n'impactent pas le temps de réponse, voire qui l'améliorent. Cependant un tiers de ces adaptations ont entrainé une dégradation de plus de 10% du temps de réponse. Nous avons proposé une évolution de cette stratégie en ajoutant la possibilité de faire des adaptations correctives. La part d'adaptations contre-productives est alors descendue à 20%.

Les expérimentations que nous avons menées nous ont permis d'évaluer un processus d'adaptation dans des conditions réelles, mais cependant assez limitées. Le nombre d'équipements impliqués est assez faible (moins d'une dizaine), la durée totale des expérimentations assez courte (environ 400 heures) et les perturbations du réseaux trop rares, à en juger par le nombre d'adaptations réalisées. Dans ces conditions, il nous est difficile d'établir si les choix faits pour

les nombreux paramètres de configuration du processus d'adaptation s'avèrent totalement judicieux. Un dispositif expérimental avec plus de moyens matériels et des expérimentations sur un temps plus long nous semble nécessaires pour aboutir à un paramètrage concluant du système.

Nous avons identifié un point d'amélioration de notre solution concernant la mise en cohérence des valeurs des matrices $R_{s(q)}$ avec le temps de réponse que nous pourrions observer dans les conditions actuelles d'utilisation. Le décalage de ces valeurs dans la solution actuelle induit des choix de placements sous-performants. La supervision de ces valeurs reste pertinente car elle nous permet d'obtenir une indication fiable sur les performances d'un service sans avoir besoin de modéliser le traitement d'une requête et l'échange entre deux équipements à partir de données plus bas niveau, comme celles de la QoS réseau. Cependant pour obtenir ces valeurs, il est nécessaire de déployer réellement le micro-service sur un équipement et de mesurer son temps de réponse depuis un second équipement.

Une première piste que nous envisageons pour améliorer notre système est de pouvoir évaluer à un même moment, dans les mêmes conditions d'utilisation, plusieurs déploiements de l'application. Cette technique, appelée *Canary deployment*[41], permet d'obtenir, grâce à ces déploiements alternatifs, de nouvelles mesures et ainsi mettre à jour plus rapidement les matrices R_s. Nous n'avons pas mis en application cette technique car nous n'avions pas à notre disposition un nombre suffisant d'équipements. Une autre solution potentielle s'appuie sur l'hypothèse que si le temps de réponse d'un service entre deux équipements augmente, il est probable qu'il en soit de même pour un autre service avec des exigences similaires en ressources de calcul et en capacité réseau. Il pourrait être ainsi envisageable de mettre à jour différentes matrices $R_{s(q)}$ à partir d'une seule mesure pour un service. Cependant cette hypothèse demande une certaine connaissance préalable de l'application, et peut-être même un apprentissage du comportement des différents services en fonction des conditions d'utilisation pour en extraire des classes d'équivalence entre services.

Chapitre 6

Conclusions et perspectives

Un expert est celui qui a pu faire toutes les erreurs possibles dans un domaine très précis.
— **Niels Bohr**, *cité par Edward Teller.*

6.1 Conclusions sur les travaux menés

Nous avons, en introduction de cette thèse, identifié les différentes problématiques soulevées par le concept d'infrastructure participative d'hébergement de service à destination d'une communauté virtuelle. Nos contributions se sont concentrées sur la recherche d'un placement des micro-services améliorant la qualité d'expérience d'une application par ses utilisateurs, ainsi que sur le maintien de cette qualité par l'adaptation du placement selon l'évolution des conditions d'utilisation. Nous allons résumer ici les points marquants de ces contributions.

6.1.1 Placement des micro-services pour optimiser le temps de réponse de l'application

Nous avons défini un modèle du temps de réponse d'une application composée de micro-services. Ce modèle s'appuie sur le temps de réponse spécifique $r_s(i, j)$ d'un micro-service selon l'équipement i qui l'héberge et l'équipement j qui le consulte. Ce temps de réponse spécifique inclue le temps de traitement de la requête par le service, ainsi que les temps de communication des données sur le réseau entre les équipements i et j. Ce temps n'inclue par contre pas le temps de traitement d'autres requêtes éventuelles du service s vers d'autres services. En connaissant ce temps de réponse, il est possible de définir le temps de réponse global de l'application comme l'addition des temps de réponse spécifiques de chaque service.

Le temps de réponse spécifique $r_s(i, j)$ ne dépend que des capacités de calcul de l'équipement i qui héberge le service et de la QoS réseau entre les équipements i et j. Cette valeur est donc

représentative de la disponibilité des ressources nécessaires à cet échange. Optimiser le temps de réponse global de l'application selon les ressources disponibles dans l'infrastructure participative revient alors à trouver un placement des différents services sur les équipements qui minimise la somme des temps de réponses spécifiques. Il est possible d'évaluer à priori les différentes valeurs de $r_s(i, j)$, soit à partir d'un modèle du temps de réponse de chaque service, soit à partir de mesures. Le choix du placement de chaque service se fait ensuite en fonction de ces valeurs.

Ce problème présente une complexité quadratique, car le placement d'un service va influer sur le temps de réponse d'autres services. Il est donc nécessaire d'utiliser une heuristique pour trouver une approximation de la solution optimale. Nous avons proposé deux heuristiques, l'une gloutonne et l'autre basée sur la méta-heuristique PSO, dont nous avons évalué les solutions de placement d'une application sur des infrastructures avec des configurations différentes. La comparaison des ces solutions avec la solution optimale montre que ces heuristiques proposent des résultats avec une approximation faible (déviation moyenne $< 5\%$ de la solution optimale), avec un temps de calcul beaucoup plus faible. Cependant l'heuristique PSO démontre une meilleure fiabilité dans ces résultats.

Bien que nos résultats n'aient été obtenus qu'à partir d'une seule application assez simple, nous pensons que l'heuristique proposée est transposable à d'autres applications. Un modèle de temps de réponse peut être construit à partir du graphe d'appel entre micro-services de l'application, avec la contrainte qu'il doit être acyclique. Les temps de réponse spécifiques de chaque micro-service sont accessibles soit par mesures dans certaines conditions d'utilisation, soit par un modèle simple basé sur un temps de traitement standard et des volumes de données échangés raisonnables. Ces valeurs, regroupées dans les matrices R_s servent ensuite de base au choix du placement par l'heuristique PSO.

6.1.2 Évaluation dans des conditions réelles d'utilisation

L'étape suivante de notre travail est d'évaluer la variation du temps de réponse de l'application dans des conditions réelles d'utilisation. L'objectif est d'évaluer si le gain obtenu par l'optimisation proposée dans la première partie de cette thèse n'est pas effacé dès que ces conditions changent. Nous nous sommes intéressés aux variations des paramètres de QoS réseau : bandes passantes disponibles montante et descendante, latence et perte de paquet. Ces paramètres sont amenés à être modifiés notamment à cause du trafic concurrent à celui de l'application hébergée. Une première phase de ce travail est donc de définir, pour chacun de ces paramètres, des intervalles caractéristiques de conditions réelles d'utilisation. Nous nous sommes appuyés pour

cela sur des résultats disponibles dans la littérature, que nous avons complété par des mesures effectuées sur des réseaux d'accès résidentiels de foyers volontaires.

Nous avons ensuite cherché à caractériser la variation du temps de réponse de l'application lorsque chacun de ces paramètres varie dans l'intervalle que nous avons défini. Nous avons utilisé pour cela une approche par émulation qui nous permet d'utiliser les mêmes micro-services que ceux déployés dans l'infrastructure participative, mais de les interconnecter par un réseau virtuel dont nous maitrisons les caractéristiques. Nous définissons ainsi dans ce réseau des liens avec une bande passante et une latence réalistes par rapport aux technologies des réseaux d'accès résidentiels. Cette topologie nous permet de déduire, à travers notre heuristique, un placement dans le réseau proche de l'optimal pour chacun des micro-services de l'application. Nous avons ensuite mesuré le temps de réponse de l'application avec des valeurs différentes des paramètres de QoS réseau. Ces mesures ont montré qu'avec des variations de ces paramètres représentant des conditions réelles d'utilisation, le temps de réponse ne se dégrade pas de manière importante. Ces résultats nous permettent donc de conclure que l'optimisation réalisée grâce à l'heuristique reste pertinente, même lorsque les conditions d'utilisation du réseau évoluent.

Nous avons ensuite reproduit cette étude sur une infrastructure participative réelle, constituée d'équipements placés dans le réseau domestique de participants volontaires. Ces équipements sont équipés de l'outillage nécessaire au déploiement et à l'interconnexion des micro-services pour permettre le fonctionnement de l'application. Cette infrastructure nous a permis d'étudier, sur un temps suffisamment long, le comportement de l'application avec un trafic utilisateur réel. Nous avons mesuré en parallèle le temps de réponse de l'application et les paramètres de QoS réseau des réseaux d'accès résidentiels des participants. Les résultats de ces mesures corroborent ceux obtenus sur une infrastructure émulée. Nous avons cependant constaté que certains résultats montrent une dégradation du temps de réponse de l'application dans des conditions d'utilisation pourtant favorables. Une analyse plus détaillée de ces résultats a révélé qu'ils étaient obtenus lorsque les conditions d'utilisation sur un réseau d'accès en particulier étaient perturbées. L'équipement localisé sur ce réseau hébergeant un des micro-services de l'application, ces perturbations ont entrainé une dégradation générale du temps de réponse.

Ce constat illustre la difficulté d'expliquer les variations du temps de réponse à partir de valeurs moyennes des paramètres de QoS. Une perturbation localisée, peu représentée dans une moyenne globale, a des répercussions importantes lorsqu'elle affecte un des services de l'application. Une solution pour trouver une explication pourrait être de superviser l'ensemble de ces paramètres pour détecter ces perturbations. Mais un tel dispositif passerait difficilement à l'échelle de plusieurs dizaines d'utilisateurs et de services. Une alternative, inspirée du modèle que nous avons proposé dans la première partie de cette thèse, est de superviser les temps de réponse spéci-

fiques de chaque micro-service. Une variation d'une de ces valeurs, induite par une variation de la disponibilité des ressources de calcul ou de réseau, nous permettrait de localiser l'origine des variations constatée pour le temps de réponse de l'application.

6.1.3 Adaptation du placement de micro-services

Nous avons donc intégré cette supervision du temps de réponse spécifique de chaque micro-service à notre infrastructure participative. Le dispositif utilisé se base sur l'interception des requêtes et des réponses au niveau d'un relai applicatif (*proxy*) déployé sur chaque équipement hébergeant un service. Les mesures des temps de réponse de chaque requête effectuée par ces relais doivent être centralisées pour en extraire les temps de réponse spécifiques de chaque micro-service. Ces valeurs nous permettent d'analyser le temps de réponse de l'application comme une somme de temps de réponse spécifique mesurée entre les micro-services. Cette analyse permet ainsi de détecter une évolution locale des conditions d'utilisation qui dégrade la qualité d'expérience.

Pour répondre à cette dégradation et essayer de maintenir la qualité d'expérience, une solution est d'adapter le placement des micro-services, en déplaçant par exemple le service impacté vers un autre équipement. Il nous faut d'abord pour cela détecter quand une adaptation du placement est pertinente. Ensuite le choix d'un nouveau placement doit être guidé par la connaissance des conditions actuelles pour que l'adaptation n'aboutisse pas à une dégradation de l'expérience utilisateur. Nous avons donc proposé une extension de notre architecture d'orchestration des micro-services qui intègre une chaine de décision autonome d'adaptation du placement. La décision d'adaptation est pilotée par une stratégie considérant certains indicateurs clés. Le choix du nouveau placement se base sur l'infrastructure de supervision du temps de réponse spécifique.

Nous avons testé ce processus d'adaptation sur notre infrastructure participative réelle, en utilisant une première stratégie simple consistant à adapter le placement à intervalle de temps régulier. Nous avons constaté que cette stratégie a des résultats aléatoires, la probabilité qu'elle aboutisse à une dégradation des performances étant assez forte (proche de 33%). Ces résultats s'expliquent par un mauvais choix du nouveau placement, lié à une connaissance incomplète des conditions d'utilisation. Nous avons donc proposé une évolution de cette stratégie qui essaie de corriger ces mauvais choix. Lorsqu'un tel placement est utilisé, les mesures de temps de réponses spécifiques viennent actualiser les matrices $R_{s(q)}$, utilisées ensuite pour le choix du nouveau placement. L'adaptation suivante profite donc de cette information pour proposer un nouveau placement, potentiellement plus performant. Les tests effectués avec cette nouvelle stratégie ont montré une amélioration par rapport à la stratégie précédente, les adaptations entrainant une dégradation ne représentant plus que 20%.

Ces résultats sont satisfaisants mais pas encore tout à fait concluants. Il reste encore une marge d'amélioration pour que l'adaptation du placement puisse aboutir au moins à un maintien de la qualité d'expérience, au mieux à une amélioration. Il existe bien sûr des cas, lorsque l'ensemble des réseaux participants sont perturbés, pour lesquels il sera difficile de maintenir cette expérience par l'adaptation du placement. Dans d'autres cas cependant, notre processus d'adaptation s'appuie sur une connaissance partielle des conditions courantes d'utilisation conduisant au choix d'un placement sous-performant. Ce choix s'appuie sur les mesures les plus récentes concernant les micro-services déployés et un historique de mesures provenant de déploiements précédents, dans des conditions d'utilisation potentiellement différentes. La solution idéale serait de mesurer l'ensemble des temps de réponse spécifiques des différents services sur les différents équipements, mais cette tâche est bien sûr trop complexe. Une piste que nous envisageons serait de superviser les temps de réponse de déploiements alternatifs en parallèle du déploiement choisi, permettant ainsi d'obtenir des mesures complémentaires de l'historique.

Nous avons publié à partir de ces travaux trois articles dans des conférences nationales et internationales :

— Dans l'article *"Vers une plate-forme communautaire d'hébergement à base de micro-services collaboratifs"* (COMPAS 2017) [90], nous explicitons le contexte et les problèmes que nous nous proposons de résoudre dans cette thèse,
— Dans l'article *"Optimizing the Performance of a Microservice-Based Application Deployed on User-Provided Devices"* (ISPDC 2018) [88], nous décrivons notre modèle du temps de réponse et proposons l'heuristique PSO comme solution d'optimisation,
— Dans l'article *"QoS-aware Autonomic Adaptation of Microservices Placement on Edge Devices"* (CLOSER 2020) [89], nous exposons les résultats de notre architecture d'adaptation du placement des micro-services.

6.2 Perspectives ouvertes

Les travaux que nous avons menés dans cette thèse ouvrent plusieurs perspectives d'études. Nous proposons ainsi d'approfondir nos solutions en remettant en cause des hypothèses que nous avons formulées. Le concept d'infrastructure participative que nous avons étudié ici, pourra aussi faire l'objet de recherche selon des approches que nous n'avons pas explorées dans le cadre de cette thèse.

6.2.1 Extension des contributions

Dans notre étude, nous avons émis l'hypothèse que l'application ne traite qu'une seule requête utilisateur. Nous ne prenons donc pas en compte de délai supplémentaire d'attente de traitement des requêtes au niveau de chaque micro-service. Ce délai est incompressible et ne peut se réduire en modifiant le placement des services. Une solution serait alors de multiplier le nombre d'instances de chaque micro-service pour traiter ces requêtes concurrentes en parallèle au sein de l'infrastructure participative. Le système est ainsi capable de traiter un nombre plus important de requêtes simultanées. Ce débit applicatif (*Throughput*) peut être considéré comme un nouveau paramètre de QoS à prendre en compte dans l'optimisation du déploiement de l'application. L'orchestrateur doit alors décider à la fois du nombre d'instance de chaque service en plus du placement de chaque instance. La supervision du nombre et du type de requêtes en cours de traitement pourra ensuite contrôler dynamiquement ce nombre d'instance pour s'adapter à la demande utilisateur.

Une autre hypothèse formulée dans cette étude est que l'ensemble des clients utilisés pour l'évaluation de la qualité d'expérience est connu et stable. Nous sommes en effet capables, dans le contexte d'une communauté virtuelle, de connaitre l'ensemble des utilisateurs potentiels de l'application. L'optimisation du placement est ainsi faite pour ces utilisateurs. Or ceux-ci n'accèdent pas forcément à l'application dans la même période de temps. Il est donc envisageable d'évaluer la qualité d'expérience non plus pour l'ensemble des utilisateurs potentiels mais pour les utilisateurs actifs pendant un certain intervalle de temps. Le placement des micro-services de l'application s'adapterait ainsi dynamiquement à l'évolution dans le temps de la demande des utilisateurs. Cette approche pourrait ouvrir des perspectives de transposition de nos solutions dans des contextes d'hébergement de services en bordure pour des utilisateurs mobiles, comme les transports intelligents.

Nous avons enfin considéré dans notre étude le déploiement et l'adaptation de micro-services ne participant qu'à une seule application. Or l'un des intérêts des micro-services est qu'une instance puisse participer simultanément à plusieurs applications. Il serait possible de considérer l'orchestration de chacune de ces applications de manière totalement indépendante en déployant pour chacune ses propres instances de micro-services. La problématique devient plus complexe si ces instances sont susceptibles d'être utilisées par plusieurs applications. Le placement de ces micro-services va influer sur le temps de réponse de plusieurs applications. La décision de ce placement pour optimiser ses temps de réponse devra être prise en considérant plusieurs fonctions objectifs. L'utilisation d'instances de micro-services communes à plusieurs applications demandera peut-être une reformulation du problème et des modèles que nous avons proposés.

6.2.2 Étude sur l'infrastructure participative

Passage à l'échelle

Nous avons considéré dans notre étude une infrastructure participative limitée à une dizaine d'équipements, dont la distribution à l'échelle de l'Internet était assez réduite. Nous pensons possible d'envisager des infrastructures participatives à plus grande échelle qui permettraient à une large communauté d'utilisateurs, répartie dans plusieurs pays, de collaborer grâce aux services hébergés sur des équipements partagés par ses membres. Une telle solution devra être suffisamment robuste pour assurer la continuité des services en cas de rupture de connectivités à large échelle (*churns*). Le placement des services et son adaptation devra aussi être capable de prendre en compte un nombre important d'équipements.

Pour ces raisons, notre solution n'est pas directement utilisable dans un tel cas d'usage. L'un des principaux verrous à ce passage à l'échelle de notre solution est qu'elle centralise les fonctions de supervision, de décision et d'orchestration des services. Ces fonctions sont réalisées dans un composant du système ayant une connaissance de l'ensemble du système : équipements disponibles, clients actifs, temps de réponses mesurés. La création d'une telle vue globale deviendra de plus en plus difficile à réaliser à mesure que l'infrastructure va comporter plus d'équipements, d'utilisateurs et de services. Cette vue globale étant nécessaire pour prendre des décisions d'adaptation, celles-ci risquent d'intervenir trop tard par rapport aux variations des conditions d'utilisation.

Une approche pour contourner cette difficulté, qui pourrait constituer une perspective d'étude intéressante, consiste à déléguer l'orchestration des services au niveau des équipements participants. Chacun de ces équipements aurait une certaine autonomie dans la décision de déployer (ou non) des micro-services de l'application, à partir d'une vue locale de l'état du système. Cette approche, sans coordination centralisée, a notamment été développée dans le concept de chorégraphie de service [15]. Le défi que pose cette approche est de définir une stratégie locale de décision au niveau de chaque équipement qui aboutisse à une amélioration globale de l'expérience utilisateur.

Sécurité

Une infrastructure participative reposant sur le réseau ouvert qu'est l'Internet, elle est donc susceptible d'être compromise par des actions malveillantes provenant de l'extérieur. Nous avons déjà exposé en introduction de cette thèse les risques que représente l'utilisation de cette solution pour l'hébergement des services d'une communauté. La recherche de solutions pour réduire ces

risques nous semble un axe de recherche intéressant car abordant des problématiques originales dans le domaine de la sécurité.

Une question à résoudre par exemple sera d'assurer un contrôle sur les équipements participant à l'infrastructure et éviter d'y introduire des équipements malveillants risquant de perturber son fonctionnement. Dans l'infrastructure participative que nous avons utilisée, la participation des équipements est contrôlée à travers le service de découverte Consul. Un nouvel équipement peut s'y enregistrer en connaissant le point d'entrée et une clé partagée. Ce mécanisme n'est pas satisfaisant et de meilleures solutions seront nécessaires pour sécuriser l'ajout de nouveaux participants. L'un des défis sera de proposer une solution pouvant fonctionner de manière décentralisée, étant donné la nature spontanée de l'infrastructure participative.

Les échanges entre micro-services de l'application sont eux aussi exposés sur l'Internet et rendent ainsi les informations transportées vulnérables à une captation ou une corruption des données. Une solution garantissant la confidentialité et l'intégrité de ces échanges est donc souhaitable. Dans notre cas, ces échanges utilisent le protocole HTTP non chiffré. L'utilisation de sa version chiffrée HTTPS nous semble difficile dans notre cas. En effet le chiffrement repose alors sur un certificat lié au service, à son emplacement et une autorité. L'emplacement du service pouvant changer dans notre cas, il nous semble difficile de générer dynamiquement auprès de l'autorité des certificats à chaque déploiement du service. Il nous parait donc intéressant d'étudier l'intégration de mécanismes de chiffrements décentralisés dans les communications entre micro-services.

BIBLIOGRAPHIE

[1] Michael ADEYEYE et Paul GARDNER-STEPHEN, « The Village Telco Project : A Reliable and Practical Wireless Mesh Telephony Infrastructure », in : *EURASIP Journal on Wireless Communications and Networking* 2011.*1* (août 2011), p. 78, ISSN : 1687-1499, DOI : 10.1186/1687-1499-2011-78.

[2] AERIS, *Auto-Hébergement, Fausse Bonne Idée*, Vannes, 2017.

[3] Arif AHMED et al., « Fog Computing Applications : Taxonomy and Requirements », in : *arXiv:1907.11621 [cs]* (juill. 2019), arXiv : 1907.11621 [cs].

[4] Babar ALI et al., « A Volunteer-Supported Fog Computing Environment for Delay-Sensitive IoT Applications », in : *IEEE Internet of Things Journal* 8.*5* (mars 2021), p. 3822-3830, ISSN : 2327-4662, DOI : 10.1109/JIOT.2020.3024823.

[5] M. ALICHERRY et T. V. LAKSHMAN, « Network Aware Resource Allocation in Distributed Clouds », in : *2012 Proceedings IEEE INFOCOM*, mars 2012, p. 963-971, DOI : 10.1109/INFCOM.2012.6195847.

[6] D. P. ANDERSON, « BOINC : A System for Public-Resource Computing and Storage », in : *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop On*, nov. 2004, p. 4-10, DOI : 10.1109/GRID.2004.14.

[7] David P. ANDERSON et al., « SETI@Home : An Experiment in Public-Resource Computing », in : *Commun. ACM* 45.*11* (nov. 2002), p. 56-61, ISSN : 0001-0782, DOI : 10.1145/581571.581573.

[8] ARCEP, *Baromètre Annuel de La Transition Vers IPv6 En France*, rapp. tech., déc. 2020.

[9] ARCEP, *Observatoire Haut et Très Haut Débit : Abonnements et Deploiements (T1 2021)*, rapp. tech., juin 2021.

[10] Ram Govinda ARYAL et Jörn ALTMANN, « Dynamic Application Deployment in Federations of Clouds and Edge Resources Using a Multiobjective Optimization AI Algorithm », in : *Third International Conference on Fog and Mobile Edge Computing (FMEC)*, avr. 2018, DOI : 10.1109/FMEC.2018.8364057.

[11] Rafael Aschoff et Andrea Zisman, « QoS-Driven Proactive Adaptation of Service Composition », in : *Service-Oriented Computing*, Springer, 2011, isbn : 978-3-642-25535-9.

[12] Ozalp Babaoglu, Moreno Marzolla et Michele Tamburini, « Design and Implementation of a P2P Cloud System », in : *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, New York, NY, USA : ACM, 2012, p. 412-417, isbn : 978-1-4503-0857-1, doi : 10.1145/2245276.2245357.

[13] Tayebeh Bahreini et Daniel Grosu, « Efficient Placement of Multi-Component Applications in Edge Computing Systems », in : *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, ACM, 2017, isbn : 978-1-4503-5087-7, doi : 10.1145/3132211.3134454.

[14] Roger Baig et al., « Guifi.Net, a Crowdsourced Network Infrastructure Held in Common », in : *Computer Networks*, Crowdsourcing 90 (oct. 2015), p. 150-165, issn : 1389-1286, doi : 10.1016/j.comnet.2015.07.009.

[15] A. Barker, C. D. Walton et D. Robertson, « Choreographing Web Services », in : *IEEE Transactions on Services Computing* 2.2 (avr. 2009), p. 152-166, issn : 1939-1374, doi : 10.1109/TSC.2009.8.

[16] Adam L Beberg et al., « Folding@ Home : Lessons from Eight Years of Volunteer Distributed Computing », in : *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium On*, IEEE, 2009, p. 1-8.

[17] Flavio Bonomi et al., « Fog Computing and Its Role in the Internet of Things », in : *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, Helsinki, Finland : Association for Computing Machinery, août 2012, p. 13-16, isbn : 978-1-4503-1519-7, doi : 10.1145/2342509.2342513.

[18] Bart Braem et al., « Analysis of End-User QoE in Community Networks », in : *Proceedings of the 2015 Annual Symposium on Computing for Development*, DEV '15, New York, NY, USA : Association for Computing Machinery, déc. 2015, p. 159-166, isbn : 978-1-4503-3490-7, doi : 10.1145/2830629.2830639.

[19] Valeria Cardellini et al., « MOSES : A Platform for Experimenting with QoS-Driven Self-Adaptation Policies for Service Oriented Systems », in : *Soft. Eng. for Self-Adaptive Systems*, Springer, 2017, isbn : 978-3-319-74183-3, doi : 10.1007/978-3-319-74183-3_14.

[20] Valentina Casola et al., « An Automatic Tool for Benchmark Testing of Cloud Applications : » in : *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, Porto, Portugal : SCITEPRESS - Science and Technology Publications, 2017, p. 729-736, isbn : 978-989-758-243-1, doi : 10.5220/0006379507290736.

[21] K. Chard et al., « Social Cloud Computing : A Vision for Socially Motivated Resource Sharing », in : *IEEE Transactions on Services Computing* 5.*4* (Fourth 2012), p. 551-563, ISSN : 1939-1374, DOI : 10.1109/TSC.2011.39.

[22] Eric K. Clemons, « The Complex Problem of Monetizing Virtual Electronic Social Networks », in : *Decision Support Systems*, Information Product Markets 48.*1* (déc. 2009), p. 46-56, ISSN : 0167-9236, DOI : 10.1016/j.dss.2009.05.003.

[23] CNC, *Le Marché Du Jeu Vidéo En 2018 | CNC*, https://www.cnc.fr/professionnels/etudes-et-rapports/etudes-prospectives/le-marche-du-jeu-video-en-2018_1072060, 2018.

[24] Bastien Confais, Adrien Lèbre et Benoît Parrein, « Quel système de stockage pour les architectures Fog ? », in : *Compas'2016*, Lorient, France, juill. 2016.

[25] CSA, *L'équipement audiovisuel des foyers aux 1er et 2e trimestres 2019 (TV) - CSA - Conseil supérieur de l'audiovisuel*, https://www.csa.fr/Informer/Collections-du-CSA/Panorama-Toutes-les-etudes-liees-a-l-ecosysteme-audiovisuel/Les-observatoires-de-l-equipement-audiovisuel/L-equipement-audiovisuel-des-foyers-aux-1er-et-2e-trimestres-2019-TV, 2019.

[26] V. D. Cunsolo et al., « Volunteer Computing and Desktop Cloud : The Cloud@Home Paradigm », in : *IEEE International Symposium on Network Computing and Applications*, juill. 2009, DOI : 10.1109/NCA.2009.41.

[27] Vincenzo D. Cunsolo et al., « Cloud@Home : Bridging the Gap between Volunteer and Cloud Computing », in : *Emerging Intelligent Computing Technology and Applications*, sous la dir. de De-Shuang Huang et al., Lecture Notes in Computer Science, Berlin, Heidelberg : Springer, 2009, p. 423-432, ISBN : 978-3-642-04070-2, DOI : 10.1007/978-3-642-04070-2_48.

[28] Xiaohong Deng et al., « Measuring Broadband Performance Using M-Lab : Why Averages Tell a Poor Tale », in : *2015 International Telecommunication Networks and Applications Conference (ITNAC)*, nov. 2015, p. 24-29, DOI : 10.1109/ATNAC.2015.7366784.

[29] Salvatore Distefano, Giovanni Merlino et Antonio Puliafito, « An OpenStack-Based Implementation of a Volunteer Cloud », in : *Advances in Service-Oriented and Cloud Computing*, sous la dir. d'Antonio Celesti et Philipp Leitner, Communications in Computer and Information Science, Springer International Publishing, sept. 2015, p. 389-403, ISBN : 978-3-319-33312-0 978-3-319-33313-7, DOI : 10.1007/978-3-319-33313-7_30.

[30] Daniel J. Dubois et al., « Mycocloud : Elasticity through Self-Organized Service Placement in Decentralized Clouds », in : *2015 IEEE 8th International Conference on Cloud Computing*, juin 2015, p. 629-636, DOI : 10.1109/CLOUD.2015.89.

[31] R. Eberhart et J. Kennedy, « A New Optimizer Using Particle Swarm Theory », in : , *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995. MHS '95*, oct. 1995, p. 39-43, doi : 10.1109/MHS.1995.494215.

[32] S. Egger et al., « Time Is Bandwidth ? Narrowing the Gap between Subjective Time Perception and Quality of Experience », in : *IEEE International Conference on Communications*, juin 2012, doi : 10.1109/ICC.2012.6363769.

[33] Pau Escrich et al., « Community Home Gateways for P2P Clouds », in : *IEEE P2P 2013 Proceedings*, sept. 2013, p. 1-2, doi : 10.1109/P2P.2013.6688732.

[34] European Commission, Directorate-General for the Information Society and Media et SamKnows Limited, *Quality of Broadband Services in the EU : Final Report : October 2014.* Luxembourg : Publications Office, 2015, isbn : 978-92-79-49486-4.

[35] Rob Flickenger, *Building Wireless Community Networks*, 1st ed, Sebastopol, CA : O'Reilly, 2002, isbn : 978-0-596-00204-6.

[36] Martin Fowler et James Lewis, « Microservices », in : *martinfowler.com* (avr. 2014).

[37] Yu Gan et Christina Delimitrou, « The Architectural Implications of Cloud Microservices », in : *IEEE Computer Architecture Letters* 17.*2* (juill. 2018), p. 155-158, issn : 1556-6064, doi : 10.1109/LCA.2018.2839189.

[38] Pradipta Ghosh, Quynh Nguyen et Bhaskar Krishnamachari, « Container Orchestration for Dispersed Computing », in : *Proceedings of the 5th International Workshop on Container Technologies and Container Clouds*, WOC '19, Davis, CA, USA : Association for Computing Machinery, déc. 2019, p. 19-24, isbn : 978-1-4503-7033-2, doi : 10.1145/3366615.3368354.

[39] Astrid GIRARDEAU, « Tout le monde a intérêt à transformer Internet en Minitel », in : *Libération* (fév. 2009).

[40] Nikolay Grozev et Rajkumar Buyya, « Multi-Cloud Provisioning and Load Distribution for Three-Tier Applications », in : *ACM Trans. Auton. Adapt. Syst.* (oct. 2014), issn : 1556-4665, doi : 10.1145/2662112.

[41] Robert Heinrich et al., « Performance Engineering for Microservices : Research Challenges and Directions », in : *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*, ICPE '17 Companion, New York, NY, USA : Association for Computing Machinery, avr. 2017, p. 223-226, isbn : 978-1-4503-4899-7, doi : 10.1145/3053600.3053653.

[42] F. Henri et B. Pudelko, « Understanding and Analysing Activity and Learning in Virtual Communities », in : *Journal of Computer Assisted Learning* 19.*4* (2003), p. 474-487, issn : 1365-2729, doi : 10.1046/j.0266-4909.2003.00051.x.

[43] Julia Hielscher et al., « A Framework for Proactive Self-Adaptation of Service-Based Applications Based on Online Testing », in : *Towards a Service-Based Internet*, Springer, déc. 2008, doi : 10.1007/978-3-540-89897-9_11.

[44] Toke Høiland-Jørgensen et al., « Measuring Latency Variation in the Internet », in : *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '16, New York, NY, USA : Association for Computing Machinery, déc. 2016, p. 473-480, isbn : 978-1-4503-4292-6, doi : 10.1145/2999572.2999603.

[45] Markus C. Huebscher et Julie A. McCann, « A Survey of Autonomic Computing : Degrees, Models, and Applications », in : *ACM Computing Surveys (CSUR)* (août 2008).

[46] Boris Iglewicz et David Caster Hoaglin, *How to Detect and Handle Outliers*, ASQC Quality Press, 1993, isbn : 978-0-87389-247-6.

[47] INSEE, *Accès et Utilisation de l'internet Dans l'Union Européenne | Insee*, https://www.insee.fr/fr/statistiques/2385835, 2019.

[48] Meron Istifanos et Israel Tekahun, *Performance Evaluation of Raspberry Pi 3B as a Web Server*, rapp. tech., Blekinge Institute of Technology, 2020, p. 48.

[49] C. Jatoth, G. R. Gangadharan et R. Buyya, « Computational Intelligence Based QoS-Aware Web Service Composition : A Systematic Literature Review », in : *IEEE Transactions on Services Computing* PP.*99* (2015), p. 1-1, issn : 1939-1374, doi : 10.1109/TSC.2015.2473840.

[50] Paridhika Kayal et Jörg Liebeherr, « Distributed Service Placement in Fog Computing : An Iterative Combinatorial Auction Approach », in : *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, juill. 2019, p. 2145-2156, doi : 10.1109/ICDCS.2019.00211.

[51] Jeffrey Kephart et William Walsh, *An Architectural Blueprint for Autonomic Computing*, rapp. tech., IBM, 2003.

[52] Amin M. Khan et Felix Freitag, « On Edge Cloud Service Provision with Distributed Home Servers », in : *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, déc. 2017, p. 223-226, doi : 10.1109/CloudCom.2017.50.

[53] Amin M. Khan, Mennan Selimi et Felix Freitag, « Towards Distributed Architecture for Collaborative Cloud Services in Community Networks », in : *6th International Conference on Intelligent Networking and Collaborative Systems (INCoS'14). Salerno, Italy : IEEE*, 2014.

[54] Reihaneh Khorsand, Mostafa Ghobaei-Arani et Mohammadreza Ramezanpour, « A Self-Learning Fuzzy Approach for Proactive Resource Provisioning in Cloud Environment », in : *Software : Practice and Experience* 49.*11* (2019), p. 1618-1642, issn : 1097-024X, doi : 10.1002/spe.2737.

[55] Reihaneh Khorsand, Mostafa Ghobaei-Arani et Mohammadreza Ramezanpour, « FAHP Approach for Autonomic Resource Provisioning of Multitier Applications in Cloud Computing Environments », in : *Software : Practice and Experience* (2018), issn : 1097-024X, doi : 10.1002/spe.2627.

[56] Derrick Kondo et al., « Cost-Benefit Analysis of Cloud Computing versus Desktop Grids », in : *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium On*, IEEE, 2009, p. 1-12.

[57] Wenfeng Li et al., « Resource Virtualization and Service Selection in Cloud Logistics », in : *Journal of Network and Computer Applications* 36.*6* (nov. 2013), p. 1696-1704, issn : 1084-8045, doi : 10.1016/j.jnca.2013.02.019.

[58] J. C. R. Licklider, *The Computer as a Communication Device*, /paper/The-Computer-as-a-Communication-Device-Licklider/1b0a846dfb21e32c2f271b0d7c0bc21a5f8fe728, 1968.

[59] Renato Lo Cigno et Leonardo Maccari, « Urban Wireless Community Networks : Challenges and Solutions for Smart City Communications », in : *Proceedings of the 2014 ACM International Workshop on Wireless and Mobile Technologies for Smart Cities*, WiMobCity '14, New York, NY, USA : Association for Computing Machinery, août 2014, p. 49-54, isbn : 978-1-4503-3036-7, doi : 10.1145/2633661.2633669.

[60] Leonardo Maccari, « An Analysis of the Ninux Wireless Community Network », in : *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, oct. 2013, p. 1-7, doi : 10.1109/WiMOB.2013.6673332.

[61] Philip Mayer et al., « The Autonomic Cloud : A Vision of Voluntary, Peer-2-Peer Cloud Computing », in : *Self-Adaptation and Self-Organizing Systems Workshops (SASOW), 2013 IEEE 7th International Conference On*, IEEE, 2013, p. 89-94.

[62] Tessema M. Mengistu et Dunren Che, « Survey and Taxonomy of Volunteer Computing », in : *ACM Computing Surveys* 52.*3* (juill. 2019), 59:1-59:35, issn : 0360-0300, doi : 10.1145/3320073.

[63] Tessema M. Mengistu et al., « cuCloud : Volunteer Computing as a Service (VCaaS) System », in : *Cloud Computing CLOUD 2018*, sous la dir. de Min Luo et Liang-Jie Zhang, Lecture Notes in Computer Science, Cham : Springer International Publishing, 2018, p. 251-264, isbn : 978-3-319-94295-7, doi : 10.1007/978-3-319-94295-7_17.

[64] Dejan S Milojicic et al., « Peer-to-Peer Computing », in : (2002), p. 52.

[65] Taariq MULLINS, « Participatory Cloud Computing : The Community Cloud Management Protocol », in : (2014).

[66] Leah MUTANU et Gerald KOTONYA, « What, Where, When, How and Right of Runtime Adaptation in Service-Oriented Systems », in : *Service-Oriented Computing ICSOC 2017 Workshops*, Lecture Notes in Computer Science, Springer, 2018, p. 30-42, ISBN : 978-3-319-91764-1, DOI : 10.1007/978-3-319-91764-1_3.

[67] Sam NEWMAN, *Building Microservices : Designing Fine-Grained Systems*, "O'Reilly Media, Inc.", fév. 2015, ISBN : 978-1-4919-5033-3.

[68] Takayuki NISHIO et al., « Service-Oriented Heterogeneous Resource Sharing for Optimizing Service Latency in Mobile Cloud », in : *First International Workshop on Mobile Cloud Computing & Networking*, 2013, ISBN : 978-1-4503-2206-5, DOI : 10.1145/2492348.2492354.

[69] F. PARAISO et al., « A Federated Multi-Cloud PaaS Infrastructure », in : *2012 IEEE Fifth International Conference on Cloud Computing*, juin 2012, p. 392-399, DOI : 10.1109/CLOUD.2012.79.

[70] Serge PROULX et Guillaume LATZKO-TOTH, « La virtualité comme catégorie pour penser le social : L'usage de la notion de communauté virtuelle », in : *Sociologie et sociétés* 32.2 (2000), p. 99, ISSN : 0038-030X, 1492-1375, DOI : 10.7202/001598ar.

[71] Howard RHEINGOLD, *Smart Mobs : The Next Social Revolution*, Hachette UK, mars 2007, ISBN : 978-0-465-00439-3.

[72] Howard RHEINGOLD, *The Virtual Community : Homesteading on the Electronic Frontier*, MIT Press, oct. 2000, ISBN : 978-0-262-26110-4.

[73] Daniel ROBINSON et Gerald KOTONYA, « A Runtime Quality Architecture for Service-Oriented Systems », in : *Service-Oriented Computing ICSOC 2008*, sous la dir. d'Athman BOUGUETTAYA, Ingolf KRUEGER et Tiziana MARGARIA, Lecture Notes in Computer Science, Berlin, Heidelberg : Springer, 2008, p. 468-482, ISBN : 978-3-540-89652-4, DOI : 10.1007/978-3-540-89652-4_35.

[74] M. RYDEN et al., « Nebula : Distributed Edge Cloud for Data Intensive Computing », in : *2014 IEEE International Conference on Cloud Engineering*, mars 2014, p. 57-66, DOI : 10.1109/IC2E.2014.34.

[75] Yuvraj SAHNI et al., « Edge Mesh : A New Paradigm to Enable Distributed Intelligence in Internet of Things », in : *IEEE Access* 5 (2017), p. 16441-16458, ISSN : 2169-3536, DOI : 10.1109/ACCESS.2017.2739804.

[76] Hani Sami et Azzam Mourad, « Towards Dynamic On-Demand Fog Computing Formation Based On Containerization Technology », in : *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, déc. 2018, p. 960-965, doi : 10.1109/CSCI46756.2018.00187.

[77] Adalberto R. Sampaio et al., « Improving Microservice-Based Applications with Runtime Placement Adaptation », in : *Journal of Internet Services and Applications* 10.*1* (fév. 2019), p. 4, issn : 1869-0238, doi : 10.1186/s13174-019-0104-0.

[78] Mennan Selimi et al., « A Lightweight Service Placement Approach for Community Network Micro-Clouds », in : *Journal of Grid Computing* 17.*1* (mars 2019), p. 169-189, issn : 1572-9184, doi : 10.1007/s10723-018-9437-3.

[79] Mennan Selimi et al., « Integration of an Assisted P2P Live Streaming Service in Community Network Clouds », in : *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, nov. 2015, p. 202-209, doi : 10.1109/CloudCom.2015.31.

[80] Mennan Selimi et al., « Practical Service Placement Approach for Microservices Architecture », in : *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, mai 2017, p. 401-410, doi : 10.1109/CCGRID.2017.28.

[81] Mennan Selimi et al., « Towards Network-Aware Service Placement in Community Network Micro-Clouds », in : *Euro-Par 2016 : Parallel Processing*, Springer, Cham, août 2016, p. 376-388, doi : 10.1007/978-3-319-43659-3_28.

[82] Vishal Sharma, Dushantha Nalin K. Jayakody et Marwa Qaraqe, « Osmotic Computing-Based Service Migration and Resource Scheduling in Mobile Augmented Reality Networks (MARN) », in : *Future Generation Computer Systems* 102 (jan. 2020), p. 723-737, issn : 0167-739X, doi : 10.1016/j.future.2019.09.008.

[83] Jessica Silver-Greenberg, « After a Data Breach, Visa Removes a Service Provider », in : *The New York Times* (avr. 2012), issn : 0362-4331.

[84] Sukhpal Singh et Inderveer Chana, « QoS-Aware Autonomic Resource Management in Cloud Computing : A Systematic Review », in : *ACM Comput. Surv.* 48 (déc. 2015), issn : 0360-0300, doi : 10.1145/2843889.

[85] Olena Skarlat et al., « A Framework for Optimization, Service Placement, and Runtime Operation in the Fog », in : *2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)*, déc. 2018, p. 164-173, doi : 10.1109/UCC.2018.00025.

[86] Edward Snowden, *Mémoires Vives*, Média Diffusion, sept. 2019, isbn : 978-2-02-144105-5.

[87] Moritz Steiner et al., « Network-Aware Service Placement in a Distributed Cloud Environment », in : *SIGCOMM Comput. Commun. Rev.* 42.*4* (août 2012), p. 73-74, issn : 0146-4833, doi : 10.1145/2377677.2377687.

[88] Bruno Stevant, Jean-Louis Pazat et Alberto Blanc, « Optimizing the Performance of a Microservice-Based Application Deployed on User-Provided Devices », in : *2018 17th International Symposium on Parallel and Distributed Computing (ISPDC)*, juin 2018, p. 133-140, doi : 10.1109/ISPDC2018.2018.00027.

[89] Bruno Stévant, Jean-Louis Pazat et Alberto Blanc, « QoS-Aware Autonomic Adaptation of Microservices Placement on Edge Devices », in : *Proceedings of the 10th International Conference on Cloud Computing and Services Science*, Prague, Czech Republic : SCITEPRESS - Science and Technology Publications, 2020, p. 237-244, isbn : 978-989-758-424-4, doi : 10.5220/0009319902370244.

[90] Bruno Stévant, Jean-Louis Pazat et Alberto Blanc, « Vers une plate-forme communautaire d'hébergement à base de micro-services collaboratifs », in : *COMPAS 2017 - Conférence d'informatique en Parallélisme, Architecture et Système*, juin 2017.

[91] Tarik Taleb et Adlen Ksentini, « Follow Me Cloud : Interworking Federated Clouds and Distributed Mobile Networks », in : *IEEE Network* 27.*5* (sept. 2013), p. 12-19, issn : 1558-156X, doi : 10.1109/MNET.2013.6616110.

[92] Mohit Taneja et Alan Davy, « Resource Aware Placement of IoT Application Modules in Fog-Cloud Computing Paradigm », in : *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, mai 2017, p. 1222-1228, doi : 10.23919/INM.2017.7987464.

[93] Zoha Usmani et Shailendra Singh, « A Survey of Virtual Machine Placement Techniques in a Cloud Data Center », in : *Procedia Computer Science*, 1st International Conference on Information Security & Privacy 2015 78 (jan. 2016), p. 491-498, issn : 1877-0509, doi : 10.1016/j.procs.2016.02.093.

[94] Luis M. Vaquero et Luis Rodero-Merino, « Finding Your Way in the Fog : Towards a Comprehensive Definition of Fog Computing », in : *ACM SIGCOMM Computer Communication Review* 44.*5* (2014), p. 27-32.

[95] Massimo Villari et al., « Osmotic Computing : A New Paradigm for Edge/Cloud Integration », in : *IEEE Cloud Computing* 3.*6* (nov. 2016), p. 76-83, issn : 2325-6095, doi : 10.1109/MCC.2016.124.

[96] Ashkan Yousefpour et al., « FOGPLAN : A Lightweight QoS-Aware Dynamic Fog Service Provisioning Framework », in : *IEEE Internet of Things Journal* 6.*3* (juin 2019), p. 5080-5096, issn : 2372-2541, doi : 10.1109/JIOT.2019.2896311.

[97] Zhi-Hui Zhan et al., « Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches », in : *ACM Computing Surveys* 47.*4* (juill. 2015), 63:1-63:33, ISSN : 0360-0300, DOI : 10.1145/2788397.

[98] Wuyang Zhang et al., « SEGUE : Quality of Service Aware Edge Cloud Service Migration », in : *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, déc. 2016, p. 344-351, DOI : 10.1109/CloudCom.2016.0061.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

INSA RENNES

Titre : Vers une infrastructure participative d'hébergement de services : Orchestration dynamique de micro-services selon les conditions d'utilisation.

Mot clés : Infrastructure participative, Calcul en bordure, Micro-services, Adaptation, Qualité de service

Résumé : Une infrastructure participative vise à fournir une solution d'hébergement de services destinés à une communauté virtuelle d'utilisateurs. Une telle communauté a des besoins qui ne sont pas entièrement satisfaits par les solutions d'hébergement dans les centres de données. Nous proposons donc une nouvelle approche où les membres de la communauté hébergent ces services sur leurs équipements domestiques. Une telle infrastructure participative soulève des problèmes liés à l'hétérogénéité des équipements et des réseaux participants, ainsi qu'aux variations de leurs capacités de calcul et de communication.

À travers l'étude des solutions existantes, l'architecture logicielle basée sur les micro-services a suscité notre intérêt. Elle permet un placement plus flexible des applications sur les équipements participants. La première partie de notre travail a donc consisté à trouver un placement des micro-services sur ces équipements qui optimise le temps de réponse des applications. Après avoir défini un modèle de ce temps de réponse, nous avons utilisé l'heuristique PSO pour trouver une solution proche de l'optimale.

Nous avons testé ce placement en conditions réelles d'utilisation pour évaluer l'influence des variations de la QoS du réseau sur le temps de réponse de l'application. Cette étude a montré que, sous certaines conditions, une adaptation du placement pouvait améliorer les performances mesurées. Nous avons donc ajouté à notre infrastructure participative un mécanisme capable de décider si une adaptation est nécessaire en fonction des temps de réponse mesurés, et de calculer un placement adapté aux nouvelles conditions.

Title: Towards a participatory infrastructure for hosting services: QoS-aware dynamic orchestration of microservices.

Keywords: Participatory Infrastructure, Edge Computing, Microservices, Adaptative orchestration, Quality of Service

Abstract: A participatory infrastructure attempts to provide a solution for hosting services intended for a virtual community of users. Such a community has requirements that are not fully met by hosting solutions in data centers. We therefore propose a new approach where the members of the community host these services on home equipments. Such participatory infrastructure raises issues related to the heterogeneity of the participating devices and networks, as well as to the variations of their computing and communication capabilities.

Through the study of existing solutions, microservices-based application architecture raise our interest. It allows for a more flexible placement of applications on the participating devices. The first part of our work was therefore to find a placement of microservices on these devices that optimizes the application response time. After defining a model of this response time, we used the PSO heuristic to find a solution close to the optimal one.

We tested this placement in real use conditions to evaluate the influence of network QoS variations on the application response time. This study showed that, under certain conditions, an adaptation of the placement might improve the measured performance. We therefore added to our participatory infrastructure a mechanism able to decide, based on the measured response times, if an adaptation is necessary and to compute a placement adapted to the new conditions.