# Nouvelle approche TURTLE pour le dimensionnement et la validation de systemes répartis temps réel

L.Apvrille
TELECOM ParisTech,
LabSoC, CNRS LTCI
2229 rte des Crêtes,B.P.193
06904 Sophia-Antipolis
Cedex, France
ludovic.apvrille@telecom-paristech.fr

A. Mifdaoui
Université de Toulouse
ISAE
1 place Emile Blouin
31056 Toulouse
France
Ahlem.Mifdaoui@isae.fr

P. de Saqui-Sannes
CNRS ; LAAS
7 avenue du colonel Roche
F-31077 Toulouse, France
Université de Toulouse
Université de Toulouse ; UPS,
INSA, INP, ISAE ; LAAS
F-31077 Toulouse
France
pdss@isae.fr

## ABSTRACT

Le profil UML temps réel TURTLE supporté par l'outil open-source TTool offre un cadre formel pour la modélisation et la vérification formelle de systèmes temps réel communicants. Cet article ajoute à la méthode TURTLE un volet "calcul réseau" adapté au traitement des systèmes temps réel répartis à large échelle. Ce volet permet de dimensionner le réseau en prenant en compte les trafics des différents nœuds, puis d'injecter les résultats de dimensionnement dans les modélisations TURTLE. Cette approche permet au niveau des modèles TURTLE de n'explorer le système que pour un nombre réduit de nœuds du système réparti. Un système de vidéo-conférence au sein d'un campus universitaire sert d'étude de cas.

## Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Distributed systems

## General Terms

Performance, Design, Languages, Verification.

## Keywords

Calcul réseau, UML, vérification formelle, systèmes répartis.

## 1. INTRODUCTION

Le concept de "profil UML" permet de personnaliser la notation UML [10] pour répondre aux besoins de modélisation d'un domaine d'application. Ainsi, le profil UML temps réel TURTLE [7] a été appliqué aux systèmes répartis et une méthode dédiée à ces systèmes lui a été associée en [6]. Un grief opposable à cette méthode réside dans le fait que le problème de la distribution est posé en aval de la conception objet. En termes UML, cela revient à dire que le diagramme de classes caractéristique de l'architecture du système et les diagrammes d'activités décrivant les comportements des nœuds sont entièrement décrits avant que l'on ne répartisse les classes sur les nœuds d'un diagramme de déploiement.

Cet article propose de prendre le contre-pied à l'approche présentée en [6] en plaçant la construction d'un diagramme de déploiement en préalable à celle des diagrammes de classes et de comportement. Prenant très tôt en compte le facteur "distribution", un diagramme de déploiement au pouvoir d'expression enrichi (par rapport à (norme UML) et [6]) permet de représenter des configurations regroupant des centaines ou milliers de nœuds.

Que faire de ce nouveau "dessin" que constitue un diagramme de déploiement étendu? Condamne-t-il toute velléité de vérification formelle à base d'analyse d'accessibilité? A la première question nous répondons que le diagramme de déploiement est enrichi de suffisamment d'informations pour autoriser la mise en œuvre d'une technique d'analyse à notre connaissance jamais appliquée à UML ou SysML : le calcul réseau ou *network calculus* [15] en anglais. Cette approche utilisée avec succès pour le dimensionnement de bus avioniques militaires [16] permet par exemple de calculer des majorants qui seront ensuite utilisés pour paramétrer le modèle abstrait du réseau de communication utilisé dans les autres diagrammes de conception du même modèle TURTLE. L'étude de cas traitée dans l'article, à savoir un système de vidéo-conférence réparti sur plusieurs immeubles d'un campus, en fournit un exemple.

Cette analyse par calcul réseau ne relègue pas la vérification formelle aux oubliettes. En fait, après avoir dimensionné le système réparti et donné la priorité au caractère "distribué" de ce système, le travail de conception se poursuit en modélisant les composants logiciels répartis sur les nœuds de ce système et en leur appliquant l'approche de vérification formelle antérieurement développée pour les diagrammes de classes/objets et de comportement TURTLE [6]) [7].

L'article est structuré de la manière suivante. La section 2 donne le point de départ de l'étude, à savoir le profil UML temps réel TURTLE et le calcul réseau. La section 3

étend la méthode associée à TURTLE et introduit des diagrammes de déploiement dans TURTLE. La section 4 présente en détail la syntaxe et la sémantique de ces nouveaux diagrammes. La section 5 applique l'approche proposée à un système de vidéo-conférence sur un campus universitaire. La section 6 positionne cet article par rapport aux travaux du domaine. Enfin, la section 7 conclut l'article.

# 2. L'EXISTANT

## 2.1 TURTLE

TURTLE (Timed UML and RT-LOTOS Environment [7]) est un langage graphique pour le recueil des exigences [9], l'analyse fonctionnelle à base de cas d'utilisation et scénarios [5], la conception objet en termes d'architecture et comportements [7] et finalement, le prototypage de systèmes temps réel et distribués [6].

TURTLE puise ses racines dans les normes UML [10] et SysML [11] d'une part et l'algèbre de processus temporisée RT-LOTOS d'autre part [12] [8]. Supporté par l'outil *open-source* TTool [13], TURTLE est associé à une méthode de développement de systèmes temps réel et distribués qui met l'accent sur la vérification formelle d'exigences temporelles [9].

## 2.2 Calcul réseau

Le dimensionnement pire cas du système est basé sur l'utilisation du formalisme calcul réseau (*Network Calculs*), introduit par Cruz [17] et amélioré par la suite par Le Boudec [15]. Cette théorie est très adaptée dans le cas de sources de trafic contrôlées et d'éléments réseaux connus, pour obtenir des bornes maximales déterministes sur les différentes grandeurs réseau comme le délai ou encore la taille des files d'attente. Cette nouvelle théorie diffère de la théorie classique des files d'attente dans la modélisation du trafic à l'entrée du réseau. En effet, la théorie des files d'attente se base sur les processus stochastiques tandis que le calcul réseau nécessite simplement de connaître "l'irrégularité du trafic".

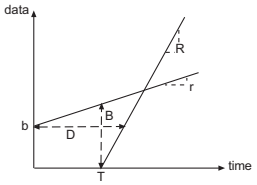


**FIG. 1 –** ***Calcul des bornes maximales sur le délai et la taille de file d'attente***

Les bornes maximales sur le délai et la taille des files d'attente dépendent de l'arrivée du trafic décrite par ce qu'on appelle *courbe d'arrivée* $\alpha$ et la disponibilité du nœud traversé décrite par une *courbe de service* $\beta$. Comme le montre la Figure 1, le délai $D$ est borné par la distance horizontale entre $\alpha$ et $\beta$, tandis que la taille de la file d'attente $B$ au sein du nœud traversé est bornée par la distance verticale. Le calcul de ces bornes est simplifié dans le cas de courbes affines et les courbes les plus utilisées dans ce cas sont:

- la courbe d'arrivée $\alpha(t) = b + rt$ où $b$ est la taille maximale d'une rafale et $r$ le débit maximal du trafic (on dit que le trafic est ($b$,$r$)-contraint). Cette enveloppe est obtenue grâce à l'utilisation d'un dispositif de contrôle de la forme "seau percé" (*Leaky Bucket*);
- la courbe de service $\beta(t) = \max(0, R(t-T))$ où $T$ est le délai d'attente imposé par le nœud avant le début du traitement du trafic et $R$ la capacité minimale de traitement.

Les bornes obtenues dans ce cas sont simplement $\frac{b}{R} + T$ pour le délai et $b + r * T$ pour la taille de la file d'attente. Enfin, il existe un théorème important qui décrit l'évolution de la contrainte d'irrégularité du trafic à la traversée d'un nœud: supposons un flux admettant $\alpha_{in}$ comme courbe d'arrivée à l'entrée d'un nœud et subissant un délai maximal de traitement $D$, alors le flux à la sortie admet une courbe d'arrivée $\alpha_{out}$ où $\alpha_{out}(t) = \alpha_{in}(t + D)$.

# 3. NOUVELLE MÉTHODE TURTLE

La notation UML normalisée par l'OMG est une notation et en aucun cas une méthode. Il appartient aux développeurs de profils UML de proposer une méthode supportée par des outils eux-mêmes en charge d'implanter la sémantique formelle du profil. Ce travail a été réalisé pour le profil UML temps réel TURTLE et la méthode que [6] dédie aux systèmes répartis propose ainsi un séquencement d'étapes désormais classique: recueil des exigences, analyse à base de cas d'utilisation avec première vérification formelle sur des scénarios pertinents, conception objet avec définition d'architecture et vérification formelle de comportements et, finalement, distribution de composants sur plusieurs nœuds.

Cette méthode prône une construction incrémentale des diagrammes de conception et s'appuie sur une architecture à trois niveaux d'usage fort répandue en modélisation de protocoles et algorithmes répartis. Les entités de protocole à développer s'appuient sur un réseau sous-jacent pour offrir à leur tour un service aux utilisateurs qui forment la couche supérieure (cf. Figure 2). Alors que les entités de protocole doivent être détaillées, les applications utilisatrices et le réseau sous-jacent sont modélisés à un haut niveau d'abstraction et de manière à préjuger le moins possible des implantations réelles des applications utilisatrices et du réseau sous-jacent. Concernant ce dernier, le problème se pose de caractériser son délai de transmission. Bien souvent, le concepteur ou la conceptrice du modèle prend des valeurs trouvées dans la littérature ou estimée de manière ad-hoc.

La méthode proposée par la Figure 3 formalise le choix de la valeur du délai de transmission dans le médium. En effet, après avoir recueilli les exigences et terminé la phase d'analyse, on démarre la conception par la construction d'un diagramme de déploiement - appelé, en TURTLE, diagramme de dimensionnement - qui contient suffisamment d'informations (voir sections suivantes) pour autoriser une analyse de type "calcul réseau" et aboutir à l'obtention d'un majorant à utiliser dans la suite de la conception et très précisément dans un objet que nous nommerons ici *Service sous-jacent*.

# 4. MODÉLISATION DU DIMENSIONNEMENT

Comme expliqué à la section précédente, notre approche repose sur le fait de calculer, dans le pire cas, les latences induites par le réseau. Ce calcul nécessite de connaître les caractéristiques du système distribué, à savoir les trafics de données qui sont émis dans le réseau, la topologie du réseau et les informations de routage. Plus précisément, notre étude

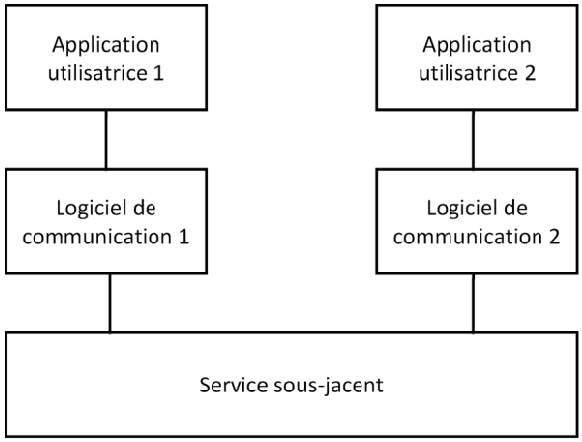


FIG. 2 – *Pattern d'architecture en couches*

de dimensionnement implique les entités suivantes :

- les équipements connectés au réseau (PCs, PDAs, etc.).
- Les différents trafics émis sur les équipements connectés au réseau.
- Les routeurs du réseau (commutateur Ethernet, borne WIFI, routeur IP, etc.).
- Les interconnexions physiques entre les équipements et les routeurs d'une part et entre les routeurs d'autre part.
- Des cas d'utilisation orientés "communication niveau réseau sous-jacent" qui identifient en particulier les destinations possibles des différents trafics.

Afin de modéliser en UML2 les éléments systèmes décrits ci-dessus, nous proposons l'utilisation de deux diagrammes.

- Pour la modélisation de la topologie du réseau, nous nous appuyons sur un diagramme de déploiement UML2. Nous rebaptisons à l'occasion ce diagramme "**diagramme de dimensionnement**".
- Pour la modélisation des communications possibles sur le système réseau distribué, nous utilisons un diagramme de cas d'utilisation UML.

## 4.1 Diagramme de dimensionnement

Un diagramme de déploiement UML [10] est constitué de nœuds d'exécution (*nodes*) qui peuvent être stéréotypés (par exemple, << *PC* >>), de liens entre les nœuds d'exécution, et enfin d'artéfacts (*artifacts*) qui sont, selon la norme UML, des éléments logiciels issus d'un processus de développement.

Nous reprenons bien entendu ces trois concepts (nœuds, liens, artéfacts) pour les appliquer à des systèmes distribués, et donc pour modéliser les équipements, les trafics, les routeurs (comprenant leur table de routage) et les interconnexions entre équipements et routeurs (cf. Figure 4).

- Les équipements connectés au réseau sont modélisés sous la forme d'un nœud d'exécution stéréotypé << *Equipment* >>.
- Les trafics sont modélisés comme des artéfacts qui peuvent être ajoutés à des << *Equipment* >>. Un trafic est caractérisé par son type (périodique, apériodique), sa limite de temps (*deadline*), sa taille de paquets minimale et maximale et, enfin, la priorité au niveau des routeurs (4 niveaux de priorité sont supportés). La Figure 5 présente la saisie sous TTool des informations pour le trafic *T3* de l'étude de cas détaillée en section 5.
- Les routeurs et commutateurs sont modélisés comme des nœuds d'exécution stéréotypés par << *Switch* >> (Par exemple, sur la Figure 4, le switch *switch0*). Les commutateurs sont caractérisés par leur politique d'ordonnancement (*static priority*, *First Come First Served*) et leur capacité exprimée en Mbs ou Gbs. Notons que cette capacité n'est prise en compte que si la capacité des interfaces des routeurs n'est pas définie.
- Les informations de routage sont modélisées au niveau des commutateurs par des artéfacts mappés sur ces commutateurs (par exemple, l'artéfact *R0* du nœud *switch1*). Un artéfact de routage peut comprendre une ou plusieurs routes. Une route se définit comme un triplet *(interface d'entrée, trafic, interface de sortie)*.
- Enfin, des liens entre nœuds permettent de modéliser les interfaces d'entrées et de sorties des équipements et des commutateurs, et les connexions entre ces interfaces. Un lien est caractérisé optionnellement par sa capacité (par défaut, la capacité d'un lien est celle des interfaces auxquelles ils est connecté) et par une multiplicité (par défaut, 1). La multiplicité permet de modéliser, sur un lien entre un équipement et un commutateur, un grand nombre d'équipements par un seul nœud << *Equipment* >> et un seul lien.

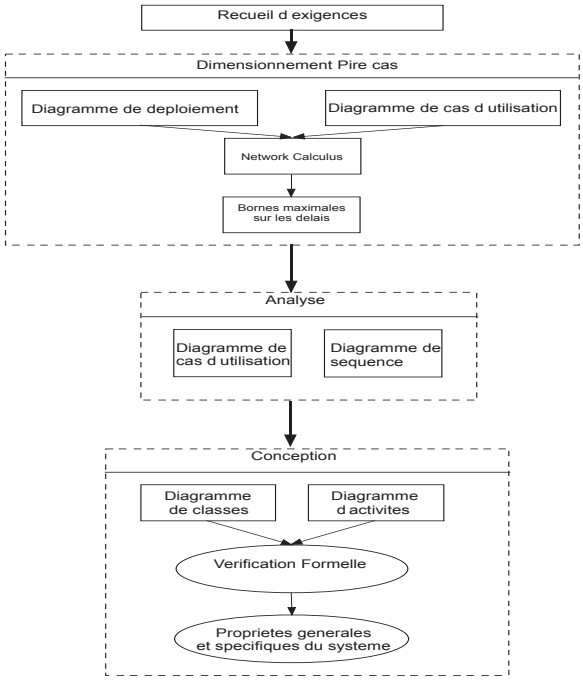


FIG. 3 – *Nouvelle méthode TURTLE*

Par exemple, la Figure 4 représente quatre équipements et trois commutateurs. Sur l'équipement *eq01* sont émis deux trafics *T1* et *T3* et sur *eq02* est émis le trafic *T2*. Le trafic *T1* traverse tout d'abord le commutateur *switch1*. Les informations de routage de la route *R0* mappée sur *switch1* indiquent que *T1* peut être routé vers l'interface *i1* et vers l'interface *i4* (cf. Figure 6). Aussi, ce trafic, en suivant les

chemins réseaux possibles, peut être routé vers les équipements *eq02* et *eq03*. Enfin, le trafic *T1* vers *eq02* peut suivre deux chemins possibles, en passant ou non par *switch2*.

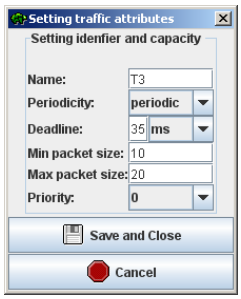


**Fig. 5 –** ***Caractéristiques d'un trafic***

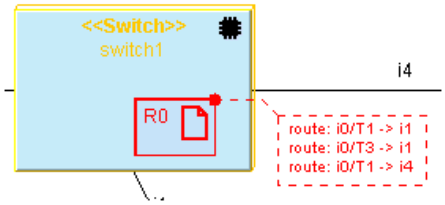


**Fig. 6 –** ***Information de routage d'un Artéfact Route***

A ces éléments de modélisation (équipements, trafics, commutateurs, routes, liens/interfaces), nous avons ajouté les propriétés suivantes, nécessaires aux calculs de dimensionnement sur un réseau :

- Il ne doit pas y avoir de cycle possible dans le routage du trafic.
- La multiplicité d'un lien ne peut concerner qu'un lien reliant un équipement à un commutateur. Cette multiplicité ne s'applique qu'à l'équipement.
- Pour tout trafic émis dans le réseau, il doit exister au moins un chemin correctement constitué lui permettant d'être acheminé vers au moins un équipement. Par correctement constitué, nous entendons qu'au niveau d'un commutateur, tout trafic entrant doit posséder au moins une interface de sortie différente de son interface d'entrée, et que les informations de routage des différents commutateurs doivent permettre à tout trafic d'atteindre au moins un équipement.

## 4.2 Cas d'utilisation

Les cas d'utilisation ont pour objectif de préciser les différentes utilisations possibles du réseau décrit au niveau du diagramme de dimensionnement. Pour décrire ces utilisations, nous proposons d'utiliser les diagrammes de cas d'utilisation d'UML. Ces diagrammes de cas d'utilisation sont constitués d'un système comprenant des fonctions - aussi appelée *cas d'utilisation* - et des acteurs représentant l'environnement du système. Lors d'un dimensionnement, le système est bien entendu le réseau lui-même; les acteurs représentent alors les applications émettrice du trafic et les cas d'utilisation représentent la fonction de routage dans le réseau.

A titre d'exemple, considérons, en avance de phase, le système de vidéo-conférence décrit dans la section 5.1. En quelques mots, ce système est utilisé par des étudiants situés aux différents étages de deux bâtiments. Par rapport à la topologie du réseau de ces bâtiments, nous pouvons distinguer trois cas d'utilisation du réseau :

1. Une vidéo-conférence entre deux étudiants du même étage (*com1*);
2. Une vidéo-conférence entre deux étudiants du même bâtiment (*com2*);
3. Une vidéo-conférence entre deux étudiants situés dans deux bâtiments différents (*com3*).

Le diagramme de cas d'utilisation de la Figure 7) décrit trois cas d'utilisation. Deux acteurs modélisent les deux applications de vidéo-conférence. Les cas d'utilisation représentent le routage des informations selon que les applications sont situées sur le même étage, dans le même bâtiment, et dans deux bâtiments différents.

## 4.3 Outillage et méthode associée

Reprenant la méthode présentée à la section 3, et les diagrammes supports présentés juste avant dans cette même section, nous avons mis en place un outillage basé sur l'outil TTool [13] et un outil externe de calcul réseau. En quelques mots, TTool est un outil *open-source* supportant des profils UML2, et notamment le profil TURTLE. Tout profil supporté par TTool comporte une sémantique formellement définie et pour laquelle un générateur de code permet à un modeleur de réaliser des vérifications formelles sans aucune connaissance spécifique sur les techniques de vérification sous-jacente : un simple clic permet d'obtenir des résultats de vérification formelle présentés de façon conviviale. Par exemple l'atteignabilité d'une action ou la caractérisation du service rendu par une couche de protocole peut-être obtenu simplement en utilisant des techniques de minimisation de graphes d'accessibilité étiquetés ; pour cela, des outils externes [2] [1] [3] de façon transparents depuis TTool.

Reprenant cette approche de vérification, nous avons implantés dans TTool les diagrammes de dimensionnement (principale contribution de cet article) et les digrammes de cas d'utilisation (voir Figure 8). A partir de ces diagrammes, TTool peut générer une spécification *xml* qui sert d'entrée à un outil de calcul réseau développé à l'ISAE (le lien entre les deux outils n'est pas encore achevé). Cette spécification *xml* n'est bien entendu générée que si le réseau modélisé avec le diagramme de dimensionnement suit les propriétés énoncées précédemment (absence de cycle, informations de routage correctes, etc.). Si la spécification n'a pu être générée, un rapport d'erreurs est fourni à l'utilisateur. Dans le cas contraire, l'outil de calcul réseau permet, pour chaque cas d'utilisation, de calculer une latence maximale subie par les trafics. Ces latences peuvent ensuite être réutilisées, selon le souhait du concepteur, au niveau des diagrammes de conception afin de spécifier correctement les délais au niveau des médiums de communication ou plus généralement du service sous-jacent au logiciel de communication à développer. Une fois les diagrammes de conception TURTLE mis à jour (i.e. diagrammes de classes et diagrammes d'activités), le système peut, comme d'habitude sous TTool, être vérifié formellement par utilisation des outils UPPAAL [3], RTL [2] et CADP [1]. Encore une fois, l'ensemble du processus de calcul de dimensionnement et de vérification formelle

est caché à l'utilisateur ; seuls les résultats de ces calcul sont présentés.

# 5. APPLICATION À UN SYSTÈME RÉPARTI DE VIDÉO-CONFÉRENCE

## 5.1 Cas d'étude

Une résidence universitaire dispose d'un réseau facilitant la communication entre les étudiants équipés d'ordinateurs. La résidence est divisée en 2 immeubles de 4 étages. L'architecture du réseau est la suivante : 2 commutateurs Ethernet full-duplex sont installés à chaque étage pour connecter les 28 chambres de l'étage et les deux commutateurs sont reliés entre eux (chaque commutateur dispose de 16 ports à 100 Mbps). Les 8 commutateurs d'immeuble sont eux mêmes reliés à un commutateur de 8 ports à 100 Mbps et d'un port à 1 Gbps (ce dernier port sert à relier les deux immeubles entre eux) comme le montre la Figure 9.

Grâce à sa webcam, chaque étudiant peut communiquer avec un autre étudiant en vidéo-conférence. Dans une telle application, on considère que deux flux audio-vidéo (un dans chaque sens) traversent le réseau, chacun d'eux ayant un débit de 120 kbps (des trames de 1500 octets sont envoyées de manière périodique).

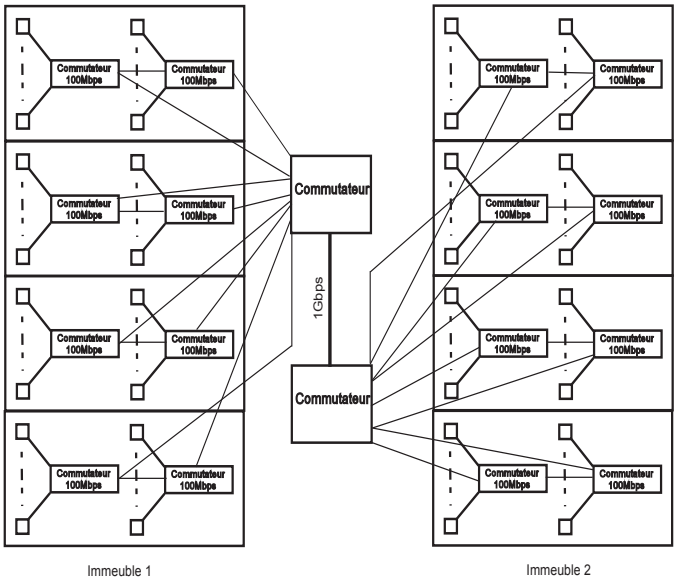


FIG. 9 – *Cas d'étude*

## 5.2 Diagramme de dimensionnement

Nous avons modélisé le système décrit ci-dessus en utilisant le diagramme de dimensionnement TURTLE introduit dans cet article (cf. Figure 10). Ce diagramme met en évidence :

- Les 8 commutateurs d'étages situés dans chacun des deux bâtiments.
- Un commutateur propre à chaque bâtiment (*switch_building0*, *switch_building1*). Ces deux commutateurs sont reliés entre eux par un lien gigabit.
- 14 équipements par commutateur d'étage. Pour modéliser ces 14 équipements, nous avons utilisé un lien avec multiplicité entre les équipements d'étage et les commutateurs d'étage. Ainsi, sur le diagramme de dimensionnement, il n'apparaît qu'un seul équipement par commutateur d'étage.
- Des liens pour connecter équipements et commutateurs.
- Des tables de routage au niveau des commutateurs. Ces tables de routage spécifient que chaque équipement peut communiquer avec n'importe quel autre équipement du réseau.

Depuis ce diagramme de dimensionnement, TTool vérifie les bonnes propriétés du réseau (par exemple, l'absence de cycles dans le routage), calcule et présente une vue synthétique du réseau (cf. Figure 11) sous la forme de l'ensemble des chemins possibles (*paths*), puis génère une spécification *xml* qui sert de point d'entrée pour la calcul réseau présenté à la sous-section suivante.

Notons que malgré la complexité du réseau (224 équipements, 224 trafics émis, 18 commutateurs, environs 50000 paths), la représentation UML reste compréhensible et gérable d'un point de vue graphique.

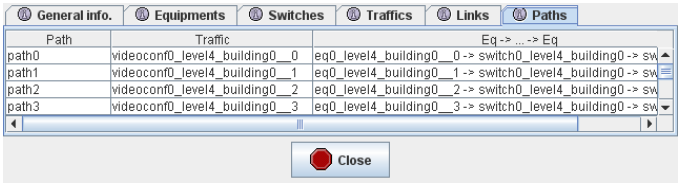


FIG. 11 – *Synthèse du réseau présentée par TTool*

## 5.3 Bornes obtenues par calcul réseau

Les flux audio-vidéo envoyés sur le réseau sont des flux périodiques avec une longueur de paquet maximale de $L = 12000$ bits et un débit maximal de $r = 120$ Kbps. Ainsi, tout flux audio-vidéo $i$ admet une courbe d'arrivée affine $\alpha_i(t)$ telle que $\alpha_i(t) = L + r * t$.

D'autre part, les commutateurs utilisés sont des commutateurs du commerce basés sur une politique de service simple de type *First Come First Served* (FCFS) et admettant une latence technologique négligeable. Ainsi, la courbe de service $\beta(t)$ offerte par un commutateur de capacité $C$ (dans notre cas $C \in \{100Mbps, 1Gbps\}$) est telle que $\beta(t) = C * t$. Par conséquent, la borne maximale sur le délai de traversée $D$ est obtenue en déterminant la distance horizontale maximale entre la courbe de service $\beta(t)$ et la courbe d'arrivée du flux agrégé $\alpha(t) = \sum_i \alpha_i(t)$ à l'entrée du commutateur. D'où, $D = \frac{\sum_i L}{C}$.

Étant donné la taille du réseau et le nombre d'utilisateurs, la description de la méthode de calcul des bornes maximales sur les délais de bout en bout est simplifiée grâce à l'algorithme 1. Tout d'abord, l'ensemble des flux à l'entrée du réseau est identifié (ligne 1). Puis, pour chaque flux identifié, on détermine son chemin parcouru (de la source à la destination finale) composé des différents commutateurs (ligne 4) et leurs courbes de service respectives (ligne 5). Par la suite, le calcul des bornes sur les délais est fait par propagation d'un commutateur à un autre en appliquant le théorème d'évolution des contraintes d'irrégularité du trafic (lignes 6-13). En effet, on identifie l'ensemble des flux à l'entrée de chaque commutateur réseau appartenant au chemin du flux

en question (ligne 7) et leurs courbes d'arrivée respectives (ligne 8). On détermine par la suite la courbe d'arrivée du flux agrégé en sommant les courbes d'arrivées des flux individuels (ligne 9). Puis, connaissant la courbe d'arrivée du flux agrégé et la courbe de service du commutateur traversé, la borne maximale sur le délai de traversée est obtenue en calculant la distance horizontale maximale entre les deux courbes (ligne 10). Ce délai est utilisé pour déterminer la courbe d'arrivée à la sortie du commutateur qui va être elle même la courbe d'arrivée à l'entrée du prochain commutateur (ligne 11). Les bornes sur les délais de traversée des différents commutateurs étants connues, la borne sur le délai de communication de bout en bout est obtenue en faisant la somme de ces dernières (ligne 12).

**algorithme 1** Calcul des bornes des délais de bout en bout

1: $S \leftarrow \{s_1, s_2 ... s_{n_{streams}}\}$
2: $EED \leftarrow$ NULL-VECTOR(S.length)
3: **for** $i = 1$ to $n_{streams}$ **do**
4: Path $\leftarrow$ Vector-crossed-components($s_i$)
5: $\beta \leftarrow$ Vector-service-curves(Path)
6: **for** $k = 1$ to Path.length **do**
7: $R \leftarrow$ Vector-rcv-streams($Path(k)$)
8: $\alpha \leftarrow$ Vector-arrival-curves($R$)
9: $\alpha_{global} \leftarrow \sum_{j \in R} \alpha(j)$
10: $D \leftarrow$ Delay-calculus ($\alpha_{global}$,$\beta(k)$)
11: $\alpha_{global} \leftarrow$ Left-shift-curve ($\alpha_{global}$,$D$)
12: $EED(i) \leftarrow EED(i) + D$
13: **end for**
14: **end for**

Dans notre cas d'étude, nous avons identifié trois types de communication et les bornes maximales sur les délais de bout en bout respectives sont données dans le tableau 1. Ces trois cas d'étude ont été modélisés dans un diagramme de cas d'utilisation (cf. Figure 7) :

- communication entre deux étudiants du même étage (**com1**) où le flux audio-vidéo va parcourir deux commutateurs au pire;
- communication entre deux étudiants du même immeuble (**com2**) où le flux audio-vidéo va parcourir trois commutateurs au pire;
- communication entre deux étudiants de la même résidence (**com3**) où le flux audio va parcourir quatre commutateurs au pire.

**Tab. 1** – ***Bornes maximales sur les délais de bout en bout obtenues en appliquant le calcul réseau***

| Type de communication | Délai de bout en bout (ms) |
|---|---|
| com1 | 1,68 |
| com2 | 2,70 |
| com3 | 28,80 |

## 5.4 Conception et résultats de vérification formelle

La conception du système de vidéo-conférence en TURTLE amène à décrire une architecture d'objets qui respecte le modèle à trois niveaux de la Figure 2. On y trouve les applications de vidéo-conférence et le réseau sous-jacent d'acheminement des trafics issus de ces applications.

En l'absence de diagramme de dimensionnement, nous aurions été amené à modéliser l'ensemble du réseau d'interconnexion afin de prendre en compte la latence de ce réseau lors de la phase de vérification formelle. Grâce aux résultats issus du calcul réseau, une seule classe *Medium*, détaillée par la suite, modélise ce réseau d'interconnexion.

Au final, l'architecture du système est constituée de cinq classes (cf. Figure 13) :

- une classe pour chaque entité émettrice et réceptrice qui modélise le protocole d'établissement de la communication, d'envoi et de réception des données et de gestion des déconnexions (classe *VCManager*).
- Une classe de capture des données audio / vidéo et d'affichage de ces données (classe *VCDataExchange*). Cette classe est instanciée deux fois, une fois par application.
- Enfin, une classe qui modélise le réseau d'interconnexion (*Medium*).

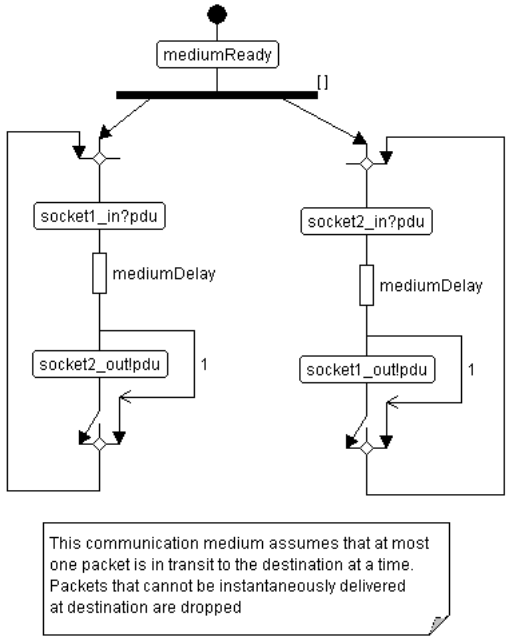


**Fig. 12** – ***Diagramme d'activités de la class Medium***

Par la suite, nous nous intéressons au diagramme d'activités de *Medium*, qui prend en compte les résultats de dimensionnement (cf. Figure 12). Ce diagramme d'activités modélise simplement que tout paquet transitant par le medium subit une latence de 29 ms (résultat de dimensionnement). Le diagramme comporte deux sous-activités en parallèle pour gérer les deux interfaces d'entrée / sortie. Tout paquet entrant sur une interface est réémis après le délai *mediumDelay* (29 ms). Notons que l'offre limitée dans le temps (par exemple, l'action sur la porte *socket1_out*) modélise le fait que l'entité réceptrice doit accepter un paquet dès son émission sur l'interface de sortie, sinon, ce paquet est détruit.

Lors de la vérification formelle, nous avons pu prouver le

"bon fonctionnement" du protocole d'ouverture de connexion, d'échange de données et de fermeture de la connexion. Bien entendu, cette vérification n'est valable que pour la latence spécifiée au niveau du médium.

A titre d'exemple de vérification que nous avons pu mener, nous avons généré un graphe d'accessibilité comprenant environ 3000 états et 10000 transitions. Ce chiffre aurait été sans nul doute très supérieur - et aurait sans doute explosé - sans le résultat du dimensionnement. Notons enfin que le graphe d'accessibilité comporte un ratio de transitions par états assez élevé ; cela tient au fait que l'échange des données peut-être interrompu à tout moment par une des deux entités impliquées dans l'échange de données. Enfin, par utilisation de techniques de minimisation de graphes, nous avons pu prouver certaines propriétés, par exemple le fait que tout échange de données est forcément précédé d'une ouverture de connexion, mais aussi qu'aucun paquet de données n'est perdu et, enfin, que la fermeture de connexion ne peut intervenir qu'après une ouverture de connexion.

## 6. TRAVAUX DU DOMAINE

Cet article allie un langage de modélisation (UML) et son outil de support (TTool) à une analyse à base de calcul réseau. L'idée est à notre connaissance nouvelle dans son application à UML, bien que des articles tels que [19] utilisent ponctuellement un diagramme UML pour documenter l'étude (en l'occurrence des diagrammes de séquences pour [18]). Par contre, l'idée d'associer le calcul réseau à une modélisation a déjà été exploitée. Ainsi, [18] intègre du calcul réseau à Matlab/Simulink à des fins d'analyse d'ordonnançabilité. Contrairement à cet article, [18] ne traite ni de systèmes distribués ni de vérification formelle. Le choix de Simulink centre le travail sur la phase de conception.

En termes de méthode, cet article prend pour référence l'architecture à trois niveaux de la Figure 2 et exploite le calcul réseau pour donner une valeur "réaliste" au délai de communication du service sous-jacent. Cette recherche de majorant relève de l'analyse de pire cas, approche pratiquée en ingénierie des systèmes embarqués et mise en oeuvre par des outils industriels tels que SCADE [14]. A notre connaissance, cette recherche de pire cas en termes de délais de transmission n'a pas encore été associée à un langage de modélisation outillé en vérification formelle et applicable à des systèmes répartis.

## 7. CONCLUSIONS ET PERSPECTIVES

La méthode [6] jusqu'ici associée au profil UML temps réel TURTLE et son application aux systèmes répartis via l'outil TTool, considèrent une architecture de conception en couches dans laquelle les délais de transmission du médium ou plus largement du service sous-jacent sont déterminées de manière empirique voir arbitraire. Cet article propose d'améliorer la méthode associée à TURTLE en donnant au délai de transmission du médium une valeur de type "majorant". Ce dernier s'obtient par application du calcul réseau sur un diagramme de déploiement étendu dont l'article précise la syntaxe compatible avec le méta-modèle UML 2. Un diagramme de cas d'utilisation permet de distinguer entre plusieurs services de communication qui recevront chacun un majorant de leur délai de transmission. Les valeurs ainsi obtenues s'utilisent dans l'architecture objet et les comportements associés qui servent de point d'entrée à la vérification formelle. Un système de vidéo-conférence illustre l'approche proposée en suivant les étapes identifiées sur la Figure 3.

A ce jour, l'éditeur de diagrammes de l'outil TTool permet de construire les diagrammes de déploiements tels que celui de la Figure 10. Un tel diagramme est sauvegardé dans un format fichier *.xml* qui sert de point d'entrée à un programme Java de calcul réseau. Pour l'instant ce programme Java n'est pas appelé depuis TTool.

L'introduction d'un diagramme de déploiement étendu en amont de la conception repose le problème de la cohérence des vues exprimées par les différents diagrammes d'un même modèle TURTLE. Là où l'introduction de règles trop coercitives nuirait à l'applicabilité de TTool à une large classe de systèmes soumis à des contraintes de temps, nous pensons poursuivre sur l'idée exposée en [4] d'un assistant méthodologique dédié aux systèmes répartis.

## 8. REFERENCES


[1] The CADP toolkit. http://www.inrialpes.fr/vasy/cadp.

[2] The RTL toolkit. http://www.laas.fr/RT-LOTOS/index.html.en.

[3] The UPPAAL toolkit. http://www.uppaal.com/.

[4] L. Apvrille and P. de Saqui-Sannes. Adding a Methodological Assistant to a Protocol Modeling Environment. In *Proceedings of the 8th Annual International Conference on New Technologies of Distributed Systems (NOTERE'2008)*, Lyon, France, June 2008.

[5] L. Apvrille, P. de Saqui-Sannes, and F. Khendek. Synthèse d'une conception UML temps-réel à partir de diagrammes de séquences. In *Colloque Francophone sur l'Ingéniérie des Protocoles*, Bordeaux, France, Mars 2005.

[6] L. Apvrille, P. de Saqui-Sannes, R. Pacalet, and A. Apvrille. Un environnement UML pour la conception de systèmes distribués . *Annales des Télécommunications*, 61:11/12:1347–1368, Novembre 2006.

[7] L. Apvrille, C. Lohr, J.-P. Courtiat, and P. de Saqui-Sannes. TURTLE: A Real-Time UML Profile Supported by a Formal Validation Toolkit. In *IEEE transactions on Software Engineering*, volume 30, pages 473–487, July 2004.

[8] J. Courtiat, C. Santos, C. Lohr, and B. Outtaj. Experience with RT-LOTOS, a Temporal Extension of the LOTOS Formal Description Technique. *Computer Communications*, 23:1104–1123, 2000.

[9] B. Fontan. Méthodologie de conception de systèmes temps réel et distribués en contexte UML/SysML. In *Doctorat de l'Université de Toulouse délivré par l'Université Paul Sabatier*, January 2008.

[10] O. M. Group. UML 2.0 Superstructure Specification. In *http://www.omg.org/docs/ptc/03-08-02.pdf*, Geneva, 2003.

[11] O. M. Group. UML Profile for Systems Engineering, SysML, Version 1.0. In *http://www.omg.org/cgi-bin/apps/doc?formal/07-09-01.pdf*, Geneva, Sept. 2007.

[12] ISO-LOTOS. A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. In *Draft International Standard 8807, International Organization for Standardization - Information Processing Systems - Open Systems Interconnection*, Geneva, July 1987.

[13] LabSoc. The TURTLE Toolkit. In *http://labsoc.comelec.enst.fr/turtle/ttool.html*.

[14] T. le Sergent, R. Heckmann, and D. Kastner. Methodology and benefist of Timing Verification for Safety-Critical Embedded Software. In *DO-178 White paper*, http://www.esterel-technologies.com/DO-178B/request/whitepaper, 2008.

[15] J. Leboudec and P. Thiran. *Network Calculus*. Springer Verlag LNCS volume 2050, 2001.

[16] A. Mifdaoui, F. Frances, and C. Fraboul. Full-Duplex Switched Ethernet For Next Generation "1553B"-based Applications. In *The 13th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS07)*, Bellevue, WA, USA, 2007.

[17] R.Cruz. A calculus for network delay, part 1: network elements in isolation. *IEEE transactions on information theory*, 37, January 1991.

[18] H. Schioler, H. P. Schwefel, and M. B. Hansen. Cync: a matlab/simulink toolbox for network calculus. In *ValueTools '07: Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[19] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse. System architecture evaluation using modular performance analysis: a case study. *Int. J. Softw. Tools Technol. Transf.*, 8(6):649–667, 2006.

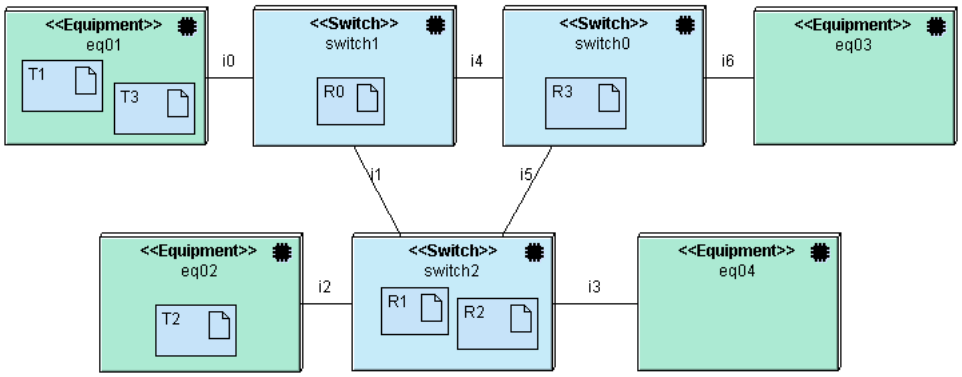


FIG. 4 – *Exemple d'un diagramme de dimensionnement*

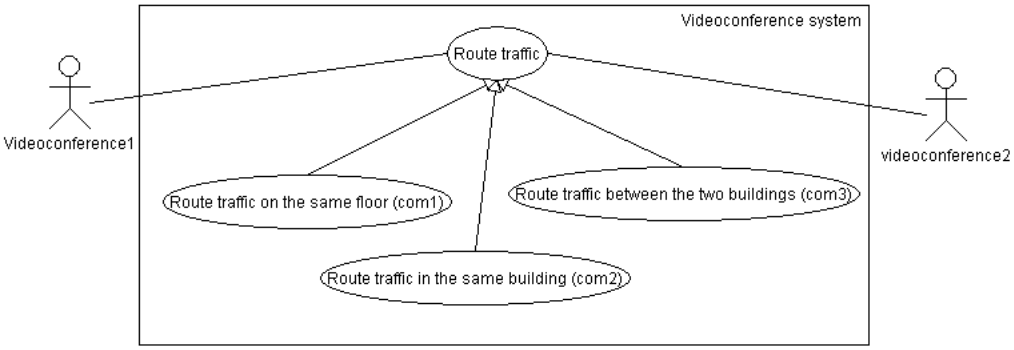


FIG. 7 – *Cas d'utilisations du système de vidéo-conférence*

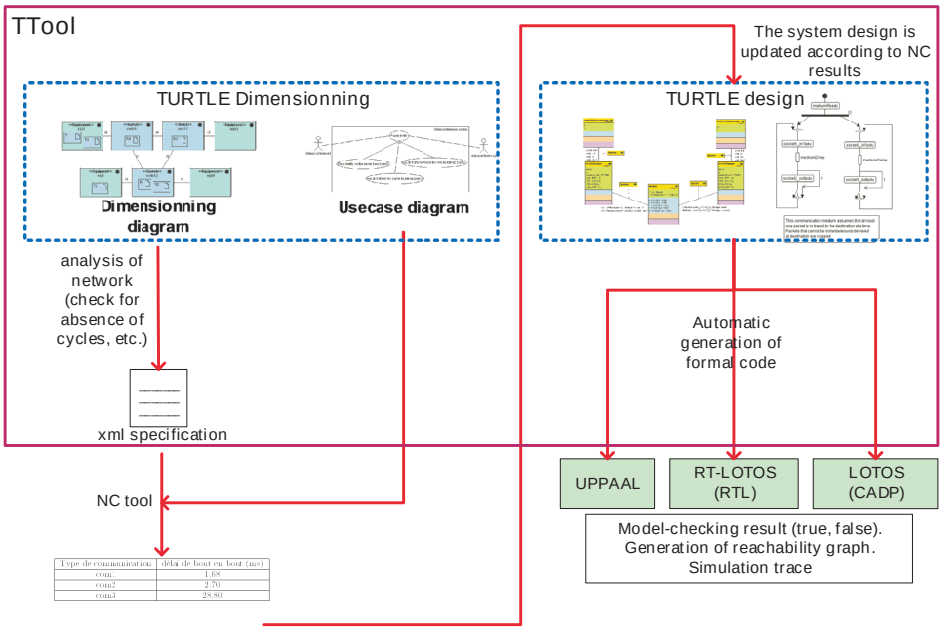


FIG. 8 – *Outillage support au dimensionnement et à la conception*

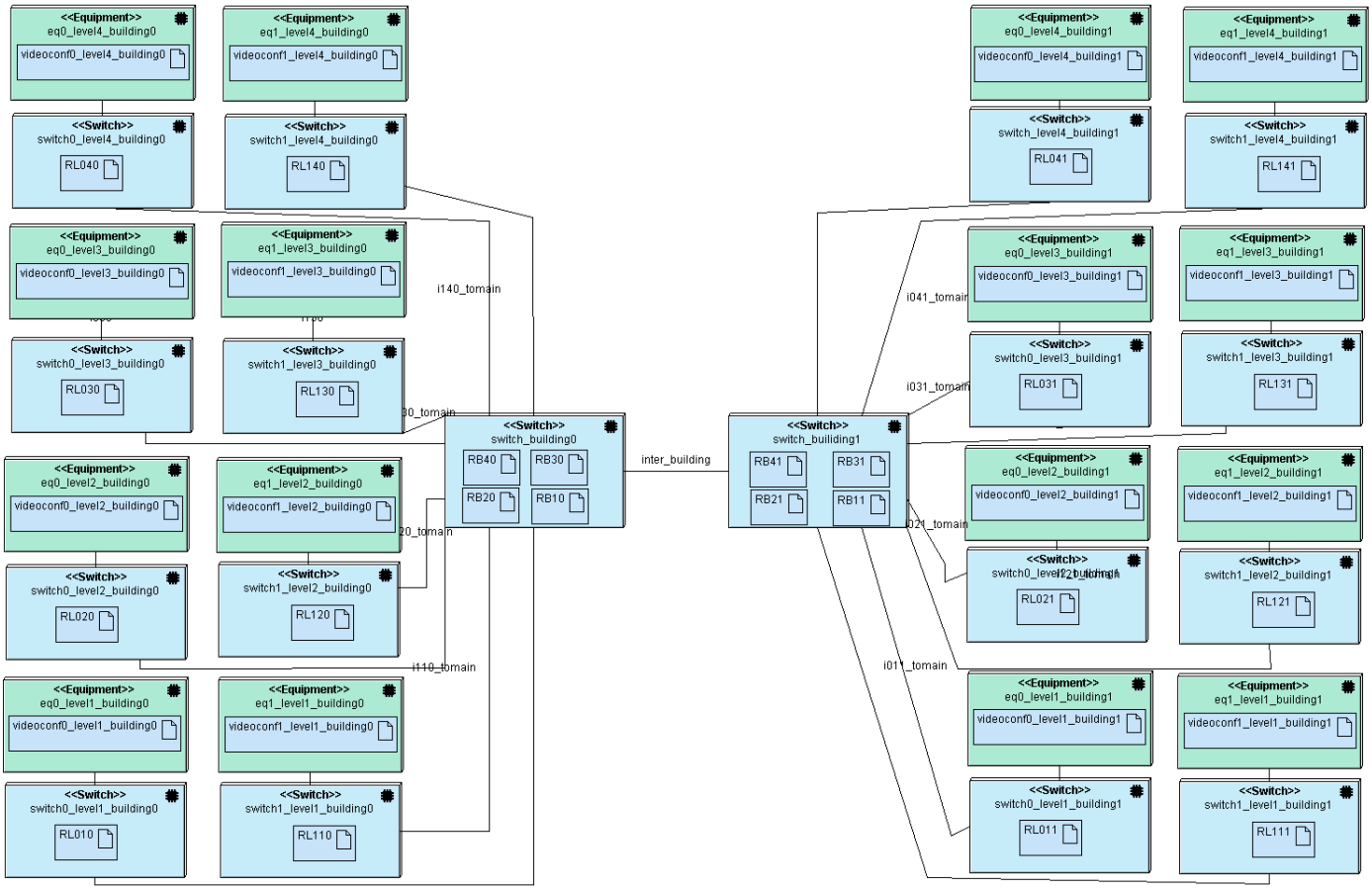


FIG. 10 – *Diagramme de dimensionnement du cas d'étude*

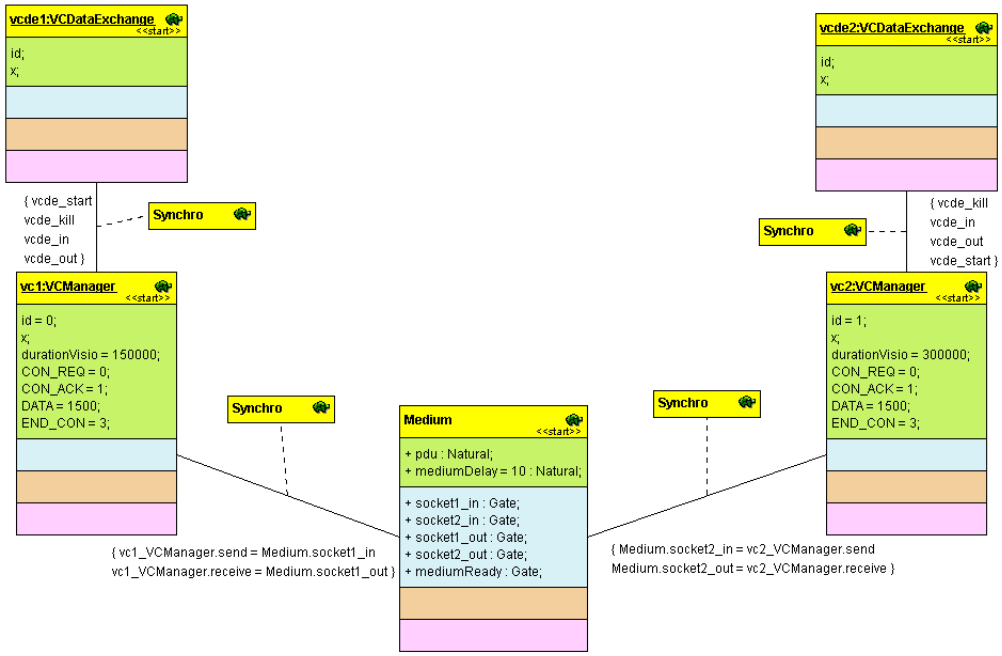


FIG. 13 – *Diagramme de classes du cas d'étude*