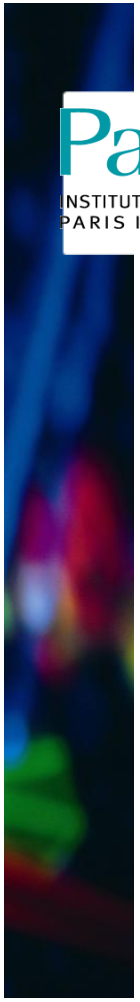
# Video conference based on enterprise desktop grid

Roman Sorokin



ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2017-ENST-006

EDITE ED 130

# Doctorat ParisTech

# T H È S E

**pour obtenir le grade de docteur délivré par**

# Télécom ParisTech

## Spécialité “ Informatique et Réseaux ”

*présentée et soutenue publiquement par*

**Roman SOROKIN**

le 24 février 2017

# Vidéoconférence basée sur les ressources internes de l'entreprise

Directeur de thèse : **Jean-Louis ROUGIER**

**Jury**

| | |
|---|---|
| **M. André-Luc BEYLOT**, Professeur, IRIT / ENSEEIHT | Président du jury |
| **M. Nadjib ACHIR**, Maître de Conférences HDR, L2TI - Institut Galilée, Université Paris 13 | Rapporteur |
| **M. Yacine GHAMRI-DOUDANE**, Professeur, L3i, Université de La Rochelle | Rapporteur |
| **Mme. Annie GRAVEY**, Professeur, Telecom Bretagne | Examinateur |
| **M. Khaled BOUSSETTA**, Maître de Conférences, L2TI - Institut Galilée, Université Paris 13 | Examinateur |
| **M. Nicolas TRANQUART**, Ingénieur de Recherche, ALE International | Invité |

T H È S E

**Télécom ParisTech**
**école de l’Institut Mines Télécom – membre de ParisTech**
46, rue Barrault – 75634 Paris Cedex 13 – Tél. + 33 (0)1 45 81 77 77 – www.telecom-paristech.fr

# Résumé


## Contexte et Motivation

Conférence vidéo est un sujet de télécommunications bien connu, qu'on étudiait pour les décades. Récemment ce sujet a reçu une nouvelle pulsion grâce à la bande passante accrue de réseau local et réseau étendu, et l'apparition de l'équipement vidéo de bon marché. Au même temps le vidéo de bonne qualité, comme "Full HD", peut demander les ressources computationnelles significatives pour son traitement. Le traitement vidéo pour les conférences comprend quelques manipulations nécessaires pour obtenir une expérience utilisateur avancée (mélange de plusieurs flux vidéo ou de passer l'image au participant qui parle actuellement), ainsi que les opérations causées par l'incompatibilité des paramètres, par exemple transcodage dans le cas où les participants utilisent différents codecs vidéo.

Actuellement, deux architectures distinctes pour le traitement de ces tâches de manipulation de vidéo sont utilisés .

La solution traditionnelle utilise Multipoint Control Unit (MCU) [1]. MCU est un composant puissant qui centralise toutes les opérations de traitement vidéo et distribue les flux résultant. MCU peut être mis en œuvre comme une unité matérielle intégrée avec Digital Signal Processors (DSP) ou un composant logiciel installé sur les serveurs type Commercial Off-The-Shelf (COTS). Aussi MCU peut être déployé dans le nuage où au mode local. Dans tous les scénarios de déploiement, MCU représente une ressource dédiée, qui doit être acheté ou loué.

Une autre solution consiste à utiliser des clients vidéo en tant que ressources pour le traitement vidéo. Ceci peut être réalisé en exploitant de la stratégie Peer-to-Peer (P2P) ou de Selective Forwarding Unit (SFU) [2]. L'approche P2P a été soigneusement étudié, mais cette technique n'a pas gagné du terrain dans les communications de l'entreprise, car elle ne fournit pas de moyens faciles d'intégration avec les applications d'entreprise, ainsi que la mise en œuvre des exigences des entreprises importantes comme l'accès aux annuaires LDAP hétérogènes. SFU est un composant logiciel qui transmet les paquets vidéo basés sur les limites des capacités des clients vidéo. Il ne fait pas de traitement des médias sur les flux vidéo, en effet il filtre et relaie les paquets. Par conséquent, la capacité d'un système de vidéoconférence entraîné par SFU dépend des capacités des clients vidéo. Si paramètres sont incompatibles en termes de codecs alors SFU n'est pas utile, en tant que les clients vidéo n'ont normalement pas fonctionnalité transcodage.

Dans cette thèse un système de Desktop Grid Conferencing (DGC) est proposé, qui utilise les ressources de la grille des ordinateurs d'entreprise (PCs, ordinateurs portables, etc. déployés dans le réseau de l'entreprise) pour l'attribution des services de traitement vidéo nécessaires pour l'organisation de vidéoconférences. Les

recherches antérieures sur les grilles des ordinateurs d'entreprise [3] montre qu'une quantité importante de ressources CPU de PC utilisés dans les entreprises ne sont pas occupés à toute activité.

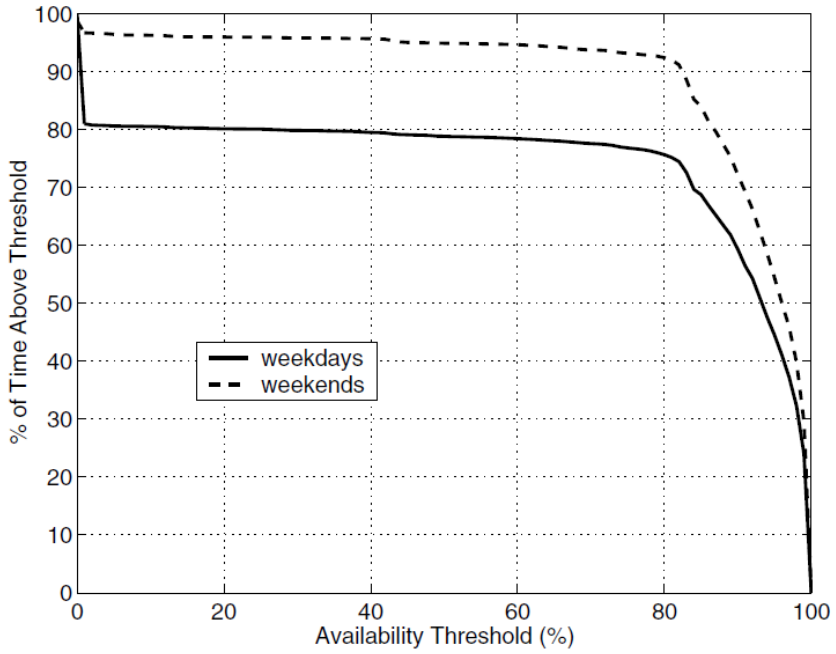



Figure [1]: Pourcentage du temps lorsque la disponibilité du processeur est supérieure à un seuil donné (à partir de [3]).

Ces ressources pourraient être utilisées pour le traitement vidéo, de façon similaire à la notion de "l'Informatique en Brouillard" [12]. Bien sûr, en raison de la nature dynamique des ressources de la grille, fournissant Service Level Agreement (SLA) est difficile, par rapport à MCU dédié. Dans la pratique, le système DGC peut être soutenu par un service de la vidéoconférence en nuage, qui sera utilisé lorsque le système DGC n'a pas assez de ressources. En combinant le système DGC avec un service de vidéoconférence en nuage, on peut obtenir des avantages financiers évidents, comme la grille existe déjà avec aucune dépense supplémentaire nécessaire.

Le système est conçu pour les topologies typiques de réseau d'entreprise, contenant des sites avec la réseau locale rapide inter-relié par potentiellement plus lents liens Internet. Les algorithmes proposés analysent les caractéristiques du réseau, tels que le retard entre les sites et la bande passante Internet requis pour les flux vidéo ainsi que les caractéristiques de nœud de réseau, comme la charge CPU, le type de connectivité réseau et le type d'alimentation afin de fournir la meilleure possible Quality of Experience (QoE) dans les circonstances actuelles. Pour comparer d'autres variantes de la répartition des tâches, une méthode de Multi Attribute Decision Making (MADM) spécialement adaptée à ce cadre est introduit.

Prenons un exemple du système DGC déployé sur les trois sites. Plusieurs utilisateurs organisent une conférence vidéo. Trois ordinateurs sont enregistrés dans le système, leurs caractéristiques sont comparées et le système décide de déployer le traitement vidéo nécessaire à la conférence sur l'un d'entre eux (Fig. [2]).

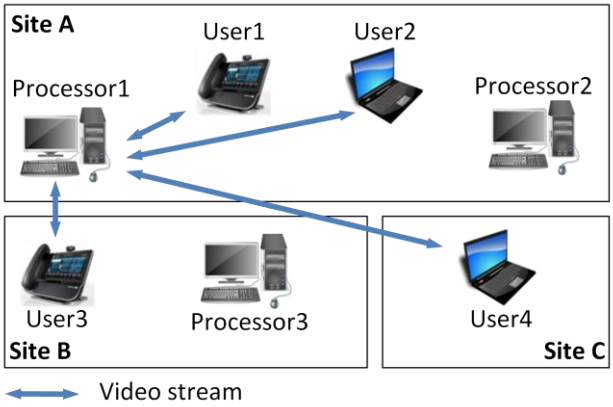



Figure [2]: le traitement de la conférence vidéo est hébergé sur Processeur 1

À un certain moment, un processus tiers, consommant beaucoup de puissance CPU, est lancé sur le PC, qui héberge le traitement de la conférence de sorte que le système décide de re-hôte le traitement de la conférence à un autre PC (Fig. [3]).

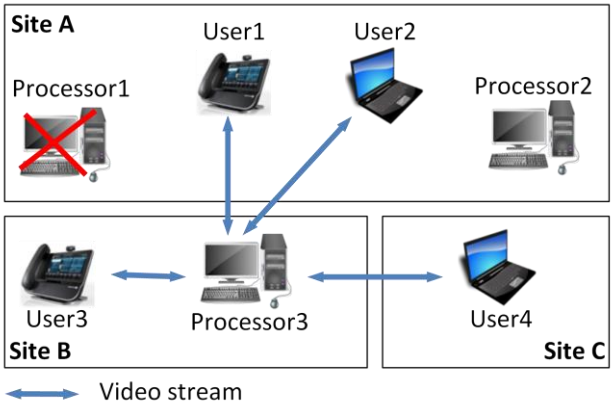



Figure [3]: Le traitement de la conférence vidéo est accueilli à Processeur 3 en raison de l'augmentation de la charge CPU sur Processeur 1

## La description du système

La description du système DGC repose sur deux notions principales: Tâches et Processeurs.

Tâche est une activité sur les flux médias, traditionnellement fourni par un MCU ou un serveur multimédia logiciel: mixage vidéo, basculement le flux vidéo sur la personne qui parle, transcodage ou d'autres manipulations sur les flux vidéo. Les flux audio accompagnent traditionnellement les flux vidéo et sont simplement mélangés ensemble par le même serveur de médias. Par exemple, une tâche associé à la

conférence vidéo représenté sur la Fig. [2] est une mélange vidéo de 4 flux en un seul flux résultant (généralement avec l'accent sur la personne en cours de parler) et potentiellement transcodage, en cas de codecs incompatibles des terminaux vidéo d'utilisateurs.

Le Processeur est un serveur média déployé sur une plate-forme générale comme un PC. Les utilisateurs peuvent activer/désactiver leur PC et de lancer des applications tierces qui consomment la puissance du CPU, ainsi que le démarrage/arrêt des appels et des conférences au hasard. Cela se traduit par l'imprévisibilité des ensembles de Tâches et Processeurs, qui doit être prise en compte par le système.

La logique principale de l'architecture proposée est de distribuer et, le cas échéant, de redistribuer les Tâches sur les Processeurs prenant en compte les changements dans l'ensemble des Tâches, ensemble de Processeurs et des contraintes externes (qui sont énumérés ci-dessous). Le résultat de la distribution doit être «optimale» dans certaines conditions.

Critères d'optimisation peuvent être divisés en deux groupes: ceux du réseau et de la plate-forme.

Critères de réseau qui doivent être prises en compte comprennent:

1) Bande passante de WAN consommée par une Tâche. Le but est d'essayer d'économiser la bande passante de WAN qui est généralement à la charge (par opposition à la bande passante de LAN qui est considéré comme gratuit et donc pas contrôlé).

2) Délai de bout en bout entre les terminaux vidéo. Le délai est très important caractéristique représentant le niveau de QoE, comme le délai important rend difficile une conversation interactive, voire impossible.

Les critères de la plate-forme sont liés aux Processeurs qui sont disponibles dans le système:

1) La connectivité réseau: prend en compte le fait que la plate-forme utilise la connectivité réseau filaire ou sans fil (Wi-Fi). Les connexions filaires offrent généralement la plus grande stabilité et moins de retard, ce qui les rend préférables pour les communications vidéo interactives par rapport aux connexions sans fil.

2) Alimentation: prend en compte le fait que la plate-forme est alimenté par le circuit électrique ou par sa batterie. Il est particulièrement important que les opérations de traitement vidéo sont très intensives au niveau de CPU.

3) Le partage des ressources: prend en compte le fait que la plate-forme (PC) héberge uniquement Processeur (serveur média) ou elle est partagé avec d'autres activités de l'utilisateur sans rapport avec le système DGC. Ce critère donne la

préférence aux plates-formes où aucune applications des utilisateurs exécutent. Telle préférence donne la stabilité aux système DGC, comme la consommation CPU est plus prévisible. Dans le même temps, cette logique empêche de déployer les Tâches sur les plates-formes utilisées activement par les utilisateurs afin de ne pas les déranger.

4) La charge CPU: fournit l'estimation de la charge CPU prévue après une Tâche donnée est déployée sur un Processeur donné. Le système tente de répartir les Tâches de manière à ce que la charge CPU sur chaque plate-forme serait minimisé afin de sécuriser les processus si leur demande de ressources CPU devait augmenter.

Potentiellement d'autres critères d'optimisation peuvent être facilement intégrés dans la logique de la répartition des Tâches, sur la base de l'expérience de l'utilisation de la mise en œuvre réelle du système DGC.

Tous les critères d'optimisation sont différents dans leur importance, ce qui nous permet de choisir une approche MADM (Multi-Attribute Decision Making), où chaque critère sera associé à un poids.

Les changements dans le système qui nécessitent la distribution des Tâches ou, dans certaines circonstances, la redistribution forment une file d'attente des événements de changement d'état (State Change Event = SCE). Il existe plusieurs types de tels événements:

1) La Tâche est ajoutée: par exemple, une nouvelle conférence est créé et la Tâche de mixage vidéo doit être distribué à un certain Processeur (voir la figure [2]).

2) La Tâche est supprimée: une Tâche déployée sur certain Processeur est plus nécessaire dans le système. Suppression d'une Tâche peut entraîner la redistribution d’autres Tâches dans le système pour l'optimisation globale.

3) Le Processeur est ajouté: un nouveau Processeur est ajouté au système. Certaines Tâches peuvent être redistribuées en prenant en compte le Processeur ajouté.

4) Le Processeur est supprimé: un Processeur est retiré du système. Si des Tâches ont été déployées sur ce Processeur alors ces Tâches doivent être redistribuées à d'autres Processeurs.

5) Valeur d'un critère d'optimisation est changé: la configuration du système a été modifiée, par exemple, la connectivité réseau d'un Processeur a été modifiée à partir de Wireline à Wireless. Dans ce cas, certaines Tâches peuvent être redistribuées, si nécessaire.

La file d'attente de SCE fonctionne comme un file d'attente FIFO (First In, First Out) avec des priorités strictes. Les priorités sont les suivantes (par ordre décroissant):

1) Processeur est supprimé (avec des Tâches déployées sur lui).

2) La charge CPU est augmentée d'une manière telle qu'elle peut bloquer l'exécution des Tâches.

3) Tâche est supprimée, Processor est ajouté, Processor est retiré (sans tâches sur lui), la charge CPU est réduite, d'autres critères d'optimisation (à savoir pas de charge CPU) sont modifiés.

4) Tâche est ajoutée.

La plus haute priorité est réglé sur l'événement "Processeur est supprimé" comme certaines Tâches sont bloquées dans cette situation, ce qui conduit à une mauvaise expérience utilisateur. La deuxième priorité est réglé sur l'événement "La charge CPU est augmentée" pour la même raison potentiellement aggraver l'expérience utilisateur. L'événement "Tâche est ajoutée" a la priorité la plus faible car il a le sens de prendre en compte tous les changements dans le système avant de distribuer une nouvelle Tâche afin d'éviter les redistributions consécutifs.

Au cours de traitement des SCEs les Tâches sont déployées / redéployé une par une. Voilà une fois qu'une décision est prise sur le déploiement / redéploiement, la Tâche est effectivement déployée / redéployée et le système attend jusqu'à ce que la Tâche se met à consommer des cycles CPU (le système est alors dans un état stable). Ensuite, le déploiement / redéploiement d'une Tâche suivante peut être traitée en fonction de la nouvelle valeur de la charge CPU.

L'objectif de la procédure d'optimisation consiste à calculer une valeur d'estimation numérique, en tenant compte de la diversité des critères, ce qui permettrait la comparaison des distributions possibles des Tâches sur les différents processeurs. La Tâche sera ensuite déployée sur le processeur avec la valeur cible optimale. Une méthode MADM volontairement créée en utilisant «normalisation au courant de contexte» est appliquée pour calculer la valeur d'estimation.

Une des spécificités des algorithmes de MADM est la nécessité de normaliser les valeurs des attributs. Dans le cas général, aucune hypothèse ne peut être faite sur eux. Plusieurs méthodes de la normalisation des valeurs dans la matrice de MADM sont bien connus ($S_{ij}$ sont des éléments de la matrice d'origine):

- $\alpha_{ij}=S_{ij}/\sum(S_{ij})$
- $\alpha_{ij}=S_{ij}/\max(S_{ij})$
- $\alpha_{ij}=(S_{ij}-\min S_{ij}) / (\max S_{ij}-\min S_{ij})$
- $\alpha_{ij}=S_{ij}/\sqrt{\sum(S_{ij})^2}$

Dans toutes ces méthodes, processus de normalisation implique des opérations sur les attributs de tous les cas possibles (par exemple somme des valeurs, valeur maximale, etc.). Cela signifie que lorsque l'ensemble des alternatives est modifié (à savoir Processeur est ajouté / supprimé ou la valeur du critère d'optimisation est modifiée), le processus de normalisation devrait être ré-exécuté. En tenant compte de la nature dynamique du système DGC, il serait hautement souhaitable de pouvoir effectuer les calculs nécessaires pour chaque alternative, indépendamment des autres. Une telle approche permet d'appliquer la procédure de MADM uniquement lorsqu'un Processeur est ajouté au système ou un attribut spécifique du Processeur est modifié. En d'autres termes, aucun calcul serait nécessaire pour un Processeur donné, quelles que soient les modifications appliquées à d'autres Processeurs.

Dans le contexte spécifique de notre problème, nous introduisons le processus de normalisation simple qui élimine ces dépendances. Nous savons en fait la nature de tous les attributs, leurs valeurs optimales et limites pratiques. Considérons les attributs de MADM utilisés dans le système DGC.

1) Retard End-to-end: La valeur optimale de retard est évidemment 0 (si l'on compte en millisecondes). Pour une valeur de retard normalisé, nous utilisons l'expression suivante:

normalized_delay = real_delay / delay_threshold

delay_threshold peut être défini de différentes façons. Par exemple, recommandation l'UIT-T G.114 peut être utilisé. Cette recommandation indique les retards vocaux acceptables dans des applications interactives. Retard inférieur à 150 ms est considérée comme acceptable, plus grand que 400 ms comme inacceptable et les valeurs entre les deux signifie qu'il y aura des problèmes de qualité. Une telle manière que nous pouvons définir la valeur 400 comme delay_threshold et cela signifie que tous les retards de plus de 400 ms ne seront pas distingués les uns des autres parce que tous les valeurs de normalized_delay plus grandes que 1 sont arrondies à 1.

2) Bande passante WAN utilisée: La valeur théorique optimale pour la bande passante WAN (WBW = WAN Bandwidth) utilisée par une Tâche est également 0, il est atteint lorsque tous les terminaux et le Processeur sont dans le même réseau local. Pour la bande passante WAN normalisée (normalized_WBW) la valeur que nous allons considérer est défini par l'expression suivante:

normalized_WBW = real_WBW / max_WBW

La valeur de max_WBW peut être considérée comme la somme des largeurs de bande de tous les flux vidéo d'une Tâche donnée. Cette valeur est connue au moment de la création de la Tâche.

3) Les critères de la plate-forme: Tous les critères de la plate-forme, à l'exception de la charge CPU (à savoir la connectivité réseau, l'alimentation, le partage des

ressources) sont binaires par leur nature, c'est-à-dire ils sont «positif» ou «négatif». Positifs sont:

- La connectivité réseau = filaire
- Alimentation = circuit électrique
- Le partage des ressources = dédié

Négatifs sont:

- La connectivité = réseau sans fil
- Alimentation = batterie
- Le partage des ressources = partagé

Pour la conformité nous avons mis la valeur "0" pour le cas positif et la valeur "1" pour le cas négatif. Grace à cela nous avons la situation quand variante idéale de la valeur de l'attribut est "0" et la normalisation n'est pas nécessaire.

La valeur de critère de charge de CPU est présentée dans les pourcentages d'utilisation du CPU prise après une Tâche donnée ont été déployés sur un Processeur donné. Il donne la valeur théorique optimale de "0" (pas réalisable dans la pratique) et la pire valeur de "100". Pour la valeur de la charge CPU normalisée, nous allons considérer l'expression suivante:

normalized_CPU_load = REAL_ CPU_load / 100

Critère de la charge CPU a quelques particularités, qui sont décrites ci-dessous.

Tous les critères d'optimisation utilisées dans les calculs sont représentés dans le tableau [1].

Tableau [1]: Critères d'optimisation

| **Attribute name** | **Ideal value** | **Worst value** | **Normalization divisor** |
|---|---|---|---|
| **End-to-end delay** | *0* | ∞ | *400, if delay <= 400*<br>*delay, if delay > 400* |
| **WAN bandwidth** | *0* | *Sum of all video streams* | *Sum of all video streams* |
| **Network connectivity** | *0* | *1* | *Not needed* |
| **Power supply** | *0* | *1* | *Not needed* |
| **Resource sharing** | *0* | *1* | *Not needed* |
| **CPU load** | *0* | *100* | *100* |

Le problème est finalement formulé sous la forme d'une méthode "Simple Additive Weighting" (SAW), mais inversée et normalisée:

$$OR_j = \sum_{i=1}^{M} w_i a_{ij} \Big/ M \qquad (1)$$

où:

$OR_j$ : Résultat Objectif pour Processeur $j$

$w_i$ : poids du critère $i$

$a_{ij}$ : valeur normalisée du critère $i$ sur le Processeur $j$

$M$: nombre des critères

La méthode SAW inversée signifie que nous devons prendre comme résultat la valeur la plus petite d'$OR_j$ au lieu de la plus grande. La méthode SAW normalisée signifie que la valeur $OR_j$ est dans l'intervalle [0, 1]. Cette formule implique que $OR_j$ est calculée pour chaque Processeur indépendamment et uniquement lorsque le Processeur apparaît dans le système ou la valeur d'un critère d'optimisation est modifié.

La charge de CPU du Processeur est différent des autres critères d'optimisation parce que sa valeur change en continu par rapport aux changements plutôt rares d'autres valeurs des critères. Du point de vue de la mise en œuvre pratique, cela signifie que nous pouvons calculer $OR_j$ pour tous les critères sauf la charge de CPU et le stocker dans un cache pendant que nous devons observer la valeur de la charge de CPU en temps réel.

En outre, afin d'être en mesure de calculer l'impact d'un type particulier de Tâche sur la charge de CPU d'un Processeur particulier, un processus de qualification préliminaire est nécessaire. Le processus de qualification signifie que le fournisseur du système DGC installe un Processeur sur une plate-forme particulière, tous les types des Tâches sont exécutées et les niveaux de consommation de CPU sont collectés et stockés. Ensuite, ces valeurs pré-collectées peuvent être utilisées comme une estimation du besoin de ressources de CPU lorsque le système DGC simule la distribution d'une Tâche sur un Processeur installé sur la plate-forme qualifiée sur un site du client.

Afin d'améliorer la perception de la conférence, nous introduisons le taux de redéploiements d'une Tâche, définie comme le nombre de fois que la Tâche existante est transférée de l'un Processeur à l'autre. Redéploiements conduiront à des interruptions dans les flux de médias, il est donc hautement souhaitable de les minimiser.

Un paramètre spécial "Redeployment Penalty" est utilisé par les algorithmes afin de réguler le nombre de redéploiements potentiels. Lorsqu'un Processeur est considéré comme un candidat à l'accueil d'une Tâche, le gain en "Objective Result" doit être au-dessus de ce seuil, afin que la Tâche d'être redéployé sur ce Processeur.

Notez également qu'un mécanisme simple d'hystérésis est appliqué sur la charge de CPU pour éviter des redéploiements cycliques lorsque la charge du CPU change de façon sporadique.

## Intégration avec Cloud

Comme il a été mentionné dans "Contexte et Motivation", le système DGC elle-même ne peut pas garantir SLA (Service Level Agreement) approprié parce que ses ressources sont contrôlées par les utilisateurs finaux, et non par le système lui-même. Pour résoudre ce problème système DGC peut être combiné avec le système de conférence dans le Cloud afin de fournir à la fois SLA et des avantages de coûts en même temps. Dans ce chapitre, nous utilisons le terme "Fog" pour les ressources de la grille de bureau afin de souligner son opposition à "Cloud".

Afin d'obtenir encore plus d'avantages, nous combinons une approche Cloud/Fog avec les différents types de serveurs de médias, notamment MCU et SFU, qui fournissent différentes caractéristiques d'exploitation. Toutes les combinaisons et les circonstances possibles, dans lesquelles leur utilisation donne le plus d'intérêt, sont pris en compte dans ce chapitre.

Récemment, conférence vidéo dans le Cloud est devenu populaire grâce à un certain nombre de propriétés utiles, telles que la flexibilité et modèle pay-per-use. Dans le même temps, du point de vue du fournisseur de conférence vidéo dans le Cloud, il existe un certain nombre de problèmes avec cette approche, nous mentionnons ici deux d'entre eux:

- Nécessité de ressources de traitement importantes dans le Cloud parce que tous les calculs sont concentrés en un seul endroit
- Augmentation du délai de bout en bout parce que les données sont envoyées à partir du client vers le Cloud et retour, souvent via des liaisons Internet lentes et peu fiables

Ces deux problèmes peuvent être résolus par le choix approprié du type de conférence vidéo établie par le fournisseur. Le type peut être la conférence traditionnelle "MCU" qui a besoin de plus de décodage/encodage sur le serveur et résulte dans une plus grande consommation de CPU et retard bout en bout, ou il peut être conférence SFU qui n'a pas besoin de traitement des flux vidéo. Dans le même temps un concept de "Fog" peut être engagée afin de choisir le type de ressources sur lesquelles le serveur multimédia sera déployé.

En combinaison dynamique MCU/SFU et Cloud/Fog on est capable de:

- Economiser des ressources de fournisseur de conférence vidéo dans le Cloud en termes de cycles de CPU et la consommation de bande passante réseau
- Réduire de manière significative le retard bout en bout, pour améliorer le QoE final

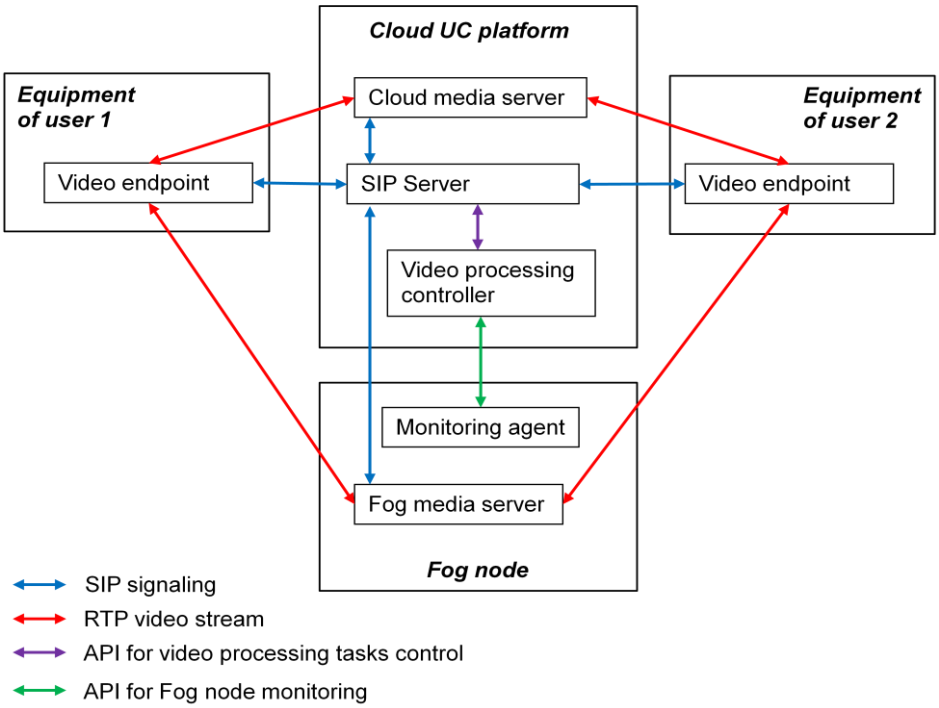



Figure [4]: Structure du système DGC intégré avec Cloud

L'idée est la suivante: nous combinons les techniques mentionnées ci-avant et obtient quatre approches possibles pour la vidéoconférence. Serveur de signalisation peut être dans le Cloud ou sur le site. Nous parlons ici que de serveur de médias et des flux de médias.

SFU dans le Fog:
'+': Pas d'utilisation de CPU dans le Cloud
'+': Pas d'utilisation de WAN
'+': Pas de décodage/encodage supplémentaire

MCU dans le Fog:
'+': Pas d'utilisation de CPU dans le Cloud
'+': Pas d'utilisation de WAN
'-': Décodage/encodage supplémentaire

SFU dans le Cloud:
'+/-': Modéré utilisation de CPU dans le Cloud
'-': Utilisation de WAN étendue
'+': Pas de décodage/encodage supplémentaire

MCU dans le Cloud:
'-': Grande utilisation de CPU dans le Cloud
'-': Utilisation de WAN étendue
'-': Décodage/encodage supplémentaire

Nous pouvons voir que du point de vue des deux parties (fournisseurs et clients) les quatre approches peuvent être priorisés (du meilleur au pire):

1. SFU dans le Fog
2. MCU dans le Fog
3. SFU dans le Cloud
4. MCU dans le Cloud

L'utilisation des ressources de Fog sur les locaux sont prescrits par les politiques qui sont négociés entre le fournisseur et le client. Les conditions peuvent varier de permissives ("toutes les ressources non occupées peuvent être utilisées pour la conférence") à restrictive ("client interdit l'utilisation des ressources sur site", qui se traduit par une solution de Cloud pure).

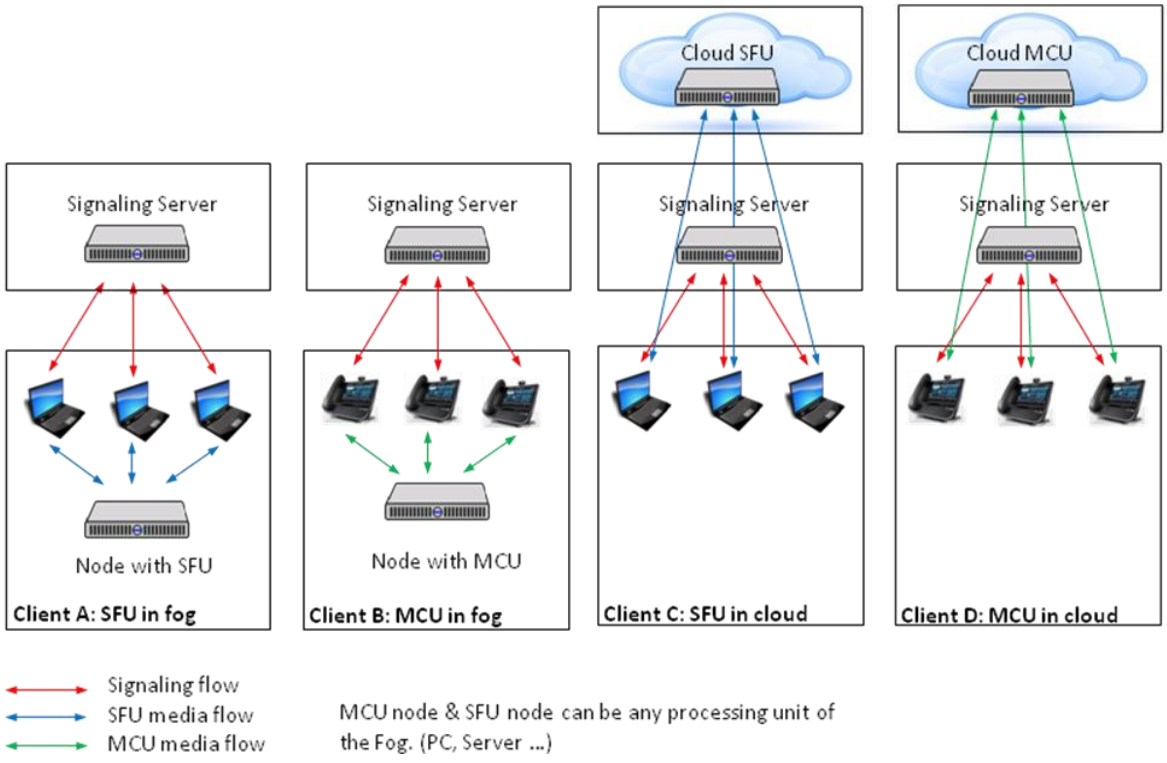



Figure [5]: Les types possibles de vidéoconférence

## Évaluation par simulation

L'objectif de la simulation était notamment de découvrir comment la valeur de Redeployment Penalty affecte différents aspects de la solution. Pour la raison de la performance l'exécution des calculs sont mises en œuvre dans des nombres entiers avec toutes les valeurs normalisées dans l'intervalle [0, 100].

Le premier point important que nous avons abordé est le nombre de redéploiements des Tâches au cours de leur exécution. Chaque redéploiement représente un compromis entre l'optimisation d'Objective Result et la perturbation de l'expérience utilisateur provoquée par ces redéploiements, comme les flux vidéo

doivent être réacheminés vers un nouveau Processeur. Dans la Fig. [6] est représenté le nombre de Tâches déployées (pour chaque simulation avec le Redeployment Penalty donné) et le nombre de Tâches redéployées. Pour Redeployment Penalty > 60, il n'y a plus de redéploiements dans le système.

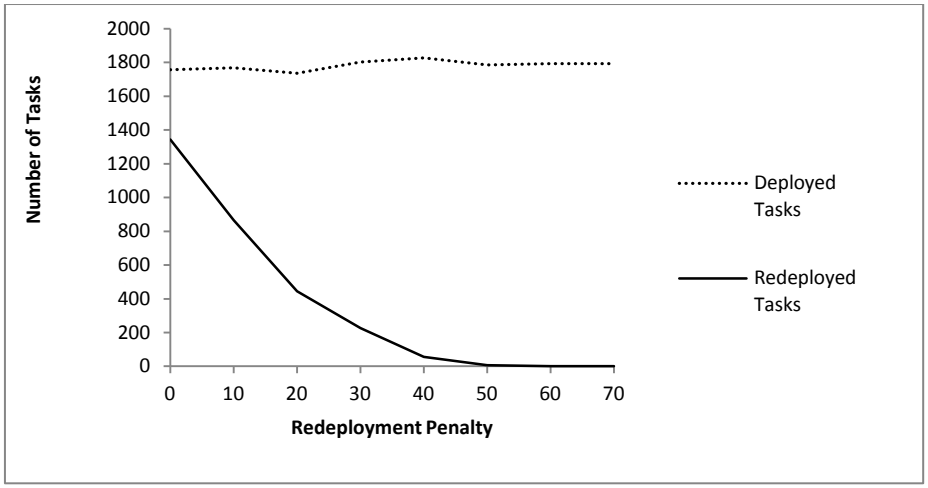



Figure. [6]. Nombre de Tâches déployées et redéployées en fonction de Redeployment Penalty

Le deuxième point que nous avons considéré est le delta entre Factual Result (FR) et Ideal Result (IR). FR est le résultat de l'application des algorithmes décrits ci-dessus. IR est une sortie de l'algorithme qui, après l'arrivée de chaque State Change Event, prend tous les Processeurs, Toutes les Tâches et calcule le déploiement théorique qui minimise la somme des Objective Results de toutes les Tâches. Dans la topologie limitée que nous avons considéré, l'IR peut être simplement calculée par une énumération exhaustive (comparant tous les déploiements possibles). La valeur IR représente la répartition optimale des Tâches sur les Processeurs, ne tenant pas compte de leur ordre d'arrivée. Dans la figure [7] nous pouvons observer le compromis entre une valeur basse de Redeployment Penalty (provoquant une certaine perturbation de l'expérience utilisateur en raison de redéploiement des Tâches) mais au même temps les valeurs proches de FR et IR; et une valeur haute de Redeployment Penalty causant faible perturbation de l'expérience utilisateur, mais écart augmenté entre FR et IR.

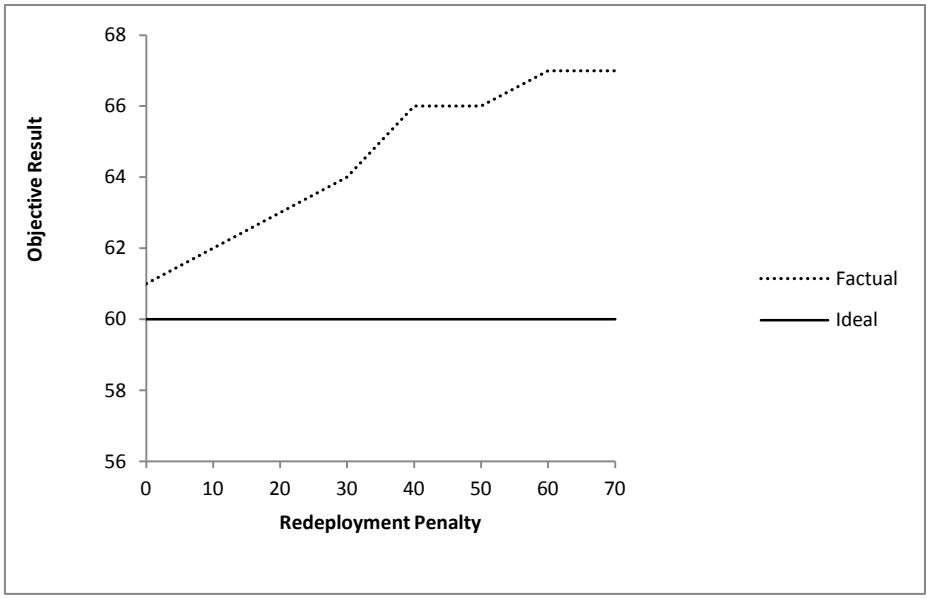



Figure [7]. Factual Result et Ideal Result en fonction de Redeployment Penalty

Ces simulations montrent un compromis clair entre optimalité du système et le nombre de redéploiements. Dans ces figures, Factual Result peut approcher Ideal Result, même sans trop de redéploiements. Cependant, quelle valeur de Redeployment Penalty devrait être pris dans l'exploitation réelle ne peut être déterminée que avec des paramètres réalistes (consommation CPU qualifié, le poids des MADM accordés), qui peuvent être disponibles seulement après l'essai de la mise en œuvre du système basé sur la plate-forme matérielle réelle avec le vrai serveur multimédia.

## Conclusions et Directions des Travaux Futures

Dans cette thèse, nous avons étudié une approche novatrice pour l'organisation de vidéoconférences. De nos jours, la vidéoconférence dans les entreprises est organisée principalement à l'aide de MCU centrales. MCU est responsable du contrôle de la conférence ainsi que des tâches de traitement vidéo, telles que le mixage ou le codage trans. En raison du fait que les MCU sont généralement conçus sous la forme de matériel spécialisé, ils sont un équipement coûteux. Les MCU de logiciels purs existent également, elles peuvent être utilisées en mode Cloud. Cependant, en raison des opérations complexes avec les flux média, ils consomment beaucoup de ressources de serveur. Dans le même temps, les approches Overlay Network existent pour la vidéoconférence: Application Layer Multicast et Peer-To-Peer. Ces approches sont conçues pour les relais vidéo, tandis que les tâches de mixage vidéo sont directement traitées aux points finaux. Par conséquent, si un point final n'est pas capable de mélanger plusieurs flux vidéo, en raison de certaines limitations matérielles / logicielles, il ne bénéficiera pas de l'expérience de téléprésence moderne.

Le problème est donc de fournir une expérience vidéo enrichissante, disponible aujourd'hui grâce à des MCU dédiés, sans utiliser de matériel dédié et sans surcharger les serveurs existants avec des opérations de traitement des médias.

La solution proposée consiste à distribuer des MCU sur Enterprise Desktop Grid, qui comprend tous les PC disponibles dans l'entreprise, avec suffisamment de ressources pour accepter les tâches de traitement vidéo. Les recherches antérieures montrent que beaucoup d'ordinateurs personnels dans une entreprise ne sont pas utilisés pendant de longues périodes, même pendant les heures de travail. Dans la terminologie moderne, cette approche est connue sous le nom de «Cloud computing» contrairement au «Cloud computing» centralisé.

Les exigences pour construire une telle MCU distribuée:

- L'architecture du réseau devrait s'appliquer à la topologie d'entreprise typique, contenant des sites avec un réseau local rapide connecté par Internet potentiellement lent

- L'architecture doit prendre en compte la nature dynamique de Enterprise Desktop Grid, en particulier le fait que les PC peuvent être arbitrairement arrêtés ou que les processus tiers peuvent être lancés par les utilisateurs finaux

Le système de conférence Desktop Grid Conferencing (DGC) que nous proposons consiste en un ensemble de serveurs multimédia (aborder les tâches de traitement vidéo), distribués sur un cluster de matériel de bureau ordinaire (PC, ordinateurs portables, etc.). La description du système DGC repose sur deux notions principales: les tâches et les processeurs.

La tâche est une activité liée aux médias, fournie traditionnellement par un MCU ou un serveur multimédia logiciel: mélange vidéo, commutation vidéo, codage trans, trans-mise à l'échelle ou d'autres manipulations sur les flux vidéo.

Processor est un serveur multimédia déployé sur un matériel général, tel qu'un PC. Les utilisateurs peuvent activer / désactiver leurs PC et lancer des applications tierces consommant de l'énergie CPU ainsi que des appels de démarrage / arrêt et des conférences au hasard. Il en résulte une imprévisibilité des ensembles de tâches et de processeurs, qui doit être pris en compte par le système.

La logique principale de l'architecture proposée est de distribuer et, si nécessaire, de redistribuer les Tâches sur les Processeurs en tenant compte des changements dans l'ensemble des Tâches, ensemble de processeurs et contraintes externes. Le résultat de la distribution devrait être «optimal» dans certaines conditions.

Les critères d'optimisation peuvent être divisés en deux ensembles: les réseaux et les plateformes. Les critères de réseau qui devraient être pris en compte comprennent la bande passante WAN consommée par une tâche et un délai de bout en bout entre les points finaux. Les critères de la plate-forme sont liés aux

processeurs disponibles dans le système: connectivité réseau, alimentation, partage des ressources, chargement de la CPU.

Tous les critères d'optimisation sont différents selon leur nature et leur importance, ce qui nous conduit à choisir une démarche MADM (Multi-Attribute Decision Making), où chaque critère est associé à un poids. L'application d'une méthode MADM donne une métrique intégrale d'un déploiement d'une tâche donnée à un processeur donné, appelé Objective Result. Une méthode MADM dédiée utilisant "normalisation contextuelle" a été conçue pour calculer le résultat objectif. Dans cette méthode, la normalisation est dérivée de la nature des attributs. Une telle approche permet d'appliquer la procédure MADM uniquement lorsqu'un processeur est ajouté au système ou qu'un attribut spécifique du processeur est modifié. En d'autres termes, aucun calcul n'est nécessaire pour un processeur donné, quels que soient les changements appliqués aux autres processeurs, ce qui est très important, compte tenu de la nature dynamique en temps réel du système DGC.

Le système DGC lui-même ne peut garantir un accord de niveau de service (SLA) approprié car ses ressources sont contrôlées par les utilisateurs finaux et non par le système lui-même. Pour résoudre ce problème, le système DGC peut être combiné avec le système de conférence dans le Cloud pour fournir à la fois SLA et avantages de coûts en même temps. Nous avons développé les algorithmes, combinant l'approche Cloud / Fog avec différents types de serveurs multimédias. Le résultat fournit une solution de conférence optimisée en termes de coût tant pour le fournisseur que pour le consommateur, ainsi que sur l'expérience de l'utilisateur final.

Afin de tester les algorithmes de distribution de tâches, la logique respective a été implémentée à l'aide d'une approche de simulation d'événement discrète.

Le premier point abordé dans la simulation est le nombre de redéploiements de Tâches lors de leur exécution. Chaque redéploiement représente un compromis entre l'optimisation de Objective Result et la perturbation de l'expérience de l'utilisateur qui accompagne le redéploiement.

Le deuxième élément que nous avons considéré est le delta entre le résultat factuel et le résultat idéal. Le résultat factuel est le résultat de l'application des algorithmes, calculant Objective Result dans la situation actuelle du système. Le résultat idéal est une sortie de l'algorithme qui, après l'arrivée de chaque événement de changement d'état, prend tous les processeurs, toutes les tâches et calcule le déploiement théorique qui minimise la somme des résultats objectifs de toutes les tâches. La valeur du résultat idéal représente la répartition optimale des tâches sur les processeurs, sans tenir compte de leur ordre d'arrivée.

Ces simulations montrent un compromis clair entre l'optimisation du système et le niveau d'expérience de l'utilisateur, affecté par les redéploiements de tâches. En fait, le résultat factuel peut s'approcher du résultat idéal, même sans trop de redéploiements. Cependant, la logique, responsable de la décision sur le

redéploiement, ne peut être déterminée qu'avec des paramètres réalistes (consommation de CPU qualifiée, pondérations MADM accordées), qui peuvent être disponibles uniquement après un test intensif de l'implémentation du système en fonction du serveur multimédia réel déployé sur le Plates-formes matérielles réelles.

Ensuite, nous avons étudié dans quelle mesure un PC peut être utilisé comme plate-forme pour héberger un serveur multimédia et comment la charge CPU de cette plate-forme affecte la qualité du flux vidéo résultant. Pour cela, nous avons créé un banc d'essai avec un serveur multimédia open source, déployé sur un ordinateur portable habituel, et connecté plusieurs téléphones portables vidéo jouant le rôle de terminaux de conférence.

Pour l'un des points d'extrémité, nous avons connecté un outil de mesure de la qualité de la vidéo, qui nous a fourni un indice d'opinion moyen prévu. Nous avons appliqué cet outil à un flux vidéo, généré par un serveur de médias de vidéoconférence. Le serveur, déployé sur un ordinateur portable de commodité, a été perturbé par un processus de tierce partie, qui a consommé différentes quantités de puissance de l'UC. En conséquence, nous avons démontré que le matériel de bureau de commodité peut vraiment être utilisé comme une plate-forme pour les serveurs de médias, transportant une charge de travail limitée dans la portée de notre système de conférence Enterprise Desktop Grid.

Deux applications de l'apprentissage par machine peuvent être envisagées afin d'améliorer la qualité du système.

Pour une plate-forme donnée avec des poids initiaux définis en exécutant un nombre limité de tests manuels, lors de son exploitation, nous pouvons:

- Basé sur une configuration / état de plate-forme donnée pour essayer de prédire dynamiquement des poids qui maximiseront la QoE (ex: la panne de Wi-Fi augmente le poids de "Connectivité réseau"). Pour cela, nous avons besoin de recueillir continuellement des informations sur différents aspects de l'environnement système: état du réseau et de l'équipement de réseau, types d'ordinateurs personnels utilisés, etc. Ensuite, après la corrélation de cette information avec QoE résultante, nous pouvons déduire les poids des critères existants ou nouvellement créés afin de maximiser la QoE qui en résulte.

- Pour tenir compte de l'historique du fonctionnement du système pour les distributions futures (ex: les observations de nœud stables / non stables présentent le "notation" du nœud). De cette façon, nous pourrions créer une sorte de «profils de ressources», c'est-à-dire les caractéristiques typiques de l'utilisation et du comportement des ressources, ce qui affecte la stabilité globale du système. En corrélatant ces modèles avec des jours de semaine, de temps et d'autres informations sur l'environnement, nous pourrons prévoir dans une certaine mesure le comportement de ressources données dans le futur.

# Contents

# Glossary

***CPU Load Qualification Matrix*** – matrix containing information on CPU load for a given type of Tasks deployed on Processor installed on a given type of platform

***Desktop Grid Conferencing (DGC)*** – general name of the overall system

***DGC stable state*** – a state of DGC system when all the Tasks are deployed and consume CPU resource or waiting in SCE queue, that is there is no distribution process ongoing

***Dynamic Simulation*** – procedure of simulation of CPU load for a given Task being distributed to a given Processor

***Dynamic Simulation Result (DSR)*** – numerical result of dynamic simulation

***Endpoint*** – a user's device which originates/terminates media streams (PC, laptop, tablet, deskphone, mobile device, conference specialized hardware, …)

***Full Simulation Result (FSR)*** – numerical value reflecting integrated evaluation of distribution of a given Task to a given Processor. SSR and DSR are used for calculation of FSR

***Leg*** – a connection between an Endpoint and a Conference. Consists of two flows: signaling flow (SIP) and media flow (RTP)

***Node*** – hardware resource on which a Processor can be installed (PC, laptop, tablet, …)

***Objective Result (OR)*** – integral metric of a deployment of a given Task to a given Processor

***Optimization criteria*** - the rules that determine by which criteria the variants of distribution are compared with each other in order to understand which one is better

***Processor*** – software component (media server) which is installed on a Node and which executes Tasks by processing media streams

***Real CPU Load (RCL)*** – real CPU load of a given Processor

***Real Deployment Result (RDR)*** – numerical value calculated by the same formula as FSR but with RCL used instead of DSR. Such a way RDR reveals the integrated characteristic of real deployment of a given Task on a given Processor contrary to FSR which reveals the integrated characteristic of simulation of such a deployment

***SCE queue*** – prioritized queue of State Change Events. Priorities reflect the urgency with which a given event should be taken into account

***State Change Event (SCE)*** – an event reflecting the change in the set of Tasks, set of Processors or in external conditions which causes Task distribution or consideration of Tasks re-distribution

***Static Simulation*** - procedure of calculation of a numerical value reflecting how “good” is a given Processor for deploying a given Task in terms of location in network and characteristics of hardware of the platform on which Processor is installed. Several optimization policies are estimated by a Multi-attribute Decision Making Method to get the final value.

***Static Simulation Result (SSR)*** – numerical result of static simulation

***Task*** – a set of manipulations executed by Processor on input video streams with a target to produce required output video streams. Manipulations include video mixing, video switching, trans-coding, trans-scaling, streams relay, etc

***Task deployment*** – a process of actual assigning of a Task to a Processor, that is instructing Endpoints to send their video streams to the given Processor for execution of the Task

***Task distribution*** – a process of choosing of an appropriate Processor for a Task based on static and dynamic simulation results

# Chapter 1

# Introduction

Video conferencing is a well-established area of communications, which have been studied for decades. Recently this area has received a new impulse due to significantly increased bandwidth of Local and Wide area networks and appearance of low-priced video equipment. At the same time high quality video images such as Full HD may require significant computational resources for their processing. Video processing for conferencing includes some manipulations necessary to get advanced user experience (mixing together several video streams or switch the image to the currently speaking participant) as well as operations caused by the incompatibility of endpoints, e.g. trans-coding in the case when participants use different video codecs.

Currently, two distinct architectures for handling these video-processing tasks are used.

The traditional solution is using Multipoint Control Unit (MCU) [29]. MCU is a powerful component that centralizes all video processing operations and distributes the resulting streams. MCU can be implemented as a hardware unit with integrated Digital Signal Processors (DSP) or a software component installed on commercial off-the-shelf (COTS) servers. Also MCU can be deployed in both cloud and on-premises mode. In all deployment scenarios, MCU represents a dedicated resource, which needs to be purchased or leased.

Another solution is to use endpoints as resources for video processing. This can be achieved by exploiting Peer-to-Peer (P2P) or Selective Forwarding Unit (SFU) [29] strategy. P2P approach has been thoroughly researched but this technique has not gained traction in enterprise communications, as it doesn't provide easy means of integration with business applications as well as implementation of important enterprise requirements like access to heterogeneous LDAP directories. SFU is a software component, which forwards video packets based on endpoints capabilities. It doesn't perform any media processing on video streams, it only filters and relays packets. As a result, the capacity of a conferencing system driven by SFU depends on the capabilities of endpoints. If endpoints are incompatible in terms of codecs then SFU is not useful, as endpoints normally don't have trans-coding functionality.

In this thesis a Desktop Grid Conferencing (DGC) system is proposed, which uses resources of the enterprise desktop grid (PCs, laptops etc. deployed within the enterprise network) for allocating video processing services needed for organizing videoconferences.

Previous research on enterprise desktop grids [39] demonstrates that a significant amount of CPU resources of PCs used within enterprises are not occupied with any activity.

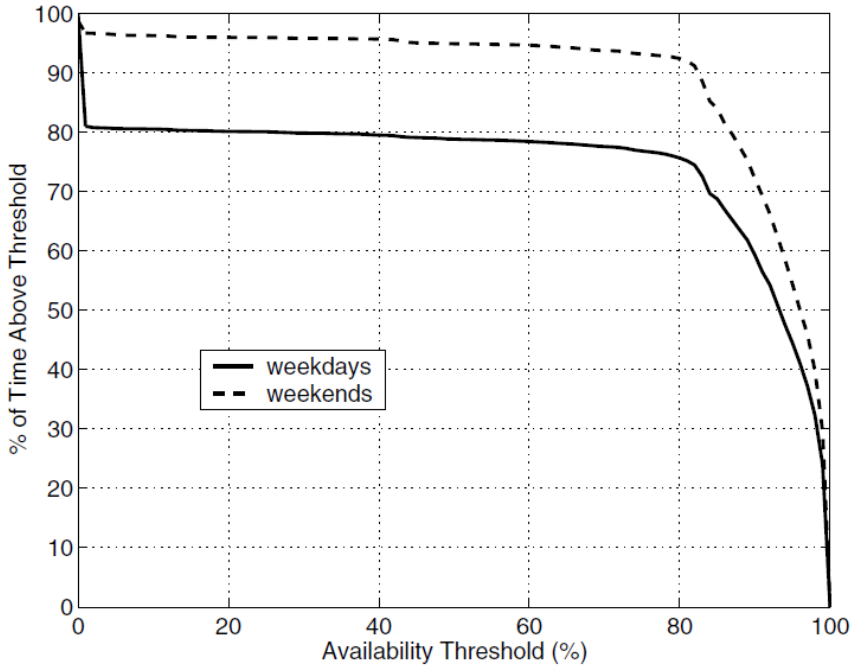



Figure 1.1: Percentage of time when CPU availability is above a given threshold (from [39]).

These resources could be used for video processing, similarly to the concept of “Fog Computing” [38]. Of course, due to the dynamic nature of the grid resources, providing Service Level Agreement (SLA) is challenging, as compared to dedicated MCU. In practice, the DGC system can be backed by a cloud video conferencing service, which will be used when the DGC system doesn’t have enough resources. Combining the DGC system with a cloud conferencing service, one can obtain clear financial benefits, as the grid already exists with no extra expenditure needed.

The system is designed for typical enterprise network topologies, containing sites with fast LAN inter-connected by potentially slower Internet links. The proposed algorithms analyze network characteristics, such as delay between sites and Internet bandwidth required for video streams as well as grid node characteristics, such as CPU load, network connectivity type and power supply type in order to provide the best possible Quality of Experience (QoE) under current circumstances. To compare alternative variants of task distribution, a Multi Attribute Decision Making (MADM) method specially customized to this framework is introduced.

Let’s consider an example of the DGC system deployed on three sites. Several users organize a video conference. Three PCs are registered in the system, their characteristics are compared and the system decides to deploy video processing needed for the conference on one of them (see Figure 1.2).

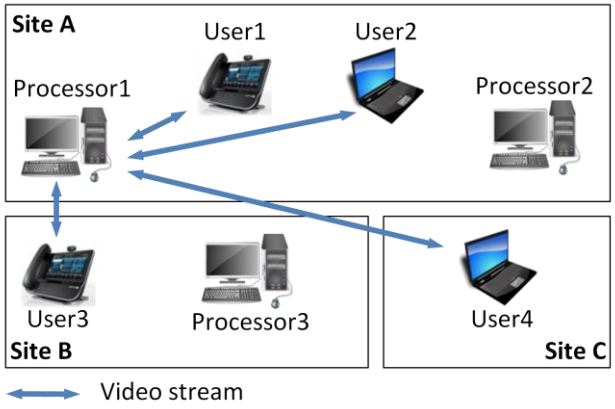



Figure 1.2: Video conference processing is hosted on Processor1

At some moment a third-party process, consuming a lot of CPU power, is started on the PC hosting video conference processing so the system decides to re-host conference processing to another PC (see Figure 1.3).

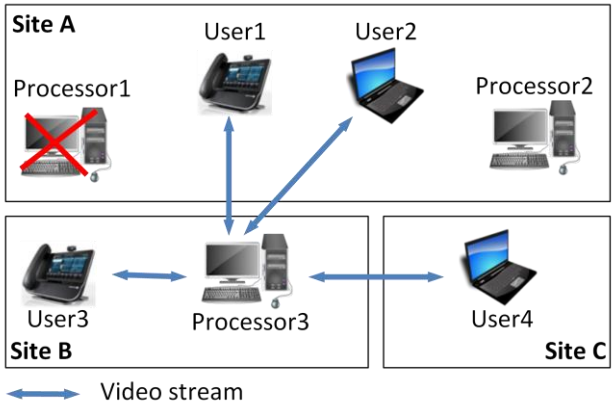



Figure 1.3: Video conference processing is re-hosted to Processor3 due to the increase of CPU load on Processor1

***Structure of the thesis***

In Chapter 2 we provide an overview of industrial and academic state-of-the-art of video conferencing. Methods and technologies, employed in modern video conferencing solutions are comprehensively introduced, and the positioning of a proposed approach is highlighted among them. In academic research overview we list the topics, elaborated by the present time, noting that the research focuses mostly on Internet topologies, oriented to general public, with no results applicable to the specific enterprise grade solution needs.

Chapter 3 describes the algorithms that allow manipulations on video conferences and desktop grid resources, on which the conferences are deployed. The chapter begins with the list of criteria, that we consider in order to provide an optimized solution for the task of deploying conferencing on grid nodes. Then we present a customized Multi-Attribute Decision Making (MADM) method, which combines proposed criteria in one resulting value in order to make the nodes comparable between each other. The difference between our

MADM method and the traditional ones is that our approach is applicable in real-time to constantly changing set of options without the need of extra calculations. And finally we elaborate detailed algorithms for all possible events, that can occur in a grid based conference system.

In chapter 4 we present the architecture of the solution. In the first part we describe the standalone system, which uses only grid resources. The problematic of delay estimation is tackled. The static design view with all necessary interfaces as well as dynamic design view with several key workflows are elaborated. In the second part we provide the description of the grid based system combined with the cloud based system. This combination allows providing guaranteed level of the solution service, even when grid resources are lacking.

In chapter 5 we discuss the simulation, which verifies the algorithms from chapter 3, based on implementation described in chapter 4. We use the statistics of utilization of a real conference system, deployed at an enterprise, and a conference topology, typical for an enterprise grade conferencing system. In simulation we demonstrate the trade-off between the optimality of the system at any given moment and its stability, that is the frequency of redeployments of conference activities from one node to another. As a result we show that good results in the terms of system optimality can be achieved without significant disturbance of user experience.

In chapter 6 we consider the question of using a PC as a platform for a soft media server. For that we demonstrate which level of conference quality can provide a PC, being loaded at the same time by third party processes. For that we use several types of software, which allow loading the CPU in a controlled manner, and a hardware solution, which is capable to estimate the quality of a video stream from the end-user point of view.

Chapter 7 concludes the thesis and contains suggestions for further work.

Annex A presents the standalone DGC system design and Annex B presents the design of the Cloud integrated DGC system.

# Chapter 2

# Video conferencing state-of-the-art

This chapter presents the main technologies behind the modern IP-based videoconferencing services, with a particular focus on codecs, network protocols and architectures. Traditional industrial disposition as well as modern innovative approaches are both addressed. Results of academic research on video conferencing are also presented.

Legacy analog/digital technologies, together with the gateways between the traditional and the IP videoconferencing systems, are not considered.

The proposed Desktop Grid Conferencing system is also regarded in line with other technologies in order to demonstrate its potential positioning in the industry.

## 2.1. Video conferencing industry

### 2.1.1. Introduction

Video conferencing is a two-way interactive communication, delivered over networks of different nature, which allows people from several locations to participate in a meeting.

Conference participants use video conferencing endpoints of different types. Generally a video conference endpoint has a camera and a microphone. The video stream, generated by the camera, and the audio stream, coming from the microphone, are both compressed and sent to the network interface. Some additional info like instant messages, the shared screen or a document can be also exchanged between participants.

IP video conferencing, considered in this tutorial, is based on the TCP/IP technology as a transport network for all these flows. In the past, specially designed analog lines and digital telephonic lines (ISDN) had been employed for that purpose. IP started to be used in the 1990s and has become the prominent vehicle for video conferencing since then.

Today IP video conferencing is a well known and widely used service. However, most users might not realize that it has a notably complex architecture, involving a wide range of technologies. This tutorial aims at providing an overview of possible architectures and technologies, involved in the realization of videoconferencing services.

### 2.1.2. Functional architecture example

There exist two fundamental means to set up videoconference calls between participants. Basic conference functions can be offered in peer-to-peer mode, in which all the participants are connected directly with each other (see Figure 2.1).


Figure 2.1: Peer-to-peer video conference

Conferences, which provide more services, such as central management of participants or conference recording, generally make use of a central point of control (“Middlebox”) in order to implement these additional services (see Figure 2.2).


Figure 2.2: Video conference with a middlebox

In this section, we present an example of a possible functional architecture of a conferencing solution, using several dedicated servers (defined hereafter), as depicted in Figure 2.3. These servers play a role of the “middlebox” in the centralized conferencing architecture. Such architecture is typical for advanced video conferences in enterprises.

#### 2.1.2.1. Functional elements

**Endpoint:** A software application or dedicated hardware equipment, which allows a user to participate in a conference. It consists of the following elements:

- Equipment for capturing and rendering both audio and video: a screen, a microphone and a loudspeaker or headphones
- Audio/video coder and decoder, in order to limit the throughput of streams sent on the network
- A signaling protocol stack, which is responsible for the registering the user in the conferencing system, joining or leaving a conference, negotiation of media formats, etc.

- A media transport protocol stack, which delivers encoded media over the network between endpoints and the middlebox

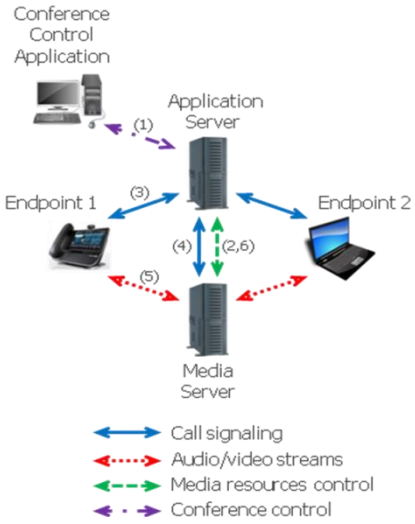



Figure 2.3: An example of a video conference functional architecture

**Conference Control Application:** A Graphic User Interface application, which allows the conference leader to fulfill different operations on the conference, such as reserving media resources for a future conference, inviting new participants or removing existing participants. Web technologies are often used for this type of applications, which can also be integrated with the endpoint software.

**Media Server:** Software component or hardware appliance, which comprises resources for media processing, like:

- Audio mixing, that allows the voices of conference participants to be mixed into one stream, that can be sent to all the participants
- Video mixing, that allows the images of several participants to be shown simultaneously on the screen (see Figure 2.4)

Figure 2.4: Multi image video conference endpoint

Basic media processing facilities, like media mixing, can be integrated into endpoints and can thus be used in peer-to-peer mode. The use of a centralized Media Server gives some advantages like:

- Media trans-coding if media is encoded by different endpoints in incompatible formats
- Generating additional media streams based on the conference ones, for example for recording

**Application server:** A software component or hardware appliance, which plays the central management role between other functional parts of the video conferencing solution (Endpoints, Conference Control Application and Media Server). Its functionality encompasses:

- Localization of Endpoints (Endpoints register in the Application Server so their network locations are known) and management of Call Signaling sessions
- Conference management: processing the requests of the conference leader made with the Conference Control Application (inviting/removing users, etc.) and translating them to Call Signaling session commands towards respective Endpoints
- Media Server management: based on the logic of the conference Application Server, the Media Server applies different media treatment to the conference participants, like playing a voice message to the participants, recording the session, etc.

Having management functions centralized allows the conference to continue smoothly even while some Endpoints leave the conference --- which is hardly possible in the case when management logic resides on one of the Endpoints. Furthermore, centralized management facilitates integration with different enterprise software, like corporate directory with information about employees, shared calendars, etc.

Application and Media Servers can be combined in one box with specifically selected hardware optimized for delivering high quality audio/video experience.

#### 2.1.2.2. Workflow example

The dynamic view of the architecture, presented in Figure 2.3, is demonstrated with the scenario below. The technologies and the protocols, used for this demonstration, are quite typical for videoconferencing, deployed in modern enterprises: SIP (Session Initiation Protocol)[10] is used for call signaling, RTP (Real-time Transfer Protocol)[9] is used for streaming media on the network and MSML (Media Server Markup Language)[15] for media resources control. All these technologies are highlighted below. The scenario, depicted in Figure 2.3 follows several steps:

1. The conference leader creates a conference using the Conference Control Application (step 1).

2. The Application Server sends a MSML command to the Media Server with instructions on how the conference must be configured and which resources are needed (step 2).

3. At this stage, Endpoint1 wants to join the conference. It sends the SIP request to the Application Server to join the video conference (step 3). The request includes information about the media session parameters.

4. The Application Server forwards the request towards the Media Server (step 4), which in turn sends a SIP answer to Endpoint1, with its own media session parameters.

5. A connection between Endpoint 1 and the Media Server is now opened (step 5). This is a direct RTP session as the Application Server does not process media streams. The connection between Endpoint1 and the Media Server is operational but nothing is transported yet, as we need to attach this session to a source of media in the Media Server (for instance to a video mixing function).

6. The Application Server sends a MSML command, which allows the connection of Endpoint1 to be attached to the videoconference session (step 6). From this moment, the media flows generated by Endpoint1 will be mixed with the streams of other participants.

If another endpoint (Endpoint2) joins the conference, the procedure will be exactly the same as described in steps 3-6 above.

### 2.1.3. Video coding

#### 2.1.3.1. Why video coding

End user's device for capturing video (i.e. web camera integrated into laptop) produces raw (uncompressed) digital video stream. Video processing (i.e. video mixer in media server) and video rendering (i.e. video conferencing endpoint) also require uncompressed digital video streams. However, raw digital video streams are usually too heavy (i.e. they consume too much bandwidth) to be sent through the network, so they should be compressed.

Video encoding is a process of converting raw digital video to a compressed format, video decoding is the opposite process. A hardware or software component that fulfills encoding and decoding is called "codec" (which is a concatenation of "coder" and "decoder") [1].

The format of the compressed streams normally conforms to some video compression standards. The standards typically define lossy compression, meaning that the compressed stream loses some of the original information present in the raw stream. As a result, compressed/decompressed streams have lower quality than the original ones.

### 2.1.3.2. Types of video coding

The codecs differ by the quantity of the data, needed to transport the video stream (which is called “bitrate”), the complexity of the encoding and decoding algorithms, robustness to data losses and errors, which occur when the stream traverses the network, end-to-end delay, and a lot of other parameters.

User endpoints vary in their capabilities to accept and process video streams. These differences can be explained by:

- Different bandwidth capabilities
- Different decoding complexity and power constraints

For example, a specialized hardware based video conferencing endpoint, which is normally installed in a meeting room, is typically able to process high quality video streams. However, a participant using a smart phone is only able to process low quality streams using small bitrates. Generally, this problem is resolved by a “Transforming middlebox”, which can adjust streams to the recipients’ needs (see Figure 2.5).

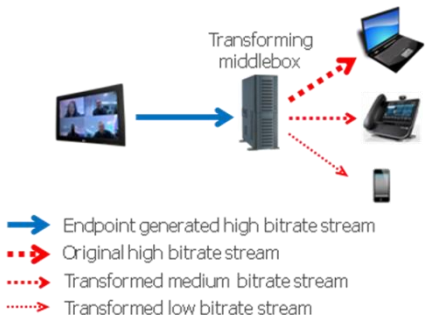



Figure 2.5: Logic of transforming middlebox

Scalable Video Coding (SVC) is another approach, allowing different types of devices to participate in the same video conference. With SVC, a high-quality video stream is divided into several layers of quality. For instance in Figure 2.6, the three layers are sent on the network. The mobile terminal (with poor network reception) will only receive the base layer, which corresponds to the lowest video quality. The other terminals, which can benefit from a better network throughput and/or CPU power, can receive additional layers (on top of the base layer) in order to get a better video quality.

The advantage of this technique is that processing at the SVC middlebox is extremely light, as the middlebox just needs to filter the different built-in layers, and processing of the content of the video stream is not required.

One of the following methods can be used to build the different layers:

- Temporal scalability (frame rate): a low frame rate is used for the base layer, while additional frames are added on advanced layers, providing more fluidity to the video.

- Spatial scalability (picture size): the base layer has a low image resolution, while advanced layers add additional pixels increasing it.
- Quality scalability (coding quality): the coding quality corresponds to the number of bits associated with each pixel. The base layer is coded with the low coding quality, advanced layers with the better one.

It is also possible to combine several of the above scalability techniques.

The flexibility of SVC comes at a price, since the layer encoding adds a bandwidth overhead of roughly 10% - 20%, as compared with a non SVC stream of the same quality.

Unfortunately, SVC technique is not currently fully standardized, so implementations of different vendors are not compatible with each other, except for the base low bitrate layer, which is coded as standard stream (and can thus be decoded by decoders which understand only standard coding).

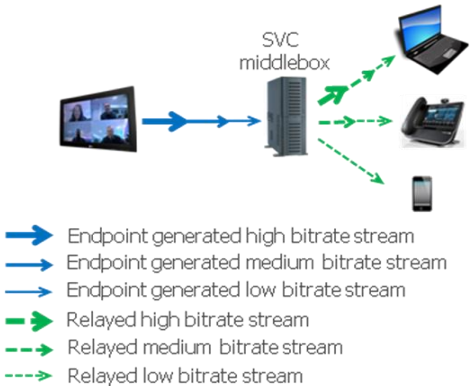



Figure 2.6: Logic of SVC middlebox

In order to overcome this obstacle, another method called Simulcast Video Coding was proposed. Simulcast Video Coding is the parallel encoding of multiple independent video streams with different quality strategies (see Figure 2.7). Each endpoint in the video conference chooses the most appropriate stream which it can process. This allows traditional endpoint, which doesn't support Scalable Video Coding technology to participate in the conference with the appropriate quality level (compatible with their bandwidth and CPU limitations).

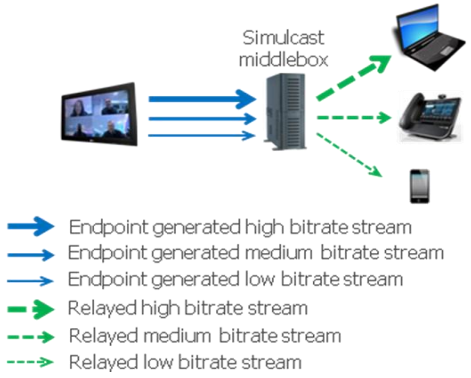



Figure 2.7: Logic of Simulcast middlebox

In Table 2.1 traditional, scalable and simulcast coding methods are compared by the following parameters:

- upstream bandwidth: the bandwidth needed to pass the client streams to the middlebox
- middlebox processing load: the amount of computations, the middlebox needs to execute in order to prepare the resulting streams for the clients
- downstream bandwidth: the bandwidth needed to pass the streams, prepared by the middlebox, to the client
- interoperability: to which extent the standard clients, which don't support a given technology, are able to receive the streams, coded by using this technology

Table 2.1: Comparison Of Coding Methods

| | **Upstream bandwidth** | **Middlebox processing load** | **Downstream bandwidth** | **Interoperability** |
|---|---|---|---|---|
| **Standard Coding** | Low | High | Low | High |
| **SVC** | Low (with small overhead) | Low | Low (with small overhead) | Low (except base layer) |
| **Simulcast** | High | Low | Low | High |

### 2.1.3.3. Codecs

Several codecs are used in modern IP video conferencing. They are divided into several families.

#### 2.1.3.3.1. H.264/H.265

H.264 and H.265 are the codecs standardized by ITU-T and ISO.

H.264 Advanced Video Coding (AVC) [2] was standardized in 2003. It provides a standard (non scalable) encoding with different quality levels, associated with the different sets of constraints imposed by decoder performance. The level defines the maximum picture resolution, frame rate and bitrate, that a decoder may use.

The H.264 standard was designed to be used by a wide variety of video applications, such as video conferencing, mobile video and high definition broadcast. Different types of target applications are addressed by profiles, which represent sets of coding tools and algorithms, used by the specific application (independently from the levels). Videoconferencing is typically based on the so called Baseline Profile (BP) or Constrained Baseline Profile (CBP).

H.264 is widely accepted in all areas and implemented in both hardware and software. H.264 is protected by a group of patents, which are managed by a holding of patent holders called “MPEG-LA”.

H.264 Scalable Video Coding (SVC) [3] was standardized in 2007 and provides H.264 implementation of Scalable Video Coding.

H.265 High Efficiency Video Coding (HEVC) [4] is a successor to H.264 AVC. It was standardized in 2013. H.265 generally doubles the data compression ratio compared to H.264 AVC at the same level of video quality or, if used at the same bitrate, it substantially improves quality. H.265 SHVC (HEVC Scalability Extension) provides implementation of Scalable Video Coding technique and was published in 2015.

#### 2.1.3.3.2. VP8/VP9/VP10

VP8, VP9 and VP10 are owned by Google. In 2010 Google exposed the source code of VP8 [5] under a 3-clause BSD license and the VP8 bitstream format is published by IETF. VP8 only supports temporal scalability.

VP9 [6] is a successor of VP8. It is also open and royalty free. Its bitstream description was published by IETF in 2013. The main improvement over VP8 is close to that of H.265 vs. H.264 – roughly half of bitrate is needed to deliver the same video quality. Scalable Video Coding version of VP9 is under development. VP10 is the latest evolution of this family which is in the early stage of development.

#### 2.1.3.3.3. NETVC

In 2015 an Internet Video Codec (NETVC) working group was created at IETF with the target to produce a high-quality video codec meeting following requirements:

- competitive in terms of performance with best-of-the-breed existing codecs
- optimized for use in interactive web applications
- patent and royalty free allowing wide implementation and deployment

As for the mid of 2016 two codecs were submitted to IETF NETVC group: Daala and Thor.

Daala [7] is a free open source codec under development by Xiph.Org Foundation. The codec is developed based on the new principles compared to existing widely adopted codecs, which will allow avoiding patent infringement. The ideas of the codec are covered by some patents which are freely licensed to everybody.

Thor [8] is a free open source codec under development by Cisco. The target is to propose a codec of moderate complexity to allow real-time implementation in software on common hardware, as well as new hardware designs. Thor is based on technologies used in currently widespread standards.

#### 2.1.3.3.4. Alliance for Open Media

The Alliance for Open Media [37] was founded by leading media related companies in 2015. The first target of the alliance is developing a new open royalty-free video codec specification and open-source implementation. VP10, Daala and Thor are considered for the development of this new codec.

#### 2.1.3.3.5. Other codecs

In existing deployments, a wide variety of other codecs is present such as legacy codecs (H.261, H.263, …) and non-standard private codecs (Microsoft RTV, …).

### 2.1.4. Video processing

Given the set of video streams produced by conference participants' endpoints, the conferencing software needs to apply necessary processing in order to guarantee that all participants receive the streams that they are able to render. Processing generally consists of two parts: video presentation and video transformation.

#### 2.1.4.1. Video presentation

Video presentation combines the streams, generated by the participants, in order to propose necessary user experience to stream recipients. Video presentation takes place in the middlebox or in the recipient endpoint. Today several types of user experience can be offered, depending on the capabilities of conferencing hardware/software and on the type of the conference considered.

#### 2.1.4.1.1. Video mixing (Continuous presence)

Continuous presence mode is the most common method used in virtual meetings. Usually in this mode the screen is split into one large and several smaller surrounding windows. The conferencing software sends the video of the current speaker to the large window and other participants to the small ones. It's also possible to use equal windows for all the participants. If the number of participants is too large to show them all, only the latest speakers are displayed.

#### 2.1.4.1.2. Video switching

Voice switch mode has only one window to which the conferencing software switches the current speaker.

#### 2.1.4.1.3. Lecturer mode

In lecturer mode, the lecturer is shown all the time in the sole window. This mode is used for lectures and presentations.

#### 2.1.4.1.4. Chair mode

In chair mode, a human moderator manually controls who "owns the floor", that is who can speak and who is shown on the screen at any given time.

#### 2.1.4.1.5. Augmented reality

In augmented reality mode, participants are put into virtual meeting room by replacing background, and some additional video effects (like manipulation of the objects) can be added. At the present time, this mode is considered as experimental and is not widely implemented in the commercial products.

### 2.1.4.2. Video transformation

Video transformation is needed in order to adjust streams to the receivers' needs in the case when they can't be accepted in the original form. Video conferencing middlebox can process video streams on the level of stream content or on the level of stream packets, as explained hereafter.

#### 2.1.4.2.1. Content transformation

Content processing means that some changes are introduced to the content of the video stream. This type of processing requires two-step. The first step is decoding, that is the stream encoded in original format is transferred to an uncompressed format. The second step is re-encoding, that is the uncompressed stream is encoded in a new format, taking into account the necessary changes, which should be introduced to the stream. During this two-step process the quality of the video stream suffers as lossy codecs are used in videoconferencing.

Content processing includes:

- TransCoding: change codec format in the case when the consumer doesn't use the same codec as the producer
- TransScaling: change the video frame size in the case when the receiver can't process big frames
- TransFrameRating: decrease video frame rate in the case when the receiver can't process too frequent frames
- TransBitRating: decrease the video codec bitrate which is the result of a decreased picture quality (i.e. bit per pixel).

The last three techniques are used in the case of scarce receiver resources or available network bandwidth.

#### 2.1.4.2.2. Packet transformation

Packet processing implies that the middlebox processes IP packets without decoding/encoding the stream. Such processing contains:

- Packet filtering
- Packet forwarding

- Packet header correction

Packet processing mode can be used only when all the endpoints are compatible in the terms of codecs and their configuration, as packet processing server only decides which streams should be sent to each participant, filters necessary packets, changes header if necessary and forwards the packets to the respective endpoints. This technique is used extensively in Selective Forwarding Unit (which is described in section VI.C).

### 2.1.5. Protocols

The protocols, used in videoconferencing systems, can be considered on three levels:

- Media plane
- Signaling plane
- Media resources control

#### 2.1.5.1. Media plane

The media plane (or data plane) consists of a set of protocols, used for transportation of audio and video streams on an IP packet network. Video conferences use Real-time Transport Protocol (RTP) [9] as a means of delivery of audio and video between endpoints and middleboxes. RTP is an application layer protocol based on UDP. The specificity of real-time audio and video favors the speed of delivery of the packets over reliability.

TCP is generally used to get a reliable transfer, and is used for video streaming for instance. However, retransmission mechanisms of TCP introduce additional delay and jitter, which significantly lowers the quality of real-time interactive media sessions. That's why UDP is generally preferred for video conferencing. However, UDP itself is not sufficient and RTP provides facilities for jitter compensation and detection of out of sequence arrival in data, which are common during transmissions on an IP network. This is ensured by the addition of timestamps and sequence numbers.

RTP is used with its pair protocol "RTP Control Protocol" (RTCP), which provides statistics on the Quality of Service (QoS) and control information for an RTP session.

To overcome potential packet loss, some advanced technologies may be used in addition to RTP.

**Packet Loss Concealment (PLC)** is a technique to mask the fact that some video stream packets are lost, corrupted or arrived too late to be rendered. PLC uses info from neighboring parts to the lost segment of the frame, and/or previous frames and future frames, in order to estimate the lost content.

**Forward Error Correction (FEC)** is a technique, which adds redundant information to the video stream that can be used in order to recover lost packets of the stream.

**Packet Retransmission (RTX)** is a technique for retransmitting lost RTP packets by the source. This approach has limited use as it adds end-to-end delay, which consists of the time to make a request for retransmission and the time for a retransmitted packet to be delivered to a destination. Good Quality of Experience (QoE) [30] prescribes maximum one way delay of 150 ms in order to provide good media quality. If the resulting delay, introduced by packet retransmission, is bigger – this technique can't be employed.

#### 2.1.5.2. Signaling plane

The signaling plane contains the protocols, that negotiate the creation/modification/termination of calls between the endpoints and the middleboxes. There exist both standard as well as proprietary signaling protocols.

Session Initiation Protocol (SIP) [10] is a text based protocol standardized by IETF. Its design is close to HTTP. Nowadays SIP is the main standard protocol for new developments in video conferencing area. Session Description protocol (SDP) is used by SIP as a means to exchange media related information, like the list of supported codecs, etc.

H.323 [11] is a binary protocol standardized by ITU-T. Before the wide adoption of SIP, it was the main protocol for videoconferencing, so many existing videoconferencing installations are still based on H.323. Currently, a lot of equipment supports dual SIP/H.323 stack.

Jingle [12] is an extension of eXtensible Messaging and Presence Protocol (XMPP), originally developed for chat services, which provides peer-to-peer signaling for multimedia sessions. It was developed by Google and the XMPP Standards Foundation, and used in a list of products, first of all open sourced.

Many well known products use their own proprietary signaling, for example Skype.

Network Announcement (NETANN) [13] provides a user with a possibility to call a conference (i.e. a virtual meeting room) using SIP. NETANN provides a way to use standard SIP messages, which were initially designed to locate and call a user, in order to locate and invoke a conferencing service. NETANN covers only very basic functionalities, not allowing rich conferencing user experience.

In the scope of the SIPPING working group, the IETF presented a more advanced conference framework, based on SIP and described in [14]. The framework defines the logical entities and terminology used for conferencing. By the way, it was stated that while some conference management requirements can be implemented with SIP, some can't be implemented. Hence additional means are needed, as presented in the next section.

#### 2.1.5.3. Media resources control

Media resource signaling takes place between the application server and the media server functions, in order to provide advanced conference control, such as:

- in-conference user interaction, like playing a voice message to the participants
- managing sub-conferences
- recording
- modification of the volume of a participant
- muting a participant
- conference event reporting (new participant is added, etc.)

There have been efforts to standardize the centralized control of a video conference at the IETF XCON working group [17]. There has also been work to standardize the control of a media server at the IETF MEDIACTRL working group [18]. However, these approaches are not widely implemented in the commercial products.

Media Server Markup Language (MSML) [15] is a XML based language, which is used to control conferencing features, such as video layout and audio mixing, configure media streams, create sidebar conferences or personal mixes. MSML was described by IETF in 2010.

Media Server Control Markup Language (MSCML) [16] is another XML based language, which also provides features to manage a conference similar to MSML. MSCML was described by IETF in 2006. MSCML and NETANN are related languages, as MSCML is an extension of NETANN.

In both cases, the XML messages are exchanged using the SIP protocol. Both MSML and MSCML are royalty-free and well adopted by the industry.

There also exists a “JSR 309: Media Server Control API” [19], which exposes media server control concepts in a form of Java API.

Finally, it’s worth noting that the IETF CLUE working group (ControLling mUltiple streams for tElepresence) [20] has been created to develop a standardized approach to control immersive telepresence systems (see section VII.A). The management of such a system is much more complex, with its large number of screens and cameras, and requires specific exchange of capabilities and layout of the meeting rooms (e.g. spatial relationships of cameras, displays and microphones).

#### 2.1.5.4. Protocols security

Network connections, used by video conferences, in particular when used over the Internet, are vulnerable to different security threats. Besides well known threats, such as Denial of Service (DoS) attacks, there exist specific security attacks, associated with the protocols listed above.

**Signaling/media eavesdropping:** this is an interception of signaling messages or media packets and extracting their content, which allows an attacker to know who is in communication with whom and what they are talking about.

**Signaling/media spoofing:** if signaling or media messages are intercepted, and their content is extracted, it becomes possible for an attacker to substitute the original participant with another one and to send false media streams to the conference participants — with eventually offending content for instance.

In order to avoid these problems, the video conferencing connections should be secured with the following requirements:

- authentication: in order to insure that each participant know with whom exactly it talks to
- encryption: packets should be encrypted in order to make it impossible to read the contents
- integrity: packets should be resistant to any changes introduced by an attacker

Secure version of SIP is called SIPS and consists in SIP over TLS, which gives SIP all necessary characteristics.

The media plane of video conferences can't be protected by TLS because it uses UDP (while TLS is based on TCP). The Secure RTP (SRTP) protocol has thus been defined at IETF [21]. SRTP requires participants to exchange cryptographical keys, and several mechanisms have been proposed: ZRTP [22], MIKEY [23], SDES [24] and DTLS-SRTP [25].

#### 2.1.5.5. Protocols network border traversal

Enterprise network borders are normally protected by a firewall, which provides Network Address Translation (NAT) and traffic filtering functions. Firewalls pose certain problems for both video conferencing signaling and media traffic, which can't bypass the network border.

Special protocols were introduced for traversing a NAT device. Session Traversal Utilities for NAT (STUN) protocol [26] is used by an endpoint to determine public IP address and port allocated to it by a NAT device. Traversal Using Relays around NAT (TURN) protocol [27] is used, when knowledge of a public IP address is not sufficient in order to traverse a NAT device, and an external server, which relays media streams is needed. TURN allows an endpoint to control the operation of such a relay.

STUN and TURN are combined in Interactive Connectivity Establishment (ICE) protocol [28], which enables the procedure of discovery and exchange of the info, which is needed to establish a media connection in the presence of NAT.

The filtering function for video conferencing is usually executed by a Session Border Controller (SBC). SBC is deployed at the network border and provides firewall function for

both signaling and media traffic. SBC can provide a video transformation function (see section IV.B) as well.

#### 2.1.5.6. Protocols compatibility

One of the main problems of the industry has been incompatibility of the solutions of different vendors for a long time. Signaling protocols as well as configuration of codecs in use were implemented differently within the limits of the respective standards.

Several industry organizations, like International Multimedia Telecommunications Consortium (IMTC) [34] and International IP Interconnection Forum (i3forum) [35] unite leading video conferencing players in order to eliminate discrepancies in protocols implementation.

### 2.1.6. Types of media servers

Media servers, used in videoconferencing, can be of several types, reflecting their design and functionality.

#### 2.1.6.1. Multipoint Control Unit

Multipoint Control Unit (MCU) [29] is a hardware appliance or software component, which provides both video presentation and video transformation functionalities. It means that it is capable of providing any type of stream presentation (mixing, switching, other types of presentation, requiring stream content processing) as well as stream modification (trans-coding, etc). An MCU can also be referred to as a ”conference bridge”.

The standard MCU functionality is to decode all incoming media streams, compose a particular stream for each conference participant and encode these streams to send them through the network. MCU performance is counted in ports, one port being capable to receive one video stream.

Traditionally, hardware MCUs utilize Digital Signal Processors (DSP) in order to make operations over video streams more efficient. Evidently, hardware MCUs are not easily scalable, as if you need more ports you need to buy more physical instances of the hardware.

A Software MCU is a software component, which is deployed on standard general purpose server hardware. Latest advancements in CPU technologies and associated programming libraries, such as the Intel Integrated Performance Primitives (Intel IPP) library [36], have made it possible for general purpose CPUs to be used for video processing. Contrary to a hardware MCU, Software MCU provides efficient scalability and flexibility. Software MCUs can be deployed in the cloud with the access to the exact amount of hardware resources, which are needed. That makes the operation of adding and removing of MCU ports very simple. Furthermore, Software MCUs benefit from easy operations of update and upgrade compared to hardware MCUs.

Hardware and software MCU logic is depicted in Figure 2.8.


Figure 2.8: Logic of MCU

### 2.1.6.2. Video gateway

A video gateway is a hardware appliance or a software component, which provides video transformation functions, being a mediator between two incompatible video conferencing systems (e.g. from two different vendors). The incompatibility may be on signaling and/or media level. Such a way video gateway can provide:

- signaling: connecting different protocols (e.g. H.323/SIP), or aligning different flavors of SIP
- transport: changing the transport protocol (TCP/UDP)
- media: trans-coding, adjusting modes of the same codec
- security: interworking between secure side (SRTP) and insecure side (RTP)

Video gateway can also be used for some additional services, which are based on information passing through it, for example for conference recording.

### 2.1.6.3. Selective Forwarding Unit

Selective Forwarding Unit (SFU) [29] is a software component, which relays the received video streams to the different conference participants. For that purpose SFU can apply packet transformation functionalities. As SFU doesn't provide content processing, it doesn't consume a lot of CPU cycles as compared to MCU.

Often SFU may receive different resolutions of the stream from one conference participant based on SVC or simulcast technology, so it should apply some logic in order to decide which resolution to send to each recipient. This logic can use the physical characteristics of the recipient in order to choose appropriate resolution. It may also be used to detect active speaker among conference participants in order to send her stream with better resolution than others.

SFU can be based on:

- standard video coding, in the case when the endpoints are all supporting this encoding
- Scalable Video Coding

- Simulcast Video Coding

In Figure 2.9 an SFU based conference with three participants is depicted. All the three participants send two levels of SVC coding. Two non-active participants, using devices with relatively small screens, receive one high quality stream with the active speaker and one low quality stream with the other non-active participant. Active speaker, using a device with a big screen, is able to benefit from receiving all the streams in high quality level (even if these are two stream of non speaking participants).

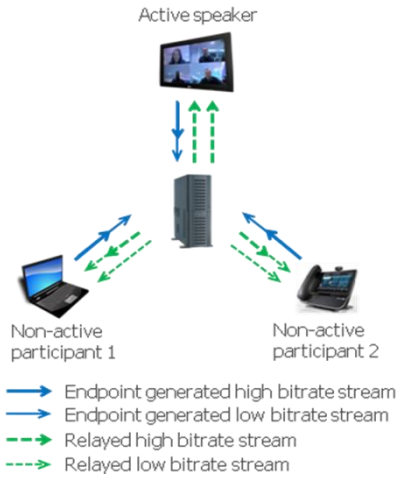



Figure 2.9: Logic of SFU based on Scalable Video Coding

### 2.1.7. Types of clients

#### 2.1.7.1. Immersive telepresence

Immersive telepresence is the most advanced (and also the most expensive) form of the video conferencing, providing the users with the experience that all the conference participants are located in the same room. Purposely designed conferencing rooms equipped with several large screens and a variety of cameras are usually used for immersive telepresence.

#### 2.1.7.2. Hardware clients

Video conferencing hardware clients are dedicated appliance, that can be used in meeting rooms or at working desks in order to participate in a conference. Usually such a client comprises a webcam and a loudspeaker. Also it can be equipped with a screen or it can use a 3rd party screen.

#### 2.1.7.3. Video desk phones

Modern desk phones have big screens, which can host an image produced by a videoconference. Integrated or external webcam is used with such a solution.

#### 2.1.7.4. PC clients

PC clients are installed on user's computers and use webcam and screen of this computer. The quality of video stream which is produced/consumed by such a client depends on CPU of the hosting computer.

#### 2.1.7.5. Browser clients

The software clients, which don't need installation, but are instead downloaded from a web site and ready to be used immediately. This category is now gaining traction especially after the introduction of the WebRTC technology. WebRTC standardizes interaction between conferencing web application and web browser, based on JavaScript API [32]. Also it provides a full media stack (protocols, codecs, …) [33].

#### 2.1.7.6. Mobile clients

With the rise of smart phones and tablets, mobile devices have become a popular platform for video conferencing clients. Taking into account that such clients are battery powered, the codec implementation shout pay special attention to energy consumption.

### 2.1.8. Topologies

#### 2.1.8.1. Dedicated on-premises

Videoconference systems are traditionally implemented as a hardware appliance, deployed in the LAN of an enterprise, or as a software component, installed on one or more servers in the data center of an enterprise.

#### 2.1.8.2. Hosted

A Hosted deployment means that hardware appliance is physically located in the data center of a service provider and operated by its IT team (see Figure 2.10). Deployed hardware can be used by several clients or it can be locked to a single client in order to provide more security. Hosted deployment should not be confused with Cloud, as the former still uses hardware appliances, and, as such, doesn't provide easy scalability.

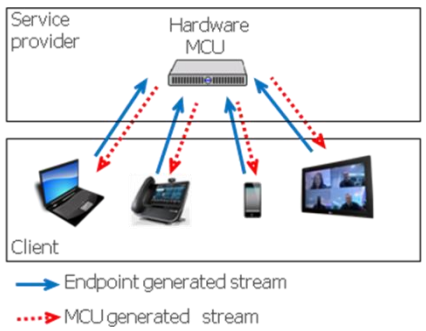



Figure 2.10: Hosted hardware MCU

### 2.1.8.3. Cloud

A Cloud deployment refers to a software component in the form of a virtual machine, which is physically located in the data center of a service provider (see Figure 2.11). All types of middleboxes (MCU, SFU, gateway) can be deployed in the Cloud. Service is offered either as a subscription (customer pays for each registered user) or on a usage basis (cost per port per minute).

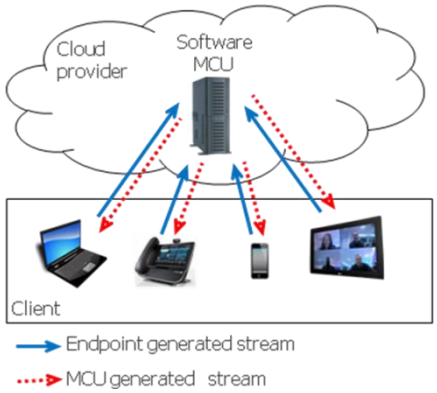



Figure 2.11: Software MCU in the Cloud

A Cloud deployment has, as defined in [31], the following properties:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

### 2.1.8.4. Enterprise Desktop Grid

An Enterprise Desktop Grid deployment means that video processing software component is deployed on the grid of usual office hardware, that is desktop and laptop PCs (see Figure 2.12). The machines, which are not occupied with other activities, are selected to host videoconferencing processing tasks. All types of middleboxes (MCU, SFU, gateway) can be deployed on Enterprise Desktop Grid. This is an experimental type of deployment, which is under investigation in this thesis.

The different approaches to video conference deployments are compared in Table 2.2. The following criteria are considered:

- Expenditure : how the conference service is paid for
- Support: who is responsible for day-by-day operations and administration

- Data: where and how the data, related to the conferencing solution (recorded video, logs, …), is stored and who can have access to it
- Elasticity: to which extent the conferencing solution can be scaled


Figure 2.12: Video conferencing deployed on Enterprise Desktop Grid

Table 2.2: Comparison Of Deployment Approaches

| | **Hosted** | **Cloud** | **On-premises hardware** | **On-premises software** | **Enterprise Desktop Grid** |
|---|---|---|---|---|---|
| **Expenditure** | Operational (OPEX) | | Capital (CAPEX) | Capital (CAPEX), general purpose servers are used | Free, as already existing hardware is reused. |
| **Support** | Service provider | | Local IT | | |
| **Data** | Questionable when contract is stopped, potentially can be accessed by 3rd parties | | Always accessible by owner, not accessible by 3rd parties | | |
| **Elasticity** | Limited by hardware MCU | Not limited | Limited by hardware MCU | Limited by servers in datacenter | Depends on load of the grid |

### 2.1.8.5. Conference endpoints

The video presentation logic can be located in the conference endpoints. In this case a conference is organized without a middlebox, in a peer-to-peer fashion (see Figure 2.13).

Advantages:

- No additional resources are required

Disadvantages:

- It is not possible to support many participants, as any endpoint is required to keep a separate video connection with all the other participants.
- It is not possible to support heterogeneous endpoints: all the participants must use the same type of client software or hardware so that codecs and their parameters (resolution, frame rate, etc) match. There is actually no gateway functionality in peer-to-peer topology, by construction.
- Absence of enterprise conferencing features: enterprise use cases often require more advanced features like conference recording, possibility to add/remove participants by a moderator, etc., which is difficult to offer in this distributed environment.


Figure 2.13: Full mesh conference topology

In order to optimize video streams, multicast technology can be utilized. Two types of multicast technologies are available: Application Level Multicast (ALM), where the distribution of flows towards the different receivers is performed by endpoints themselves (i.e. at the application layer) (see Figure 2.14), or directly by the network (IP Multicast).

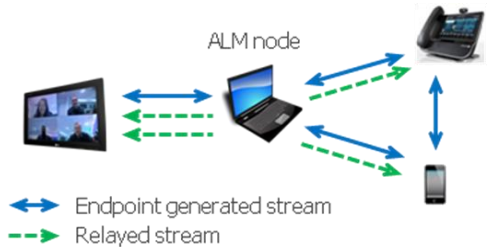



Figure 2.14: Conference topology based on Application Level Multicast

IP multicast has, however, not gained traction, because:

- Internet Service Providers usually don't offer IP multicast services, by reason of management complexity, scalability concerns and security risks, associated with this technology. As a consequence, this technology is not available when the videoconference takes place over the Internet (involving participants from different networks).

- In an intranet context, multicast is much widely available. However, in the enterprise environment, additional services are needed, which can only be provided by a middlebox. ALM is thus more natural in this context.

Selective Forwarding Units can also be used in the context, when video processing is integrated in the endpoints:

Advantages:

- Often the most cost-effective choice

Disadvantages:

- No trans-coding of different video and audio codecs
- Limited number of participants (usually less than 6)
- Bandwidth requirements aggregate to increase demand on the organizer's network

#### 2.1.8.6. Hybrid topologies

Different types of hybrid topologies are possible. Particularly on-premises + cloud deployments are popular nowadays. If a company has got through mergers and acquisitions a fleet of several MCUs of different vendors, it can use a cloud multi-vendor interoperability service, which allows using these MCUs in the same conference. Or a company can decide to physically protect its data in the form of recordings of the meetings, and place the recording and storage server in their data center while the conferences consume resources in the cloud.

### 2.1.9. Current trends

Currently, several trends of video conferencing evolution can be identified.

In the codec area, there is a significant move towards open royalty-free codecs, which is supported by a great number of industry players through "Alliance for Open Media" and IETF NETVC working group. Further increase of resolution from today's standards HD (1280×720) and Full HD (1920×1080) towards 4k (4096×2180) also gains traction for big screens.

Pure software technologies, on which both server infrastructure and clients are based, are extensively developed nowadays.

On the server side they allow virtualization, which is a necessary requirement in order to place video conferencing to the cloud and expose it in the form of VCaaS (Video Conferencing as a Service).

At the same time, we can wait evolutions of a Fog approach [38], which moves processing burden from the cloud to the resources in proximity of the end users, for

example to the network edge devices. The Fog technique can potentially spare needed WAN bandwidth as well as reduce end-to-end delay.

On the client side, software technologies allow using commodity hardware (PC screens, integrated cameras and microphones), instead of purchasing expensive one-purpose videoconferencing endpoints. This trend is especially important with wide adoption of mobile clients (smart phones, tablets) as video conferencing endpoints for the workers, which are not attached to a fixed working place.

WebRTC technology becomes very popular, as it enables browser based clients, which are accessed by the users as standard web pages. This removes the necessity of client installation, significantly improving end user experience. Being oriented to numerous web developers, the technology is very popular and benefits from wide support of the community.

In today's world, video conferencing is less considered as a stand-alone technology and more as a part of "collaboration", which is a wider notion, comprising different means of meeting organization like instant messaging and screen sharing as well as integration of video conferencing functionality to business and vertical applications, which prescribes its efficient usage in the context of business and industrial processes of the organization.

## 2.2. Academic research of video conferencing

Generally, distributed on end user devices video conferencing is well researched in two forms:

- Application Level Multicast
- Peer-To-Peer

The literature on these two approaches is analyzed in the following sections, the state of research as of 2011 is done in [43].

Active efforts were also dedicated to research of video streaming, supported by the distributed "helpers". The respective articles are analyzed below as well.

Energy consumption of media processing operations, which is very important for battery greedy mobile devices is considered in [83], [84], [85].

Cloud-assisted video conferencing is analyzed in [90], [91], [103], [104].

Conferences, based on IP Multicast technology, are regarded in [96-98], [107].

Scalable Video Coding based conferences are described in [99], [105], [108-112].

### 2.2.1. Application Level Multicast

In particular, the structure and efficiency of Application Level Multicast trees for video distribution are well elaborated in the literature.

For instance, the construction of optimal overlay video distribution trees has been well investigated. In [44], it is considered as a utility maximization problem in the context of multi-rate video coding, where the utility function is represented by the Peak-Signal-to-Noise-Ratio (PSNR) of the decoded video.

In [45], the authors propose to establish an overlay network of specialized Media Control Servers responsible for a transmission path selection. User Endpoints can also be engaged in this activity if they are capable enough. Several methods for selecting a media transmission path are considered. All these works concentrate on video routing but video processing (e.g. mixing, trans-coding) is not considered.

[77] proposes a method for creating ALM trees used for video conferencing. The suggested approach considers the number of clients, the available bandwidth, the available RAM, processor speed of the peers and hop distance to sort the various nodes that participate in a conference. The most powerful participants play the role of LAN gateways through which the video stream is passed from the source to the participants of the given LAN which is applicable to our situation too. The subject of this paper is video routing, video mixing is not analyzed.

[78] proposes bandwidth fair algorithm for ALM distribution tree construction and a new protocol for ALM packet replication and distribution. Neither video mixing nor media negotiation is analyzed.

[79] describes an ALM approach to video conferencing and lists some criteria for the ALM tree building. DNS server is used as a controller of the tree. Some criteria interesting in scope of our research are mentioned, for example choosing a peer in the same LAN. The tree is used for video distribution only, the question of video mixing is not covered.

[80] describes decentralized P2P conferencing and presents an ALM-based architecture that is enhanced by leveraging conferencing specific behavior of the participating peers with different capabilities. The approach proposes to use the most powerful nodes as media mixers. The idea is pretty close to the one I’m going to research but the paper describes only audio streams without analyzing video specific behavior which in nature very different from the audio.

### 2.2.2. Peer-to-Peer

In the context of Peer-to-Peer networking, P2P conferencing systems have been considered, such as in [46]. The proposed method takes into account the capability of peers while assigning video distribution tasks. Here, again, no media processing is considered.

The proposed in [76] approach improves P2P video conferencing introducing “helpers” which address bandwidth deficiency inherent to pure P2P conferencing. So the authors study the optimal bandwidth sharing in multiparty P2P video-conferencing systems with the helpers. Video mixing is out of the scope of this paper, only video routing is considered.

A pretty close to our topic paper was presented in 2015 by Hossain and Khan [88]. They describe an election protocol for a P2P conference, which allows Multipoint Control Units to dynamically migrate among peers when new peer joins or leaves. But their algorithms take into account only video traffic, minimizing the necessary amount of hops as the metric. While it will work for geographically distributed servers, for which we consider the resources of the platform itself are big enough, it's not applicable for our problem, where we should base our calculations not only on network characteristics, but also on platform ones, like CPU load.

P2P based conferences are also researched in [92-95], [100-102], [106].

### 2.2.3. Video streaming

The strategy of using distributed "helpers" can be applied to video streaming functionality, where helpers are used to compose the final stream.

[81] considers a multi-source streaming network with distributed mixers, where streams originated from multiple sources are mixed before presented to distributed users. The proposed approach minimizes the worst-case delay from the source to users via the mixers. An adaptive and distributed protocol called "MixN-Stream" is proposed, which continuously reduces the network diameter in the presence of churns.

[82] investigates the question of composable services in media gateways. A user can request a computation to be performed on a set of media streams. The system then distributes the computation over multiple gateways for execution. An algorithm for decomposing the computation into sub-computations and an application-level protocol that locates appropriate media gateways to run these sub-computations is presented.

Different technological approaches to distributed video streaming are also considered in [86], [87], [89].

Generally the proposed methods are applied to broadcasted video streaming and doesn't take into account interactive video conference requirements, first of all strict end to end delay. Such a way they are not directly applicable to our problematic.

### 2.2.4. Conclusion

To summarize, no results were found on the issue of distributing of video processing in the context of enterprise desktop grid. However, we believe that nowadays it is crucial to have a video processing entity in the middle of a conference, given the diversity of codecs and features, implemented by different vendors and often non-compatible with each other.

# Chapter 3

# Node selection algorithms

## 3.1. General system description

The DGC system consists of a set of media servers (tackling video processing tasks), distributed on a cluster of ordinary office equipment (PCs, laptops, etc).

The description of the DGC system is based on two main notions: *Tasks* and *Processors*.

*Task* is a media related activity, traditionally provided by a MCU or a software media server: video mixing, video switching, trans-coding, trans-scaling or other manipulations on video streams. Audio streams traditionally accompany video streams and are just mixed together by the same media server. For example, the *Task* associated to the video conference depicted in Figure 1.2 is video mixing of 4 streams into a single stream (with typically an emphasis on the current speaker) and potentially transcoding, in case of incompatible codecs of the user terminals.

*Processor* is a media server deployed on a general purpose hardware platform like a PC. Users can turn on/off their PCs and launch third party applications consuming CPU power as well as start/stop calls and conferences randomly. This results in unpredictability of the sets of *Tasks* and *Processors*, which should be taken into account by the system.

The main logic of the proposed architecture is to distribute and, if necessary, to redistribute *Tasks* on *Processors* taking into account changes in the set of *Tasks*, set of *Processors* and external constraints (that are enumerated below). The result of distribution should be “optimal” under some conditions.

## 3.2. Optimization criteria

Optimization criteria can be divided into two sets: the Network and Platform ones.

Network criteria that should be taken into account include:

1) *WAN bandwidth* consumed by a *Task*. The goal is to try to economize WAN bandwidth which is generally chargeable (as opposed to LAN bandwidth which is considered free of charge and thus not controlled).

2) *End-to-end delay* between endpoints. Delay is very important characteristic representing the level of QoE, as large delay makes an interactive conversation difficult or even impossible.

Platform criteria are related to the *Processors* that are available in the system:

*1) Network connectivity:* takes into account whether the platform uses wired or wireless (Wi-Fi) network connectivity. Wired connections generally provide more stability and less delay, which makes them preferable for interactive video communications as compared to wireless connections.

*2) Power supply:* takes into account whether the platform is powered by electric circuit or by its battery. It is particularly important as video processing operations are very CPU intensive.

*3) Resource sharing:* takes into account whether the platform hosts only *Processor* or it is shared with other user activities unrelated to the DGC system. This criterion gives the preference to the platforms where no users applications run. Such a preference gives stability to the DGC system, as CPU consumption is more predictable. At the same time, this prevents from deploying *Tasks* on the platforms actively employed by the users in order not to disturb them.

*4) CPU load:* provides the estimation of planned CPU load after a given *Task* is deployed on a given *Processor*. The system tries to distribute *Tasks* in such a way that CPU load on each platform would be minimized in order to secure the processes if their demand of CPU resources were to increase.

Potentially other optimization criteria could be easily integrated into the logic of *Task* distribution, based on the utilization experience of the real implementation of the DGC system.

All the optimization criteria are different in their importance, leading us to choose a MADM (Multi-Attribute Decision Making) approach, where each criterion will be associated with a weight. Before presenting the chosen MADM algorithm, we need to analyze the system behavior.

## 3.3. State Change Events

The changes in the system that require *Task* distribution or, under some circumstances, redistribution form a queue of State Change Events (SCE). There are several types of such events:

*1) Task* is added: for example a new conference is created and video mixing *Task* should be distributed to some *Processor*.

*2) Task* is removed: a *Task* deployed on some *Processor* is no more needed in the system. Removal of a *Task* may cause some *Tasks* redistribution for overall optimization.

*3) Processor* is added: a new *Processor* is added to the system. Some *Tasks* may be redistributed taking added *Processor* into account.

*4) Processor* is removed: a *Processor* is removed from the system. If any *Tasks* were deployed on this *Processor* then these *Tasks* should be redistributed to other *Processors.*

5) Value of an optimization criterion is changed: the configuration of the system has been modified, for example the network connectivity of a *Processor* has been altered from Wireline to Wireless. In this case, some *Tasks* redistribution may be performed, if needed.

State Change Events queue functions as a FIFO (First In, First Out) queue with strict priorities. The priorities are the following (from highest to lowest):

*1) Processor* is removed (with *Tasks* deployed on it).

*2)* CPU load is increased such a way that it may block the *Tasks* execution.

*3) Task* is removed, *Processor* is added, *Processor* is removed (with no *Tasks* on it), CPU load is decreased, other (i.e. not CPU load) optimization criteria are changed.

*4) Task* is added.

The highest priority is set to "*Processor* is removed" SCE as some *Tasks* are blocked in this situation, leading to bad user experience. The second priority is set to "CPU load is increased" SCE for the same reason of potentially worsening user experience. The "*Task* is added" SCE has the lowest priority as it has sense to take into account all changes in the system before distributing a new *Task* in order to avoid consecutive redistributions.

During State Change Events processing *Tasks* are deployed/redeployed one by one. That is once a decision is taken about deployment/redeployment, the *Task* is actually deployed/redeployed and the system waits until the *Task* starts consuming CPU cycles (the system is then in a stable state). Then deployment/redeployment of the next *Task* can be processed based on the new value of CPU load.

## 3.4. MADM approach

The target of the optimization procedure is to calculate a numerical estimation value, taking into account the variety of criteria, which would allow comparing potential distributions of the *Tasks* on the different *Processors*. The *Task* will then be deployed on the *Processor* with the optimal target value. A purposely-created MADM method using "context aware normalization" is applied to calculate the estimation value.

One of the specificities of MADM algorithms is the requirement to normalize values of attributes. In the general case, no assumptions can be made on them. From the state-of-the-art [41], several methods of normalization of values in MADM matrix are well-known ($s_{ij}$ are elements of original matrix):

- $\alpha_{ij}=S_{ij}/\sum(S_{ij})$
- $\alpha_{ij}=S_{ij}/\max(S_{ij})$
- $\alpha_{ij}=(S_{ij}-\min S_{ij}) / (\max S_{ij}-\min S_{ij})$
- $\alpha_{ij}=S_{ij}/\sqrt{\sum(S_{ij})^2}$

In all these methods, normalization process involves operations on attributes of all the possible cases (e.g. sum of values, maximum value, etc). It means that when the set of alternatives is changed (i.e. *Processor* is added/removed or value of optimization criterion is changed), the normalization process should be re-executed. Taking into account dynamic nature of the DGC system, it would be highly desirable to be able to make the necessary computations for each alternative independently from the other ones. Such an approach allows applying the MADM procedure only when a *Processor* is added to the system or a specific attribute of the *Processor* is changed. In other words, no computation would be needed for a given *Processor*, whatever are the changes applied to other *Processors*.

In the specific context of our problem, we introduce in the following a simple normalization process that eliminates such dependencies. We actually know the nature of all the attributes, their optimal values and practical limitations. Let's consider the MADM attributes used in the DGC system.

*1) End-to-end Delay*: The optimal value of delay is evidently 0 (if we count it in milliseconds). For normalized delay value we use the following expression:

*normalized_delay = real_delay / delay_threshold*

Delay threshold can be defined in different ways. For example, ITU-T recommendation G.114 [47] can be used. This recommendation states acceptable voice delays in interactive applications. Delay smaller than 150 ms is considered as acceptable, bigger than 400 ms as inacceptable and values between the two imply that there will be some quality issues. Such a way we can set the value 400 as *delay_threshold* and it will mean that all delays more than 400 ms will not be distinguishable from each other as all the *normalized_delay* values bigger than 1 are rounded to 1.

*2) Used WAN Bandwidth*: The optimal theoretical value for WAN Bandwidth (WBW) used by a *Task* is also 0, it's achieved when all the endpoints and the *Processor* are in the same LAN. For normalized WAN bandwidth value we will consider the following expression:

*normalized_WBW = real_WBW / max_WBW*

The value of *max_WBW* can be taken as the sum of bandwidths of all the video streams of a given *Task*. This value is known at the moment of creation of the *Task*.

*3) Platform criteria*: All platform criteria, except CPU load (i.e. network connectivity, power supply, resource sharing) are binary by their nature that is they are "positive" or "negative". Positives are:

- Network connectivity = wireline
- Power supply = electric circuit
- Resource sharing = dedicated

Negatives are:

- Network connectivity = wireless
- Power supply = battery
- Resource sharing = shared

For conformity we set “0” value for positive case and “1” value for negative case. Thanks to that we have the situation when ideal variant of attribute value is “0” and normalization is not needed.

CPU load criterion value is presented in percents of CPU usage taken after a given *Task* were deployed on a given *Processor*. It gives the optimal theoretical value of “0” (while not achievable in practice) and the worst value of “100”. For normalized CPU load value we will consider the following expression:

*normalized_CPU_load = real_ CPU_load / 100*

CPU load criterion has some particularities, which are described below.

All optimization criteria used in calculations are represented in Table 3.1.

Table 3.1: Optimization Criteria

| **Attribute name** | **Ideal value** | **Worst value** | **Normalization divisor** |
|---|---|---|---|
| **End-to-end delay** | *0* | ∞ | *400, if delay <= 400*<br>*delay, if delay > 400* |
| **WAN bandwidth** | *0* | *Sum of all video streams* | *Sum of all video streams* |
| **Network connectivity** | *0* | *1* | *Not needed* |
| **Power supply** | *0* | *1* | *Not needed* |
| **Resource sharing** | *0* | *1* | *Not needed* |
| **CPU load** | *0* | *100* | *100* |

The problem is finally formulated as an inverse normalized Simple Additive Weighting (SAW) method [42]:

$$\boldsymbol{OR_j = \sum_{i=1}^{M} w_i a_{ij} \Big/ M} \qquad (1)$$

where:

$OR_j$ : *Objective Result* for *Processor j*

$w_i$ : weight of criterion *i*

$a_{ij}$ : normalized value of criterion *i* on *Processor j*

*M* : number of criteria

Inverse SAW means that we need to take as the result the smallest $OR_j$ instead of the biggest one. Normalized SAW means that $OR_j$ value is in the range [0, 1]. This formula implies that $OR_j$ is calculated for each *Processor* independently and only when the *Processor* appears in the system or value of an optimization criterion is changed.

## 3.5. CPU load criterion

The *Processor* CPU load is different from other optimization criteria as its value changes continuously compared to rather rare changes of other criteria values. From the point of view of the practical implementation, this means that we can calculate $OR_j$ for all the criteria except CPU load and store it in a cache while we need to observe the value of CPU load in real time.

Furthermore, in order to be able to compute the impact of a particular type of *Task* on the CPU load of a particular *Processor,* a preliminary *Qualification process* is needed. *Qualification process* means the vendor of the DGC system installs a *Processor* on a particular platform, then all types of *Tasks* are executed and CPU consumption level is collected and stored. Then these pre-collected values can be used as an estimation of necessary CPU resource when the DGC system simulates distribution of a *Task* on a *Processor* installed on the qualified platform at a customer site.

Table 3.2: Example of Load Qualification Matrix

| **Platform**<br>**Type of Task** | PC Intel core Quad2 CPU @ 2.40 GHz | IPhone 5S | Alcatel-Lucent 8082 deskphone |
|---|---|---|---|
| Mixing 3 H264 streams in HD | *X1% CPU Load* | *Not possible* | *Not possible* |
| Trans-coding H264 to VP8 in HD | *X2% CPU Load* | *Not possible* | *Y1% CPU Load* |
| Trans-scaling H264 from HD to CIF | *X3% CPU Load* | *Not possible* | *Y2% CPU Load* |

Comparing columns of Load Qualification Matrix we can understand which of two Tasks is more resource-intensive so we can sort all the Tasks by their CPU consumption – it's needed in some algorithms.

Merging information from Load Qualification Matrix with information about real CPU load of Processors we obtain the Deployment matrix, which is used in some algorithms described further.

Table 3.3: Example of Deployment matrix (bold means real values, not simulated)

| | **Proc1**<br>(Real CPU Load: 70) | **Proc2**<br>(Real CPU Load: 30) | **Proc3**<br>(Real CPU Load: 80) |
|---|---|---|---|
| **Task1**<br>(Deployed on Proc1) | *Load Qualification: 30*<br>*SSR: 40*<br>***RCL: 70***<br>***RDR: 55*** | *Load Qualification: 20*<br>*SSR: 70*<br>*DSR: 50*<br>*FSR: 60* | *Load Qualification: 20*<br>*SSR: 50*<br>*DSR: 100*<br>*FSR: 75* |
| **Task2**<br>(Deployed on Proc1) | *Load Qualification: 40*<br>*SSR: 40*<br>***RCL: 70***<br>***RDR: 55*** | *Load Qualification: 30*<br>*SSR: 50*<br>*DSR: 60*<br>*FSR: 55* | *Load Qualification: 30*<br>*SSR: 50*<br>*DSR: 110*<br>*FSR: 80* |
| **Task3**<br>(Deployed on Proc2) | *Load Qualification: 20*<br>*SSR: 40*<br>*DSR: 90*<br>*FSR: 65* | *Load Qualification: 15*<br>*SSR: 50*<br>***RCL: 30***<br>***RDR: 40*** | *Load Qualification: 15*<br>*SSR: 50*<br>*DSR: 95*<br>*FSR: 72* |

How numbers in the matrix are produced:

Task1 on Proc1 (Real CPU Load: 70)

*Load Qualification: 30* = got by qualification process

*SSR: 40* = calculated by Static Simulation

***RCL: 70*** = 40 [CPU load by other processes] + 30 [Load qualification]

***RDR: 55*** = (40 [SSR] + 70 [RCL]) / 2

Task1 on Proc2 (Real CPU Load: 30)

*Load Qualification: 20* = got by qualification process

*SSR: 70* = calculated by Static Simulation

*DSR: 50* = 20 [RCL] + 30 [Load Qualification]

*FSR: 60* = (70 [SSR] + 50 [DSR]) / 2

## 3.6. MADM Example

### 3.6.1. Description

The proposed approach is illustrated by the following example.

Let's we have 2 Endpoints on 2 sites:

- Endpoint1 on Site1
- Endpoint2 on Site2

These endpoints use different video codecs so for direct communication trans-coding is needed.

Also there are the following Nodes in the system:

- Node1 on Site1: PC with circuit supply using wireline connection
- Node2 on Site1: Laptop with battery supply using Wi-Fi connection
- Node3 on Site3: PC with circuit supply using wireline connection

One way delay between any 2 sites is 30 ms. Delay of trans-coding executed on PC is 5 ms. Delay of trans-coding executed on laptop is 12 ms. Delay of Wi-Fi link is 7 ms. (Note: values are just illustrative, they may differ from values in real life).

Bandwidth consumed by both codecs is 1 mb/s.

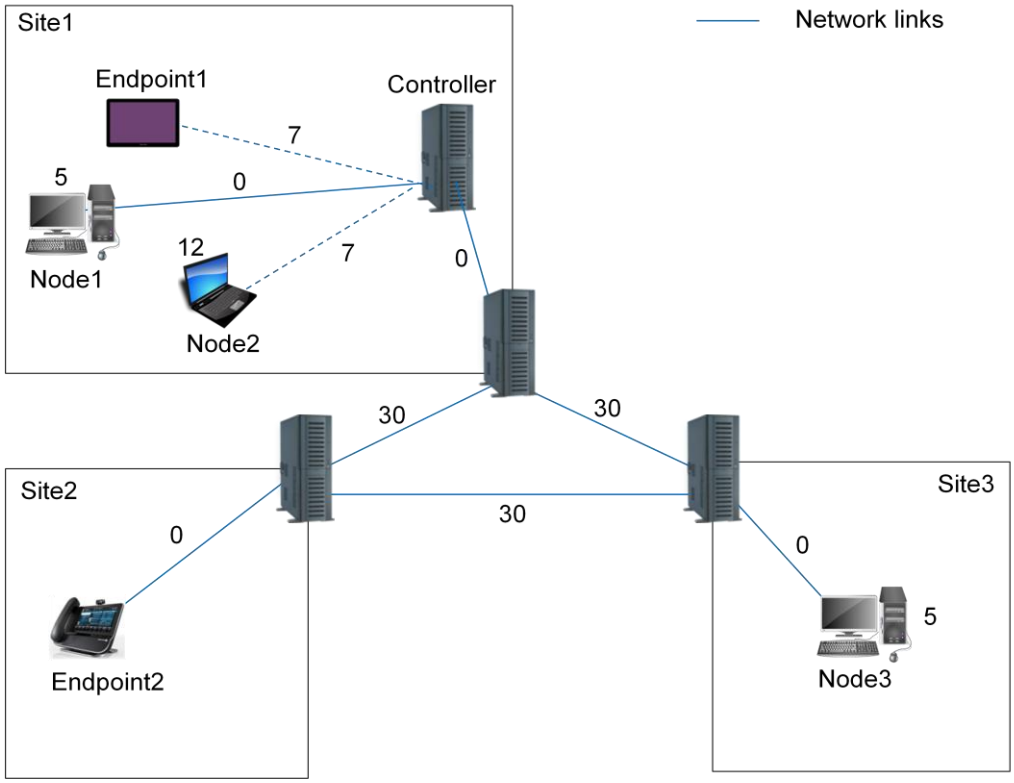



Figure 3.1: Topology fog MADM illustration

### 3.6.2. Calculations

The calculations in MADM are made using the standard decision table (see Figure 3.2):

| | | $A_1$ | ... | $A_n$ |
|---|---|---|---|---|
| $w_1$ | $C_1$ | $a_{11}$ | ... | $a_{1n}$ |
| ... | ... | ... | ... | ... |
| $w_m$ | $C_m$ | $a_{m1}$ | ... | $a_{mn}$ |

- $A_x$ – alternatives, i.e. Processors which can accept the task
- $C_x$ – optimization criteria (end-to-end delay, bandwidth consumption, ...)
- $w_x$ – weights, i.e. the importance of a criterion
- $a_{xy}$ – performace, i.e. the value of a given criterion on a given alternative

Figure 3.2: MADM decision table format

Taking into account our modifications which introduce a predefined norm for each attribute, we add an additional column with the norm.

Let's see which results we will have for described above topology while modifying some values and weights. In bold are the performances which we are going to modify.

Let's for beginning chose the weights that put stress to Bandwidth and Delay, and especially to Power:

| Weight | Attribute | *Node1* | *Node2* | *Node3* | Norm |
|---|---|---|---|---|---|
| **20** | ***Bandwidth*** | 2 | 2 | 4 | 4 |
| **20** | ***Delay*** | 42 | 63 | **72** | 400 |
| **10** | ***Network*** | WIRELINE | WIRELESS | WIRELINE | 1 |
| **40** | ***Power*** | CIRCUIT | **BATTERY** | CIRCUIT | 1 |
| **10** | ***Shared*** | SHARED | DEDICATED | DEDICATED | 1 |
| | | | | | |
| | **Result** | 22 | 63 | 23 | |

We see that Node1 and Node3 are very close, while Node2 is much worse, mostly because it's powered by a battery.

Now we change the type of alimentation of Node2 from Battery to Circuit:

| Weight | Attribute | *Node1* | *Node2* | *Node3* | Norm |
|---|---|---|---|---|---|
| **20** | ***Bandwidth*** | 2 | 2 | 4 | 4 |
| **20** | ***Delay*** | 42 | 63 | **72** | 400 |
| **10** | ***Network*** | WIRELINE | WIRELESS | WIRELINE | 1 |
| **40** | ***Power*** | CIRCUIT | **CIRCUIT** | CIRCUIT | 1 |
| **10** | ***Shared*** | SHARED | DEDICATED | DEDICATED | 1 |
| | | | | | |
| | **Result** | 22 | 23 | 23 | |

The result is that all the three nodes are almost equal.

Now let's imagine that Site3 becomes very far from the other two sites and delay becomes 500 ms:

| Weight | Attribute | *Node1* | *Node2* | *Node3* | Norm |
|---|---|---|---|---|---|
| **20** | ***Bandwidth*** | 2 | 2 | 4 | 4 |
| **20** | ***Delay*** | 42 | 63 | **500** | 400 |
| **10** | ***Network*** | WIRELINE | WIRELESS | WIRELINE | 1 |
| **40** | ***Power*** | CIRCUIT | **CIRCUIT** | CIRCUIT | 1 |
| **10** | ***Shared*** | SHARED | DEDICATED | DEDICATED | 1 |
| | | | | | |
| | **Result** | 22 | 23 | 40 | |

This logically leads to the worsening of the result of the Node3.

Now let's change the weights, putting additional stress to Power and reducing all other weights. In this case the initial version logically has even more expressive results:

| Weight | Attribute | *Node1* | *Node2* | *Node3* | Norm |
|---|---|---|---|---|---|
| **15** | ***Bandwidth*** | 2 | 2 | 4 | 4 |
| **15** | ***Delay*** | 42 | 63 | **72** | 400 |
| **5** | ***Network*** | WIRELINE | WIRELESS | WIRELINE | 1 |
| **60** | ***Power*** | CIRCUIT | **BATTERY** | CIRCUIT | 1 |
| **5** | ***Shared*** | SHARED | DEDICATED | DEDICATED | 1 |
| | | | | | |
| | **Result** | 13 | 74 | 17 | |

With such disposition the battery powered node is significantly worse that the other two. Even if we put again the big delay for the Node3 it can't change the situation radically:

| Weight | Attribute | *Node1* | *Node2* | *Node3* | Norm |
|---|---|---|---|---|---|
| **15** | ***Bandwidth*** | 2 | 2 | 4 | 4 |
| **15** | ***Delay*** | 42 | 63 | **500** | 400 |
| **5** | ***Network*** | WIRELINE | WIRELESS | WIRELINE | 1 |
| **60** | ***Power*** | CIRCUIT | **BATTERY** | CIRCUIT | 1 |
| **5** | ***Shared*** | SHARED | DEDICATED | DEDICATED | 1 |
| | | | | | |
| | **Result** | 13 | 74 | 30 | |

The Node2 remains the worst one.

Such a way we see how changes in weights and attribute values can affect significantly the decision which node to select for *Task* deployment.

## 3.7. Redeployment Penalty

In order to improve user's perception of the conference, we introduce the rate of *Task* redeployments, defined as the number of times the existing *Task* is redeployed from one *Processor* to another. Redeployments will lead to interruptions in the media streams, so it's highly desirable to minimize them.

A special parameter "Redeployment Penalty" is employed by the algorithms in order to regulate the number of potential redeployments. When a *Processor* is considered as a candidate to host a *Task*, the gain in *Objective Result* must be above this threshold in order for the *Task* to be redeployed to this *Processor*. Note also that a simple hysteresis mechanism is applied on the CPU load to prevent from cyclic redeployments when the CPU load changes sporadically.

## 3.8. Algorithms of Processor selection

### 3.8.1. Notation conventions for algorithms

A *Task* is denoted by *T*, a set of *Tasks* is denoted by *{T}*. For identification purpose, the *Task I* is denoted by $T_i$. Similarly are used *P*, *{P}* and $P_i$ for *Processors*.

*SR* stands for *Simulation Result* and equals *OR* (see (1)) resulting from the simulation process, that is from calculating *OR* for a *Task* in application to a *Processor* but the *Task* is not really deployed on this *Processor*. *DR* stands for *Deployment Result* which is *OR* of a given *Task* really deployed on a given *Processor* and consuming CPU cycles of this *Processor*. *DR* values are stored in the system.

*SR* of *Task* $T_i$ simulated on *Processor* $P_j$ is denoted by $SR_{ij}$, *DR* uses the same notation for deployed *Tasks*.

Redeployment Penalty, denoted by $\lambda$, is a constant in the range [0, 1] which reflects the threshold of the difference between *SR* and *DR* of a *Task* that triggers its redeployment. That is if $DR_{im} - SR_{in} > \lambda$, then *Task* $T_i$ is redeployed from *Processor* $P_m$ to *Processor* $P_n$.

### 3.8.2. Task is added

1. Compute *SR* for added *Task* $T_a$ on all the *Processors* $\{P_j\}$ registered in the system. The result is $\{SR_{aj}\}$.
2. Deploy *Task* $T_a$ on *Processor* $P_d$ for which $SR_{ad} = min_j \{SR_{aj}\}$. If no *Processor* is able to accept *Task* $T_a$ then the *Task* is considered as lost.

### 3.8.3. Task is removed

Generally to remove a *Task* we need only to remove appropriate objects from the program. No *Tasks* redeployment is triggered by *Task* removal itself. However, CPU resources are freed and this fact will be notified to the DGC system, which might in turn trigger redeployment (see algorithm *IV.G "Processor CPU load is decreased"*).

### 3.8.4. Processor is added

1. Compute $\{SR_{ia}\}$ for all the *Tasks* in the system $\{T_i\}$ simulated on added *Processor* $P_a$.
2. Retrieve from the system $\{DR_{ij}\}$ for all the *Tasks* $\{T_i\}$ deployed on the respective *Processors* $\{P_j\}$.
3. Calculate $\{D_{ia}\} = \{DR_{ij} - SR_{ia}\}$ for *Tasks* $\{T_i\}$.
4. Let's $D_{xa} = max_i\{D_{ia}\}$ which corresponds to *Task* $T_x$. If $D_{xa} > \lambda$ then redeploy *Task* $T_x$ from *Processor* $P_j$ on which it's currently deployed to *Processor* $P_a$.
5. If *Task* $T_x$ was redeployed on step 4 then wait until system is in stable state and return to step 1.

### 3.8.5. Processor is removed

1. Using the results of *Qualification process*, select the most resource-intensive *Task* $T_x$ from the set $\{T_f\}$ of *Tasks* deployed on the removed *Processor* $P_r$. Compute $\{SR_{xj}\}$ for *Task* $T_x$ simulated on all the *Processors* $\{P_j\}$ registered in the system.
2. Let's $SR_{xy} = min_j\{SR_{xj}\}$ which corresponds to *Processor* $P_y$. Redeploy *Task* $T_x$ from *Processor* $P_r$ to *Processor* $P_y$. Remove *Task* $T_x$ from $\{T_f\}$. If no *Processor* is able to accept *Task* $T_x$ then the *Task* is considered as lost.
3. If $\{T_f\}$ is not empty then wait until the system is in stable state and return to step 1.

### 3.8.6. Processor CPU load is increased

1. Retrieve from the system $\{DR_{ic}\}$ for *Tasks* $\{T_i\}$ deployed on *Processor* $P_c$ for which CPU load is increased.
2. Compute $\{SR_{ij}\}$ for *Tasks* $\{T_i\}$ simulated on all *Processors* registered in the system $\{P_j\}$ except $P_c$.
3. Calculate $\{D_{ij}\} = \{DR_{ic} - SR_{ij}\}$ for *Tasks* $\{T_i\}$.
4. Let's $D_{xy} = max_{ij}\{D_{ij}\}$ which corresponds to *Task* $T_x$ and *Processor* $P_y$. If $D_{xy} > \lambda$ then redeploy *Task* $T_x$ from *Processor* $P_c$ to *Processor* $P_y$.
5. If *Task* $T_x$ was redeployed on step 4 then wait until system is in stable state and return to step 1.

### 3.8.7. Processor CPU load is decreased

1. Retrieve from the system $\{DR_{ij}\}$ for *Tasks* $\{T_i\}$ on respective *Processors* $\{P_j\}$ on which they are deployed.
2. Compute $\{SR_{ic}\}$ for *Tasks* $\{T_i\}$ on *Processor* $P_c$ for which CPU load is decreased.
3. Calculate $\{D_{ic}\} = \{DR_{ij} - SR_{ic}\}$ for *Tasks* $\{T_i\}$.
4. Let's $D_{xc} = max_i\{D_{ic}\}$ which corresponds to *Task* $T_x$. If $D_{xc} > \lambda$ then redeploy *Task* $T_x$ from *Processor* $P_j$ to *Processor* $P_c$.
5. If *Task* $T_x$ was redeployed on step 4 then wait until system is in stable state and return to step 1.

## 3.9. Possible extensions

### 3.9.1. Media stream relays

One of the main requirements that are posed to DGC system is sparing WAN bandwidth. To reduce WAN bandwidth the situations when several media streams delivering the same content, originating on one Network Location and terminating on another Network Location should be optimized. Such situations are possible when several stream consumers, located on one site, use the same output of a Processor located on another site. For example 4 persons participate in a conference: users A and B on Site 1, users C and D on Site 2. Processor 1 is located on Site 1 and plays the role of video mixer. Each conference participant receives the image of all 4 participants. Such a way without a Relay the resulting video stream is sent twice over the WAN from Site 1 to Site 2.

The role of Relay is to accept the stream and fork it to all necessary recipients. In our case with a Relay deployed on Site 2 the resulting stream is sent only once and then forked by the Relay for Users C and D.

Without a Relay the streams are sent twice over WAN:

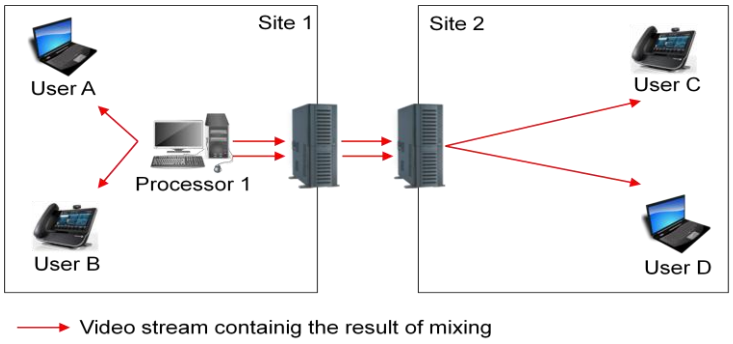



Figure 3.3: Topology without Relay

With a Relay only one stream is sent over WAN and then forked locally:

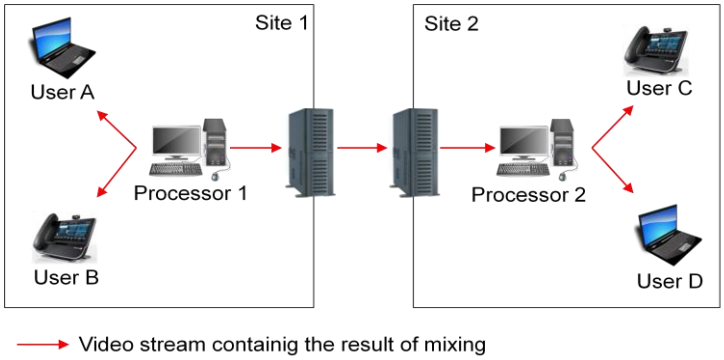



Figure 3.4: Topology with Relay

Trigger: a new Task $T_a$ is added to the system and should be distributed to some Processor taking into account that additional Processors can be used as Relays.

Basic algorithm:

1. For each Processor $P_j$ on which Task $T_a$ can be deployed consider respective topology of video streams distribution and determine all pairs of Network Locations in a form $NetLocPair_k = [Src_k, Dst_k]$ where at least two streams from Network Location 'Src' to Network Location 'Dst' have the same contents. As a result a set $\{NetLocPair_{jk}\}$ is formed for each Processor $P_j$.
2. For each pair $NetLocPair_{jk}$ find the best Relay located in Network Location $Dst_k$. Best relay is determined by application of static and dynamic simulations but from the set of optimization policies used for static simulation the network policies are removed as the scope of a Relay is one Network Location. As a result the set of Relays $\{P_{jk}\}$ is defined for each Processor $P_j$.
3. Execute static simulation, dynamic simulation and calculate Full Simulation Results for Task $T_a$ on all Processors $\{P_j\}$ taking into account spared WAN bandwidth thanks to respective set of Relays $\{P_{jk}\}$. As a result the set $\{FSR_{aj}^{relay}\}$ is formed.
4. Task $T_a$ is deployed on Processor $P_d$ for which $FSR_{ad}^{relay} = \min_j \{FSR_{aj}^{relay}\}$.

### 3.9.2. Taking RTCP feedback into account

The standard way for the sender of a video stream to receive a feedback on how successfully it was received is RTCP feedback described in IETF RFC 3611. Normally video encoder takes into account info on degraded user experience caused by increased delay/jitter/packet loss and reduces size/frame rate/ quality of the video stream.

In DGC system it's also possible to try to resolve the problems with the stream delivery by changing Processor if there are some alternatives. To implement this logic a new type of SCE should be added: "Network problems reported in real-time". When the system receives this SCE via SCE queue then it considers the variants of re-distribution.

The idea is to remove problematic link from network optimization policies, calculate new FSR value and re-deploy Task $T_p$ which experiences problems from the Processor $P_f$ on which it's factually deployed to another Processor according to the new FSR value.

Basic algorithm:

1. Retrieve Real Deployment Result for Task $T_p$ on Processor $P_f$: $RDR_{pf}$.
2. Remove problematic link from network optimization policies.
3. Execute static simulation, dynamic simulation and calculate Full Simulation Results for Task $T_p$ on all Processors $\{P_j\}$ from $\{P\} \setminus P_f$. As a result a set $\{FSR_{pj}\}$ is formed.
4. Select $FSR_{py} = \min_j \{FSR_{pj}\}$ which corresponds to Processor $P_y$.
5. If $(FSR_{py} - RDR_{pf}) <$ DISTURBANCE_RATE then re-deploy Task $T_p$ to Processor $P_y$.

## 3.10. Analysis of the process

### 3.10.1. Scalability

The situation is possible when an enterprise where DGC system is installed is big enough to experience problems in supporting the full Deployment Matrix and using it in all the algorithms. To make DGC system scalable and to limit the amount of necessary calculations the following approach is proposed.

The idea is to limit the number of Processors considered for distribution of a given Task. Using this approach N Processors with smallest SSR values are selected and only these Processors are used in further calculations.

Let's consider the use case of Added Task in the context of Scalability. A new Task $T_a$ is added to the system with big number of Processors and should be distributed to some Processor.

Basic algorithm:

1. Execute static simulation of Task $T_a$ on all the Processors from {P} and select N smallest values of SSR. These values form the set $\{SSR_{aj}^{S}\}$.
2. Select the set of Processors $\{P_j^{S}\}$ corresponding to the values in $\{SSR_{aj}^{S}\}$.
3. Execute dynamic simulation of Task $T_a$ on Processors from $\{ P_j^{S}\}$. As a result a set of Dynamic Simulation Results $\{DSR_{aj}^{S}\}$ is created.
4. A set of Full Simulation Results $\{FSR_{aj}^{S}\}$ is created using elements of $\{SSR_{aj}^{S}\}$ and $\{DSR_{aj}^{S}\}$.
5. Task $T_a$ is deployed on Processor $P_d$ for which $FSR_{ad}^{S} = \min_j \{FSR_{aj}^{S}\}$.

### 3.10.2. Algorithms complexity

Analysis of algorithms complexity is performed by application of the standard method of loops execution number counting.

Let's introduce the following notation:

- P: number of Processors
- $T_s$: number of Tasks in the system, value can be potentially big
- $T_p$: number of Tasks deployed on one Processor, value is small
- C: number of clients in a given Task
- N: number of Network Locations in the system

The complexities:

- Static Simulation O(1)
- Dynamic Simulation O(1)
- Use case: Task is added O(P)

| | |
|---|---|
| - Use case: Task is removed | O(1) |
| - Use case: Processor is added | $O(T_s^2)$ |
| - Use case: Processor is removed | $O(T_p^2 + P)$ |
| - Use case: CPU load is increased | $O(PT_p^2)$ |
| - Use case: CPU load is decreased | $O(T_p^2)$ |
| - Media Stream Relays | O(P + C + N) |
| - Taking RTCP feedback into account | O(P) |
| - Scalability | O(P) with coefficient lesser than for the use case “Task is added” |

# Chapter 4

# Solution architecture

We consider the static and dynamic aspects of DGC architecture, implemented while using two distinctive approaches.

At first, we see how the DGC system looks like and functions when it's implemented as a standalone system (i.e. in "fog" mode). That is when only the resources available at enterprise premises can be used for organizing a conference. The system based on this approach evidently can't guarantee any predefined SLA (Service Level Agreement) as it depends on quantity of available resources at enterprise premises.

To avoid this limitation, the DGC system can be coupled with Cloud based conferencing, which we consider at second. In this case on-premises resources are utilized at available extent, all the requests beyond this capacity are served by Cloud.

## 4.1. Standalone DGC system

### 4.1.1. DGC architecture

The structure of components of the DGC system is depicted in Figure 4.1.

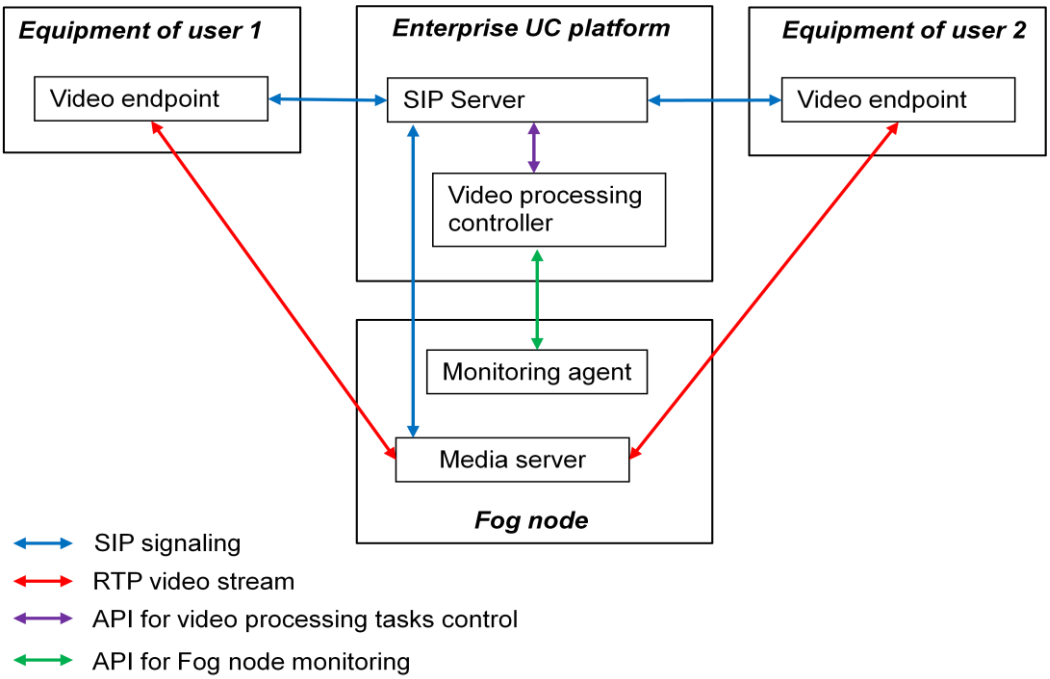



Figure 4.1: Structure of DGC standalone system

Components:

- ***User equipment:*** a device, employed by a user, in order to participate in a conference (PC, laptop, tablet PC, smart phone)
    - *Video endpoint:* a software application capable of treating SIP call signaling and RTP flow delivering audio and video. It's not part of our research - we use existing software.

- ***Enterprise communications platform:*** a centralized dispatcher of clients requests, responsible of deciding which Tasks are executed on which Processors
    - *SIP server:* processes standard telephony/conferencing signaling for call establishing and tearing. It's not part of our research - we use existing software.
    - *Video processing controller:* distributes Tasks and observes status of Nodes

- ***Fog node:*** a hardware platform (PC, laptop) used to host media server
    - *Monitoring agent:* responsible for observing the state of the node, first of all CPU load, and reporting this info to *Video processing controller.* This reporting is combined with Heartbeat functionality (that is with the regular check that the Node is still functioning and not halted). As well it measures delay to different network locations, these results are used by Video Processing Controller in Tasks distribution logic.
    - *Media server:* the component which actually executes media processing tasks like audio/video stream mixing. A general purpose off-the-shelf media server can be used, no development specifically done for DGC is needed. It's not part of our research - we use existing software.

Interfaces:

- ***SIP signaling:*** standard SIP signaling is utilized, no special modifications for DGC are needed

- ***RTP stream:*** standard RTP flows are employed, no special modifications for DGC are needed

- ***API for video processing tasks control:*** through this API SIP server notifies Video Processing Controller about creating/removing/updating of conferences and Video Processing Controller notifies SIP server which manipulations with endpoints should be executes.

- ***API for fog node monitoring:*** through this API Monitoring Agent notifies Video Processing Controller about the fact that a Fog node joins/disjoins the Desktop Grid and regularly updates the info about CPU load as a part of Heartbeat mechanism. Also through this API Video Processing Controller requests Monitoring Agent to measure delay to some network location.

### 4.1.2. Delay estimation

Information about the characteristics of the links between the enterprise sites is critically important for efficient functioning of DGC system. This information comprises several factors:

- delay
- bandwidth
- cost

A possible solution for this problematic is ALTO (Application-Layer Traffic Optimization) [64]. ALTO provides a possibility for a network element to ask a server, which is aware about network characteristics, to get access to this info in convenient form. ALTO provides information in the form of a network map and a cost map.

Network map defines groups of network elements which are close to each other so that they can be considered as a one logical entity. An identifier is assigned to each group. In our context these are nodes and endpoints which are physically located on one enterprise site and connected by LAN.

Cost map provides costs between groups defined in network map. Cost map is a generic notion that can reflect a variety of different link characteristics, like distance between network location expressed in number of hops. Some integer number is used to express the link cost so the costs can be compared with each other.

As of the year 2016 the IETF working group has defined several main ALTO documents and continues working on details. Such a way we have a chance to see the results of this work in production in some time.

While there are no industrial deployments of ALTO services for the moment, we can estimate delay between enterprise sites and remote endpoints using other means. The simplest method is just direct probing. We can ask endpoints to ping different nodes in order to know exact round-trip time.

There also exist an academic research on the topic of delay estimation. The two general approaches can be applied on application layer to our problem:

- Coordinates-based Systems (GNP [65], Vivaldi [66], PIC [67], Sequoia [68], ICS [69], Pharos [70])
- Path Selection Services (IDMaps [71], Meridian [72], OASIS [73])

Comprehensive overview of practical aspects of existing methods of discovering and using network topology information for estimating network delays is done in [113]. Based on this overview it seems that Meridian is the most appropriate as it doesn't need any additional infrastructure and gives exact values instead of estimated.

### 4.1.3. Example of ALTO usage

Let's consider example of integration of ALTO info into our algorithms. Let's we have 2 Endpoints on 2 sites and one remote Endpoint connected to the third site. Three endpoints join the conference. Each participant wants to see two other participants mixed into one image.

There are also 2 Nodes which can be used for video processing on the first and the third sites.

(Note: described video transformations are chosen for demonstration purpose only so the result doesn't necessarily expose the real life transformations required for good end user experience.)

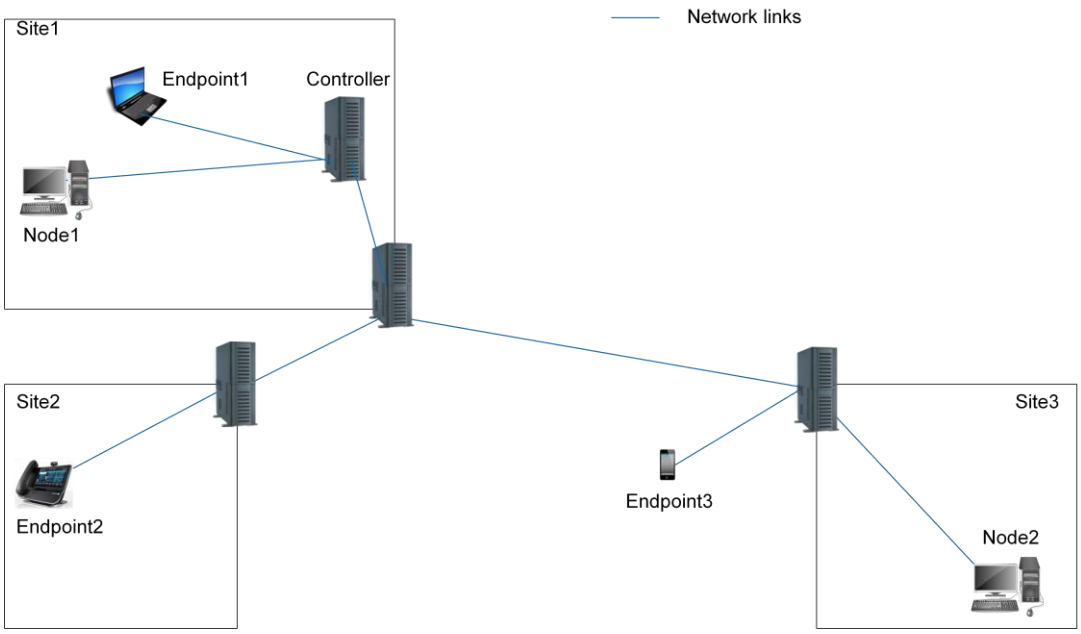



Figure 4.2: Topology for example of ALTO usage

Let's the configuration of endpouints is the following:

Endpoint1

- Produces: H264 HD
- Consumes: H264 VGA

Endpoint2

- Prod: H264 VGA
- Cons: H264 HD

Endpoint3

- Produces: VP8 CIF
- Consumes: VP8 CIF

ALTO Service provides costs between network locations. In our case we are interested in WAN links so network locations are sites and remote devices.

Network map (Identifiers mapped to network locations, in our case sites and remote devices):

- PID1: {Site1 = [Endpoint1, Node1]}
- PID2: {Site2 = [Endpoint2]}
- PID3: {Site3 = [Node2]}
- PID4: {Endpoint3}

Cost map (Integers, reflecting link costs between network locations):

- PID1: {PID1:1, PID2:3, PID3:5, PID4:4}
- PID2: {PID1:3, PID2:1, PID3:6, PID4:4}
- PID3: {PID1:4, PID2:6, PID3:1, PID4:2}

- PID4: {PID1:4, PID2:4, PID3:2}

Then we need to provide Tasks definition, that is which exactly operations on video streams should be executed in order to provide appropriate user experience.

On the streams addressed to Endpoint1 (Ep1):

- Stream from Ep2 must be mixed with stream from Ep3
- Stream from Ep3 must be transcoded VP8->H264, upscaled CIF->VGA and then mixed with stream from Ep2

On the streams addressed to Endpoint2 (Ep2):

- Stream from Ep1 must be mixed with stream from Ep3
- Stream from Ep3 must be transcoded VP8->H264, upscaled CIF->HD and then mixed with stream from Ep1

On the streams addressed to Endpoint3 (Ep3):

- Stream from Ep1 must be transcoded H264->VP8, downscaled HD->CIF and then mixed with stream from Ep2
- Stream from Ep2 must be transcoded H264->VP8, downscaled VGA->CIF and then mixed with stream from Ep1

Taking all the above info we need to distribute these Tasks. Distribution of the tasks is possible with different policies, it depends whether only delay, delay + bandwidth or some other parameters should be optimized. In this example let's consider bandwidth as not lacking/expensive resource so we optimize only delay.

To do that we should avoid extra encodings as the most expensive operation[114]. The best place to fulfil a task in this case is on the consumer immediately before rendering.

In our example only Ep1 can execute necessary stream processing. It means that Ep2 and Ep3 just send their streams directly to Ep1.

Ep2 can't fulfil necessary stream transformations so they are done on Node1 and resulting stream is sent to Ep2.

For Ep3 we need to transcale and downscale streams from Ep1 and Ep2. The best place for it is Site1 as streams from Ep1 and Ep2 are already there. But if Node1 is not capable to fulfil the second task (in parallel with processing streams for Ep2) we need to send streams from Ep1 and Ep2 to Node2 on Site3 and then send resulting stream to Ep3.

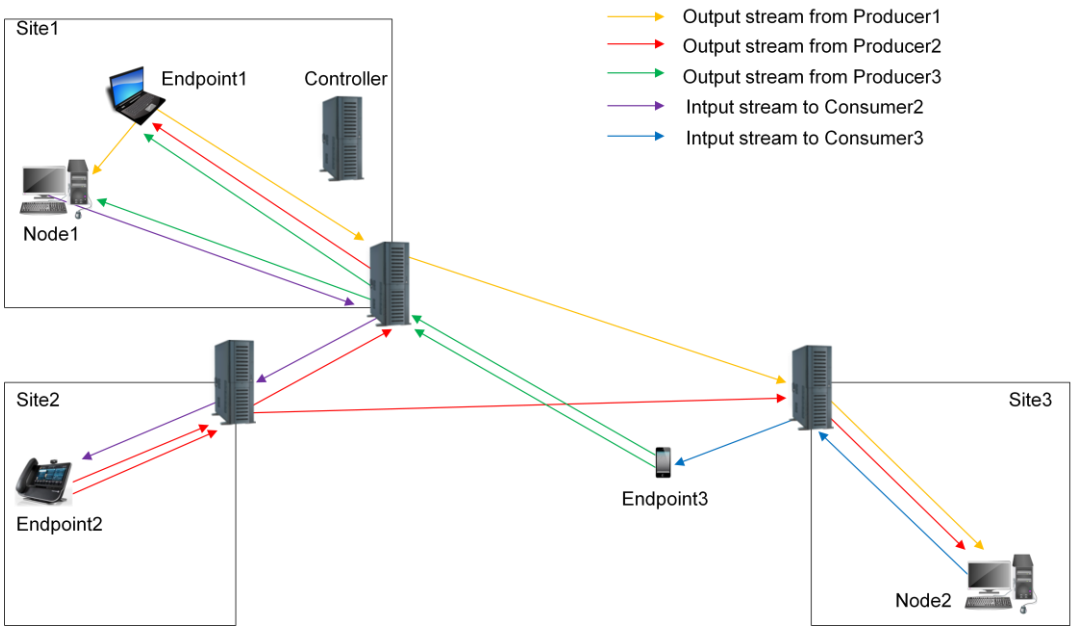



Figure 4.3: Video streams disposition in example of ALTO usage

## 4.2. Cloud integrated DGC system

As it was mentioned in “Introduction”, DGC system itself can’t guarantee appropriate SLA (Service Level Agreement) as its resources are controlled by the end users, and not by the system itself. To resolve this problem DGC system can be combined with the cloud conferencing system in order to provide both SLA and cost benefits at the same time.

In order to get even more benefits, we combine Cloud/Fog approach with the different types of media servers, notably MCU and SFU, which provide different exploitation characteristics. All the possible combinations and circumstances, under which their usage gives the most interest, are considered in this chapter.

Recently the cloud video conferencing has gain traction thanks to a number of useful properties, such as flexibility and pay-per-use model. At the same time, from the point of view of the cloud video conferencing provider, there exists a number of problems with this approach, we mention here two of them:

- Need of significant processing resources in cloud as all calculations are concentrated in one place
- Increased end-to-end delay as data is sent from the client to the cloud provider data center and back, often via slow and unreliable Internet links

The two problems may be resolved by the proper choice of the type of video conference established by the provider. The type may be traditional “MCU” conference which needs additional decoding/encoding on the server and results in greater CPU consumption and end-to-end-delay, or it may be “SFU” conference which doesn’t need video stream processing. At the same time a concept of “Fog” can be engaged in order to choose the type of resources on which media server will be deployed.

Combining dynamically MCU/SFU and Cloud/Fog approaches we can:

- Spare cloud video conferencing provider resources in terms of CPU cycles and network bandwidth consumption
- Significantly reduce end-to-end delay, improving end user QoE

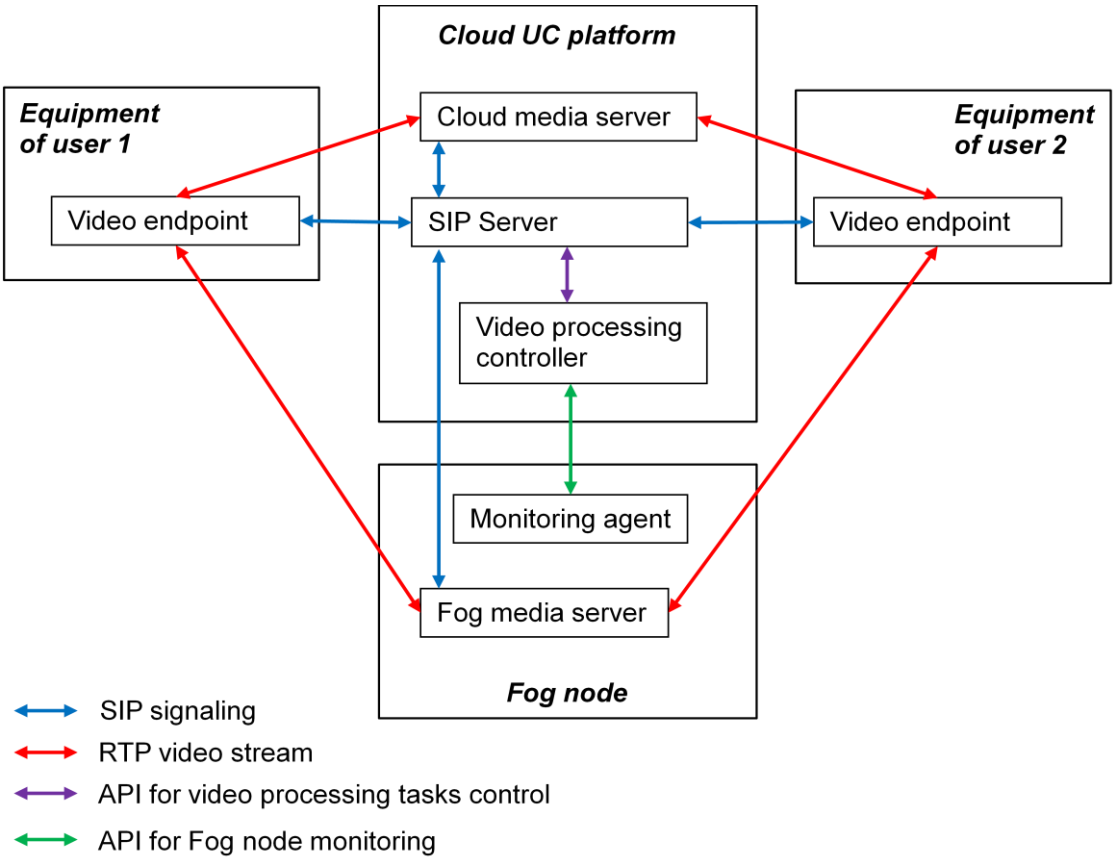



Figure 4.7: Structure of DGC system integrated with Cloud

The idea is the following: we combine before mentioned techniques and obtain four possible approaches to video conferencing. Signaling server can be in Cloud or on-premises. We are talking here only about media server and media flows.

SFU in Fog:

‘+’: No Cloud CPU utilization
‘+’: No WAN utilization
‘+’: No additional decoding/encoding

MCU in Fog:

‘+’: No Cloud CPU utilization
‘+’: No WAN utilization
‘-’: Additional decoding/encoding

SFU in Cloud:

‘+/-’: Moderate Cloud CPU utilization

‘-’: WAN utilization
‘+’: No additional decoding/encoding

MCU in Cloud:
‘-’: Extensive Cloud CPU utilization
‘-’: WAN utilization
‘-’: Additional decoding/encoding

We can see that from the point of view of the both stakeholders (provider and client) the four approaches can be prioritized (from the best to the worst):

1. SFU in Fog
2. MCU in Fog
3. SFU in Cloud
4. MCU in Cloud

Cloud provider keeps its revenue as it controls the overall solution and charges for its usage. Utilization of on-premises fog resources are prescribed by policies which are negotiated between the provider and the client. Policies may vary from permissive (“all not occupied resources may be used for the conferencing”) to restrictive (“client prohibits using any on-premises resources”, which results in pure cloud solution).

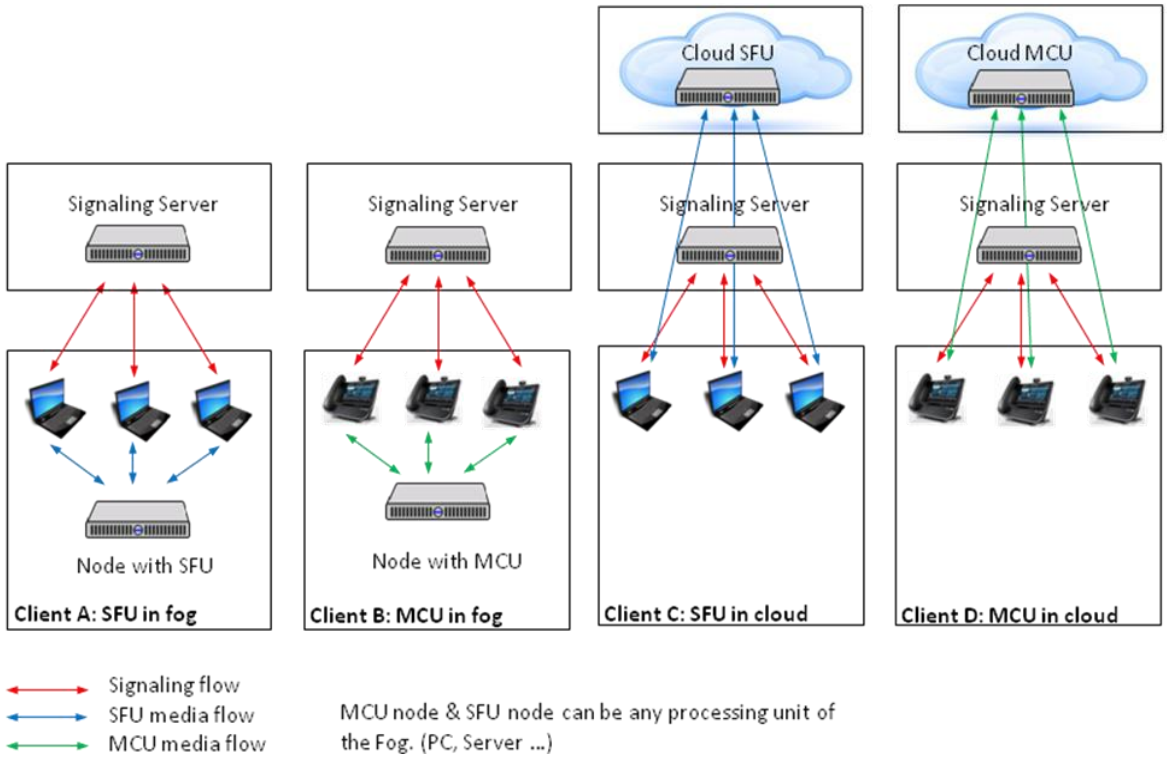



Figure 4.8: Possible types of video conferencing

In Figure 4.8:

- The cloud SFU & the cloud MCU are located in the cloud. The use of these resources will introduce extra charges depending on the policies of the cloud provider.
- The signaling server can be located in the cloud for the cloud based solution using Fog resources.
- The signaling server can be located in the Fog, for the on-premise based solution using cloud based MCU/SFU resources.
- Fog part will not introduce extra charges.

When a new video conference is going to be organized, an algorithm is applied to choose dynamically which type of conference will be performed (SFU or MCU), and where the media server is instantiated (Fog or Cloud).

Media server can be moved dynamically from Fog to Cloud or from Cloud to Fog according to the available resources in the Fog. And it can be switched from SFU mode to MCU mode according to the capacity of the end users equipment.

In the proposed solution SIP signaling and RTP video streams are standard, that is no extensions, designed specifically for our solution, are needed. Video processing controller calculates the possibilities of deploying a given video processing Task to a Fog node or refuses the task if Fog is not capable to accept it. In the case of refusal the Task is deployed to the Cloud.

# Chapter 5

# Evaluation by simulation

## 5.1. Simulation input

The developed task distribution algorithms have been tested using Discrete Event Simulation approach. Discrete Events in the simulation are mapped to State Change Events reflecting the typical load of an enterprise communication system.

The characteristics of a typical enterprise communications system were collected using the platform, deployed at a real enterprise (Alcatel-Lucent Enterprise) for its internal use. In Figures 5.1 and 5.2 the collected statistics of call arrivals is depicted in two formats. We can clearly see peaks and dips during working hours, peaks are aligned with the beginning of hours (XX:00) and half-hours (XX:30) when scheduled meetings are normally planned.

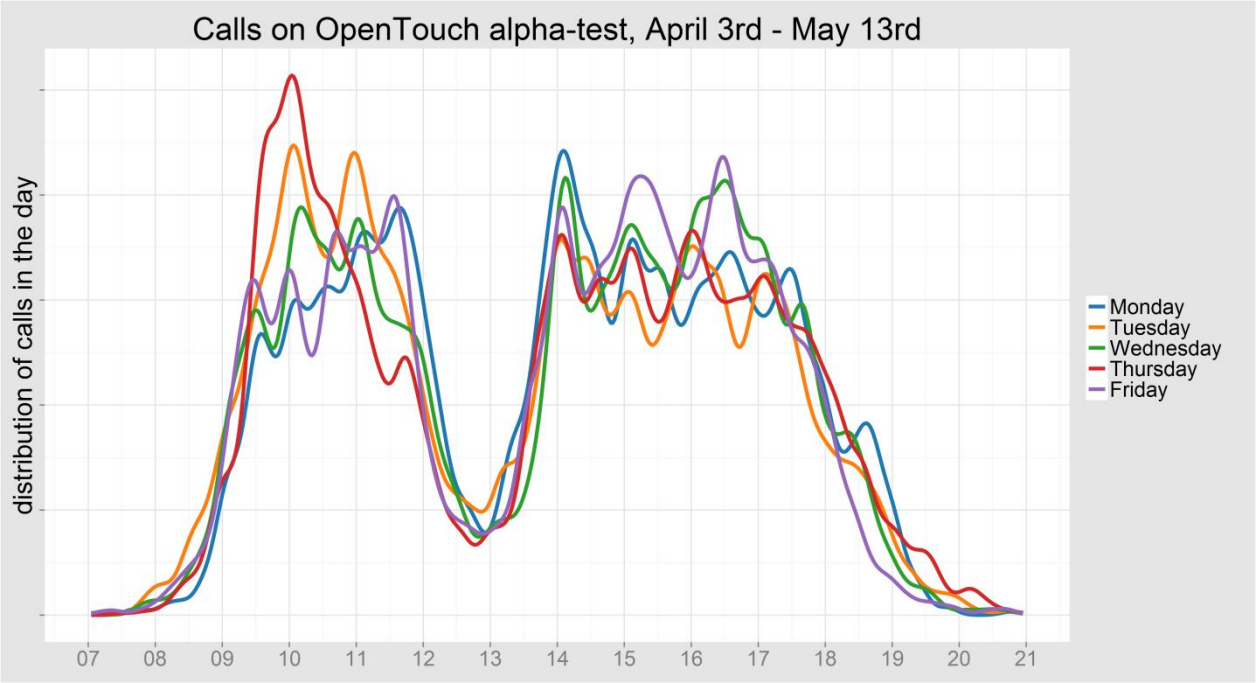



Figure 5.1: Distribution of call arrivals by day and by hour

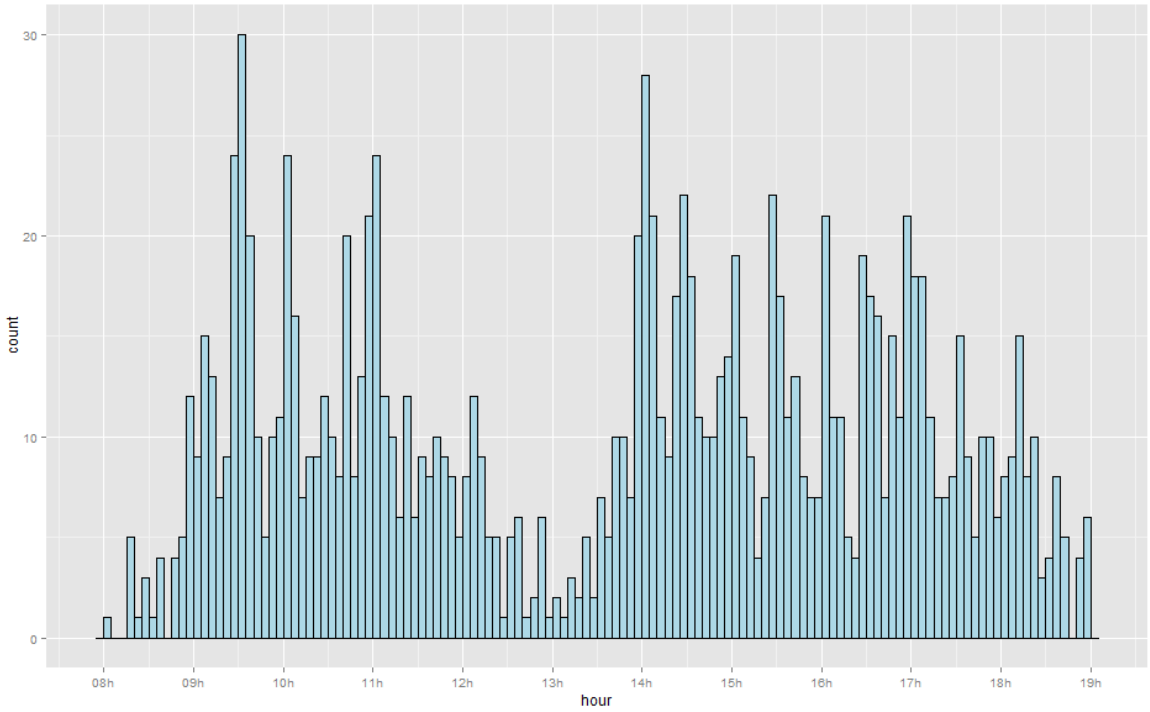



Figure 5.2: Distribution of call arrivals by hour

In Figure 5.3 the distribution of call durations is demonstrated. Most of durations are concentrated around thirty minutes and one hour calls which reflects the standard durations of regular scheduled conferences.

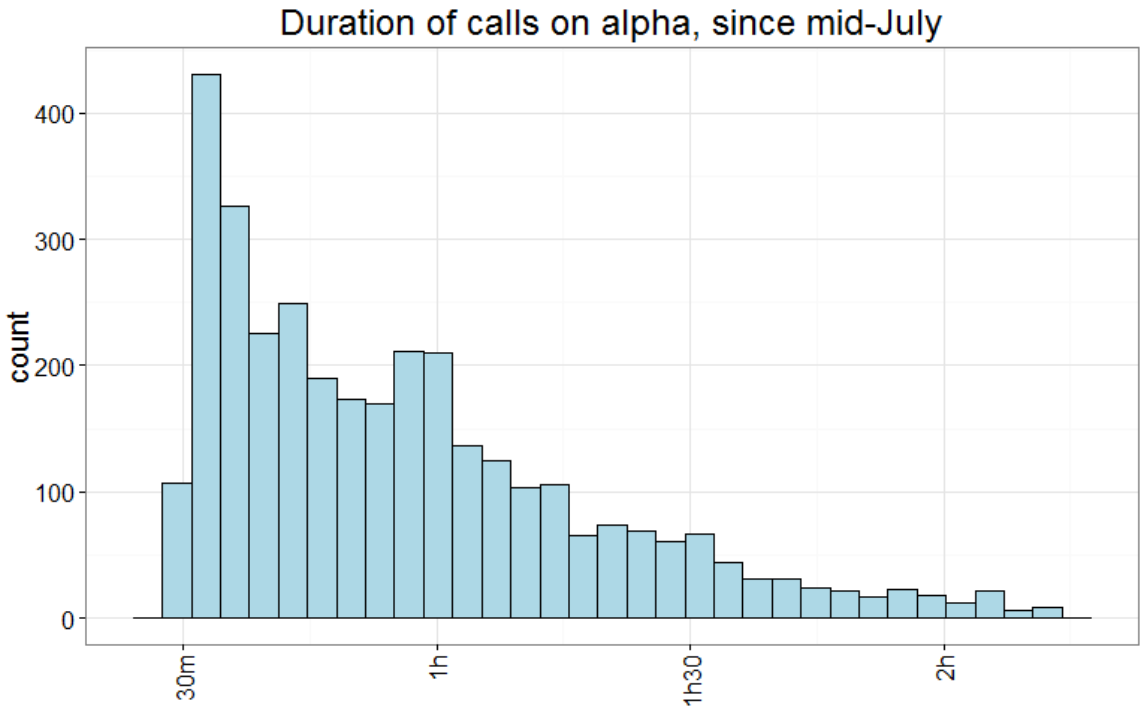



Figure 5.3: Distribution of call durations

The algorithms have been developed in Java programming language using DESMO-J library [74] for Discrete Event Simulation and JFreeChart library [75] for graphics.

## 5.2. Simulation topology

The simulation topology includes 4 sites with 2 Processors and 4 to 8 endpoints on each site:

- Site A: Processors (PA1, PA2), Endpoints (EA1, … EA4)
- Site B: Processors (PB1, PB2), Endpoints (EB1, … EB8)
- Site C: Processors (PC1, PC2), Endpoints (EC1, … EC4)
- Site D: Processors (PD1, PD2), Endpoints (ED1, … ED8)

The delays between the sites:

- Site A - Site B : 20 ms
- Site A - Site C : 40 ms
- Site A - Site D : 50 ms
- Site B - Site C : 50 ms
- Site B - Site D : 60 ms
- Site C - Site D : 80 ms

The Tasks are represented by the conferences comprising random number of participants (2 to 4) located on 1, 2 or 3 sites. All the Tasks have the same nominal type and consume CPU power proportionally to the number of conference participants. Tasks are added to the system during extended business hours (7:00 - 21:00) for a period of a month, with the duration of each Task from several minutes to 2 hours. Both arrival times and durations are generated following statistical distributions taken from a real enterprise communication system. Weights of all the optimization criteria were equal in these tests. This is somewhat arbitrary but they can only be tuned properly once real media server implementation on desktop equipment is used.

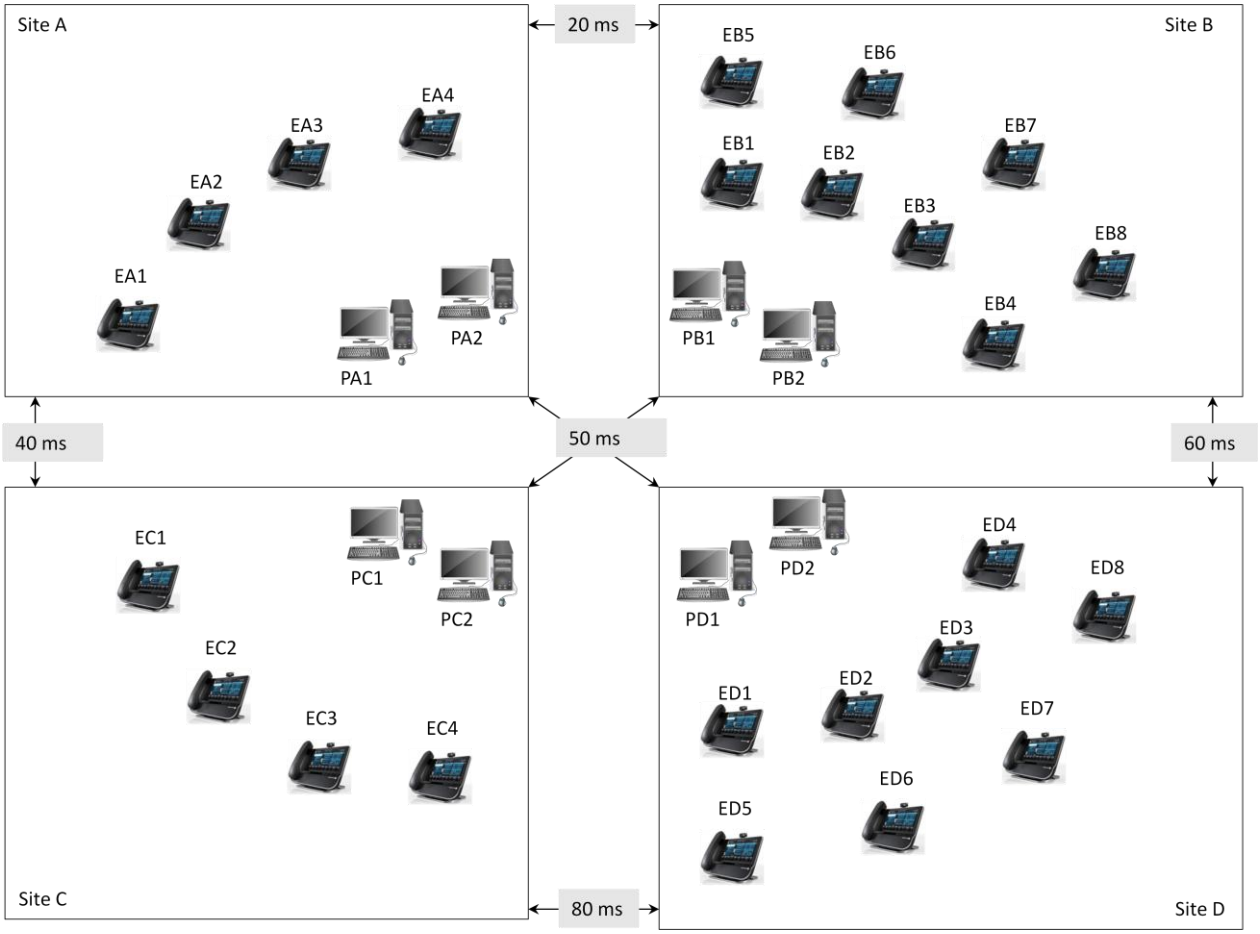



Figure 5.4: Graphical representation of simulation topology

## 5.3. Simulation results

The target of the simulation was in particular to discover how Redeployment Penalty value affects different aspects of the solution. For the performance reason calculations are implemented in integer numbers with all values normalized in the range [0, 100].

The first important point we tackled is the number of redeployments of Tasks during their execution. Each redeployment represents a trade-off between optimization of Objective Result and the perturbation of user experience caused by these redeployments, as video streams should be re-routed to a new Processor. In Figure 5.5 is shown the number of deployed Tasks (for each simulation with a given Redeployment Penalty) and the number of redeployed Tasks. For Redeployment Penalty > 60, there are no more redeployments in the system.

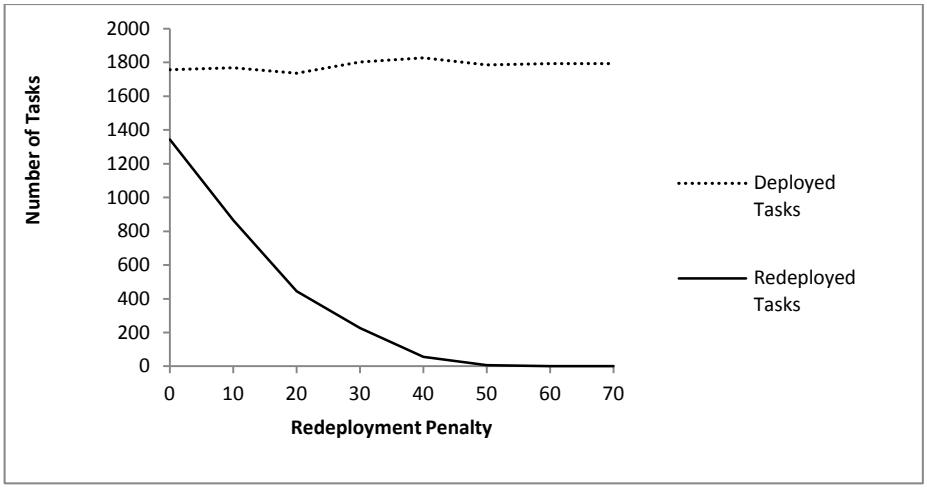



Figure 5.5: Number of Deployed and Redeployed Tasks depending on Redeployment Penalty

The second item that we considered is the delta between Factual Result (FR) and Ideal Result (IR). FR is the result of applying the algorithms described above. IR is an output of the algorithm which, after arriving of each State Change Event, takes all Processors, all Tasks and calculates the theoretical deployment which minimizes the sum of ORs of all the Tasks. In the limited topology that we considered, the IR can be simply computed by an exhaustive enumeration (comparing all possible deployments). The IR value represents the optimal distribution of the Tasks on the Processors, not taking into account their order of arrival. In Figure 5.6 we can observe the trade-off between low Redeployment Penalty (causing some perturbation of user experience due to Tasks redeployment) but at the same time close values of FR and IR; and high Redeployment Penalty causing low perturbation of user experience but increased gap between FR and IR.

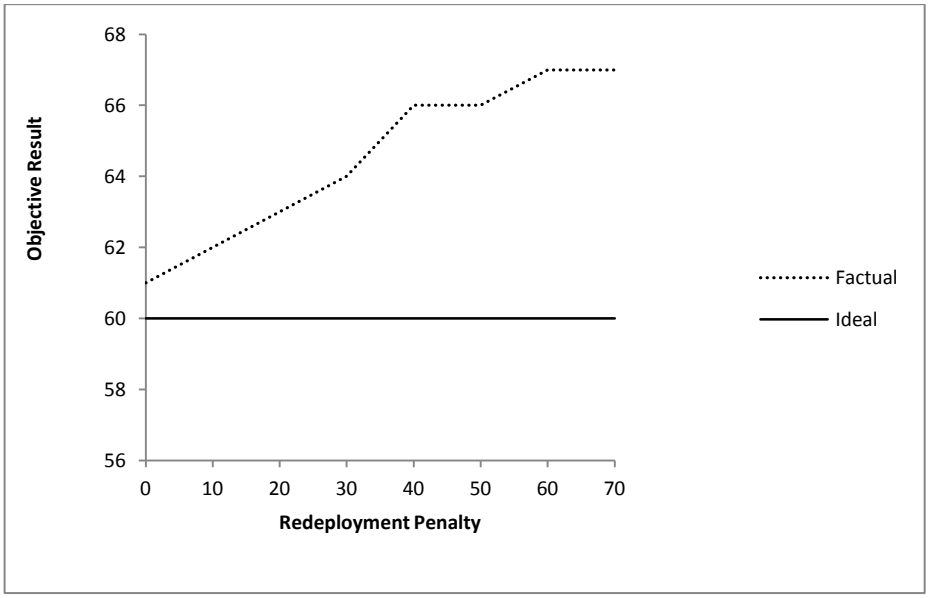



Figure 5.6: Factual Result and Ideal Result as a function of the Redeployment Penalty

These simulations show a clear trade-off between system optimality and the number of redeployments (which will affect user experience). In these figures, the Factual Result can approach the Ideal Result, even without too many re-deployments.

Of course, the results shown above are plotted for specific and somewhat arbitrary parameters (qualified CPU consumption, MADM weights). The optimal value of Redeployment Penalty and Weights can only be chosen in real exploitation conditions, which may be only available after testing of the implementation of the system with the real media server deployed on the real hardware platforms. It is out of scope of the current thesis as it is a pre-deployment engineering task that is extremely resource consuming: for instance, qualified CPU consumptions require extensive tests on a broad range of typical office hardware.

# Chapter 6

# Impact of CPU load on video quality

## 6.1. Introduction

Utilizing a PC as a platform for a media server poses a question on whether this commodity hardware can really be used for this purpose, even at limited scale, as traditionally media servers are deployed on powerful hardware in data centers. And, once the media server is deployed on a PC, how 3rd party processes, executing on the same machine and consuming the CPU resource, affect the quality of video conferencing service, which is provided by this media server. In this paper we answer these questions by elaborating an approach for estimating a quality of a video stream, which is sent to a conference participant from the media server, and fulfilling some experiments to demonstrate the results of application of this approach to a conference, hosted on a standard PC, while the CPU undergoes different levels of the load, perturbating execution of the media server.

The question of video quality is researched intensively in the presence of the complex of network impairments: delay, jitter, packet loss [60], [61], as well as separately for jitter [62] and for packet loss [57], [58]. Potential impact of video content on quality is also considered [63].

Regarding influence of CPU load the question to some extent was tackled for the video endpoints. Azzazi et al [56] describes how CPU is loaded at endpoint, processing video streams, when these streams are delivered by ATM network configured with different quality attributes.

At the same time no prior research was found on how CPU load affects the quality of video stream provided by MCU. While the MCU is a central part in modern video conferencing solutions, and we can assume that impediments to its functioning have dramatic consequences on conference quality.

Videoconference contents over packet networks can be impaired by compression coding, packet loss, delay jitter, signal decoding and reconstruction process. The end-user may perceive a loss in image clearness-sharpness (like blurring and artefacts) and fluidity impairments (like freezing and jerky motion) [53] on the visual information. These

impairments can have a negative impact on the end-user quality assessment and satisfaction of the videoconference service.

The current way of measuring the video quality is mainly based on technical parameters of the service like encoding parameters, network errors and bitrates; however, these parameters are not directly related with the end-user quality perception. For instance, given a video bitrate, perceived quality may be strongly different from content to content [54].

Given that signal integrity is not guaranteed and that the final receptor is a human, it is therefore necessary to measure video quality by taking into account end-user visual perception and judgment. In addition, given that videoconference services are in real time, the most practical approach for measuring video quality is using a No-reference metrics. In this work, we use Perceptiva VIVO [59] to measure the impact of CPU load on video conferencing perceived quality.

Perceptiva VIVO is an automatic tool for measuring and analysing video quality from end-user assessment stand point. We use the version specifically adapted to videoconference contents. The tool is a No-Reference metric because there is no need for a reference or test signal to analyze quality, its technology is based on human vision perception and customer behavior when assessing video quality. Perceptiva VIVO is a proprietary technology based on a low-level human vision model reproducing the first stage of eyes images processing combined with cognitive models of brain judgment mechanisms. Subjective quality tests, respecting the standardised SAMVIQ methodology [55], were conducted for modelling and validation proposes. This technology reproduces end-user behaviour when assessing videoconference services.

The tool receives signals from a digital video output interface. The solution then provides, in real time, a Mean Opinion Score Predicted (MOSp) representative of end-users subjective opinion. A numerical scale (0-100) for rating quality is used. This scale is related to five quality categories (bad, poor, fair, good, excellent) that are uniformly distributed. The subjective quality scale used to represent customer's opinion is in accordance to SAMVIQ [55] international standard.

## 6.2. Description of the testbed

For our experiments we use a testbed, comprising typical office hardware and an operating system. Open source MCU as well as special software, managing the CPU load, is installed on the platform. Perceived video quality measurement is conducted using Perceptiva VIVO version Visio v2.0. The equipment receives videoconference signal from the HDMI interface of the participant desktop.

As a platform a laptop Dell Latitude E5450 with the CPU Intel Core i5-5200U @ 2.20 GHz and 8 GB RAM is employed. The operating system is Windows 7 Enterprise with Service Pack 1.

OpenMCU-ru [49] of the version 4.1.4 plays the role of media server in our experiments. OpenMCU-ru is an MCU providing video mixing, trans-coding and other functionalities.

Linphone [50] of the version 3.9.1 is our choice for conference participants endpoints. Linphone is a software video phone, which is installed, for our experiments, on desktop and laptop computers with Windows 7 operating system. The computers are equipped with Logitech HD Pro C920 webcam or an integrated webcam (see Figure 6.1).

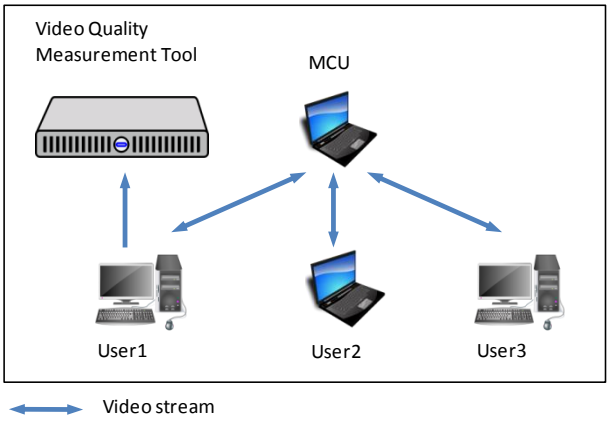



Figure 6.1: Configuration of the testbed: User1 receives video stream from the MCU and via HDMI connection sends it to the Video Quality Measurement Tool

We use Session Initiation Protocol [10] in order to connect video endpoints to the media server. The video is encoded by VP8 codec [5]. Clients send video streams with the resolution 800x600 pixels (SVGA). The CPU mixes these streams in one stream with the resolution 704x576 pixels (4CIF). The frequency is 50 fps (frame per second).

The scenario is the following: several participants connects to the same “virtual meeting room”, their video streams are mixed and the resulted stream is sent back to each participant. The video streams, sent from the video cameras of the participants, correspond to the real video conferencing picture with a person’s head in the frame, without any changes of the scene. We capture the screen of one of the participants and analyze the quality of the video stream rendered on its screen.

## 6.3. Results of the experimentation

The first thing we would like to know is to which extent a standard laptop can be used as a platform for an MCU. For that we are going to measure which CPU resources are needed to organize a “virtual meeting room” (with one participant, as a basic case), and which resources are needed in order to add a participant to an existing room. Normally the most demanding operation is video encoding, so we can expect that each new room, which needs a new encoding of the result of the mixing of all the participants streams, will increase significantly the CPU load. We see the confirmation of that in Figure 6.2, in which the X axis is marked by the following format: “r1:X, r2:Y”, where “r1” and “r2” are two “virtual meeting rooms”, X and Y are the number of participants in a respective room.

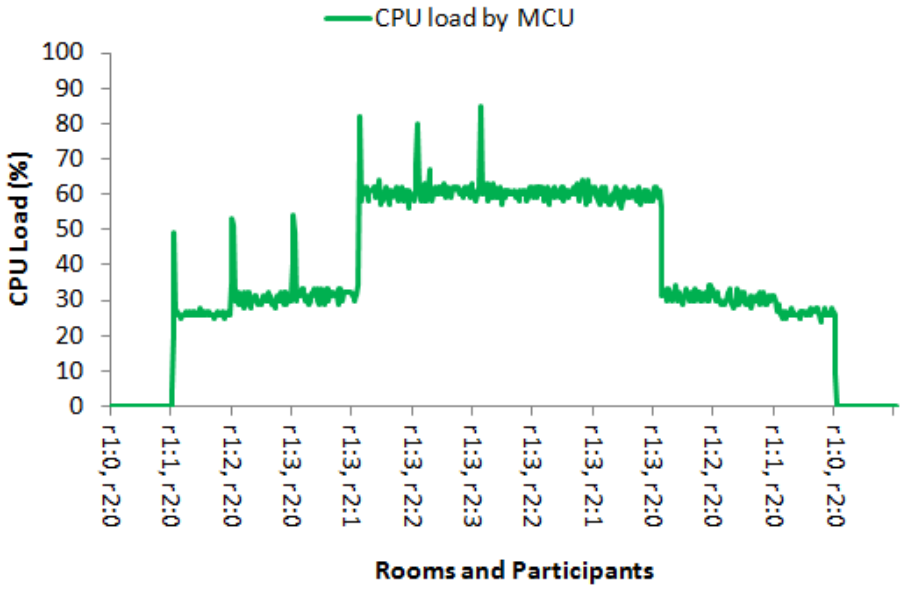



Figure 6.2: CPU load caused by video processing tasks. X axis titles “r1:X, r2:Y” mean X participants in room1, Y participants in room 2

We can see that creation of a “virtual meeting room” adds 30% - 35% of CPU load: these are resources, needed to encode the output video stream of a room. At the same time adding a participant to the existing room increases CPU load very moderately, if increases at all. This graph proves our idea that a standard user equipment can be used as a platform for media server executing a limited number of video processing tasks. In this exact case a mediocre laptop holds two conferences each with three participants (and the number of participants can be extended to some meaningful value) and consumes about 60% - 65% of CPU power, which excludes battle for the resources between the media processing tasks and leaves enough resources for 3rd party processes.

Further, we would like to know how limitation of CPU resources, dedicated to MCU, affects the quality of resulting video stream. As a media processing task, executed by media server, we organized a conference for three participants, consuming about 33% of CPU power. Then we investigate this problem by means of two experiments. The first experiment is slowing down the target process, in our case OpenMCU-ru, by making the CPU to go to idle mode for short periods of time. This functionality is implemented in an application “Battle Encoder Shirase” (BES) [51]. We use the version 1.6.2. We can call it a “hard” approach as it does not allow Windows process scheduling mechanism to intervene into the game and optimize the execution. This approach gives very visual results but it does not reflect the real modes in which processes compete for the resources.

In Figure 6.3 we see that we can reduce CPU resources, used by MCU, leaving about 10-15% as the reserve to keep Good quality (the reserve is the distance between green and red lines). When the reserve is about 5-10%, the quality becomes Poor to Fair, and when reserve goes to 0% - the quality gets Bad level.

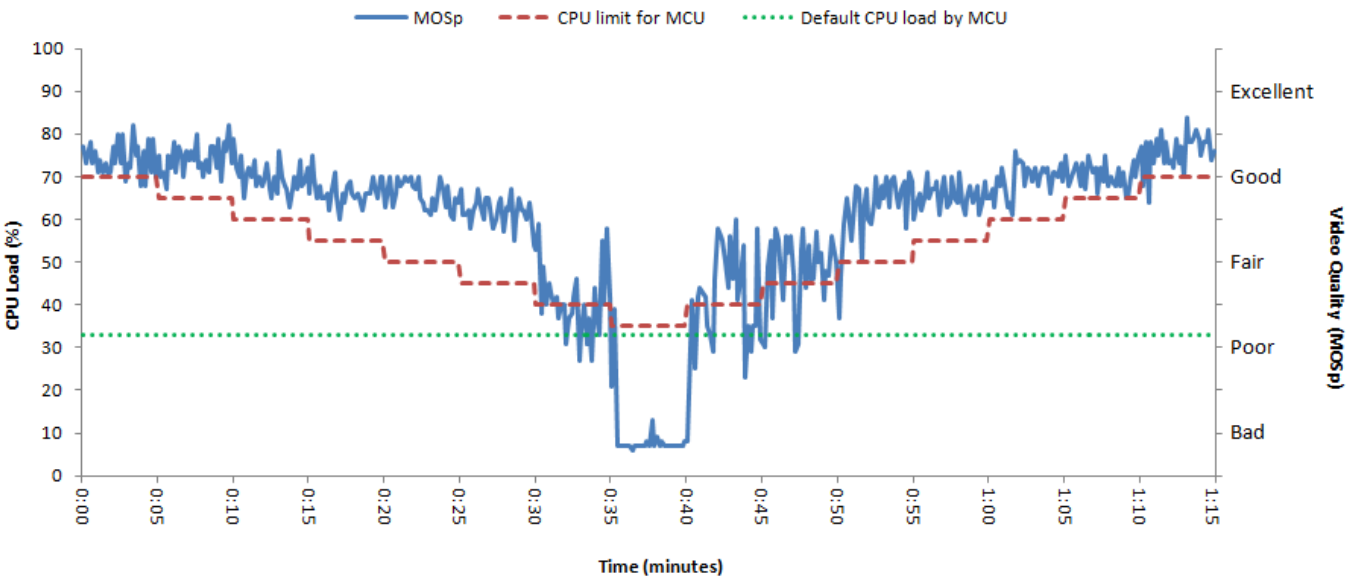



Figure 6.3: Video quality affected by MCU process CPU consumption limitation

The second experiment employs a special process, which consumes a preconfigured amount of CPU, such a way competing with the target process. The special process in our case is a “CPU Killer 3” (CPUK) application [52], in the version 1.0.7. CPUK tries to load the CPU to the preconfigured level and then Windows process scheduling mechanism assigns the real amount of CPU cycles, taking into account other processes, executed on the same CPU. Here we use the assumption that all processes have the same “standard” priority. This rather well simulates the concurrent applications launched on the same platform, we can call it a “soft” approach. It’s depicted in Figure 6.4.

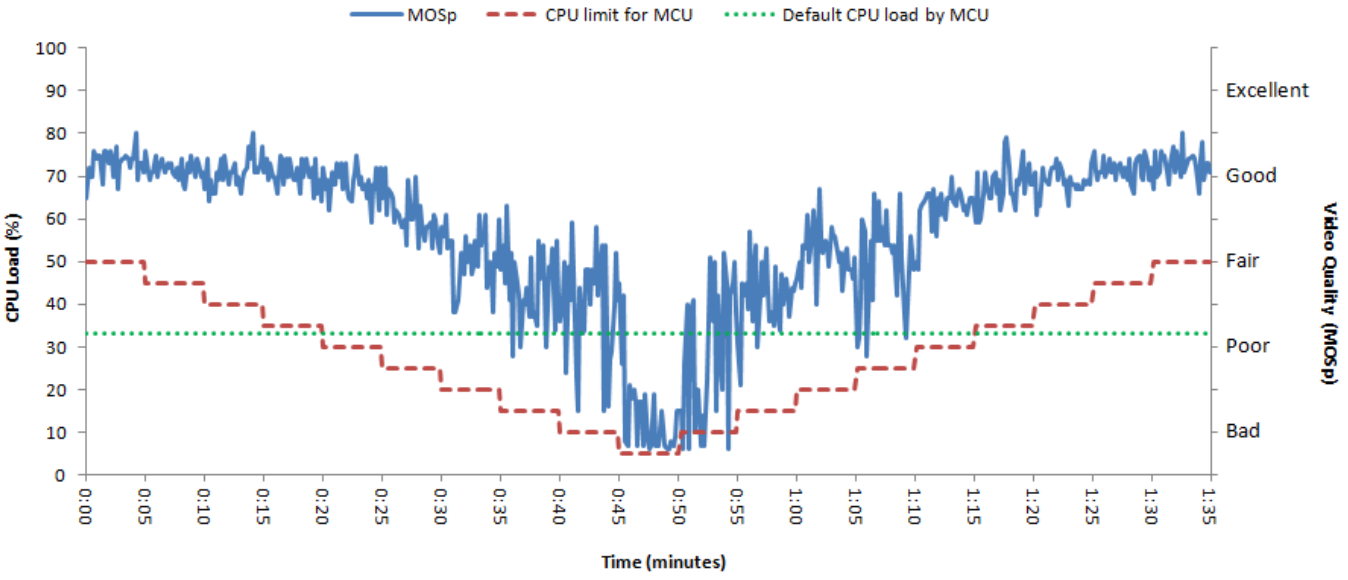



Figure 6.4: Video quality affected by launching 3rd party process competing with MCU process for CPU resources

In this experiment we see how the two processes “fight” for the CPU resources. Until CPUK process claims 65% of CPU, such a way leaving 35% of CPU, necessary for normal MCU functioning, the quality of video remains Good. Then, following the increasing CPUK demand until 85% - 90% of CPU, the quality of video gradually degrades through Fair to Poor level. This demonstrates that Windows process scheduler arbitration in a real situation of two processes, competing for the CPU resources, allows MCU to keep some “acceptable” level of quality. By the way, further increasing CPU demand from CPUK makes the video quality Bad.

In order to extract practically usable results from the graph, represented in Figure 6.4, we would like to understand how video quality depends on the rate, with which media server and a 3rd party process compete for the CPU resources. This info is drawn in Figure 6.5. The X axis on this graph is marked with the delta between “provided CPU” (that is an amount of CPU resource, which is not demanded by 3rd party process) and “needed CPU” (that is amount of CPU resource, which is demanded by media server). In fact, this CPU delta means how much CPU resource is available for the media server without competition with the 3rd party process.

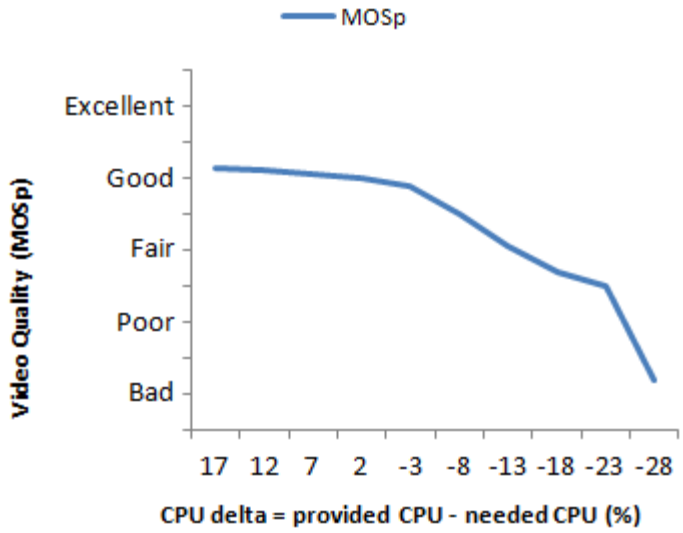



Figure 6.5: Video quality affected by the level of competition for the CPU resource

We can see that while the delta is greater than 0 (that is the processes do not compete for the CPU resource), the video quality is good. When competition starts, the quality gradually decreases until the level, when the 3rd party process starts to demand about 90% of CPU resource. This causes dramatic video quality decrease to unacceptable values, as Windows process scheduling mechanism does not have enough space to provide media server with the required CPU resource.

Such a way, we observe here the clear trade-off between the video quality (varying between Good and Poor), and the amount of CPU resource, requested by the 3rd party process. Finally, it’s up to a user to decide, which level of quality she is

ready to tolerate in order to allow the cooperative use of the CPU by several processes.

# Chapter 7

# Conclusions and future work

## 7.1. Conclusions

In this thesis we studied a novel approach to organizing video conferences. Nowadays video conferencing in enterprises is organized primarily using central MCUs. MCU is responsible for controlling the conference as well as for video processing tasks, such as mixing or trans-coding. Due to the fact that MCUs are usually designed in the form of specialized hardware, they are an expensive equipment. Pure software MCUs also exist, they can be utilized in Cloud mode. However, due to the complex operations with media streams, they consume a lot of server resources. At the same time, Overlay Network approaches exist for video conferencing: Application Layer Multicast and Peer-To-Peer. These approaches are designed for video relays, while video mixing tasks are directly handled at the endpoints. Thereby, if an endpoint is not capable to mix several video streams, due to some hardware/software limitations, it will not benefit from modern telepresence experience.

So the problem is to deliver rich video experience, available today through dedicated MCUs, without using dedicated hardware and without overloading existing servers with media processing operations.

The proposed solution is to distribute MCUs on Enterprise Desktop Grid, which consists of all the PCs, available in the enterprise, with enough resources to accept video processing tasks. Prior art research shows that a lot of personal computers at an enterprise are not used during long periods, even during work hours. In modern terminology this approach is known as “Fog computing” in contrast to centralized “Cloud computing”.

The requirements to build such a distributed MCU:

- The network architecture should be applicable to typical enterprise topology, containing sites with fast LAN connected by potentially slow Internet

- The architecture should take into account dynamic nature of Enterprise Desktop Grid, in particular the fact that PCs can be arbitrarily stopped or that 3rd party processes can be launched by end users

The Desktop Grid Conferencing (DGC) system that we propose consists of a set of media servers (tackling video processing tasks), distributed on a cluster of ordinary office equipment (PCs, laptops, etc). The description of the DGC system is based on two main notions: Tasks and Processors.

Task is a media related activity, traditionally provided by a MCU or a software media server: video mixing, video switching, trans-coding, trans-scaling or other manipulations on video streams.

Processor is a media server deployed on a general purpose hardware, such as a PC. Users may turn on/off their PCs and launch third party applications consuming CPU power as well as start/stop calls and conferences randomly. This results in unpredictability of the sets of Tasks and Processors, which should be taken into account by the system.

The main logic of the proposed architecture is to distribute and, if necessary, to redistribute Tasks on Processors taking into account changes in the set of Tasks, set of Processors and external constraints. The result of distribution should be “optimal” under some conditions.

Optimization criteria can be divided into two sets: the Network and Platform ones. Network criteria that should be taken into account include WAN bandwidth consumed by a Task and End-to-end delay between endpoints. Platform criteria are related to the Processors that are available in the system: network connectivity, power supply, resource sharing, CPU load.

All the optimization criteria are different in their nature and importance, leading us to choose a MADM (Multi-Attribute Decision Making) approach, where each criterion is associated with a weight. Application of a MADM method gives an integral metric of a deployment of a given Task to a given Processor, which is called Objective Result. A dedicated MADM method using “context aware normalization” has been designed to calculate the Objective Result. In this method normalization information is derived from the nature of the attributes. Such an approach allows applying the MADM procedure only when a Processor is added to the system or a specific attribute of the Processor is changed. In other words, no computation would be needed for a given Processor, whatever are the changes applied to other Processors, which is very important, taking into account the dynamic real-time nature of the DGC system.

The DGC system itself cannot guarantee suitable SLA (Service Level Agreement) because its resources are controlled by end users, and not by the system itself. To solve this problem DGC system can be combined with the conference system in the

Cloud to provide both SLA and cost advantages at the same time. We elaborated the algorithms, combining Cloud/Fog approach with different types of media servers, the result provides an optimized conferencing solution in the terms of cost for both provider and consumer as well as in terms of end user experience.

In order to test the Task distribution algorithms, the respective logic was implemented using a discrete event simulation approach.

The first point we tackled in the simulation is the number of redeployments of Tasks during their execution. Each re-deployment represents a trade-off between optimization of Objective Result and the perturbation of user experience accompanying redeployment.

The second item that we considered is the delta between Factual Result and Ideal Result. Factual Result is the result of applying of the algorithms, calculating Objective Result in the current situation in the system. Ideal Result is an output of the algorithm which, after arriving of each State Change Event, takes all Processors, all Tasks and calculates the theoretical deployment which minimizes the sum of Objective Results of all the Tasks. The Ideal Result value represents the optimal distribution of the Tasks on the Processors, not taking into account their order of arrival.

These simulations show a clear trade-off between system optimality and level of user experience, affected by Tasks redeployments. In fact, the Factual Result can approach the Ideal Result, even without too many re-deployments. However, the logic, responsible for the decision on redeployment, can only be determined with realistic parameters (qualified CPU consumption, tuned MADM weights), which may be only available after intensive testing of the system implementation based on the real media server deployed on the real hardware platforms.

Then we investigated to which extent a PC can be used as a platform for hosting a media server, and how CPU load of this platform affects the quality of resulting video stream. For that we established a test bed with open source media server, deployed on a usual laptop, and connected several video soft phones playing the role of conference endpoints.

To one of the endpoints we connected a hardware video quality measurement tool, which provided us with predicted Mean Opinion Score. We applied this tool to a video stream, generated by a video conferencing media server. The server, being deployed on a commodity laptop, was disturbed by a third party process, which consumed different amounts of CPU power. As a result we demonstrated that the commodity office hardware can really be used as a platform for media servers, carrying limited workload in the scope of our Enterprise Desktop Grid conferencing system.

## 7.2. Future Work

Two applications of machine learning can be considered in order to improve the quality of the system.

For a given platform with initial weights defined by executing a limited number of manual tests, during its exploitation we can:

- Based on any given platform configuration/state to try to dynamically predict weights which will maximize QoE (ex: Wi-Fi blackout increases weight of "Network Connectivity"). For that we need continuously collect information on different aspects of the system environment: state of the network and networking gear, types of utilized personal computers and so on. Then after correlation of this information with resulting QoE we can deduce weights of existing or newly created criteria in order to maximize resulting QoE.

- To take into account the history of system functioning for future distributions (ex: stable/not stable node observations introduce the node "rating"). Such a way we will be able to create some sort of “resources profiles”, that is typical paterns of resources usage and behavior, which affects overall stability of the system. Correlating thise patterns with days of week, time and other environment information we will be able to predict to some extent the behavious of given resources in the future.

# Annex A

# Standalone DGC system design

## A.1 Design static view

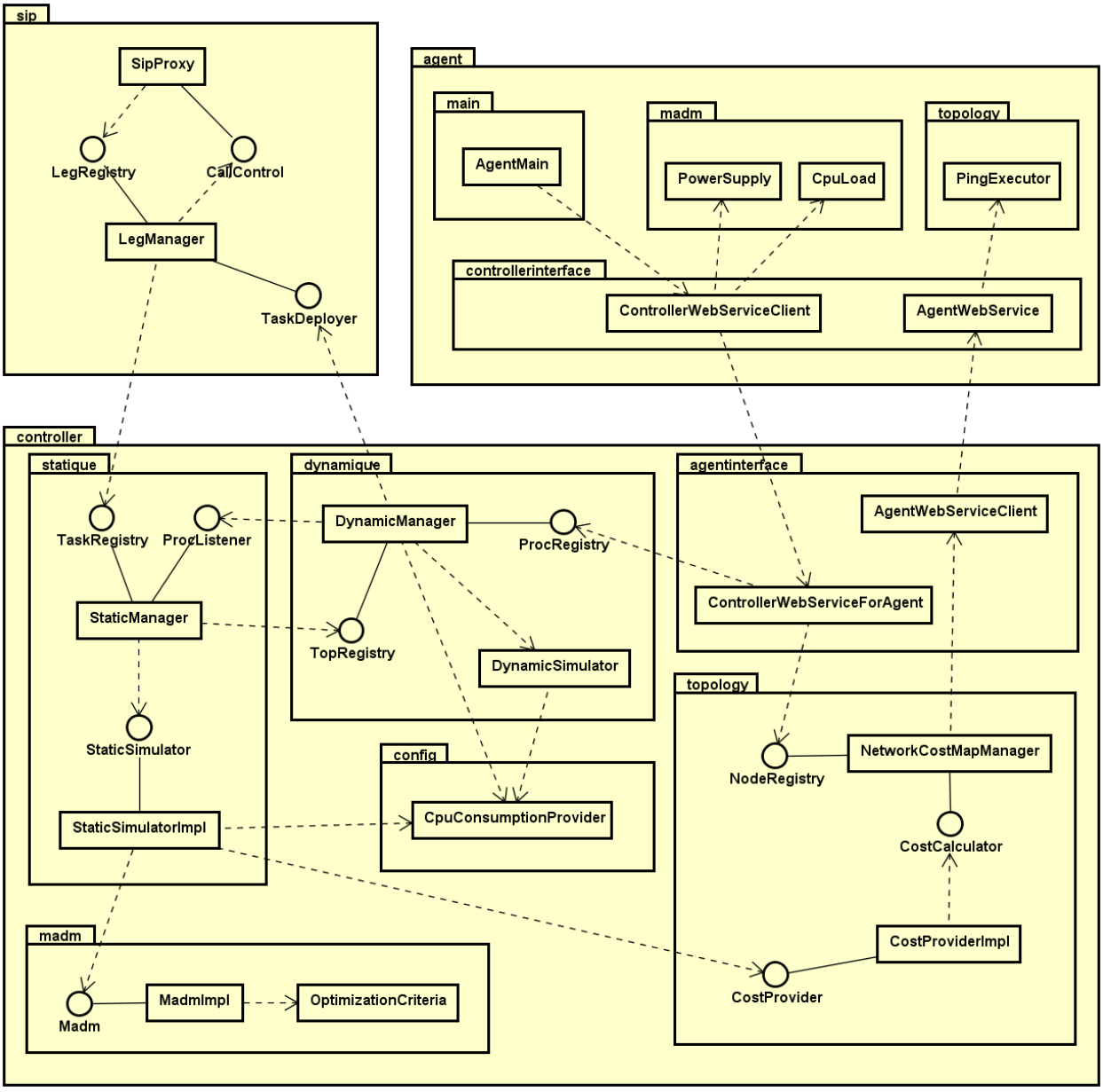



Figure A.1: Packages, interfaces and classes structure

***Structure of the description***

**`firstlevel`** – solution component

**`secondlevel`** – Java package

**`ThirdLevel`** – Java class or interface

forthLevel() – Java method

***Design description***

**`agent –`** classes and interfaces comprising *Monitoring agent*

**`main -`** classes controlling overall Agent behavior

**`AgentMain –`** registers *Processor* executed on this platform on *Video processing controller*

**`controllerinterface -`** classes used to dialog with *Video processing controller*

**`AgentWebService –`** services provided by *Monitoring agent*

**`getPingResult() –`** returns ping result from this agent to a given network location

**`ControllerWebServiceClient –`** client consuming services of *Video processing controller*

**`madm -`** classes providing values for MADM criteria

**`CpuLoad –`** provides current CPU load

**`PowerSupply –`** provides current status of power supply

**`topology -`** classes providing topology information

**`PingExecutor –`** provides value of round trip time to a given network location

**`sip –`** classes and interfaces comprising *SIP server*. The implementation of SIP server is extremely simplistic for this Proof-Of-Concept. A real SIP server should be used for real deployments.

**`TaskDeployer –`** used to instruct SIP server of which operations should be applied to Tasks

**`deployTask()`** – this method is called when a new task should be deployed: SIP Server asked Video processing controller to deploy a new Task so Video processing controller returns selected node by means of deployTask() method

**`undeployTask()`** – this method is called when the DGC system can't execute the given Task (e.g. the Processor has stopped) and can't redeploy it to another Processor due to the lack of resources

**redeployTask()** – this method is called when the existing task should be redeployed from one node to another due to the changes of the system (node has stopped, CPU is overloaded, etc)

**refuseTask()** – this method is called when SIP Server asked to distribute a new task, but Video processing controller could not find a Processor which is capable to execute a task so task is refused

**LegRegistry –** used by SIP server to notify about arrival or departure of a conference participant

**legAdded()** – a new participant connected. He should be added to an existing conference, or a new conference should be created, if this is the first participant.

**legRemoved()** – an existing participant hung up. If he is the last participant in the conference, the conference should be removed from the DGC system.

**CallControl –** when conference is redeployed to another Processor, this interface is used to instruct SIP server to make conference participants to reconnect to the conference in order to route the call to a new Processor

**refer()** – makes participants to reconnect

**LegManager –** logic of conference legs management

**SipProxy –** listens to SIP signaling from the conference participants

**controller –** classes and interfaces comprising *Video processing controller*

**agentinterface -** classes used to dialog with *Monitoring agent*

**ControllerWebServiceForAgent –** services provided by ***Video processing controller*** to *Monitoring agent*

**registerProcessor() –** a new Processor has appeared in the grid

**unregisterProcessor() –** the Processor has left the grid

**keepAlive() –** notifies that the Processor is still alive and passes current info about CPU load and power supply

**AgentWebServiceClient –** client consuming services of *Monitoring agent*

**statique -** classes and interfaces, responsible for static aspects of tasks distribution, that is all the optimization criteria, except CPU load

**TaskRegistry –** through this interface Video processing controller is notified about created and removed Tasks

**taskAdded() –** this method is called when SIP Server is requested to organize a new conference. Video processing controller calculates the node on which the given Task should be deployed.

**taskRemoved() –** this method is called when SIP Server is requested to stop the existing conference. Video processing controller updates its state according to the fact that the given Task is removed from the system.

**ProcListener –** through this interface the changes in the list or state of Processors, which affect the results of Static Simulation, are reported

**procAdded() –** this method is called when a new Processor is added to the grid and the procedure of Static Simulation should be applied to existing Tasks

**powerStatusChanged() –** this method is called when the Power Status of a given Processor is changed and it affects the results of Static Simulation, which should be re-executed in this case

**StaticSimulator –** through this interface the results of Static Sumulation are returned. Static Simulation is application of MADM to all optimization criteria, except CPU load

**calculateSsr() –** calculates SSR (Static Simulation Result) for a given Task in application to a given Processor

**StaticManager –** keeps Static Simulation Result info about Tasks on Processors

**StaticSimulatorImpl –** logic of static simulation

**dynamique -** classes and interfaces, responsible for dynamic aspects of tasks distribution, that is processing of CPU load

**TopRegistry –** through this interface information on “Task On Processor”, that is the results of Static Simulation of a given Task on a given Processor are passed from the package, responsible for static optimization criteria to the package, responsible for dynamic optimization criteria

> **addProcQueue() –** for an added Task the results of Static Simulation of this Task on all the Processors are passed to **DynamicManager** in order to apply Dynamic Simulation
>
> **removeProcQueue() –** for the removed Task **DynamicManager** is notified to remove the results of Static Simulation of this Task as they are no more needed
>
> **addTaskQueue() –** for an added Processor the results of Static Simulation of all the Tasks on this Processors are passed to **DynamicManager** in order to apply Dynamic Simulation
>
> **updateTaskQueue() –** for a Processor with changed static optimization criteria the results of Static Simulation of all the Tasks on this Processors are passed to **DynamicManager** in order to re-apply Dynamic Simulation

**ProcRegistry –** through this interface **DynamicManager** is notified about changes in the list or state of Processors

> **procAdded() –** Processor is added to the grid
>
> **procRemoved() –** Processor is removed from the grid
>
> **setCpuLoad() –** CPU load of a Processor is changed
>
> **setPowerStatus() –** power status (AC/battery) of a Processor is changed

**DynamicManager –** processes changes in Static Simulation Results and initiates Dynamic Simulation when needed. Based on both types of simulation selects the target Processor for Task deployment or re-deployment

**DynamicSimulator –** responsible for calculating Dynamic and Full Simulation Results, as well as Full Deployment Result, combining information about static and dynamic info about the system

> **calculateDsr() –** calculates DSR (Dynamic Simulation Result), which is the prediction of CPU load when a given Task is deployed on a given Processor

**calculateFsr()** – calculates FSR (Full Simulation Result), which is the combination of Static and Dynamic Simulation Results

**calculateRdr()** – calculates RDR (Real Deployment Result), which is the metric, combining Full Simulation Result and actual CPU load on the Processor, currently occupied by a given Task

**madm** - classes and interfaces, implementing MADM logic

**Madm** – provides access to MADM logic

**calculateObjectiveResult()** – calculates Objective Result based on predefined Optimization Criteria

**MadmImpl** – implementation of MADM logic

**OptimizationCriteria** – provides access to Optimization Criteria, stored in a file

**topology** - classes and interfaces, responsible for processing of topology information represented in ALTO form

**NodeRegistry** – through this interface Video processing controller is notified about created and removed Nodes. Node is a hardware platform on which Processor is deployed, the Node is used for processing topology informationj.

**nodeAdded()** – this method is called when a new Node is added to the grid and ALTO information should be updated, if needed

**nodeRemoved()** – this method is called when an existing Node has left the grid and ALTO information should be updated, if needed

**CostCalculator** – through this interface ALTO cost is requested in the case when it's not cached in **CostProviderImpl**

**calculateCost()** – calculates ALTO cost between two network elements (Endpoints or Nodes)

**CostProvider** – provides fast access to cached ALTO information

**getCost()** – provides ALTO cost between two network elements (Endpoints or Nodes)

**`NetworkCostMapManager`** – manages Nodes and calculates ALTO information

**`CostProviderImpl`** – servers as a cache for ALTO information, allowing fast access to this info during Static Simulation process

**`config`** – classes, providing configuration information, gathered from external sources

**`CpuConsumptionProvider`** – stores pre-computed values from CPU load Qualification Matrix

## A.2 Design dynamic view

In this chapter the two use case are presented in the form of UML sequence diagrams:

- Add Task
- Add Processor

## A.2.1 Add Task sequence

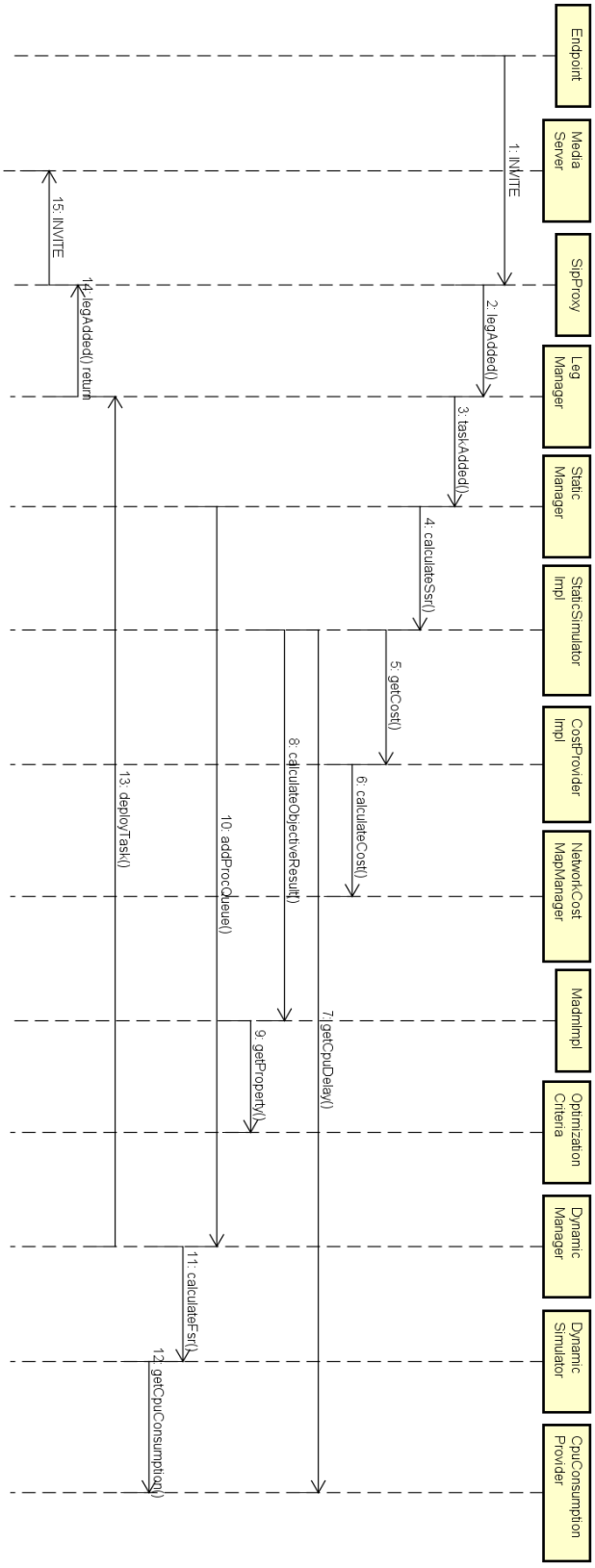



Figure A.2: add Task sequence

**1: INVITE –** *Endpoint* sends SIP INVITE to *SipProxy* in order to join a conference

**2: legAdded() –** *SipProxy* notifies *LegManager* that a new leg should be added to a given conference

**3: taskAdded() –** *LegManager* understands that this *Endpoint* is the first participant of the conference as this conference doesn't exist yet, so it should be created. *LegManager* notifies *StaticManager* that a new *Task* corresponding to this conference should be deployed to the DGC system

**4: calculateSsr() –** *StaticManager* calls *StaticSimulator* in order to calculate Static Simulation Result for the *Task* on all the registered *Processors*

**5: getCost() –** *CostProviderImpl* checks whether Cost (i.e., in the simplest case, Delay) between Network Location, to which pertains currently examined *Processor*, and the given *Endpoint* was already calculated and cached. If yes, the value is returned. If not, *NetworkCostMapManager* is called in order to calculate it.

**6: calculateCost() –** *NetworkCostMapManager* can use different approaches in order to calculate Network Cost. It can use info from ALTO provider, or it can ask *Endpoint* to directly ping *Processor* under question and return ping result to *NetworkCostMapManager*

**7: getCpuDelay() –** *StaticSimulatorImpl* requests the delay, caused by *Processor* CPU, which executes video treatment

**8: calculateObjectiveResult() –** *StaticSimulatorImpl* calls *Madm* interface in order to apply Multi Attribute Decision Making procedure and get Objective Result for static Optimization Criteria

**9: getProperty() –** *MadmImpl* calls *OptimizationCriteria* class in order to obtain a list of Optimization Criteria and their weights

**10: addProcQueue() –** *StaticManager* passes obtained list of Static Simulation Results for added *Task* to *DynamicManager*

**11: calculateFsr() –** *DynamicManager* uses *DynamicSimulator* in order to combine Static Simulation Results with CPU consumption information and calculate Full Simulation Results for added *Task* on all the *Processors*

**12: getCpuConsumption() –** *CpuConsumptionProvider* returns a value from CPU Qualification Matrix

**13: deployTask() –** *DynamicManager* selects a *Processor* and asks *LegManager* to address incoming request from Endpoint to this *Processor*

**14: legAdded() return –** *LegManager* returns this *Processor* to *SipProxy*

**15: INVITE –** *SipProxy* forwards SIP INVITE from *Endpoint* to selected *Processor*

## A.2.2 Add Processor sequence

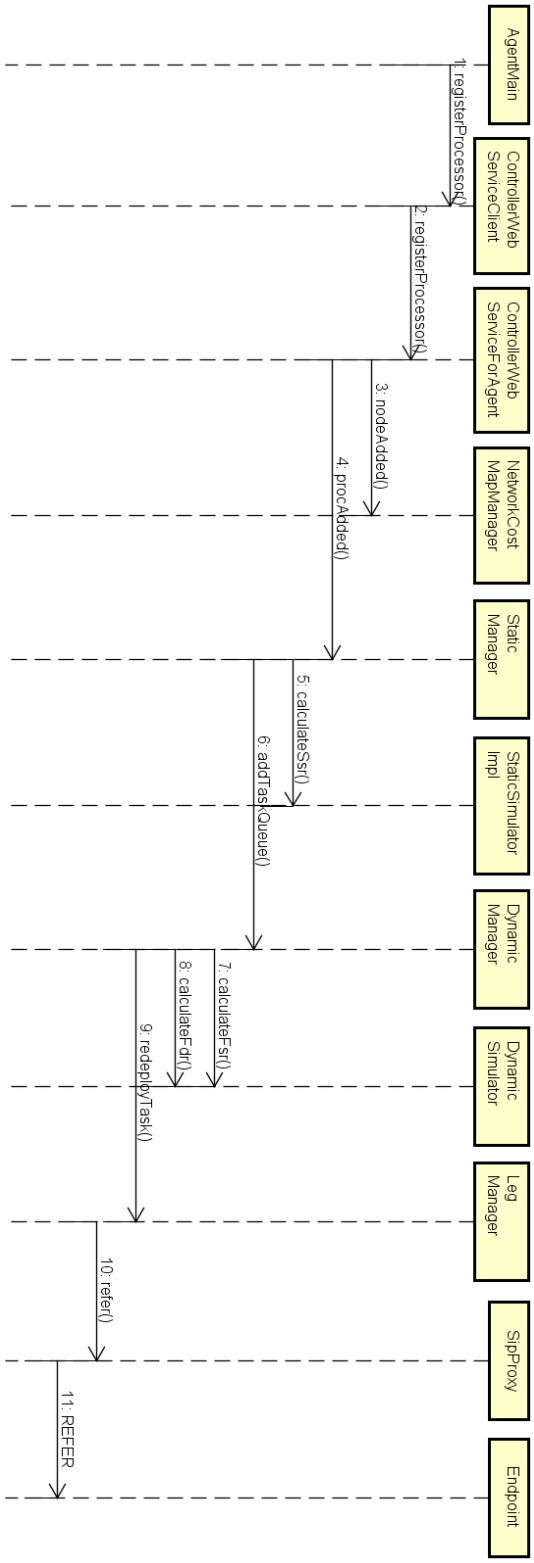



Figure A.3: add Processor sequence

**`1: registerProcessor()`** – when *Processor* starts *AgentMain* class solicits *ControllerWebServiceClient* in order to notify DGC system about its appearance

**2: registerProcessor() –** *ControllerWebServiceClient* sends this notification to *ControllerWebServiceForAgent,* which is the WebService API of DGC system

**3: nodeAdded() –** *ControllerWebServiceForAgent* uses this method in order to notify *NetworkCostMapManager* about arriving of a new *Node*.

**4: procAdded() -** *ControllerWebServiceForAgent* uses this method in order to notify *StaticManager* about arriving of a new *Processor*.

**5: calculateSsr() –** *StaticManager* calculates Static Simulation Results for all the *Tasks* on this *Processor.* This step analogous to the step 4 from the algorithm "Add Task sequence".

**6: addTaskQueue() -** *StaticManager* passes obtained list of Static Simulation Results for added *Processor* to *DynamicManager*

**7: calculateFsr() -** *DynamicManager* uses *DynamicSimulator* in order to combine Static Simulation Results with CPU consumption information and calculate Full Simulation Results for all the *Tasks* on the added *Processor.* This step analogous to the step 11 from the algorithm "Add Task sequence".

**8: calculateRdr() -** *DynamicManager* uses *DynamicSimulator* in order to obtain Real Deployment Results for the *Tasks* which are already deployed on DGC system.

**9: redeployTask() –** Comparing Full Simulation Results and Full Deployment Results *DynamicManager* decides about each already deployed *Task* whether it should be redeployed to the added *Processor.* I this is the case, *DynamicManager* calls *LegManager* and passes the *Task* that should be redeployed.

**10: refer() –** *LegManager* calls this method for each leg pertaining to the *Task* being redeployed

**11: REFER –** *SipProxy* sends SIP REFER requests to each *Endpoint* communicating with a *Task* being redeployed

# Annex B

# Cloud integrated DGC system design

## B.1 Design

Here we describe several basic use cases of the DGC system composed with the Cloud. To simplify understanding we briefly remind necessary parts of APIs of SIP server and Video processing controller.

Video processing controller:

- addTask(): Video processing controller calculates the node on which the given task can be deployed. The method is called when SIP Server is requested by means of 1PCC or 3PCC to organize a new conference.
- removeTask(): Video processing controller updates its state according to the fact that the given task is removed from the system. The method is called when SIP Server is requested by means of 1PCC or 3PCC to stop the existing conference.

SIP Server:

- deployTask(): method is called when a new task should be deployed. SIP Server called addTask() so Video processing controller returns selected node by means of deployTask() method
- refuseTask(): SIP Server called addTask() to add a new task but Video processing controller can't find a node which is capable to execute a task so task is refused

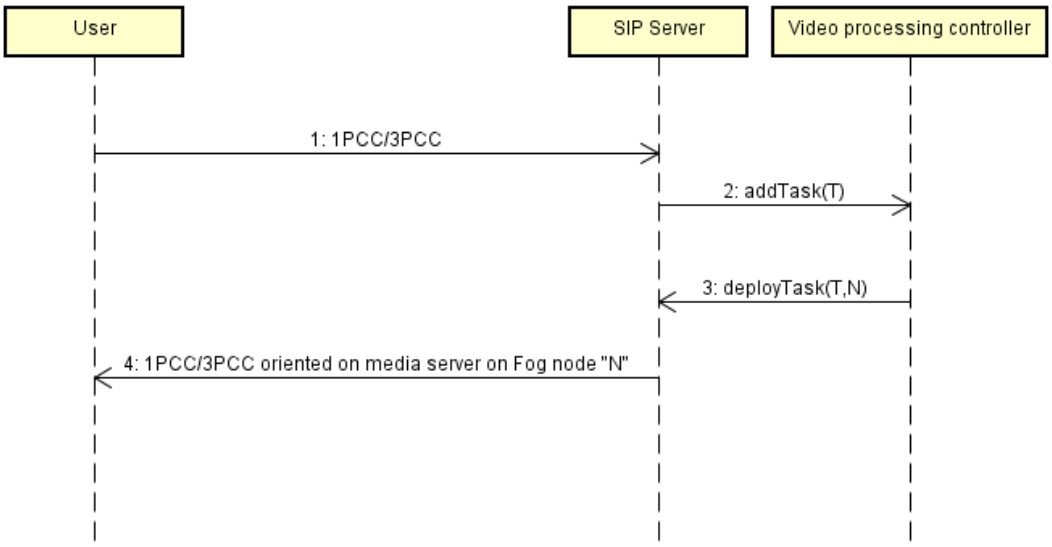


Figure B.1: Task “T” is added to the system and Video processing controller selects Fog node “N” for execution. Then SIP Server notifies conference participants that media server is on the node “N” where they should send their video streams.

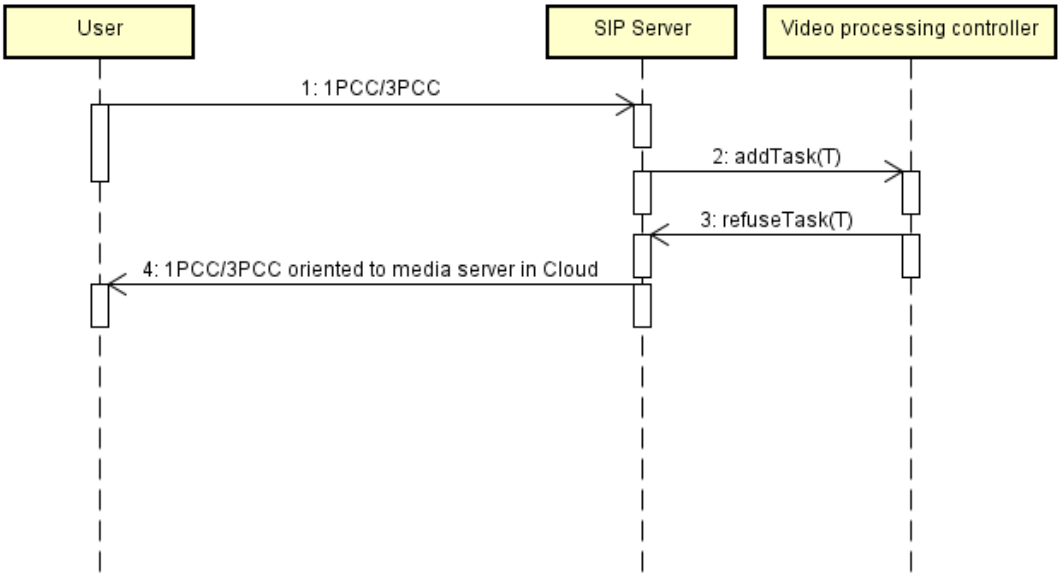



Figure B.2: Task “T” is added to the system but Video processing controller can’t find a Fog node capable to execute the task. It refuses the task so SIP Server notifies conference participants that media server is in the Cloud.

## B.2 Algorithms

Here we will describe the algorithm that allows moving Tasks from Fog to cloud (the cloud can be seen as a particular super node of the Fog) and from Cloud to Fog, in order to minimize the cost of conferences while maintaining a high level of service to the end user.

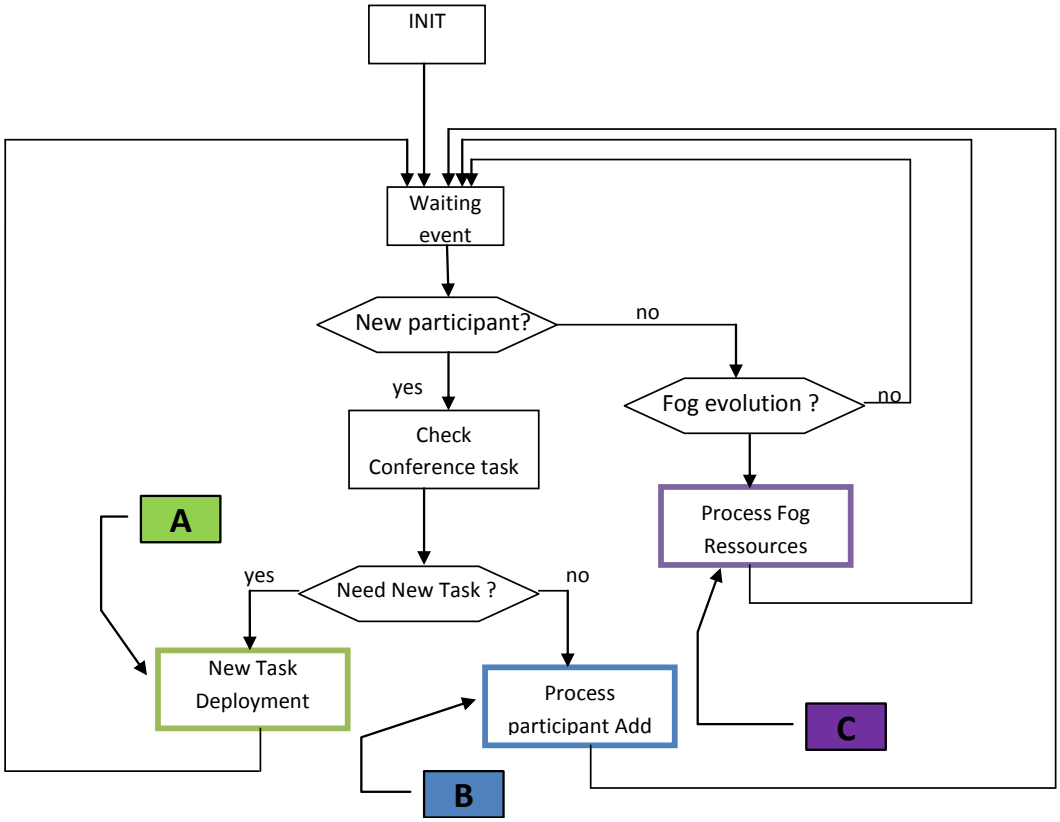



Figure B.3: High level diagram of the algorithm of moving Tasks between Cloud and Fog

The algorithm, presented in Figure B.3, deals dynamically with the evolution of resources. Adding a participant will have an impact on the management of the resources, so this operation is detailed in the algorithm. Removing a participant has low impact. It doesn't require checking if we have enough resources. So this case is not described in the algorithm.

When an event occurs (new participant wants to join a conference or leave a conference, or evolution of Fog resource), the algorithm checks this event and applies the required actions.

If a new participant asks to join a conference, the algorithm checks if a conference is already deployed (there are already participants in this conference) or no (the participant is the first joining this conference). If a new conference is required, a "New Task Deployment: **PHASE A**" is performed, else "Process Conference Add:

**PHASE B**” is performed. If an evolution of Fog resources the “Process Fog Resources: **PHASE C**” is performed.

“New Task Deployment: **PHASE A**”: this part of the algorithm checks if the conference will be started in MCU mode or in SFU mode and where to deploy the conference.

“Process Conference Add: **PHASE B**”: this part of the algorithm checks if there is enough resources to add the participant to the Fog if the conference is deployed in the Fog. If there are no more resources, the conference Task will be moved to the Cloud. If the conference is already in the Cloud, the participant is added to the Cloud conference.

“Process Fog Resources: **PHASE C**”: two main evolutions can occur:

- decrease of resources (a node is not available anymore, …)
- increase of resources (some resources are released, new node available, …)

If there is increase of resources, the algorithm will check if it can move a cloud conference to Fog. If there is decrease of resources, the algorithm will check if it’s necessary to move a conference to the Cloud.

The Phase A and Phase B of the algorithm is triggered by the messages from SIP server when a participant asks to join a conference.

The details of the **PHASE A** “New Task Deployment” are shown in Figure B.4:

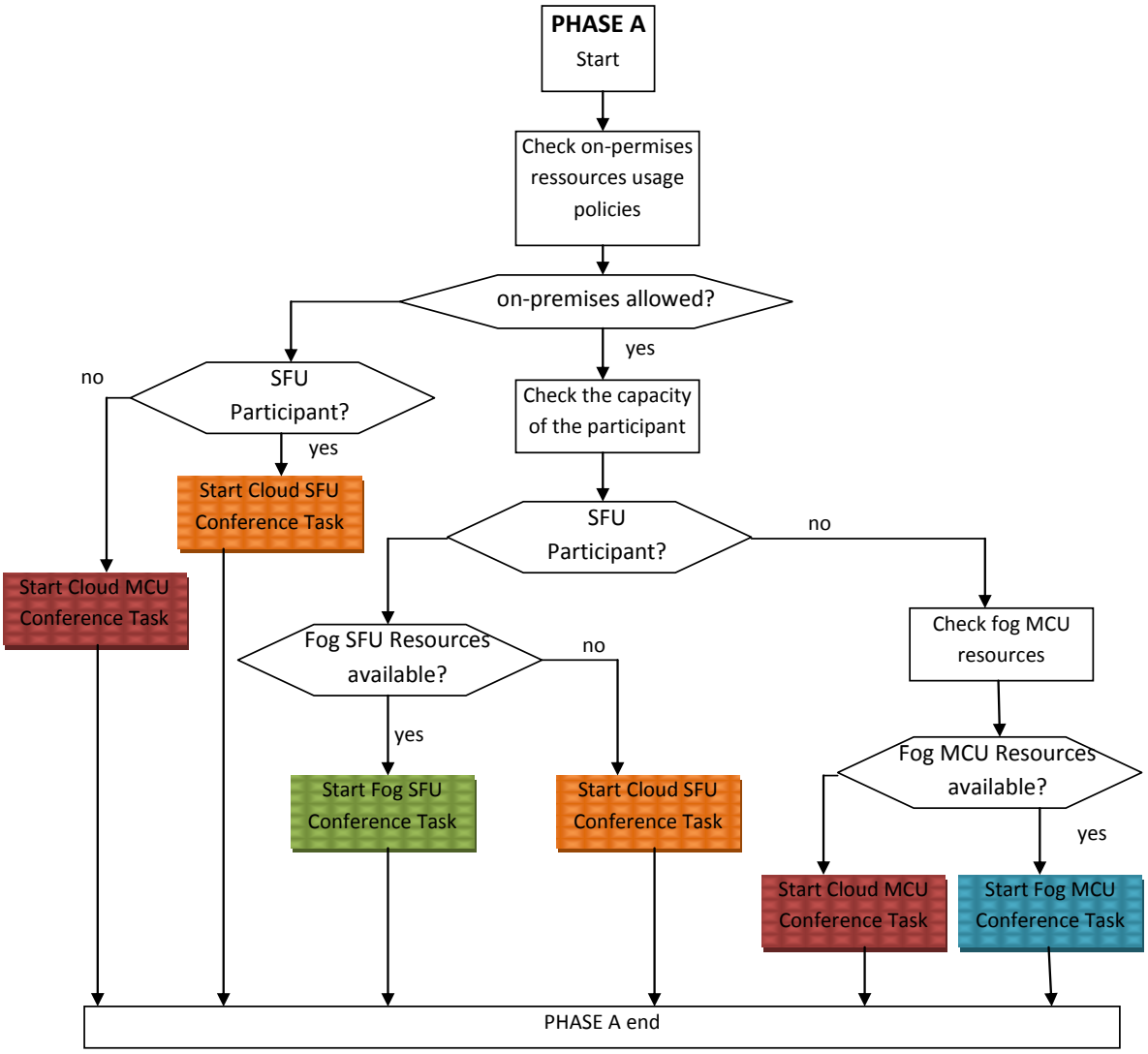



Figure B.4: PHASE A “New Task Deployment” diagram

In this algorithm, the decision to deploy the conference in the Cloud or in the Fog depends on the availability of resources in the Fog. The mode of the conference depends on the capacity of the first participant joining the conference. If the first participant support SFU mode the conference will start using SFU mode, else the conference will start using the MCU mode.

The details of the **PHASE B** “Process Conference Add” are shown in Figure B.5:

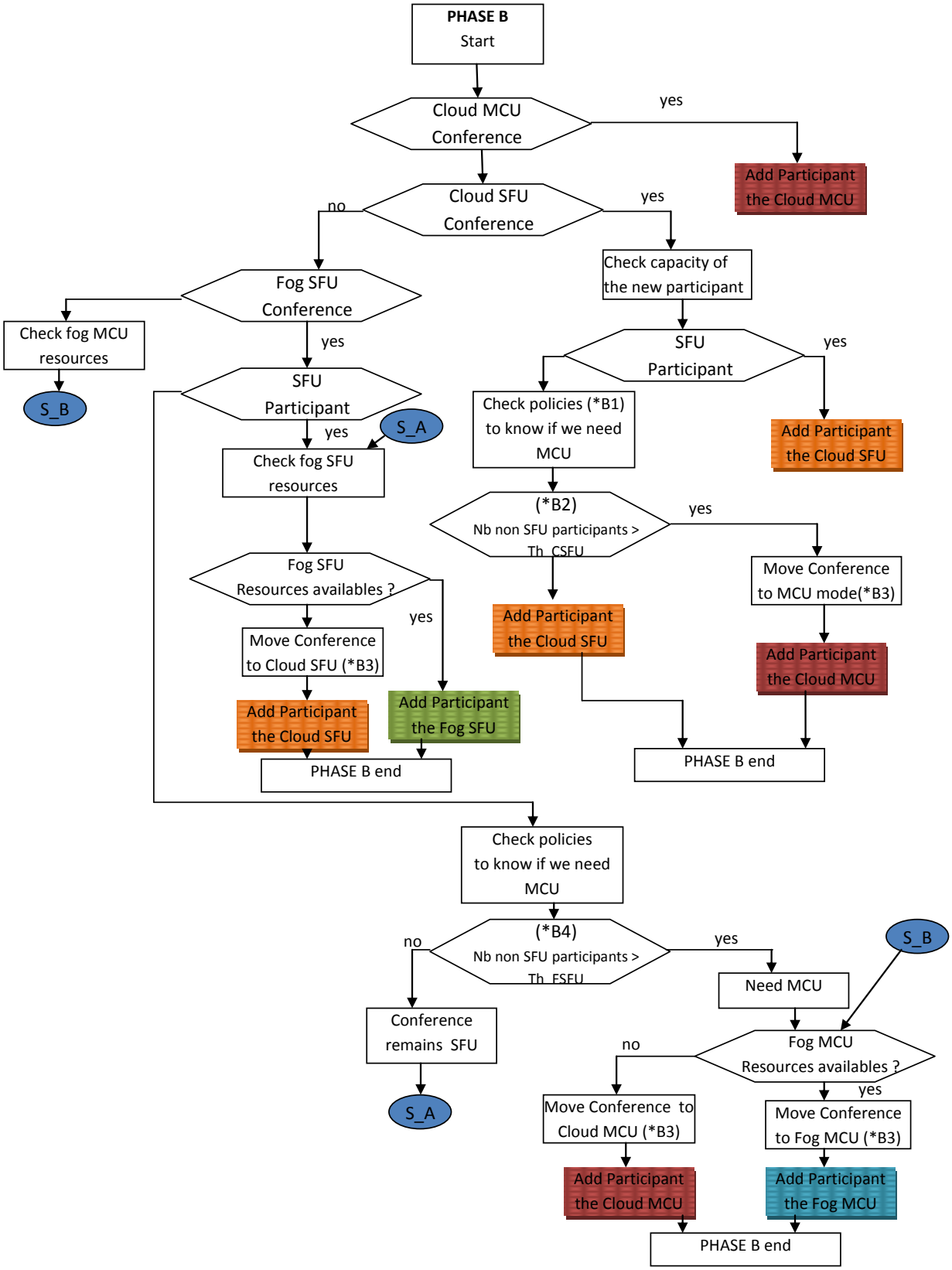



Figure B.5: PHASE B “Process Conference Add” diagram

In Figure B.5:

(*B1): Here the algorithm checks the policy to know if it will switch the conference mode from SFU to MCU when a non SFU participant is connected when the conference is performed in Cloud. The policy can be, for example, if at least one participant has no SFU capabilities, then switch to MCU mode, even if MCU mode is heavy in term of CPU load.

(*B2): Th_CSFU: threshold to switch from Cloud based SFU to Cloud based MCU, which depends on the number of non SFU participants.

(*B3): Move conference. Moving a conference from Fog/Cloud or MCU/SFU is done by the Video Processing Controller. It's in charge of making SIP server to re-invite all participants

(*B4): Th_FSFU: threshold to switch from Fog based SFU to Fog based MCU, which depends on the number of non SFU participants.

The algorithm doesn't describe moving from MCU to SFU when an SFU participant joins the MCU conference or the number of non SFU participants is under the threshold because we do not want to fall back the experience of non SFU users connected to the MCU conference.

The details of the **PHASE C** "Process Conference Add" are shown in Figure B.6. The Phase C of the algorithm is triggered by the message of Monitoring agents of Fog nodes.

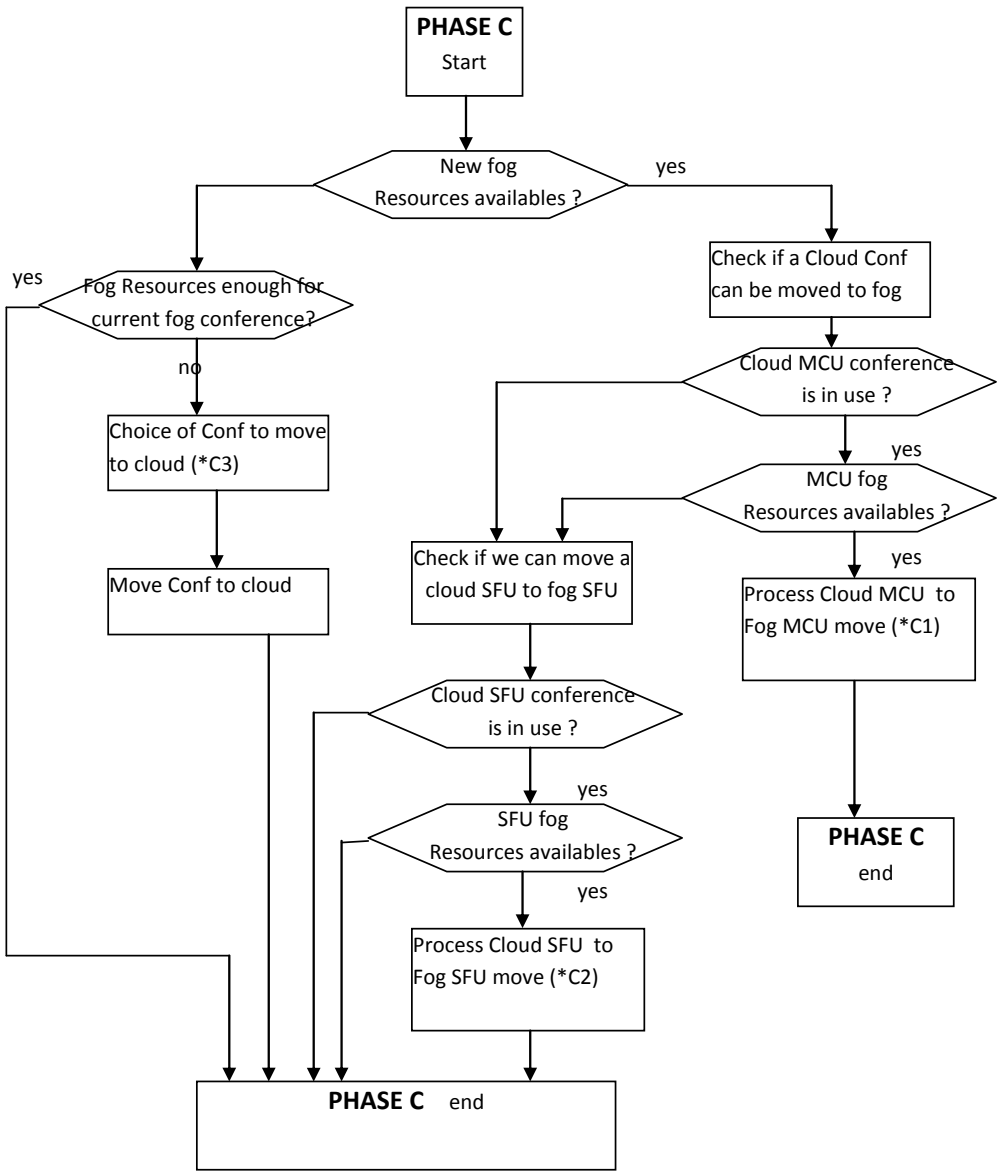



Figure B.6: PHASE C “Process Conference Add” diagram

In Figure B.6:

(*C1): Process Cloud MCU to Fog MCU move:

This block checks which cloud MCU to move to the Fog. It estimates the required CPU resources of all current Cloud MCU conferences.

The algorithm will select a conference that satisfies two criteria:

- CPU resources
- Elapsed time from the last move of the conference from Fog to Cloud

For all the conferences which satisfy resource criterion the time criterion is checked. The conference with the biggest elapsed time is chosen for the move. If

there is no conference which satisfies the time criterion, the algorithm will wait until this criterion is satisfied and check again the resources criterion.

(*C2) Process Cloud SFU to Fog SFU move:

Same behavior as in (*C1), but applied to SFU conference

(*C3) Choice of Conference to move to Cloud:

If a Fog node, on which a conference is executed, is not available anymore, or becomes overloaded, the conference should be moved from this node. If the Fog contains other nodes, which are capable to accept this conference, the conference is moved to this node. If there are no such nodes in the Fog, the conference is moved to the Cloud.

# List of publications

## Published papers

- R. Sorokin, J.-L. Rougier, “Video conference based on enterprise desktop grid”, 23rd International Conference on Telecommunications (ICT), May 2016

## Submitted papers

- R. Sorokin, J.-L. Rougier, R. Pastrana-Vidal, N. Tranquart, "Impact of CPU Load on Video Conferencing Quality", submitted to IEEE International Conference on Communications (ICC) 2017
- R. Sorokin, J.-L. Rougier, “Video Conference in the Fog: An economical approach based on Enterprise Desktop Grid”, submitted to Annals of Telecommunications, Springer
- R. Sorokin, J.-L. Rougier, “IP Video Conferencing: A Tutorial”, submitted to International Journal of Computer Science and Engineering Survey (IJCSES), AIRCC

## Patents

- R. Sorokin, M. Fadili, “A method for allocating a video conferencing task to a processing device”, Deposit number: EP15305454
- R. Sorokin, M. Fadili, S. Coulon, “Methods and nodes for controlling a conference communication”, Deposit number: US 15/076,821

# References


[1] Jin Zeng, Oscar C. Au, Wei Dai, Yue Kong, Luheng Jia, Wenjing Zhu, “A tutorial on image/video coding standards”, Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific, October 2013

[2] “H.264 : Advanced video coding for generic audiovisual services”, ITU-T, May 2003

[3] “H.264 : Advanced video coding for generic audiovisual services”, Annex G, ITU-T, May 2007

[4] “H.265 : High efficiency video coding”, ITU-T, April 2013

[5] J. Bankoski, J. Koleszar, L. Quillio, J. Salonen, P. Wilkins, Y. Xu, “VP8 Data Format and Decoding Guide”, IETF RFC 6386, November 2011

[6] J. Uberti, S. Holmer, M. Flodman, J. Lennox, D. Hong, “RTP Payload Format for VP9 Video”, IETF Work in Progress, March 2016

[7] https://xiph.org/daala/

[8] A. Fuldseth, G. Bjontegaard, S. Midtskogen, T. Davies, M. Zanaty, "Thor Video Codec", IETF Work in Progress, October 2016

[9] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications”, IETF RFC 3550, July 2003

[10] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, “SIP: Session Initiation Protocol”, IETF RFC 3261, June 2002

[11] “H.323 : Packet-based multimedia communications systems”, ITU-T, November 1996

[12] Scott Ludwig, Peter Saint-Andre, Sean Egan, Robert McQueen, Diana Cionoiu, “XEP-0167: Jingle RTP Sessions”, XMPP Standards Foundation, December 2009

[13] E. Burger, J. Van Dyke, A. Spitzer, “Basic Network Media Services with SIP”, IETF RFC 4240, December 2005

[14] J. Rosenberg, “A Framework for Conferencing with the Session Initiation Protocol (SIP)”, IETF RFC 4353, February 2006

[15] A. Saleem, Y. Xin, G. Sharratt, “Media Server Markup Language (MSML)”, IETF RFC 5707, February 2010

[16] J. Van Dyke, E. Burger, A. Spitzer, “Media Server Control Markup Language (MSCML) and Protocol”, IETF RFC 4722, November 2006

[17] M. Barnes, C. Boulton, O. Levin, "A Framework for Centralized Conferencing", IETF RFC 5239, June 2008

[18] T. Melanchuk (editor), "An Architectural Framework for Media Server Control", IETF RFC 5567, June 2009

[19] “JSR 309: Media Server Control API”, Java Community Process, December 2009

[20] R. Presta, S. Romano, "CLUE protocol", IETF Work in Progress, November 2016

[21] M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman, “The Secure Real-time Transport Protocol (SRTP)”, IETF RFC 3711, March 2004

[22] P. Zimmermann, A. Johnston, J. Callas, “ZRTP: Media Path Key Agreement for Unicast Secure RTP” , IETF RFC 6189, April 2011

[23] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, K. Norrman, “MIKEY: Multimedia Internet KEYing”, IETF RFC 3830, August 2004

[24] F. Andreasen, M. Baugher, D. Wing, “Session Description Protocol (SDP) Security Descriptions for Media Streams”, IETF RFC 4568, July 2006

[25] D. McGrew, E. Rescorla, “Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)”, IETF RFC 5764, May 2010

[26] J. Rosenberg, R. Mahy, P. Matthews, D. Wing, “Session Traversal Utilities for NAT (STUN)”, IETF RFC 5389, October 2008

[27] R. Mahy, P. Matthews, J. Rosenberg, “Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)”, IETF RFC 5766, April 2010

[28] J. Rosenberg, “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols”, IETF RFC 5245, April 2010

[29] M. Westerlund, S. Wenger, “RTP Topologies”, IETF RFC 7667, November 2015

[30] Fernando Kuipers, Robert Kooij, Danny De Vleeschauwer, Kjell Brunnström, “Techniques for Measuring Quality of Experience”, 8th International Conference WWIC 2010, Luleå, Sweden, June 2010

[31] “The NIST Definition of Cloud Computing”, National Institute of Standards and Technology, September 2011

[32] http://www.w3.org/TR/webrtc/

[33] C. Holmberg, S. Hakansson, G. Eriksson, "Web Real-Time Communication Use Cases and Requirements", IETF RFC 7478, March 2015

[34] http://www.imtc.org/

[35] http://i3forum.org/

[36] https://software.intel.com/en-us/intel-ipp

[37] http://aomedia.org/

[38] L. M. Vaquero, L. Rodero-Merino, “Finding your Way in the Fog: Towards a Comprehensive Definition of Fog Computing”, ACM SIGCOMM CCR, vol.44, no.5, October 2014

[39] D. Kondo, M. Taufer, C. Brooks, H. Casanova, A. Chien, “Characterizing and Evaluating Desktop Grids: An Empirical Study”, in Proceedings of IPDPS, 2004

[40] G. Novelli, G. Pappalardo, C. Santoro, E. Tramontana, “A Grid-based Infrastructure to Support Multimedia Content Distribution”, in Proceedings of the second workshop on Use of P2P, GRID and agents for the development of content networks, pp. 57–64, 2007

[41] S. Chakraborty, C.H. Yen, “A Simulation Based Comparative Study of Normalization Procedures in Multiattribute Decision Making”, Proceedings of the 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Vol.6, pp 102–109, 2007

[42] K. Yoon, C.L. Hwang, "Multiple Attribute Decision Making Introduction", Sage Publication, 1995

[43] A. Alalousi, A. Osman, S. Noori, A. Hussain, A. Munther, H. El-Taj, “A Study on Video Conferencing using Overlay Network”, European Journal of Scientific Research, Vol.59, Issue 3, September 2011

[44] M. Chen, M. Ponec, S. Sengupta, J. Li, P. Chou, “Utility Maximization in Peer-to-Peer Systems With Applications to Video Conferencing”, IEEE/ACM ToN, Vol.20, No.6, December 2012

[45] X. Yu, Z. Yu, “A Distributed Architecture of Video Conference Using P2P Technology”, Journal of Networks, vol.7, no.11, November 2012

[46] A. Munther, S. Noori, A. Osman, A. Hussain, I. Jasim, A. Shanoon, “Peer-to-Peer Video Conferencing Using Hybrid Content Distribution Model”, Journal of Computer Science, Vol. 8, No.7, 2012

[47] “G.114 : One-way transmission time”, ITU-T G.114, May 2003.

[48] “P.910 : Subjective video quality assessment methods for multimedia applications”, ITU-T P.910, April 2008.

[49] http://openmcu.ru/eng.htm

[50] http://www.linphone.org/

[51] http://mion.faireal.net/BES/

[52] http://www.cpukiller.com/

[53] R. Pastrana-Vidal et al., “Sporadic Frame Dropping Impact on Quality Perception”, Human Vision and Electronic Imaging IX, SPIE, San José, 2004

[54] R. Pastrana-Vidal, J.-C. Gicquel, “Looking at the relationship between video bitrates and end-user quality assessment: subjective tests approach”, ETSI Workshop Effect of transmission performance on Multimedia Quality Service, Prague, 2008

[55] ITU-R BT.1788, 207. SAMVIQ – Subjective Assessment Methodology for Video Quality, Recommendation, ITU, January 2007

[56] A. Azzazi, H. Abusaimeh, S. R. Masadeh, “CPU Utilization for a Multiple Video Streaming Over a Fiber Optic ATM-Network when Varying the Quality of Service”, Journal of Emerging Trends in Computing and Information Sciences, Vol.5, No.3, March 2014

[57] M. Goudarzi, L. Sun, E. Ifeachor, “Impact of Bursty Packet Loss on Voice and Video Quality in Wireless Networks”, Postgraduate Conference for Computing: Applications and Theory (PCCAT), June 2010

[58] A. Tarakanov, O. Gushchina, I. Nenakhov, “Influence of Packets Losses on Video Quality in Case of Using Multiple Description Coding with Time Division into Two and Three Substreams”, Proceedings of the 17th conference of FRUCT association, April 2015

[59] http://www.perceptiva-labs.com/

[60] S. Murphy, M. Searles, C. Rambeau, L. Murphy, “Evaluating the Impact of Network Performance on Video Streaming Quality for Categorised Video Content”, 14th International Packet Video Workshop, 2004.

[61] A. Kwon, J. Xiao, S. Seo, JWK Hong, R. Boutaba, “The Impact of Network Performance on Perceived Video Quality in H.264/AVC”, 2012 IEEE Network Operations and Management Symposium, 2012

[62] M. Claypool, J. Tanner, “The Effects of Jitter on the Perceptual Quality of Video”, ACM Multimedia, Vol.2, November 1999

[63] A. Khan, L. Sun, E. Ifeachor, “Impact of Video Content on Video Quality for Video over Wireless Networks”, 5th International Conference on Autonomic and Autonomous Systems (ICAS 2009), April 2009

[64] R. Alimi, R. Penno, Y. Yang (editors), “Application-Layer Traffic Optimization (ALTO) Protocol”, IETF RFC 7285, September 2014.

[65] T. S. Ng, H. Zhang, “Predicting Internet Network Distance with Coordinates-Based Approaches”, INFOCOM’02, June 2002

[66] F. Dabek, R. Cox, F. Kaashoek, R. Morris, “Vivaldi: A Decentralized Network Coordinate System”, ACM SIGCOMM, August 2004.

[67] M. Costa, M. Castro, R. Rowstron, P. Key, “PIC: practical Internet coordinates for distance estimation”, 24th International Conference on Distributed Computing Systems, 2004

[68] V. Ramasubramanian, D. Malkhi, F. Kuhn, I. Abraham, M. Balakrishnan, A. Gupta, A. Akella, “A Unified Network “Coordinate” System for Bandwidth and Latency”, Technical Report MSR-TR-2008-124, Microsoft Research, September 2008

[69] H. Lim, J.C. Hou, C. Choi, "Constructing internet coordinate system based on delay measurement", ACM IMC, October 2003

[70] Y. Chen, Y. Xiong, X. Shi, J. Zhu, B. Deng, X. Li, “Pharos: accurate and decentralised network coordinate system”, IET Communications, Vol. 3, Issue 4, April 2009

[71] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, L. Zhang, “IDMaps: A Global Internet Host Distance Estimation Service”, IEEE/ACM Transactions on Networking, Vol. 9, Issue 5, October 2001

[72] B. Wong, A. Slivkins, E. G. Sirer, “Meridian: A Lightweight Network Location Service without Virtual Coordinates”, ACM SIGCOMM, Vol. 35 Issue 4, October 2005

[73] M. J. Freedman, K. Lakshminarayanan, D. Mazières, “OASIS: Anycast for Any Service”, NSDI'06 Proceedings of the 3rd conference on Networked Systems Design & Implementation, Vol. 3, 2006

[74] http://desmoj.sourceforge.net/

[75] http://www.jfree.org/jfreechart/

[76] L. Chao, Z. Miao, L. Yong, “Optimal Bandwidth Sharing in Multiswarm Multiparty P2P Video-Conferencing Systems”, IEEE/ACM Transactions On Networking, Vol.19, No.6, December 2011

[77] T. Ruso, C. Prabhu, C. Chellappan, “NetRawALM: Network Based Resource Aware Application Layer Multicast for Multiparty Video Conference”, International Journal of Distributed and Parallel Systems, Vol.2, No.5, September 2011

[78] E. Karrupiah, E. Lin, T. Phan, N. Thoai, E. Muramoto, P. Tan, “Bandwidth Fair Application Layer Multicast for Multi-Party Video Conference Application”, Consumer Communications and Networking Conference, 2009

[79] A. Abbasi, T. Mehmood, “Dynamic Scalable Model for Video Conferencing (DSMVC) using Request Routing”, International Journal of Video & Image Processing and Network Security, Vol.9, Issue 9, October 2009

[80] X. Wu, K. Dhara, V. Krishnaswamy, “Enhancing Application-Layer Multicast for P2P Conferencing”, Consumer Communications and Networking Conference, 2007

[81] P. Yuen, G. Chan, “MixNStream: multi-source video distribution with stream mixers”, Proceedings of the 2010 ACM workshop on Advanced video streaming techniques for peer-to-peer networks and social networking, New York, USA, 2010

[82] W. Ooi, R. Renesse, “Distributing Media Transformation Over Multiple Media Gateways”, Proceedings of the ninth ACM international conference on Multimedia, 2001

[83] D. Maggiorini, D. Riboni, “Continuous Media Adaptation for Mobile Computing Using Coarse-Grained Asynchronous Notifications”, SAINT-W'05 Proceedings of the 2005 Symposium on Applications and the Internet Workshops, 2005

[84] S. E. Restrepo, P. Pinaud, J. E. Pezoa, S. Sobarzo, “Energy-aware Image Allocation for Distributed Video Processing on Handheld Devices”, IEEE 32nd International Performance Computing and Communications Conference (IPCCC), December 2013

[85] S. Mohapatra, R. Cornea, N. Dutt, A. Nicolau, N. Venkatasubramanian, “Integrated Power Management for Video Streaming to Mobile Handheld Devices”, Proceedings of the 11th ACM international conference on Multimedia, 2003

[86] R. Lienhart, I. Kozintsev, Y.-K. Chen, M. Holliman, M. Yeung, A. Zaccarin, R. Puri, "Challenges in Distributed Video Management and Delivery", Handbook of

Video Databases Design and Applications, Edited by John C . Munson, CRC Press, 2003

[87] M. Kim, Y. Cui, H. Lee, H. Lee, “Performance Evaluation of a Hadoop-based Distributed Video Transcoding System for Mobile Media Service”, Proceedings of IST 2012 International Conference, April 2012

[88] M. Hossain, J. Khan, “Dynamic MCU Placement for Video Conferencing on Peer-to-Peer Swarm”, 2015 IEEE International Symposium on Multimedia, December 2015

[89] P. Troubil, H. Rudova, P. Holub, “Media Streams Planning with Transcoding”, 12th IEEE International Symposium on Network Computing and Applications, August 2013

[90] Yu Wu, Chuan Wu, Bo Li, Francis C.M. Lau, “vSkyConf: cloud-assisted multi-party mobile video conferencing”, Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing, August 2013

[91] Yuan Feng, Baochun Li, Bo Li, “Airlift: Video Conferencing as a Cloud Service using Inter-Datacenter Networks”, Proceedings of 20th IEEE International Conference on Network Protocols, October 2012

[92] Chao Liang, Yong Liu, “Optimal Resource Allocation in Multi-Source Multi-Swarm P2P Video Conferencing Swarms”, accepted for publication in IEEE/ACM Trans. on Networking, 2011

[93] Yongxiang Zhao, Yong Liu, Changjia Chen, Jianyin Zhang, “Enabling P2P One-View Multiparty Video Conferencing”, IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 1, January 2014

[94] Han Zhao, Daniel Smilkov, Paolo Dettori, Julio Nogima, Frank A. Schaffa, Peter Westerink, Chai Wah Wu, “A Feasibility Study of Collaborative Stream Routing in Peer-to-Peer Multiparty Video Conferencing”, IEEE International Symposium on Multimedia, December 2011

[95] Daniel Smilkov, Han Zhao, Paolo Dettori, Julio Nogima, Frank A. Schaffa, Peter Westerink, Chai Wah Wu, “Non-intrusive Adaptive Multi-media Routing in Peer-to-Peer Multi-party Video Conferencing”, IEEE International Symposium on Multimedia, December 2010

[96] Li Xin, Guan Jianfeng, Zhang Hongke, “Distortion Optimized Mobile Multiparty Video Conferencing”, International Conference on Communications and Mobile Computing, January 2009

[97] Dhiman Chattopadhyay, Aniruddha Sinha, T. Chattopadhyay, “A Low Cost Multiparty H.264 Based Video Conference Solution for Corporate Environment”,

International Conference on Computational Intelligence and Communication Networks, November 2010

[98] Xuan Zhang, Chongrong Li, Xing Li, “Multi-party Videoconferencing Based on Hybrid Multicast with Peer-Forwarding”, IEEE 16th International Conference on Parallel and Distributed Systems, December 2010

[99] M. Ponec, S. Sengupta, Minghua Chen, Jin Li, P.A. Chou, “Optimizing Multi-Rate Peer-to-Peer Video Conferencing Applications”, IEEE Transactions on Multimedia, Vol.13, Issue: 5, October 2011

[100] S. Nari, H. R. Rabiee, A. Abedi, M. Ghanbari, “An Efficient Algorithm for Overlay Multicast Routing in Videoconferencing Applications”, Proceedings of 18th International Conference on Computer Communications and Networks, August 2009

[101] Xiangwen Chen, Minghua Chen, Baochun Li, Yao Zhao, Yunnan Wu, Jin Li, “Celerity: a low-delay multi-party conferencing solution”, Proceedings of the 19th ACM international conference on Multimedia, November 2011

[102] Zhi Wang, Jizhong Zhao, Wei Xi, Zhiping Jiang, “A Scalable P2P Video Conferencing System Based on VCStream Model”, 11th International Conference on Computer and Information Science, May 2012

[103] K. Singh, V. Krishnaswamy, “Building Communicating Web Applications Leveraging Endpoints and Cloud Resource Service”, IEEE 6th International Conference on Cloud Computing, June 2013

[104] Fang Zhiyuan, Li Wei, Feng Zhang, Zhou Fang, Donghang Huang, Xin Dai, “A Cloud-Based Pan-Terminal Video Conferencing System”, IEEE 10th International Conference on e-Business Engineering, September 2013

[105] Tien Anh Le, Hang Nguyen, “Application-aware cost function and its performance evaluation over scalable video conferencing services on heterogeneous networks”, IEEE Wireless Communications and Networking Conference, April 2012

[106] D. Ben Khedher, “A Peer-to-Peer self-organizing scheme for multiparty session”, IEEE International Conference on Communications, June 2012

[107] HaiYan Liu, Zhiyuan An, Yangang Lv, “Audio-video conference systems design and implementation base on P2P and multicast”, 2011 International Conference on Electronics, Communications and Control, September 2011

[108] Tien Anh Le, Hang Nguyen, “Perception-Based Application Layer Multicast Algorithm for Scalable Video Conferencing”, IEEE Global Telecommunications Conference, December 2011

[109] Tien Anh Le, Hang Nguyen, “Centralized and distributed architectures of scalable video conferencing services”, Second International Conference on Ubiquitous and Future Networks, June 2010

[110] Istemi Ekin Akkus, Oznur Ozkasap, M. Reha Civanlar, “Peer-to-peer multipoint video conferencing with layered video”, Journal of Network and Computer Applications, no.34, 2011

[111] Tien Anh Le, Hang Nguyen, “Perception-based Application Layer Multicast Algorithm for scalable video conferencing”, IEEE Global Telecommunications Conference, December 2011

[112] Tien A. Le, H. Nguyen, “Human perception-based distributed architecture for scalable video conferencing services: theoretical models and performance”, Annals of telecommunications, Vol. 69, Issue 1-2, February 2014

[113] I. Rimac, V. Hilt, M. Tomsu, V. Gurbani, E. Marocco, "A Survey on Research on the Application-Layer Traffic Optimization (ALTO) Problem", IETF RFC 6029, October 2010

[114] D. Finstad, H. Stensland, H. Espeland, P. Halvorsen, "Improved Multi-Rate Video Encoding", Proceedings of the IEEE International Symposium on Multimedia (ISM), December 2011