# Édition Coopérative et Large Échelle dans Duplex [1]

François Pacull Alain Sandoz André Schiper

Ecole Polytechnique Fédérale de Lausanne
Département d'Informatique, Laboratiore de Systèmes d'Exploitation
CH-1015 Lausanne (Suisse)
{Pacull,Sandoz,Schiper}@lse.epfl.ch

**Mots-clés:** Édition coopérative, réseau à large échelle, contrôle de concurrence.

**Résumé**

Duplex est un environnement d'édition coopérative pour des utilisateurs connectés sur Internet. La large échelle implique l'hétérogénéité, des délais de communication non bornés et des pannes. Dans ce cadre, les techniques traditionnelles utilisées pour l'édition coopérative dans un réseau local sont inefficaces. Duplex propose un modèle basé sur une décomposition du document en parties indépendantes maintenues individuellement par un noyau dupliqué. Le traitement de l'atomicité et du contrôle de duplication devient réalisable puisque confiné à un petit contexte. Les utilisateurs interagissent avec le noyau soit en récupérant une copie locale d'un objet du noyau qu'ils peuvent modifier à loisir dans leur environnement local, soit au contraire en répercutant au niveau du noyau une copie locale. Les communications sont réduites au minimun autorisant une utilisation à large échelle. Le contrôle de concurrence offre des mécanismes optimistes ou pessimistes pour chaque objet individuellement tout en garantissant une cohérence forte (la linéarisabilité).

**Keywords:** Collaborative éditing, large scale network, concurrency control.

**Abstract**

Duplex is a collaborative editor for users connected through the Internet. Large scale implies heterogeneity, unpredictable communication delay and failures La large échelle implique l'hétérogénéité, des délais de communication non bornés et des pannes and leads to inefficient implementation of techniques traditionally used for collaborative editing in local area networks. Duplex proposes a model based on splitting the document into independant parts, maintainedindividually and replicated by a kernel. Atomicity, concurrency, and replica control are confined to a manageable small context. Users interact with the kernel either by copying locally a kernel object that they can modify in their own local environment, or by flushing the modified object to the kernel. Communication is reduced to a minimum, allowing the use in a large scale distributed system. Concurrency control offers optimistic and pessimistic mecanisms for each individual object with enforcing a strong consistency (linearizability).

[1] Recherche financée par le FNS et l'OFES sous le contrat numéro 21-32210.91, dans le cadre du projet européen ESPRIT Basic Research numéro 6360 (BROADCAST).

# 1 Introduction

L'édition coopérative comprend non seulement des activités d'écriture mais encore des échanges d'information entre les co-auteurs. Ainsi un environnement d'édition coopérative est la combinaison de plusieurs outils[15]: un *éditeur coopératif* qui permet de maintenir l'état du document partagé, des *facilités de communications directes* (les destinataires d'un message sont sélectionnés par l'émetteur) ou *indirectes* (les destinataires sont ceux qui ont un interêt pour un sujet particulier) qui permettent un échange d'informations entre les co-auteurs. Chacun de ces outils permet des interactions *synchrone* (en temps-réel) ou *asynchrone*[20]. Lors d'une interaction synchrone, les modifications produites peuvent être observées en temps-réel par tous les membres de la collaboration (GROVE [5], DistEdit[9], Mule[19] proposent un éditeur coopératif synchrone).

L'environnement DUPLEX[18] incorpore un éditeur coopératif offrant une interaction asynchrone à travers un noyau partagé qui maintient les informations relatives au document. Une interaction synchrone, quand elle est nécessaire, est réalisée grâce aux facilités de communication. En effet, la phase initiale de planification du document ou la phase finale de polissage exigent des communications directes ou indirectes intensives plutôt que des facilités d'édition en temps-réel. Dans un second temps, les résultats de ces phases sont fusionnés avec le document en utilisant l'éditeur coopératif. En restreignant l'éditeur coopératif à des interactions asynchrones, un environnement puissant peut être offert aux utilisateurs sans pour autant les pénaliser par l'inhérente inefficacité de la communication et de la distribution dans une large échelle.

Parmi les éditeurs asynchrones décrits dans la littérature, le partage du document peut être réalisé soit à travers un stockage centralisé (Quilt[6], MultimETH[13], Prep[16], Mule[19]), soit à travers une duplication *locale* des données sur le site même des utilisateurs (IRIS[2],CES[7]). DUPLEX propose une solution hybride où le partage est basé (1) sur un noyau dupliqué qui maintient les informations partagées indépendamment de la localisation des utilisateurs, et (2) des caches locaux et des politiques de traitement de ceux-ci afin d'optimiser le contrôle de concurrence et le temps de réponse.

Le contrôle des accès concurrents aux données partagées est bien entendu un aspect important de l'édition coopérative. Trois approches ont été proposées: (1) le contrôle est assuré par le système, basé sur une unité de concurrence définie à partir d'une modélisation appropriée du document[2, 7, 13]; (2) le contrôle est assuré par le système, basé sur l'attribution aux utilisateurs de responsabilités sur certaines parties du document (rôles[11]) et (3) il n'y a pas de contrôle: la gestion se base sur l'auto-discipline des utilisateurs qui utilisent leurs perceptions des conflits potentiels[5].

L'approche de DUPLEX les intègre toutes les trois d'une manière flexible. Le document est décomposé en parties indépendantes comme dans l'IRIS[2], CES[7], et MultimETH [13]. Cette décomposition suit la structure du document, elle est dynamique et se base à la fois sur l'état courant du document et la responsabilité des auteurs sur les différentes parties. Cet aspect dynamique autorise un contrôle de concurrence adaptatif permettant aux auteurs de choisir le type de contrôle (exclusif, pessimiste, optimiste, etc.) qu'ils désirent employer sur les parties du document qui les concernent.

Le modèle de base de DUPLEX est présenté dans la section 2. La décomposition du document est décrite dans la section 3 et la structure et la mise en oeuvre du noyau est présenté dans la section 4. Une discussion conclut le papier.

# 2 Le Modèle Duplex

Cette section présente une vue d'ensemble de notre environnement distribué d'édition coopérative. Il présente le système distribué sous-jacent, le modèle de base de DUPLEX et les implications de ce modèle sur (1) la tolérance aux pannes, (2) le contrôle de concurrence ainsi que (3) les aspects grande échelle.

## 2.1 Le modèle de système distribué

Le système distribué que nous considérons est composé d'un ensemble de noeuds dans un réseau à grande échelle. Un noeud est soit la station de travail d'un utilisateur, soit un site de stockage pour les données correspondant à une partie du document, soit les deux. L'ensemble des noeuds évolue dynamiquement au fur et à mesure que les utilisateurs rejoignent ou quittent la collaboration ou que des parties du document sont ajoutées ou enlevées. Les noeuds que nous considérons, dans notre modèle, sont ceux qui sont impliqués par l'édition d'un même document; les noeuds impliqués par des documents différents sont gérés de façon indépendante.

La sémantique de panne d'un noeud que nous considérons est "crash-failure"[3] (i.e. toutes les activités d'un noeud, y compris la communication, cessent lors d'une panne). La communication entre les noeuds est point-à-point, asynchrone (délais de transmission non bornés) et non fiable[2]. Plus précisement, nous considérons un système dont les noeuds sont connectés à travers Internet, ce qui constitue, dans l'état actuel de la technologie, le plus large réseau possible. Ceci empêche la mise en oeuvre efficace des approches synchrones. De plus dans le contexte du travail coopératif à large échelle, l'interaction entre utilisateurs travaillant dans des fuseaux horaires différents suggèrent plutôt un mode opératoire asynchrone.

Trois problèmes majeurs ont une influence sur la disponibilité et les fonctionnalités du système: (1) gérer les pannes de sites et de communications, (2) contrôler la concurrence des opérations sur les objets partagés et (3) intégrer le système dans un environnement à large l'échelle.

## 2.2 Le modèle de base de DUPLEX

DUPLEX propose trois concepts fondamentaux:

Le premier concerne la *décomposition du document* qui spécifie un ensemble de règles pour la partition du document en parties *indépendantes*. Ceci a pour but la réduction de la granularité, offrant une meilleure concurrence d'accès pour les utilisateurs et une minimisation des effets d'une panne du système[3]. Le partitionnement du document est dicté par les rôles et les responsabilités des co-auteurs dans la coopération, ce qui permet de réduire la probabilité que plusieurs auteurs aient à travailler sur les même données concurremment.

Le second est l'existence d'un *noyau*, partagé par tous les membres de la collaboration, qui pourvoit à la persistance et la disponibilité de l'état le plus récent du document. Le noyau est composé d'*objets* (un par partie indépendante) sur lequel les utilisateurs vont invoquer des opérations. Les objets du noyau sont dupliqués afin d'assurer leur disponibilité même en cas de défaillance, ainsi qu'un meilleur accès.

[2] Les liens de communications peuvent être inopérants et le réseau peut être partitioné en sous-réseaux déconnectés.

[3] L'indisponibilité du document entier n'est plus possible.

Le troisième est un environnement *local* à l'utilisateur qui offre un espace de stockage autonome. L'environnement d'un utilisateur contient un ensemble de copies des objets du noyau qui compose sa *vue* locale du document. Une copie locale n'est pas maintenue comme un des duplicas d'un objet du noyau mais une copie propre à l'utilisateur. Ce dernier travaille localement à partir des objets de sa vue; il est déconnecté du noyau jusqu'à ce que (1) une copie d'un objet noyau soit exigée pour continuer le travail (elle est copiée dans la vue locale à partir du noyau) ou (2) l'utilisateur considère qu'un objet de sa vue locale peut être partagé (l'objet noyau correspondant est mis à jour à partir de la vue locale). Ainsi les utilisateurs travaillent indépendamment dans leur environnement local et n'interagissent avec le noyau que lorsque c'est nécessaire.

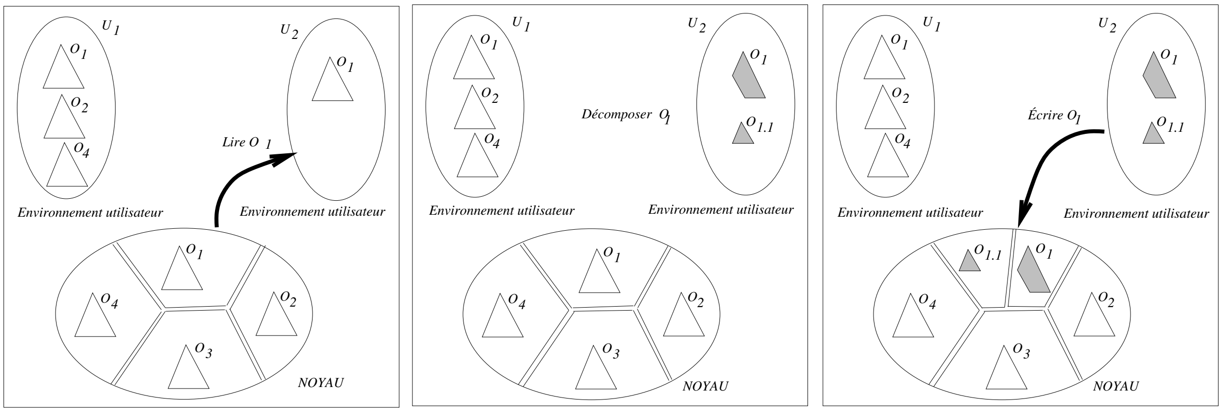


FIG. 1 - le noyau et l'environnement utilisateur

La figure 1 illustre ceci. La vue du document pour l'utilisateur $u_1$ est composée des copies des objets $o_1$, $o_2$, $o_4$. L'utilisateur $u_2$, désirant travailler sur l'objet $o_1$, copie l'objet depuis le noyau dans son environnement local (fig. 1-a). L'utilisateur $u_2$ modifie l'objet $o_1$ en le décomposant (voir plus loin) en deux objets locaux $o_1$ et $o_{1.1}$ (fig. 1-b). Finalement $u_2$ met à jour $o_1$ au niveau du noyau mettant ainsi à disposition les deux objets partagés $o_1$ et $o_{1.1}$ (fig. 1-c). La vue locale d'un utilisateur peut donc différer du noyau de plusieurs façons. Premièrement, elle peut être incomplète dans le sens où certains objets du noyau sont absents de la vue de l'utilisateur. (e.g. la vue de $u_1$ ne contient pas $o_3$). Deuxièmement, un utilisateur peut localement modifier les copies des objets (e.g. vue de $u_2$ dans la fig. 1-b). Finalement, une modification de l'état du noyau peut rendre obsolètes les vues locales des autres utilisateurs (e.g. vue de $u_1$ dans fig. 1-c).

Autoriser une divergence entre l'état du noyau et les vues locales est adapté aux interactions asynchrones entre les utilisateurs dans un environement à large échelle où il est coûteux d'assurer la consistence mutuelle. La vue locale d'un utilisateur peut être archivée localement sur son site, soit explicitement sur une requête de ce dernier, soit périodiquement, soit quand l'utilisateur quitte sa session de travail. Ainsi s'il arrivait qu'un objet du noyau soit modifié par un utilisateur $u_2$, sans qu'un utilisateur $u_1$ ayant travaillé sur cet objet précédemment n'en soit averti, les vues antérieures de $u_1$ lui permettraient de procéder à une fusion (plus ou moins automatique) entre son propre travail et les modifications apportées par $u_2$. La gestion des différences entre vues locales et le noyau est basée à la fois sur des outils au niveau du noyau (e.g. contrôle de concurrence, journalisation) et au niveau de l'environnement local des utilisateurs (e.g. archivage des vues, visualisation, dialogue).

## 2.3 Implications du modèle de base

### 2.3.1 Tolérance aux pannes.

Grâce à la séparation en deux niveaux (noyau et environnement utilisateur), les pannes dans le système distribué peuvent être proprement décomposées et leur impact sur l'application d'édition coopérative peut être confiné. Nous distinguons[4] les *noeuds du noyau* qui maintiennent un duplica d'au moins un objet du noyau, et *les noeuds utilisateurs* qui maintiennent l'environnement local d'au moins un utilisateur.

*La panne d'un noeud du noyau.* Les objets du noyau sont dupliqués afin de tolérer les pannes. Les noeuds correspondant à un objet noyau sont maintenus comme un groupe[1, 12] et les politiques de duplication sont mises en oeuvre au-dessus de ce concept du groupe. En particulier, assurer l'atomicité des opérations du noyau et le traitement de la partition du groupe de duplicas gérant un objet noyau sont traités dans le contexte de groupe[23, 24].

*La panne de communication entre les noeuds noyau et utilisateur.* Deux types de pannes peuvent avoir lieu. (1) une panne qui survient durant une opération avec le noyau, qui est traitée en imposant l'atomicité des opérations du noyau, (2) une partition responsable d'une perte de liaison entre les noeuds utilisateur et noyau, impliquant que, pour un certain laps de temps, un utilisateur se trouvera dans l'impossibilité d'accéder à certains objets du noyau. Ceci n'empêche nullement l'utilisateur de travailler sur les copies de son environnement ou bien d'atteindre d'autres objets du noyau.

*La panne d'un noeud utilisateur.* Ceci a lieu lors d'une panne matérielle ou logicielle du noeud d'un utilisateur (e.g. son système de fichier est inopérant). Si cela survient durant une opération avec le noyau, la tolérance aux pannes garantit l'atomicité de l'opération. Ainsi, les effets sont confinés au noeud de l'utilisateur et la panne n'affecte pas le travail des autres utilisateurs.

### 2.3.2 Le contrôle de concurrence

Le problème de la consistance des données se pose quand plusieurs utilisateurs font une requête sur le même objet du noyau de façon concurrente. Cette situation est inhérente à toute application distribuée utilisant des données partagées[25] notamment dans le contexte dupliqué[21]. Résoudre le problème de la consistance exige (1) de définir un critère de consistance pour les requêtes concurrentes, (2) d'adapter le critère au contexte dupliqué, et finalement, (3) d'offrir un outil de contrôle de concurrence et une politique de gestion de la duplication en conséquence. En fonction de l'application et des caractéristiques des pannes considérées dans le système sous-jacent, différentes politiques peuvent être utilisées (pessimiste vs. optimiste, ou syntaxique vs. adaptée à la sémantique de la requête[4, 10])

Dans DUPLEX, le problème de la consistance est résolu par l'introduction d'une décomposition du document réduisant l'occurrence d'opérations qui rentrent en conflit. Le critère de consistance finit un ordonnancement des opérations sur tout le noyau. Finalement, différentes politiques de contrôle de concurrence (pessimiste ou optimiste) adaptées à l'édition coopérative sont proposées aux utilisateurs, afin de leur permettre d'utiliser celle qui est la plus adaptée à la situation courante.

### 2.3.3 Intégration de la grande échelle

L'intégration de la grande échelle pose deux problèmes: (1) les délais de communication importants pénalisent les opérations entre les noeuds distants et (2) le système utilisé dans

[4] Bien qu'un noeud peut être à la fois noyau et utilisateur, les deux fonctions sont indépendantes

le contexte du document est noyé dans un très vaste système distribué.

Le premier problème est résolu en maintenant localement pour chaque auteur un riche environnement lui permettant de travailler la majeure partie du temps indépendamment du noyau. De plus, par un choix judicieux des noeuds maintenant les objets du noyau, on accélère à la fois les communications et les opérations à distance.

Le second problème est plus délicat et nécessite un service de noms gérant l'ensemble des objets du noyau distribués sur un large réseau hétérogène. Cet ensemble est petit et composé d'objets, dont les noms sont contextuels au document, alors que les duplicas sont maintenus par des systèmes de fichiers hétérogènes, dont le schéma global de désignation est indépendant de l'espace de nom de la collaboration. Ceci exige un service de noms dédié spécialement défini pour résoudre ce problème dans les applications distribuées à large échelle[14]. Combiné avec une décomposition judicieuse du document, il permet d'intégrer le noyau dans un environnemt hétérogène à large échelle.

# 3 Décomposition du document

La décomposition du document permet de réduire les conflits entre les opérations sur les objets du noyau et par là même, simplifie le contrôle (coûteux dans un système à grande échelle) nécessaire pour assurer la consistance. Informellement, si on gère le document comme un objet unique, alors chaque paire d'opérations peut potentiellement entrer en conflit. Soit $N(N-1)/2$ conflits à gérer pour $N$ opérations. Si le document est décomposé en, $k$ parties ($k \ll N$) alors le nombre des conflits[5] tombe à $N(N-k)/2k$.

Le partitionnement dans DUPLEX est dynamique et dirigé par les auteurs eux-mêmes puisqu'ils sont probablement les plus aptes à connaître les conflits potentiels (là où ils peuvent ou au contraire ne peuvent pas être évités). Le mécanisme de décomposition de DUPLEX assure que le partitionnement sera exécuté d'une manière consistante vis-à-vis des autres opérations.

La décomposition est basée sur la structure hiérarchique du document; un document est composé de chapitres, composés de sections, sous-sections, paragraphes, figures, etc... Prenons comme exemple le papier présent. Du point de vue hiérarchique, c'est un segment de *niveau-0* composé de plusieurs segments de *niveau-1* disjoints qui peut être modelisé comme une expression parenthésée bien formée[6]:
$\{_0\{_1\mathsf{1.\ Introduction}\}\{_1\mathsf{2.\ Le\ modèle\ Duplex}\}\cdots\{_1\mathsf{5.\ Discussion}\}\}$.
A son tour, chaque segment de *niveau-1* est lui-même une séquence bien formée de segments de *niveau-2*, et ainsi de suite récursivement jusqu'aux segments du document non structurés (e.g. une phrase ou un paragraphe non numéroté) qui sont appellés segments *élémentaires* et ne peuvent plus être décomposés. Dans DUPLEX, chaque segment peut être défini comme un segment *indépendant* du document et maintenu comme un objet du noyau.

Dans la figure 2 chaque boîte représente un segment. Les segments dépendants sont représentés imbriqués dans leur segment père et les segments indépendants de *niveau-k* sont représentés dans les colonnes correspondantes.

Considérons à un instant donné une partition du document $\Pi = \{S_i\}$ suivant ce type de frontière structurelle. Une description complète de segment indépendant $S_i$ de niveau $k$ est fourni par: (1) une description complète des sous-segments[7] de $S_i$ dépendants, (2) la position relative dans $S_i$ des sous-segments dépendants et indépendants, et (3) les références

[5] Si l'on considère que la probabilité d'accéder à chacune de ces partie est distribuée uniformement

[6] L'index de la parenthèse ouvrante représente le niveau du segment dans la hiérarchie du document

[7] Un sous-segment d'un segment de $niveau - k$ est un segment de niveau supérieur à $k$

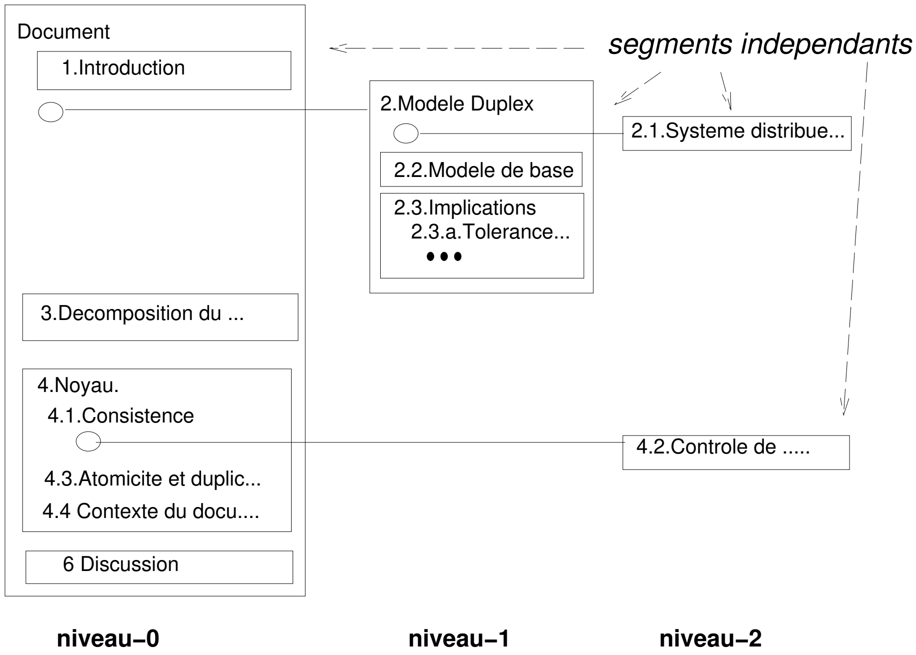


FIG. 2 - Exemple de decomposition

aux sous-segments indépendants. Ces derniers ne peuvent être que de niveau $> k+1$ (e.g. 2.1. le système distribué est un sous-segment indépendant de *niveau-2* relativement au segment 2. Le modèle Duplex de *niveau-1*).

Extraire un sous-segment indépendant de $S_i$, fusionner un sous-segment indépendant de $S_i$ (i.e. le rendre dépendant de $S_i$) sont des opérations atomiques qui impliquent seulement $S_i$ et le sous-segment considéré. Un segment indépendant $S_i$ est maintenu comme un objet du noyau. Deux opérations sur $S_i$ représentent une concurrence potentielle, et ainsi sont sujet au contrôle de concurrence. Par contre, les opérations sur $S_i$ n'entrent pas en conflit avec les opérations sur les sous-segments indépendants de $S_i$ ou sur les autres segments indépendants du document.

# 4 Le Noyau

Au cours du temps, des segments indépendants sont extraits du document et ajoutés comme nouveaux objets dans le noyau. La responsabilité du noyau est (1) de maintenir la consistance globale du document décomposé, et (2) d'assurer l'accès au document. La première fonction exige la définition d'un critère de consistance adapté à l'édition coopérative. La seconde exige la duplication des objets du noyau. Les deux fonctions mises ensemble exigent d'adapter le critère de consistance au contexte dupliqué, et de le mettre en oeuvre de manière efficace.

## 4.1 Consistance

L'idée fondamentale au niveau du critère de consistance est de *considérer* le document comme étant logiquement une version de travail unique et centralisée. Les co-auteurs travaillent sur cette version qui est passée de l'un à l'autre au fur à mesure des requêtes de lectures, de mises à jour. Ce protocole mutuellement exclusif permet de garantir que la version de travail reste toujours consistante[8].

Le critère de consistance considéré dans DUPLEX se base sur l'existence d'une séquence de toutes les mises à jour et de toutes les lectures, compatibles avec *l'ordre causal*[22] du système. Si l'état du document est décomposé en un ensemble d'objets concurrents, cette séquence s'appelle une *linéarisation*[8] du système. La linéarisabilité est un critère bien adapté aux systèmes à large échelle, puisqu'il assure que si chaque objet est linéarisable individuellement, alors le système entier est linéarisable[9]. Ainsi le contrôle de concurrence peut être mis en oeuvre indépendament pour chaque objet et assurer la consistance globale du document. Dans[17] le critère de linéarisabilité a été adapté au contexte dupliqué, à des fins de tolérance aux pannes. Dans le cas d'objets ayant de bonne propriétés, comme des segments de document par exemple, il permet une mise en oeuvre efficace du contrôle de concurrence, et ainsi fournir une implémentation du modèle de consistance choisi.

## 4.2 Contrôle de concurrence

La localité du critère de consistance fait que les mécanismes assurant la linéarisabilité des différents objets du noyau peuvent être choisis individuellement pour chacun des objets et modifiés au cours du temps. Ainsi, DUPLEX offre pour chacun des objets du noyau un ensemble de politiques de contrôle de concurrence (e.g. pessimiste basé sur des capacités, ou optimiste basé sur une détection des conflits), que l'utilisateur peut choisir suivant le degré de concurrence qu'il est prêt à autoriser sur les objets avec lesquels il est impliqué. Si l'accès exclusif offre la garantie que la dernière mise à jour d'un objet est réellement celle qui est maintenue par le noyau, une politique optimiste est plus avantageuse avec une décomposition utilisée de façon ad-hoc. Dans ce dernier cas, si une mise à jour entre en conflit, il est toujours possible de lire la dernière version du noyau et d'y reporter les modifications encore présentes dans la copie locale. En cas de conflit plus sérieux, les utilisateurs peuvent entrer dans une phase de conciliation en utilisant les facilités de communication directe ou indirecte.

## 4.3 Atomicité et duplication

Le problème de l'atomicité des opérations dans un contexte dupliqué exige des diffusions fiables au *groupe* de processus gérant les duplicas dans un modèle virtuellement synchrone[1, 24]. La localisation de ce groupe est réalisée grâce à l'utilisation d'un service de nom dédié au document[14]. Ce service lie les segments logiques du document (connu des auteurs par un nom logique établi à la création du segment) à l'ensemble des processus gérant les duplicas.

## 4.4 Contexte du document

Il reste à décrire les informations maintenues par un duplica. Elles sont modifiées atomiquement par les opérations sur un duplica du noyau. Ces informations sont: *L'information textuelle* décrit complètement le segment du document (Une suite de caractères, ou le bitmap d'une figure). *L'information structurelle* décrit la décomposition du segment en sous-segments dépendants et indépendants, et maintient les références logiques (par nom)

[8]Ceci ne veut pas dire que le *contenu* du document satisfasse tous les auteurs puisque leurs idées sur le contenu peuvent différer. Toutefois, il est de leur responsabilité de faire converger le document.

[9]Cette propriété est appelée propriété de localité

des sous-segments indépendants[10]. *Le bulletin d'information* qui assure une moyen de communication indirecte concernant le segment. C'est une liste de messages (persistant ou de durée limitée) causalement ordonnée et consistante avec l'entrelacement des discussions et les mises à jour des segments. *Le journal* garde une trace des opérations sensibles (mise à jour, destruction, etc.) accompli sur le segment. Le journal est persistant même après la destruction du segment.

# 5 Discussion

Le premier prototype de DUPLEX est opérationnel. Son objectif est d'offrir un environnement d'édition coopérative pour la grande échelle où l'hétérogénéite, les délais de communication importants et l'occurence de pannes rend inefficace les techniques classiques.

Notre approche explore deux directions pour rendre le système extensible. La première concerne le paradigme de travail coopératif et propose d'utiliser une décomposition dynamique du document combinée avec un critère de consistance adapté à l'édition coopérative pour limiter les conflits entre les auteurs. La décomposition permet aux utilisateurs de définir clairement les segments du document pour lequel ils sont le plus intéressés et d'appliquer le contrôle de concurrence qu'ils désirent sur ces segments. Ainsi le contrôle de concurrence est *adaptable*, en ce sens qu'il reste relativement lâche pendant la phase d'édition du document où les utilisateurs travaillent le plus souvent indépendamment, et devient plus restrictif lorsque les segments indépendants sont fusionnés. Le contrôle de concurrence s'applique ainsi à des parties de document plus grandes. Cette première direction met en relief la liberté d'action et le choix laissé aux utilisateurs pour les mécanismes de contrôle de concurrence.

La seconde direction est plus orientée vers le système et concerne le travail à large échelle dans des environnements hétérogènes. Par rapport à ce dernier point, DUPLEX est un ensemble de bibliothèques de fonctions s'exécutant au dessus d'UNIX et utilisant X11 et TCP/IP et par là même indépendant d'un matériel particulier. Concernant la large échelle, permettre de rendre des segments de document indépendants dans le système distribué offre beaucoup de flexibilité et permet de mettre en oeuvre le concept de noyau/environnement local d'une manière extensible. Le système est à la fois fiable et efficace dans un réseau à large échelle.

# Références


[1] K. Birman. The process group approach to reliable distributed computing. *Communications of the ACM*, 36(12), December 1993.

[2] U.M. Borghoff and G. Tegge. Application of collaborative editing to software-engineering projects. *ACM SIGSOFT*, 18(3):56–64, July 1993.

[3] F. Cristian. Understanding fault-tolerant distributed systems. *Communications of the ACM*, 34(2):56–78, February 1991.

[4] S.B. Davidson, H. GarciaMolina, and D. Skeen. Consistency in partitioned networks. *ACM Computing Surveys*, 17(3):341–370, September 1985.


[10]Ces références seront traduites en adresses dans le système distribué par le service de nom dédié.

[5] C.A. Ellis, S.J. Gibbs, and G.L. Rein. Groupware - some issues and experiences. *Communications of the ACM*, 34(1):38–58, January 1991.

[6] R.S. Fish and M.D.P. Leland R.E. Kraut. Quilt: A collaborative tool for cooperative writing. In *Proceedings of ACM Int. Conf. on Office Information Systems*, volume 9, pages 30–37, March 1988.

[7] I. Greif, R. Seliger, and W. Weihl. A case study of CES: A distributed collaborative editing system implemented with Argus. *IEEE Transactions on Software Engineering*, 18(9):827–839, September 1992.

[8] M. Herlihy and J. Wing. Linearizability a correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems*, 12:463–492, July 1990.

[9] M.J. Knister and A. Prakash. DistEdit: A distributed toolkit for supporting multiple group editors. In *Proceedings of ACM Int. Conf. on Computer-Supported Cooperative Work (CSCW '90)*, pages 343–355, October 1990.

[10] R. Ladin, B. Liskov, and L. Shrira. Lazy replication: Exploiting the semantics of distributed services. *Operating Systems Review*, 25(1):49–55, January 1991.

[11] M.D.P. Leland, R.S. Fish, and R.E. Kraut. Collaborative document production using Quilt. In *Proceedings of ACM Int. Conference on Computer-Supported Cooperative Work (CSCW '88)*, pages 206–215, September 1988.

[12] L. Liang, S.T Chanson, and G.W. Neufeld. Process groups and group communications: Classifications and requirements. *IEEE Computer*, 23(2):56–65, February 1990.

[13] H. Lubich and B. Plattner. A proposed model and functionality definition for a collaborative editing and conferencing system. In *Proceedings of IFIP WG 8.4 Conf. on Multi-User Interfaces and Applications*, pages 215–232. North-Holland, September 1990.

[14] J.-C. Lugeon and A. Sandoz. Sharing a small domain in a large distributed file system. Technical report, Swiss Institute of Technology of Lausanne (anonymous FTP ftp-lse.epfl.ch:/pub/TechReports/DILSE-6-93.ps), 1993.

[15] V.C. Miles, J.C. McCarthy, A.J. Dix, M.D. Harrison, and A.F. Monk. Computer Supported Cooperative Writing (chapter 8: Reviewing designs for a synchronous-asynchronous group editing environment). In *Computer Supported Cooperative Work*, pages 137–160. Springer-Verlag, 1993.

[16] C.M. Neuwirth, D.S. Kaufer, R. Chandhok, and J. Morris. Issues in the design of computer support for co-authoring and commenting. In *Proceedings of ACM Int. Conf. on Computer-Supported Cooperative Work (CSCW '90)*, 1990.

[17] F. Pacull and A. Sandoz. R-linearizability: An extension of linearizability to replicated objects. In *Proceedings of the 4th IEEE Workshop of Future Trends of Computing Systems*, 1993.

[18] F. Pacull, A. Sandoz, and A. Schiper. Duplex: A distributed collaborative editing environment in large scale. In *Proceedings of ACM Int. Conf. on Computer-Supported Cooperative Work (CSCW '94)*, 1994.

[19] M.O. Pendergast and D. Vogel. Design and implementation of a PC/LAN-based multi-user text editor. In *Proceedings of IFIP WG 8.4 Conf. on Multi-User Interfaces and Applications*, pages 195–206. North-Holland, September 1990.

[20] I.R. Posner and R.M. Baeker. How people write together. In *Proceedings of the 25 th Hawaii International Conference on System Sciences*, volume IV, pages 127–138, January 1992.

[21] M. Raynal and M. Mizuno. How to find his way in the jungle of consistency criteria for distributed shared memories. In *Proceedings of the 4th IEEE Workshop on Future Trends of Computing Systems*, 1993.

[22] A. Schiper, J. Eggli, and A. Sandoz. A new algorithm to implement causal ordering. In *Proceedings of the 3rd Workshop on Distributed Algorithms*, 1989.

[23] A. Schiper and A. Ricciardi. Virtually synchronous communication based on a weak failure suspector. In *Proceedings of the 23rd IEEE Int. Conf. on Fault Tolerant Computing Systems*, 1993.

[24] A. Schiper and A. Sandoz. Uniform reliable multicast in a virtuallly synchronous environment. In *Proceedings of 13th IEEE Int. Conference on Distributed Computing Systems*, May 1993.

[25] A. Sheth and M. Rusienkiewicz. Management of interdependent data: specifying dependency. *Proceedings of the 1st Workshop on the Management Of Replicated Data*, November 1990.