DÉFINITION, CONCEPTION, SPÉCIFICATION ET ANALYSE D'UNE SOLUTION PAIR-À-PAIR SENSIBLE AU CONTEXTE CAP A CONTEXT-AWARE PEER-TO-PEER SYSTEM

Ce chapitre définit, spécifie et analyse un système P2P sensible au contexte du réseau. Cette solution, baptisée CAP (Context-Aware P2P system), entend introduire une sémantique dans les identifiants des pairs et des objets. Pour cela, elle construit la DHT avec une fonction de hachage particulière, à savoir HMAC (keyed-Hash Message Authentication Code).

Le chapitre commence par un rappel sur la fonction HMAC.

4.1 - Keyed-Hash Message Authentication Code

HMAC (*keyed-Hash Message Authentication Code*) [RFC2104] est une fonction à trois paramètres : un message, une fonction de hachage et une clé. Elle est définie comme suit :

$$\text{HMAC}(h, k, m) = h(k \oplus opad \,||\, h(k \oplus ipad \,||\, m))$$

- *h* une fonction de hachage cryptographique, comme MD5 ou SHA-1 ;
- *m* est le message ;
- *k* est la clé de sécurité partagée uniquement par les entités impliquées dans l'échange du message. Si *k* est plus longue qu'*opad* et *ipad*, elle est remplacée par *h*(*k*). Si *k* est plus courte que ces constantes, elle est complétée par des zéros ;
- *opad* et *ipad* sont respectivement les octets 0x36 et 0x5C répétés chacun B fois (avec souvent B=64).
- || symbolise la concaténation ;

– $\oplus$ est l'opérateur de choix exclusif bit à bit (XOR). Il est prioritaire par rapport à la concaténation ;

HMAC est donc une fonction de hachage dont le paramètre, calculé de façon spécifique sur base d'opérations XOR et de concaténation, est en partie déjà haché. La fonction HMAC appliquée à *m* donne alors un résultat complètement différent de *h*(*m*).

HMAC a été initialement définie dans le domaine de la sécurité, et la clé sert la sécurité. En effet, le résultat de la fonction HMAC est un condensat, appelé MAC (*Message Authentication Code*), qui permet de vérifier l'intégrité du message et l'authenticité de sa source. L'émetteur et le récepteur partageant une même clé *k*, l'émetteur envoie au destinataire le message *m* et son condensat HMAC. Le récepteur applique alors la fonction HMAC au message reçu et compare le résultat ainsi obtenu avec la valeur du condensat HMAC reçu.

Afin de construire un système P2P sensible au contexte, nous proposons d'utiliser un routage contextuel. Pour cela, nous proposons de remplacer la fonction de hachage de la DHT par une fonction HMAC. Dans ce cas, la signification des paramètres en entrée de la fonction est adaptée :

– *h* devient la fonction de hachage initialement utilisée par la DHT ;
– *m* représentera selon le cas, soit l'adresse IP du pair, soit la référence de l'objet, ceux-là même qui permettent de construire les *nodeId* et *objectId* respectivement ;
– *k* sera une clé particulière, que nous appellerons *Hkey* (comme étant la clé de la fonction HMAC, différente de la clé de recherche qui est un *objectId*). Cette clé va représenter le contexte ; elle va prendre un sens et nous allons lui définir une sémantique (au paragraphe suivant). Elle sera nécessairement connue de tous les pairs d'une même DHT ; autrement aucune recherche ne pourra être effectuée dans l'espace d'identification de la DHT.

4.2 - Définition sémantique d'une Hkey

Une *Hkey* peut être simple ou composée. Elle peut être aussi dérivée et dérivable.

4.2.1 - Hkey simple

Nous disons qu'une *Hkey* est simple, si elle représente un seul critère ou un seul paramètre caractéristique.

Voici quelques exemples de ce que peut être une *Hkey* simple : une clé secrète, le numéro de l'AS ou le nom du domaine administratif d'appartenance, le nom ou l'identifiant d'un groupe de communication (sécurisé ou pas), un paramètre de qualité de service (e.g. la bande passante minimale disponible, la vitesse minimale de fonctionnement du processeur, la puissance minimale s'il s'agit d'un système mobile, la capacité de stockage minimale, etc.), un paramètre de localisation (e.g. proximité réseau ou distance métrique, localisation GPS (*Global Positioning System*), pays d'appartenance, etc.), la langue ou le type des fichiers partagés (qui peut être basé sur des métadonnées), leur taille, un mot clé, etc.

4.2.2 - Hkey composée

Nous disons qu'une *Hkey* est composée, si elle représente plusieurs critères ou paramètres. Elle est donc composée de plusieurs *Hkey*s simples. Elle est le résultat du hachage de la concaténation de ses composantes. Par exemple, si nous nous intéressons à trois métriques m_1, m_2, et m_3, nous définissons une *Hkey* composée, ayant pour valeur $h(m_1||m_2||m_3)$.

4.2.3 - Hkey dérivée

Une *Hkey* composée peut être décomposée en des *Hkey*s dérivées, tel que chacune soit formée d'un ou plusieurs des éléments simples de la forme composée. Ceci revient à former une *Hkey* composée en concaténant graduellement et partiellement plusieurs

*Hkey*s simples en *Hkey*s intermédiaires, qu'on dira dérivées de la composée.

Des *Hkey*s dérivées peuvent être aussi définies à partir d'une *Hkey* simple. Par exemple, si une certaine *Hkey* simple peut prendre quatre valeurs différentes pour un même pair (e.g. si le pair réside dans quatre zones), des *Hkey*s dérivées peuvent être définies par la combinaison de deux ou plusieurs de ces valeurs.

4.3 - Architecture de CAP

HMAC étant une fonction de hachage, le condensat est de même nature et de même longueur que le résultat de cette même fonction de hachage. L'utilisation de HMAC pour la construction d'une DHT n'affecte donc ni l'espace d'identification, ni le principe des DHTs.

Avec HMAC, chaque *nodeId* ou *objectId* n'est plus le résultat d'un $h(m)$, mais d'un $HMAC(h, k, m)$.

Si k (c.-à-d. *Hkey*) était une constante, il est évident que, les mécanismes de routage (pour la localisation d'une clé) et de maintenance (pour la (dé)connexion d'un pair) resteraient inchangés.

Maintenant que *Hkey* représente le contexte du réseau, il s'agit d'un paramètre variable. Le réseau P2P devient alors un réseau logique hétérogène. En d'autres termes, il s'agit d'une agglomération *overlay* de réseaux homogènes distincts, pour chacun desquels *Hkey* a une valeur constante. Chaque réseau homogène est appelé *zone*. Chaque zone est caractérisée par une *Hkey* et est identifiée par $h(Hkey)$ afin de respecter l'espace d'identification. Une *Hkey* sert alors d'étiquette pour l'espace d'identification de la zone concernée. Elle est donc utilisée pour le calcul (avec HMAC) de tous les *nodeId*s et *objectId*s de la zone. En d'autres termes, tous les *nodeId*s et *objectId*s d'une même zone connaissent une seule et même *Hkey*. Or ces identifiants se trouvent dans une même DHT. La DHT d'une zone est donc aussi étiquetée par cette même *Hkey*. Cette

nouvelle DHT est virtuelle par rapport à la DHT de base utilisant la fonction *h*. Elle sera appelée VDHT (pour *Virtual DHT*).

Le type des *Hkey*s sera défini dans un profil global du système. Ce profil pourrait être disponible, par exemple, en distribué ou sur un nœud d'amorce géré par le fournisseur de service ou de réseau. La manière dont il est chargé dans le système n'affecte cependant ni le routage P2P ni les opérations des différents pairs. Il ne sera donc pas détaillé dans la suite.

Dans le cas de l'existence de *Hkey*s dérivées, ce profil fournira aussi leurs structures (paramètres ou valeurs), ainsi qu'un ordre de priorité pour leurs usages dans les procédures de routage et de recherche de ressources.

Le mécanisme de détermination des *Hkey*s est toutefois extérieur à CAP. Il ferait partie d'un mécanisme de gestion de politiques du système.

Le mécanisme de l'évaluation individuelle des critères et paramètres par chaque pair est aussi extérieur à CAP. Il faudra pour cela sans doute se servir des bases d'information du système (e.g. des tables BGP [RFC1771] pour avoir les numéros des AS d'appartenance).

Ces deux types de mécanismes ne sont pas discutés dans la suite. CAP s'intéresse principalement aux mécanismes de routage et aux mouvements des pairs.

Le réseau P2P est donc une agglomération hétérogène de zones virtuelles homogènes chacune. Il peut alors se décomposer en plusieurs couches *overlay* secondaires disposées au-dessus du réseau global et caractérisée chacune par une *Hkey* précise. Ces *overlay*s correspondent aux différentes zones et sont dits locaux. C'est l'algorithme de routage (e.g. Chord [SML+01], Pastry [RD01], etc.) mis en œuvre au niveau global qui sera repris dans chacun de ces *overlay*s locaux.

Chaque pair reste actif dans le plan P2P global. Il peut en plus participer à un, plusieurs ou aucun de ces *overlay*s secondaires. Ceci dépend de la correspondance entre ses propriétés et la sémantique des différentes *Hkey*s du système, ainsi que du nombre des différentes *Hkeys* et de celui des éventuelles *Hkey*s dérivées.

Pour des raisons de simplicité, dans la suite, et sauf mention contraire explicite, nous n'aborderons pas les *Hkey*s dérivées et nous ne distinguerons pas entre une *Hkey* simple et une *Hkey* composée.

Dans le cas où le profil du système définit une seule *Hkey* (qu'elle soit donc simple ou composée, mais pas de *Hkey* dérivée), tous les *overlay*s locaux seront à un même niveau au-dessus du plan global. Et chaque pair participera à au plus deux *overlay*s : le global et un local.

La figure 4.1 illustre le cas d'un tel système. On y voit, au niveau des *overlay*s locaux, trois zones distinctes P1, P2 et P3, chacune caractérisée par une *Hkey* différente. Les pairs de chacune de ces zones se trouvent au niveau P2P global, où d'autres pairs, n'appartenant à aucune de ces zones, sont aussi actifs. Ces pairs peuvent soit faire partie d'autres zones que celles représentées, soit ne faire partie d'aucune zone s'ils ne satisfont pas les caractéristiques représentées par les *Hkey*s définies dans le profil du système. Au niveau IP, sont représentés aussi des nœuds n'appartenant pas à un réseau P2P, ou du moins pas à celui considéré dans l'exemple.

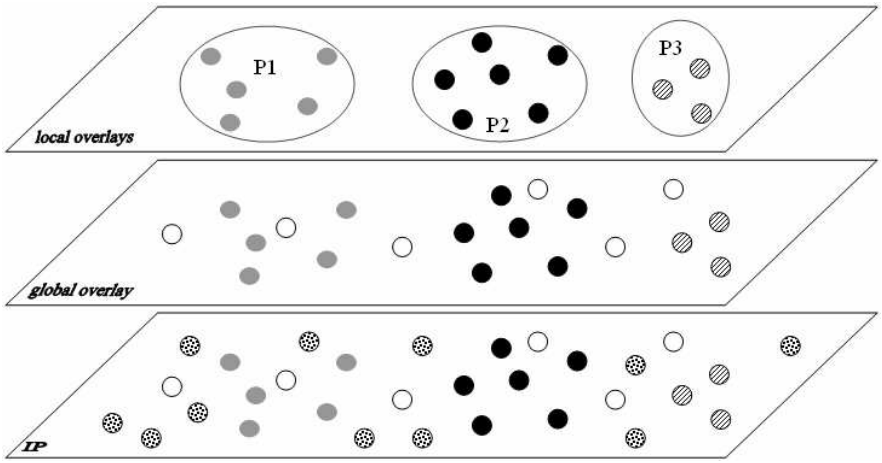

Figure 4.1 : *Vue d'ensemble du système sans Hkeys dérivées*

Chaque pair a un *nodeId* dans chaque *overlay* où il existe, et donc autant de *nodeId*s que le nombre total d'*overlay*s auxquels il participe. Ceci est analogue à un terminal ayant plusieurs interfaces réseaux et donc autant d'adresses IP : une par réseau. Chaque *nodeId* est le condensat d'une fonction HMAC utilisant la *Hkey* de l'*overlay* concerné. Dans un souci de

simplicité, nous n'exigeons pas d'unicité entre les différents espaces d'identification.

Quant au niveau global, où les pairs sont hétérogènes, le calcul des identifiants avec HMAC est générateur de conflits entre les *nodeId*s. Pour cela, deux cas sont possibles :

– soit cette nouvelle méthode est appliquée uniquement au niveau des *overlay*s locaux ;
– soit une valeur est fixée pour la *Hkey* au niveau global (e.g. 0x00 répété 64 fois) et définie dans le profil du système. Ce cas s'appliquera aux nouveaux pairs se connectant au système, et se généralisera au rythme de la dynamique du réseau. Cependant, un risque de collision dans la phase intermédiaire n'étant pas à exclure, la préférence va au premier choix qui ne modifie pas la génération des *nodeId*s au niveau global.

Comme un pair participe à plusieurs *overlay*s, il partage alors ses ressources dans chacun de ces *overlays*. Une même ressource aura donc autant d'identifiants (*objectId*s) que le pair qui la partage en a. Ceci revient à donner à un objet plusieurs pseudonymes. De même que pour les *nodeId*s nous n'exigeons pas d'unicité entre les différents espaces d'identification.

4.4 - Mécanisme de routage dans CAP

Le mécanisme de routage est le même dans chaque *overlay* local. Il s'agit de celui mis en œuvre au niveau global (e.g. Chord [SML+01], cf. paragraphe 1.4.2 p. 44). Une différence s'installe toutefois d'un *overlay* à l'autre : la *Hkey* caractéristique et du coup, les *nodeId*s et *objectId*s utilisés, ainsi que la DHT (ou VDHT) consultée.

Quand un pair veut rechercher un objet, il commence par calculer sa clé en utilisant la *Hkey* de l'*overlay* local où il réside. Puis il lance sa requête dans cet *overlay* en s'identifiant par son *nodeId* correspondant et en utilisant la VDHT correspondante. Ainsi la requête n'atteint que des pairs résidant dans le même *overlay* local.

Si la requête n'aboutit pas, elle est alors relancée dans l'*overlay* global avec les *objectId*, *nodeId* et DHT correspondants.

Quand des *Hkey*s dérivées sont définies dans le système, et que le pair réside dans plusieurs *overlay*s locaux, la requête est lancée dans chacun de ces *overlay*s. Ceci se fait à tour de rôle conformément à l'ordre défini dans le profil du système, et jusqu'à satisfaction de la requête. La requête peut aussi se lancer simultanément dans les *overlay*s locaux concernés, mais ceci génèrerait un trafic supplémentaire pas nécessairement utile. En tout cas, ce n'est qu'en dernier lieu, qu'une requête est lancée dans l'*overlay* global, si elle n'a pas encore abouti.

Il est à noter que seul le pair initiateur d'une requête peut relancer (à un niveau inférieur) une requête non aboutie. Aussi, un pair reconnaît la VDHT qu'il doit utiliser grâce au *nodeId* par lequel il a été sollicité.

Quand une ressource n'est trouvée que dans l'*overlay* global, le pair qui la récupère la remet en partage dans son *overlay* local, après lui avoir calculé son nouvel identifiant basé sur la *Hkey* correspondante, tout comme il y partagerait une nouvelle ressource.

4.5 - Mouvement des pairs dans CAP

Les mouvements d'arrivée et de départ d'un pair sont gérés de façon identique dans tous les *overlay*s du système et conformément au protocole de routage mis en œuvre dans l'*overlay* global. Cependant après s'être joint à l'*overlay* global et avant de pouvoir se joindre à un *overlay* local, un pair *n* doit vérifier l'existence d'au moins un autre pair n_1 actif dans cet *overlay* local. Pour cela, il doit effectuer certaines opérations, qui vont lui permettre de reconnaître un tel pair n_1, et à défaut, d'initier l'*overlay* en question.

4.5.1 - La 'Zone Table'

L'identifiant d'un *overlay* local est sauvegardée dans une '*zone table*' (ou *ZT*). Il s'agit d'une table de données relatives à une même zone. Pour chaque zone, il existe deux ZT : une locale,

stockée dans l'*overlay* local de la zone, et une autre globale, stockée dans l'*overlay* global.

Tableau 4.1 : *Structure de la 'zone table'*

zoneId	*zone label*	*largest nodeId*	*2nd largest nodeId*	*2nd smallest nodeId*	*smallest nodeId*

Les deux ZTs ont une structure identique en six champs (cf. tableau 4.1) :

- le *zoneId* vaut *h*(*Hkey*) et identifie la table ;
- le '*zone label*' est la valeur de la *Hkey* en chaîne de caractères. Il est particulièrement utile pour éviter d'éventuels conflits dans le cas des *Hkey*s dérivées ;
- les *nodeId*s des deux pairs aux plus grands *nodeId*s dans l'*overlay* local, et des deux autres aux plus petits *nodeId*s dans l'*overlay* local. Il s'agit des *nodeId*s dans l'espace d'identification local pour la ZT locale, et des *nodeId*s dans l'espace d'identification global pour la ZT sauvegardée au niveau global. Ce sont les pairs ayant ces *nodeId*s qui assurent la cohérence entre les deux ZTs. S'il y a moins de quatre pairs dans l'*overlay* de la zone, les champs restants sont mis à NULL.

Chaque ZT est donc gérée comme un objet identifié par le *zoneId*. Dans chaque *overlay*, la ZT est stockée sur un pair dit « nœud de rendez-vous » (ou RP – *Rendezvous Point*). Il s'agit du pair dont le *nodeId* est le plus proche numériquement du *zoneId*, conformément à la DHT mise en œuvre dans le réseau. Au niveau global, un même RP peut donc stocker plusieurs ZTs, relatives chacune à une zone différente.

La ZT est mise à jour quand un des pairs qui y sont référencés disparaît de l'*overlay* local (départ volontaire ou panne), ou bien quand un nouveau pair arrive dans cet *overlay* avec un *nodeId* tel qu'il doit être référencé dans la ZT. La mise à jour de la table se faisant donc suite à une arrivée ou une disparition de pair, nous la

présentons convenablement dans chacun des paragraphes suivants (4.5.2 et 4.5.3).

Si un RP vient à quitter le réseau ou tomber en panne, la prise en charge de la ZT est gérée comme celle des autres objets que le RP stockait. Par souci de robustesse, la ZT peut être aussi dupliquée dans l'overlay où elle se trouve. En particulier, elle pourrait l'être sur le pair ayant le plus petit identifiant dans l'*overlay*, soit le pair référencé dans le dernier champ de la ZT.

4.5.2 - Arrivée d'un pair

(Il est à noter que si des *Hkey*s dérivées sont définies dans le système et concernent le nouveau pair, celui-ci réitère le processus d'adhésion décrit dans ce paragraphe pour chaque *zone* à laquelle il veut appartenir, et ce avec la *Hkey* correspondante.)

Une fois inséré dans le réseau global, le nouveau pair prend connaissance du profil du système (qui peut lui être aussi communiqué). Il connaît alors les différentes *Hkey*s. Par un mécanisme externe au réseau P2P, le nouveau pair évalue et détermine les *Hkey*s qui lui correspondent. Il calcule alors l'identifiant *h*(*Hkey*) de chaque réseau local auquel il doit appartenir. Cet identifiant étant aussi celui de la ZT de l'*overlay* local, le nouveau pair initie une requête *zone_table_request( )* au niveau global pour récupérer la ZT. Deux cas sont alors possibles suivant que l'*overlay* local en question est déjà initialisé ou pas.

4.5.2.1 - Overlay local initialisé

Dans le premier cas, l'*overlay* local est déjà initialisé. Il a une ZT stockée sur un RP au niveau global. Le nouveau pair peut donc la récupérer. Il va ainsi connaître jusqu'à quatre pairs résidants dans la zone qu'il doit rejoindre. Il en choisit un et lui envoie dans l'*overlay* global un message de demande d'adhésion. Dans ce message, le nouveau pair

envoie la *Hkey* de la zone et *n*, son propre *nodeId* calculé avec cette *Hkey*. Le pair contacté envoie alors au nouveau pair, son *nodeId* local n_1. L'insertion du nouveau pair dans l'*overlay* local se poursuit alors conformément au protocole de routage mis en œuvre (e.g. à travers la fonction $n.join(n_1)$, si le protocole est Chord [SML+01] ; cf. paragraphe 1.4.2 p. 46).

Une fois le nouveau pair correctement inséré dans l'*overlay* local, il cherche dans celui-ci la ZT locale. Quand il la récupère, il compare son *nodeId* local, *n*, aux *nodeId*s qui s'y trouvent. Si *n* est supérieur au deuxième *nodeId* le plus grand ou inférieur au deuxième *nodeId* le plus petit, le nouveau pair met à jour la ZT locale avec *n*, et se charge de mettre à jour la ZT globale de la zone. Les champs des deux ZTs se succédant dans le même ordre, le nouveau pair remplace par son *nodeId* global le champ de la ZT globale ayant le même numéro d'ordre que le champ qu'il a modifié dans la ZT locale. Par la suite, il en informe le RP qui lui avait fournit la ZT au niveau global, afin que la mise à jour soit cohérente. Le nouveau pair peut aussi directement charger le RP global de la mise à jour de la ZT globale.

4.5.2.2 - Overlay local inexistant

Dans le second cas, l'*overlay* local n'est pas encore initié. Le nouveau pair est le premier à vouloir adhérer à cet *overlay*. Il n'y a pas de ZTs correspondantes. La requête lancée dans l'*overlay* global pour récupérer la table n'aboutit pas : soit il y a un retour de message (de la part du supposé RP) informant de l'inexistence de l'objet recherché, soit l'initiateur de la requête déduit ceci suite à l'expiration d'un jeu de temporisateurs. C'est alors au nouveau pair de créer la ZT globale de la zone qu'il voulait

rejoindre. Il le fait avec les *zoneId* et *zone label* correspondants et son propre *nodeId* global, et met les trois autres champs à NULL. Une fois la ZT globale créée, le nouveau pair la partage (elle sera stockée sur le RP) et initie l'*overlay* de sa zone. Ensuite, il crée la ZT locale, copie de la ZT globale où il remplace son *nodeId* global par son *nodeId* local calculé avec la *Hkey*. Les trois nouveaux pairs suivants dans la zone modifieront les champs initiés à NULL dans les ZTs concernées.

Dans un réseau P2P bien peuplé, ce dernier cas est peu probable, notamment avec des *Hkey*s communes (e.g. si la zone correspond à un AS, si la *Hkey* désigne un type populaire de fichiers, si la *Hkey* est la bande passante typique d'une majorité de pairs, etc.)

4.5.3 - Disparition d'un pair

La disparition d'un pair du système (départ explicite, départ silencieux, ou panne) est généralement remarquée quasi-simultanément dans tous les *overlay*s auxquels il appartenait. Elle est traitée de façon identique mais indépendante dans chaque *overlay*, et conformément au protocole mis en œuvre au niveau global.

S'il s'agit d'un des pairs référencés dans les ZTs relatives à la zone affectée par la disparition, celles-ci seront mises à jour. En effet, le RP local vérifie périodiquement l'état des pairs référencés dans la ZT (locale) dont il a la charge. Lorsqu'un d'eux ne répond plus, le RP le remplace et réordonne les *nodeId*s de la ZT. Il le remplace par le pair suivant au *nodeId* strictement plus grand ou strictement plus petit. En fait, le RP local garde en antémémoire une liste courte des pairs de la zone aux plus petits ou aux plus grands *nodeId*s. Il est peu probable que tous tombent en même temps, mais si le cas se présente, le RT met le champ restant à NULL et attend une nouvelle adhésion.

Quand le RP local a mis à jour la ZT locale, il envoie localement un message au pair qu'il vient d'ajouter pour l'informer de la table où il est référencé. Ce pair se charge alors de la mise à jour de la ZT globale, exactement comme ceci se fait dans le cas de l'adhésion d'un nouveau pair à un *overlay* local déjà initié (cf. paragraphe 4.5.2.1, p. 92).

4.6 - Analyse de CAP

L'utilisation de HMAC pour la génération des identifiants (des pairs et des objets) permet à un système P2P utilisant les DHTs d'être sensible au contexte du réseau. Ceci est réalisable à travers la clé de HMAC, baptisée *Hkey* pour ce système. L'algorithme de la DHT en soi n'est pas affecté, même si des opérations complémentaires viennent s'y greffer pour la cohésion du système.

Ce changement de fonction de hachage donne une sémantique aux identifiants. Il permet aussi de structurer un réseau P2P initialement hétérogène en des couches secondaires virtuelles homogènes chacune, avec une granularité définie par la *Hkey*. A chacune de ces couches secondaires, il y a donc une DHT virtuelle, dite VDHT, qui permet une gestion et une recherche contextuelles des données.

En effet, la recherche d'une ressource ou d'un élément de données dans CAP commence toujours dans un *overlay* local en utilisant les services de la VDHT correspondante. Or, tous les identifiants référencés dans une VDHT sont calculés avec une fonction HMAC utilisant la même *Hkey* qui représente le contexte de l'*overlay* et caractérise donc la VDHT. Ainsi un *objectId* ne peut être référencé dans une VDHT que si l'objet est bien dans l'*overlay* (local) correspondant.

Par ailleurs, toute requête qui aboutit dans l'*overlay* où elle a été initiée est garante de l'existence de l'objet dans cet *overlay*. Il en découle donc que le pair initiateur de la requête peut récupérer l'objet d'un pair partageant le même contexte et à travers un chemin conforme à ce contexte.

En garantissant la récupération d'un objet de la zone où il a été trouvé, CAP offre un cloisonnement des ressources, contextuel et efficace, qui réduit le temps de latence des messages et améliore le temps de récupération des données. De plus, la mise en partage dans un *overlay* local d'une ressource récupérée au niveau global favorise la popularisation des objets dans le système.

CONCLUSION DU CHAPITRE –

Ce chapitre a défini, spécifié et analysé CAP (Context-Aware P2P system), un système P2P utilisant des DHTs et sensible au contexte du réseau. Ce contexte est caractérisé par un ou plusieurs paramètres du système, et ne se restreint pas à la seule topologie sous-jacente. Pour spécifier cette sensibilité, CAP a introduit une sémantique dans les identifiants des pairs et des objets en construisant la DHT avec une fonction de hachage particulière, HMAC, présentée au début du chapitre.

CAP est un système générique et configurable, mais où chaque élément (pair ou objet) a plusieurs identifiants indépendants. Le chapitre suivant présentera une nouvelle solution de sensibilité au contexte, où la gestion des identifiants sera simplifiée. Le contexte sera celui d'un opérateur de réseaux.