Context dependency analysis in ubiquitous computing

Raheel Ali Baloch

Ecole Doctorale EDITE

Thèse présentée pour l'obtention du diplôme de
Docteur de Télécom & Management SudParis

Doctorat conjoint TMSP-UPMC

Spécialité : Informatique et Télécommunications

Par Raheel Ali Baloch

Context Dependency Analysis in Ubiquitous Computing

Soutenue le 17 Fev., 2012 devant le jury composé de :

Serge Garlatti	**Professeur à Télécom Bretagne**	Rapporteur
Xiaodi Huang	**Professeur à Charles Sturt Univ.**	Rapporteur
Amal El Fallah Seghrouchni	**Professeur à Paris VI**	Examinateur
Yacine Ghamri-Doudane	**Professeur Associé à ENSIIE**	Examinateur
Emmanuel Bertin	**Docteur à Orange Labs**	Examinateur
Noel Crespi	**Professeur à Télécom SudParis**	Directeur de thèse

Thèse n° 2012TELE0004

Acknowledgements

I am indebted to my PhD supervisor, Dr. Noel Crespi, for his understanding of problems and people, encouragement, optimism, and meticulous attention to all aspects of my research work. Aside from guiding me through this thesis, he has given me valuable insight into the art and science of problem solving. I would also like to acknowledge the generous support of RS2M team at ITS.

On a more personal note, I would like to thank my family for always encouraging me to do my best, and strive for excellence. I am grateful to my parents without whom this thesis would never have been possible. Their encouragement and support enabled me to ultimately achieve this milestone.

Abstract

To provide users with personalized adaptive services only using the accessible computing resources in a cloud environment, context aware applications need to assimilate both the accessed and derived context, i.e. a combination of more than one sensed data and information in the environment. Context data dependency, dependency that arises between the context data producer and consumer, may get introduced in a system due to numerous reasons. But as the number of context dependencies for a service increases, the more complex the system becomes to manage. The thesis addresses issues of how to identify context dependencies, represent such context dependencies and then reduce them in a system. In the first part of the thesis, we present two efficient approaches to determine context dependency relations among various services in ubiquitous computing environment to help better analyse the pervasive services. One approach is based on graph theory, and we have used the topological sort to determine the context dependencies. The second approach is based on solving constraint networks which determines whether an entity is affected when the state of a certain other entity has its state changed, i.e. determining the dynamic nature of context dependency. In the second part of the thesis, we present a mode for representation of context dependencies within a system. Our model that represents context dependencies is based on set theory and first-order predicate logic. The context dependency representation model also represents alternative sources for context acquisition that can be utilized in a case in which the preferred context producers are not available to service the desired context to the relevant context consumer any more. Further, we try to reduce the context dependencies by presenting the idea of profile context, which is based on the proposal of an open framework for context acquisition, management and distribution. This heuristic approach is based on the idea of utilizing mobile nodes in an ad hoc overlay network with more resources than the context

producer itself to store various contextual information under the banner of profile context, and further, provide profile context instead of each context individually based on the queries the nodes receive from the context consumers. Bringing together the context information and context updates from various sources, support for context aware decisions can be implemented efficiently in a mobile environment by addressing the issues of context dependency using profile context.

Table of Contents

Publications

R.A. Baloch, N. Crespi, "Managing Context Dependencies in Internet of Things," *Journal of Universal Computer Science*. (**under revision**).

R.A. Baloch, and N. Crespi, "Profile context management in ubiquitous computing," *International Journal of Ad Hoc and Ubiquitous Computing*, Ed. Inderscience Publishers Ltd. (**accepted**)

R.A. Baloch, and N. Crespi, "Service Dependency Analysis in Ubiquitous Environment," in *International Workshop on Ubiquitous Service Platforms at IEEE International Conference on Network Protocols*, Kyoto, Japan, October 2010.

R.A. Baloch, and N. Crespi, "Formal Model for Analysis of Context Dependencies in Ubiquitous Systems," in *Proc. 6th IEEE International Conference on Digital Information Management*, Ontario, Canada, pp. 431-436, July 2010.

R.A. Baloch, and N. Crespi, "Addressing context dependency using profile context in overlay networks," in *Proc. 7th IEEE Consumer Communication and Networking Conference*, Las Vegas, United States, pp. 668-673, January 2010.

R.A. Baloch, I. Awan and G. Min, "A mathematical model for wireless channel allocation and handoff schemes," *Special Issue on Performance Modelling and Evaluation of Telecommunication Systems, Telecommunication Systems Journal*, vol. 45, issue 4, pp. 275-287, 2010.

R.A. Baloch, I. Awan and G. Min, “Analytical Models for Complete and Partial Channel Allocation Schemes,” in *Proc. IEEE PMECT, at 17th IEEE International Conference on Computer Communications and Networks (ICCCN ‘08)*, Virgin Island, USA. pp. 702-707, 2008.

List of Figures

List of Tables

Resume

Pour fournir aux utilisateurs des services personnalisés d'adaptation en utilisant uniquement les ressources informatiques accessibles dans un environnement de cloud computing, les applications contexte, conscients besoin d'assimiler à la fois le contexte accessible et dérivés, c'est à dire une combinaison de plus d'un senti données et d'informations dans l'environnement. Contexte des données de dépendance, la dépendance qui se pose entre le contexte des données du producteur et du consommateur, peut se présenter dans un système en raison de nombreuses raisons. Mais comme le nombre de dépendances de contexte pour une augmentation des services, la plus complexe, le système devient à gérer. La thèse aborde les questions de la façon d'identifier les dépendances de contexte, représentent des dépendances de contexte tels, puis les réduire dans un système. Dans la première partie de la thèse, nous présentons deux approches efficaces pour déterminer les relations de dépendance entre les différents services du contexte dans l'environnement informatique ubiquitaire pour aider à mieux analyser les services omniprésents. Une approche est basée sur la théorie des graphes, et nous avons utilisé le tri topologique pour déterminer les dépendances de contexte. La deuxième approche est basée sur la résolution des réseaux de contraintes qui détermine si une entité est affectée lorsque l'état d'une certaine entité autre a changé son état, c.-à-détermination de la nature dynamique de la dépendance contexte. Dans la deuxième partie de la thèse, nous présentons un mode de représentation des dépendances de contexte au sein d'un système. Notre modèle qui représente les dépendances de contexte est basé sur la théorie des ensembles et la logique des prédicats du premier ordre. Le modèle de représentation contexte de dépendance représente également d'autres sources pour l'acquisition de contexte qui peuvent être utilisés dans une affaire dans laquelle les producteurs contexte privilégiées ne sont pas disponibles pour desservir le contexte souhaité pour le consommateur un contexte

pertinent, pas plus. En outre, nous essayons de réduire les dépendances de contexte en présentant l'idée du contexte de profil, qui est basé sur la proposition d'un cadre ouvert pour l'acquisition de contexte, la gestion et la distribution. Cette approche heuristique est basée sur l'idée d'utiliser les nœuds mobiles dans un réseau ad hoc avec superposition de plus de ressources que le producteur lui-même contexte pour stocker diverses informations contextuelles sous la bannière du contexte profil, et en outre, fournir le contexte profil au lieu de chaque contexte individuellement sur la base sur les requêtes des nœuds reçoivent des consommateurs contexte. Réunissant les informations de contexte et de mises à jour de contexte à partir de diverses sources, le soutien aux décisions contexte, conscients peut être mis en œuvre efficacement dans un environnement mobile en s'attaquant aux problèmes de dépendance en utilisant le contexte contexte profil.

1.1 Contexte

Applications contexte, conscients sont le fondement pour le monde d'aujourd'hui de l'informatique omniprésente. Services et applications dans un environnement ubiquitaire ont à interagir avec une variété de dispositifs mobiles, réseaux de communication et de capteurs sans fil avec la variabilité de leur environnement d'exécution respectifs. Par conséquent, les applications contexte, conscients doivent être conscients de leur contexte en cours d'exécution afin de réagir en conséquence à des scénarios d'exécution multiples. En outre, la personnalisation des services dépend des applications contexte, conscients de réaliser la vision de l'informatique omniprésente rêvé par Weiser (1991). Dans un environnement informatique ubiquitaire, divers applications hétérogènes et des dispositifs sont impliqués, la production et la consommation de l'information rapide du contexte. Les données et informations de contexte de nœuds de capteurs, qui sont disponibles dans des réseaux ad hoc, doivent être utilisés par les services de contexte, conscients et les applications. Les appareils mobiles sont également

quitter et de rejoindre des réseaux ad hoc à un rythme dynamique, ce qui rend les informations de contexte hautement imprévisible dans la nature. Une telle nature toujours changeante d'informations de contexte rend les applications plus complexes contexte, conscients de développer, maintenir et à comprendre.

Le contexte est généralement fait référence aux attributs dans un environnement informatique toujours dynamique dans lequel les divers aspects tels que l'emplacement de l'utilisateur, la situation sociale et l'interaction avec d'autres ressources sont en alternance en permanence. Peut-être en raison de son approche plus générale, la définition fournie par Dey et Abowd (1999) est le plus largement cité dans la communauté de la recherche: "Le contexte est toute information qui peut être utilisée pour caractériser la situation d'une entité." Une entité peut être définie comme «une personne, un lieu ou un objet qui est considéré comme pertinent pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application eux-mêmes."

Avant 1980, il y avait déjà quelques chercheurs qui ont pensé de l'informatique omniprésente, mais il était en 1988 que l'idée révolutionnaire de l'informatique ubiquitaire intégré dans l'environnement humain a été présenté par Mark Weiser. Auparavant, il y avait peu de recherches qui avaient touché à l'idée de l'informatique ubiquitaire. Une telle recherche a été appelé Human-Computer Interaction (HCI). HCI a été le plus proche de l'idée de l'informatique omniprésente Weiser. Par exemple, HCI système amélioré les qualités de l'apprentissage et l'efficacité de l'utilisation du système. En outre, la reconnaissance des gestes de HCI est un concept important pour l'informatique ubiquitaire. De même, il y avait le travail assisté par ordinateur coopérative (TCAO), le système de travail en ligne la participation des personnes situées à plusieurs sites physiquement. Ces sujets de HCI étaient liés à l'idée de l'informatique ubiquitaire Weiser (Brad, 1998).

Une autre recherche effectuée en 1995, appelé Things That Think (Media Lab, 2011a), par le

MIT Media Lab a étudié conceptions intelligentes pour l'interaction entre les appareils ménagers et les utilisateurs à la perspective d'améliorer l'ergonomie. Une autre recherche connexe qui est encore en usage est appelé Radio-Frequency Identification (RFID) transpondeur, qui a été construit à des fins de facilité d'utilisation et est maintenant une partie essentielle de n'importe quel scénario système omniprésent impliquant emplacement d'un objet. RFID a été initialement développé en 1940, mais perfectionné en 1973 (Landt, 2001). Une autre technologie similaire, le code à barres, a été élaboré en début des années cinquante, mais était seulement disponible dans le commerce dans les années 1980.

Une étape importante vers l'informatique omniprésente a été le développement de la première génération de téléphones intelligents qui étaient en mesure de soutenir la non-communication vocale. Il a été adopté par les utilisateurs commerciaux avec succès en raison de l'avantage de communiquer de n'importe où, avec ou sans voix. Capteurs de Berkeley et Centre de l'actionneur (BSAC, 2008) a été créé en 1986, et a pris l'avantage des capteurs »et actionneurs des technologies sortant de la PARC de Xerox. Dans les années 1990, diverses technologies ont commencé à se interconnectés comme les microcontrôleurs et les émetteurs-récepteurs deviennent plus petits, devenant ainsi le principal composant de diverses technologies. En 1998, dans le cadre de SmartDust projet (SmartDust, 2008), les technologies de détection, des microcontrôleurs et des transceivers en bandes ISM 2 ont été en rapport avec les nœuds de capteurs sans fil, donnant ainsi un fond intéressant pour le développement et la recherche pour les réseaux d'aujourd'hui de capteurs sans fil tels que TinyOS (TinyOS, 2011), arbalète (Crossbow, 2008) et de poussière (Dust, 2008). En outre, le développement du système de positionnement global (GPS) de démarrer au kick applications de localisation basés sur diverses, telles que dans les scénarios où les gens utilisent le GPS pour trouver leur emplacement actuel sur la carte pendant le voyage. L'idée de Weiser peut également être attribuée aux romans de science-fiction et des films.

Dans Ubik de Philip K. Dick, le roman 1969, l'écrivain a écrit sur un milieu riche avec des technologies différentes aider les humains dans de multiples façons. Dans le roman, la vie humaine quotidienne a été exécuté par les machines nouvelles de lecture, le nettoyage des robots et des portes de discussion. Le roman offre un bon reflet de l'avenir de l'informatique ubiquitaire. Auteurs de fiction n'ont pas leurs propres idées bien différentes de celles de Weiser. Ils ne savaient pas quoi que ce soit sur l'informatique omniprésente à partir d'un point de vue technique, mais ils étaient capables d'écrire de tels romans qui se chevauchaient l'idée proposée par Weiser, par exemple en utilisant différents appareils électroménagers pour communiquer avec l'utilisateur, de trouver un emplacement de l'utilisateur, etc Tous ces écrits de fiction ont montré à ce moment-la façon dont l'avenir pourrait ressembler à l'homme, et maintenant il a commencé à se produire dans notre vie.

1.2 Motivation

Chaque nouvelle technologie émergente doit faire face à de nombreux défis à atteindre le succès et la confiance. Initialement, les défis sont seulement dans le développement du prototype. Une fois le prototype fonctionne selon les normes requises et les chercheurs pensent que maintenant le projet est sur le point d'atteindre un certain niveau de maturité qui profitera à la collectivité dans le long terme, alors les défis à venir auxquels est confronté le projet de rendre la technologie omniprésente sont connu sous le nom de grands défis. La définition de grands défis a été formulée par le Comité du Royaume-Uni Centre de recherche informatique (Grand Challenge, 2008). Les grands défis sont les suivants:

(I) Soutenu par la communauté scientifique

(Ii) L'ambition est bien plus grande que celle réalisée par une équipe de recherche unique

(Iii) Défi s'adresse à révolutionner le projet à des produits de l'industrie prêts.

Mais l'occasion grand défi pour la plupart des projets est rarement, car il est une tâche énorme pour mener le projet à une certaine maturité de sorte qu'il peut alors révolutionner la vie de

l'homme. Pour des exemples, les grands défis peuvent être la cartographie du génome humain (accompli) ou d'unifier les théories pour les quatre types de forces en physique (objet d'une enquête).

The Ubiquitous Computing Grand Challenge (Chalmers, 2006) est l'un des actuellement neuf de tels défis, basés sur la théorie et l'ingénierie de systèmes ubiquitaires. D'une part, il s'efforce d'atteindre idée de Weiser à cette époque et, d'autre part, l'ingénierie et les questions sociales qui sont axés sur des groupes de recherche différents. Il préserve également "la théorie qui sous-tend la conception requise et l'analyse des systèmes ubiquitaires qui sont intrinsèquement de grande envergure et complexes". Il ya quelques questions importantes qui sont liées à l'informatique ubiquitaire du Grand Défi, comme "Combien d'ordinateurs allez-vous utiliser, le port, ou avez installé sur votre corps, en 2020? Combien d'ordinateurs d'autres seront-ils parler? Que vont-ils dire au sujet de vous, de faire pour vous, ou pour vous? (...) Allons-nous être capables de gérer ces systèmes à grande échelle, ou même de les comprendre? Comment les gens interagissent avec eux et comment cette nouvelle technologie ne omniprésente sur la société? Comment peut non informatiques gens de configurer et de les contrôler? Quels outils sont nécessaires pour la conception et l'analyse de ces adaptant constamment et les systèmes de l'évolution? Quelles sont les théories nous aidera à comprendre leur comportement? "(Chalmers, 2006). Les réponses à ces questions ne sont pas simple. Il est nécessaire d'observer tous les aspects du Grand Challenge Ubiquitous Computing afin de surmonter ces problèmes de recherche. Les questions peuvent être analysé à travers trois perspectives distinctes: l'expérience, de l'ingénierie et théorique. Expérience - le comportement des gens étude avec les systèmes informatiques ubiquitaires, et basé sur l'étude tels que l'amélioration du système devrait être plus orientée vers la perspective socio-technique.

Ingénierie - Les solutions aux défis auxquels nous étions confrontés sont en raison de la

nature dynamique du système omniprésent. Chaque système est tributaire de la conception, une infrastructure efficace et d'approche. Ces principes sont également applicables aux systèmes informatiques ubiquitaires. Ces systèmes doivent avoir des modèles efficaces et des infrastructures bon réseau.

Théorique - étude analytique détaillée est toujours important que ce soit avant le développement du système ou lors de la performance du système. Avec l'approche de la recherche proprement dite, de nouvelles techniques, modèles et méthodes peuvent être utilisées dans le système pour le rendre plus efficace et de haute performance qui devrait répondre aux normes requises des clients.

Outre ces perspectives, d'autres objectifs doivent également être fixés, qui sont plus orientées vers l'interaction individuelle et sociale. La bonne conception, l'utilisation de techniques et de modèles, répondant aux exigences des individus et des sociétés et la recherche continue sera remarquablement améliorer la croissance des systèmes informatiques ubiquitaires.

Pour définir la dépendance dans le contexte des systèmes ubiquitaires, il vaut mieux d'expliquer avec une aide d'un petit exemple. La plupart des applications contexte, conscients utiliser le contexte emplacement. Le contexte de localisation peut être acquise à partir de sources diverses. Les ressources communes dans les systèmes de contexte, conscients sont les capteurs physiques, qui détectent la position de la personne dans la salle de par exemple Global Positioning System (GPS), et par exemple le contexte dérivée si l'ordinateur personnel de la personne dans son bureau est actuellement en cours d'utilisation, il peut être dérivé que la personne dit, c'est dans son bureau. Ainsi, l'application contexte courant dépend de l'une de ces sources d'acquérir contexte de position. Par conséquent, il existe la dépendance de contexte entre l'application contexte, conscients à la recherche d'emplacement et le contexte de la source fournissant le contexte emplacement. Si, pour une raison quelconque, la source n'est pas disponible contexte, l'application peut demander contexte, conscients sources

contexte d'autres de fournir le contexte approprié. Donc, trouver la dépendance et la gestion des dépendances contexte contexte pertinent est important pour le bon fonctionnement de l'application contexte courant. Si une application contexte courant est dépendante de sources multiples pour l'acquisition de contexte différents différents des données contextuelles, puis un échec pour acquérir un quelconque de ceux des informations de contexte nécessaires peut entraîner l'échec d'une telle demande contexte courant. Donc, il est important de réduire la dépendance contexte de l'application contexte courant sur les sources de contexte et les producteurs qui ne sont pas suffisamment fiables pour offrir un service uniforme tel que demandé par l'application.

Le processus d'acquisition contexte est un élément vital de tout système de contexte courant. La plupart des approches de tenter d'acquérir le contexte de périphérique, par exemple, la présence d'une personne dans la salle du capteur de position, au lieu du "contexte profil". Profil contexte est le contexte agrégé à partir de sources diverses contexte qui donne une vue d'ensemble plus du sujet, par exemple, le contexte de profil, qui nous dit que l'utilisateur est occupé, peut être acquis de l'agrégation de contexte de position, de nous dire que l'utilisateur est dans le bureau, et contexte d'utilisation de son ordinateur personnel avec la ligne téléphonique occupée. Chaque dispositif détenues par un utilisateur est contacté individuellement dans le but d'acquérir des données pertinentes à la situation d'utilisateur dans des approches communes, mais notre approche fournit un contexte global sur une demande en vue de réduire les requêtes de contexte pour les dispositifs à ressources limitées. Contexte publié par un dispositif échoue individuellement pour capturer l'image globale de la situation actuelle de l'utilisateur par rapport au contexte obtenu à partir de sources multiples qui sont au courant de leur existence mutuelle, et peuvent fournir un contexte mieux l'utilisateur, travaillant en collaboration entre eux. Contexte de l'utilisateur, dans notre approche, est généralement dépendant de deux ou plusieurs dispositifs »détectés données, plutôt que juste

un contexte de périphérique unique.

En dépit de leurs potentiels des applications répandues, le contexte des applications et des services sont encore dans l'enfance parce que l'architecture d'acquisition contexte est fondée, plus ou moins, sur requête fondée sur le cadre du silo dans lequel un service ou une application demande directement pour certain contexte à partir d'une source de contexte individuel . Un tel cadre est sensible à de nombreuses questions; la plus commune d'entre eux sont des performances goulets d'étranglement et une forte probabilité d'échec, car la gestion des ressources non-efficacité du système. Si l'approche client-serveur est utilisé pour réaliser un service de contexte courant, il peut être facilement imaginé que les serveurs seront bientôt surchargé de requêtes, influant sur leur performance. La nature dynamique de l'information à traiter dans un environnement de cloud computing est énorme, et le nombre de requêtes augmente avec le temps que les utilisateurs plus rejoindre le réseau ad hoc demandant des mises à jour constantes à partir de sources contexte pour les types de contexte tels que l'emplacement.

Certaines applications simples contexte sensibiliser utiliser les données détectées qu'à partir d'une source unique de prendre des décisions appropriées personnalisés pour l'utilisateur. La plupart, cependant, d'utiliser les données détectées et informations de contexte transformés provenant de sources multiples pour obtenir de plus amples renseignements contexte de haut niveau. Cette contexte abstrait nécessite de multiples contexte de périphérique de bas niveau qui sont regroupées et traitées de nombreuses fois pour obtenir un contexte abstrait niveau. Pour fournir à l'utilisateur avec la plupart des prestations et des services personnalisés en utilisant les ressources informatiques disponibles dans un réseau ad-hoc, les applications contexte, conscients besoin de diffuser à la fois les informations de contexte accessibles et dérivés dans le réseau.

Dérivé contexte est fondamentalement dépendant des données captées ou / et d'autres

informations contextuelles transformés. Le contexte peut se la dépendance intronisé dans un système pour diverses raisons telles que l'erreur au moment du déploiement du système. Mais cette question de côté, les dépendances contexte plus une application a, le plus complexe le processus devient de développer et de maintenir une telle demande, car la défaillance d'un contexte de dépendance de rendre la demande contexte, conscients inutiles, qui sont censés être très sensible à la environnement utilisateur. En raison de la nature hautement dynamique des dispositifs mobiles dans un réseau ad-hoc, les dépendances contexte de trop nombreux peuvent gravement affecter la performance d'une application contexte courant. Réduire les dépendances de contexte ou de fournir des sources alternatives contexte est crucial pour des gains de performances dans les applications contexte, conscients.

1.3 Les contributions

La thèse contribue à l'analyse, la représentation et la réduction des dépendances de contexte dans un environnement informatique ubiquitaire. Aux fins d'analyse, deux approches sont utilisées. Une approche basée sur la théorie des graphes détermine les dépendances le contexte actuel, en utilisant l'algorithme de tri topologique. Il en résulte une complexité temporelle linéaire en termes de Big O. L'autre approche adoptée pour trouver la nature de la dépendance contexte est fondée sur les réseaux de contraintes. Mais jusqu'à présent, la nature détermination des dépendances entre différentes variables dans un réseau de contraintes a été considéré comme NP-complet. La thèse a présenté une approche qui réduit l'analyse de la dépendance en utilisant le réseau de contraintes à la complexité polynomiale, tant en termes de complexité spatiale et temporelle.

La deuxième contribution de cette thèse est le modèle de représentation pour les dépendances de contexte dans un système. Il n'a pas été un système de représentation du contexte qui a été développé dans le seul but de représenter les dépendances de contexte, en général, et encore moins, les dépendances transitives dans les systèmes pervasifs. Le modèle de représentation

contexte présenté dans cette thèse est basée sur la logique des prédicats et la théorie des ensembles. Le modèle rend le processus de représenter les dépendances contexte assez facile et intuitive. L'approche perspicace qui est adoptée dans l'élaboration du modèle de dépendance contexte rend l'analyse des dépendances contexte simples à comprendre en termes de types de contexte et les sources de contexte, aux côtés de la prévoyance des sources autre contexte qui peut être utilisé par une application dépend du contexte. Dans la troisième contribution de cette thèse, une approche visant à réduire les dépendances de contexte est présentée, qui visent à combiner l'idée de contexte et le profil des réseaux de recouvrement ainsi que dans le but de proposer un cadre ouvert pour l'acquisition de contexte qui abordent également la question de la dépendance contexte. Cette approche permet à la notion de «Le contexte est toujours disponible", basée sur la fonction de ce contexte de l'utilisateur est accessible même s'il est hors ligne ou ses appareils sont éteint. Cela est possible parce que le contexte le profil distribue la dernière cadre d'un dispositif dans un réseau de recouvrement à travers des noeuds super, et le contexte est mis à la disposition même après que le dispositif a été mis à pied ou a quitté le réseau de recouvrement. Ainsi, une application peut toujours accéder à la dernière mise à jour contexte de la surcouche réseau, évitant résultante retard de l'appareil étant submergés par les demandes de contexte à partir d'applications multiples contexte de consommation.

1.4 Organisation

La contribution de thèse est organisée en trois sections principales: la recherche, la représentation et la réduction des dépendances de contexte. Dans la première partie de la thèse, nous présentons une solution efficace pour trouver les dépendances valides dans un environnement pervasif. Plus tard, nous présentons une approche représentation du contexte pour représenter ces dépendances de contexte valide, et de leurs substituts. Dans la dernière partie de la thèse, nous visons à réduire les dépendances de contexte dans un environnement

ad hoc en combinant l'idée de contexte et le profil réseau de recouvrement ainsi que dans le but de proposer un cadre ouvert pour l'acquisition de contexte. Cette approche permet à la notion de «Le contexte est toujours disponible", basée sur la fonction que la "dernière" de l'utilisateur contexte dans cet environnement est accessible, même s'il est hors ligne ou ses appareils sont éteint. Cela est possible parce que le contexte le profil conserve la dernière cadre du dispositif dans le réseau overlay, et est disponible même après que le dispositif a été mis à pied. Ainsi, une application peut toujours accéder aux dernières contexte, en évitant tout retard qui en résulte de la dépendance contexte d'une source contexte qui n'est plus accessible.

Les chapitres de la thèse sont organisées comme suit: Chapitre 2 donne les antécédents et les définitions fondamentales de l'informatique contexte courant; chapitre 3 revue de la littérature de détails concernant le problème de la recherche, le chapitre 4 présente les approches conçues pour déterminer les dépendances de contexte et de leur nature de la dépendance entre différentes applications; Le chapitre 5 présente un modèle de représentation contexte pour représenter ces dépendances contexte, le chapitre 6 présente l'approche contexte profil en détail dans le cadre de réseaux superposés pour réduire les dépendances de contexte; chapitre 7 utilise les résultats, basés sur des simulations et la modélisation mathématique, de mettre en évidence les avantages de notre approche de réduction des contexte de dépendance, et le chapitre 8 conclut la thèse en mettant l'accent sur les travaux futurs qui est prévu. Dans le chapitre suivant, l'arrière-plan de l'informatique contexte courant est détaillé.

2.1 Conditions d'arrière-plan

L'informatique ubiquitaire n'est pas un concept standard unique que tout le monde doit suivre. Il peut changer en fonction de la pensée personnelle, et par conséquent, il a été appelé avec beaucoup d'autres noms dans la littérature. Certains des noms alternatifs d'informatique omniprésente sont "pervasive computing", "l'informatique mobile distribuée", "informatique

mondiale», «l'Internet des objets», «informatique contexte courant", "intelligence ambiante", "wearable computing", "médias tangibles »,« physique informatique »,« everyware »,« l'ordinateur à disparaître "," technologie calme », etc Parce que des différentes façons de penser l'informatique omniprésente aujourd'hui, la première idée de l'informatique omniprésente présenté par Mark Weiser n'est pas exactement semblable à l'établi un aujourd'hui.

Everyware Greenfield (2006) fait le point que la référence à tous ces accents dont le nom diffère, «on se rappelle encore et encore de la parabole des six hommes aveugles décrivant un éléphant". Dans l'interprétation du livre, la parabole va comme ceci: "Six anciens sages du village ont été invités à décrire la vraie nature de l'animal qui avait été porté devant eux; malheureusement, l'âge et les infirmités avaient réduit tous à une dépendance à l'égard de la faculté de toucher. Un sage, tentatives et des échecs pour envelopper ses bras autour de la circonférence de la jambe ridée massif de la bête, a répondu que cela doit sûrement être parmi le plus puissant des arbres. Un autre discerné une grande tortue dans la douceur courbe d'une défense, tandis qu'un autre, à la rencontre sinueuse de l'éléphant, le tronc musculaire, pensait qu'il pouvait à peine ont été la manipulation rien d'autre que le roi des serpents. Aucun des six, en fait, pourrait venir n'importe où près d'un accord en ce qui concerne ce que c'était qu'ils éprouvent. "

Malgré les différentes versions de l'idée d'informatique omniprésente, l'idée centrale est toujours le même que celui présenté par Wesier. Toutes ces variations dans le calcul ubiquitaire offrent une bonne opportunité pour l'expansion des thèmes de recherche et leurs applications avec des vues et des objectifs différents. En raison des divers points de vue présentés, il ya des noms différents décrivant l'informatique ubiquitaire, mais pour les utilisateurs finaux ce n'est pas vraiment important. Comme les fonctionnalités de tous les systèmes ubiquitaires sont presque les mêmes, donc les utilisateurs finaux peuvent ne

remarquerez aucune différence notamment en termes de définitions et pour eux toutes les conceptions de systèmes ubiquitaires, les noms, les fonctionnalités sont exactement les mêmes.

Cette section suivante décrit omniprésente et le contexte informatique courant. Il couvre également les développements et les défis rencontrés par l'informatique contexte courant.

2.1.1 Informatique ubiquitaire

L'informatique ubiquitaire est communément connu sous le nom informatique omniprésente. L'omniprésent mot est défini comme «existant ou étant partout à la fois; constamment rencontrées; répandue." Le concept principal de l'informatique omniprésente est que les dispositifs informatiques différents sont interconnectés les uns avec les autres pour effectuer des tâches spécifiques, par exemple bureau de contrôle de la température lorsque l'utilisateur est dans le bureau. La informatique ubiquitaire concentre son adaptation en fonction des besoins et exigences des utilisateurs. Ces appareils eux-mêmes de gérer automatiquement en fonction des changements dans les besoins de l'utilisateur. Mark Weiser, le pionnier de l'informatique omniprésente, a déclaré: "les technologies les plus profondes sont celles qui disparaissent, ils tissent eux-mêmes dans le tissu de la vie quotidienne jusqu'à ce qu'ils se confondent avec elle", Weiser (1991).

A partir des définitions ci-dessus, à la suite des caractéristiques importantes peuvent être déduites de l'informatique ubiquitaire:

• Les appareils informatiques sont éparpillés un peu partout autour de nous, et ils sont interconnectés les uns aux autres,

• Ils sont autonomes et ne nécessitent pas une attention continue active de l'utilisateur, et

• Ils deviennent invisibles en étant parfaitement intégré dans l'environnement. Mark Weiser était contributeur majeur à l'idée de l'informatique omniprésente. Il était le chef scientifique au Xerox Palo Alto Research Centre, et dans les années 1990 le PARC de Xerox

déjà eu de nombreuses contributions à l'informatique moderne comme Ethernet et de calcul distribué, le logiciel pour le poste de travail informatique personnelle, la programmation orientée objet, etc En outre plusieurs nouvelles technologies telles que IPv6, etc ont été mis au point par Xerox (1996). Weiser fut le génie de l'innovation et visionnaire. Il a dirigé l'équipe de recherche de contribuer efficacement à des innovations dans le 21e siècle. Weiser et son équipe sont considérés comme le premier à expliquer l'utilité de la technologie informatique du point de vue humain.

Phase I - L'ère Mainframe

Le mainframe terme "[rappels] la relation des gens avec des ordinateurs qui ont été la plupart du temps gérés par des experts derrière des portes closes. Chaque fois qu'un ordinateur est une ressource rare, et doit être négociée et partagée avec d'autres, notre relation est celle de l'ère mainframe, "(Weiser, 1996) à savoir« de 1940 à 1980 environ. "

Phase II - L'ère de l'ordinateur personnel

"En 1984, le nombre de personnes utilisant des ordinateurs personnels a dépassé le nombre de personnes utilisant les ordinateurs partagés. La relation personnelle est l'informatique personnelle, voire intime, (...) et vous interagissez directement et profondément avec elle. L'ordinateur personnel est le plus analogue à l'automobile -. Une spéciale, élément relativement coûteux, ce que (...) nécessite une attention considérable à exploiter »(Weiser, 1996)

Transition - L'Internet et l'informatique distribuée

«[Les gens] et leur information sont devenus interconnectés. [Le] Internet rassemble des éléments de l'ère mainframe et l'ère du PC. Il est client-serveur informatique à une échelle massive, avec des clients Web les PC et les serveurs Web des mainframes. "(Weiser, 1996) L'informatique ubiquitaire n'est pas un nouveau concept ou une technologie, c'est une idée d'amener toutes les technologies actuelles de travailler ensemble dans le même

environnement. Par exemple, quelques années en arrière les téléphones mobiles étaient chers et grands dans la taille, mais maintenant, ils ont évolué dans des dispositifs plus minces et plus petits. Ce concept peut également être appliquée à l'informatique. Maintenant, les serveurs de données sont grandes, mais à l'avenir, ils vont également évoluer dans les ressources plus petites et efficaces comme les téléphones mobiles, et ne seront donc pouvoir être intégrés dans des dispositifs différents, comme les appareils ménagers et du matériel de bureau, de les interconnecter entre eux et avec l'Internet . Une telle ère sera vraiment appelé l'informatique omniprésente lorsque tous les appareils seront interconnectés ensemble. Phase III - L'ère de l'informatique ubiquitaire "[Cette] ère aura (...) les ordinateurs qui partagent chacun de nous. Certains d'entre eux (...) nous pouvons accéder au cours de quelques minutes de navigation sur Internet. D'autres seront intégrés dans les murs, les chaises, des vêtements, les interrupteurs, les voitures-en tout. [Ubicomp] se caractérise fondamentalement par la connexion des choses dans le monde avec le calcul. Cette rencontre aura lieu sur des échelles beaucoup, y compris le microscope. "(Weiser, 1996) et," activer le monde. Fournir des centaines de dispositifs informatiques sans fil par personne et par bureau, de toutes les échelles (...). Cela a exigé de nouveaux travaux dans les systèmes d'exploitation, les interfaces utilisateur, les réseaux, sans fil, les écrans, et de nombreux autres domaines. Il est invisible, partout l'informatique qui ne vivent pas sur un dispositif personnel d'aucune sorte, mais il est dans le travail du bois partout. "(Weiser, 1991) Les technologies qui sont utilisées aujourd'hui également en charge le concept de l'informatique omniprésente. Par exemple, IPv6 est conçu pour accueillir des dispositifs beaucoup plus que IPv4 pourrait. En étendant l'espace d'adressage en IPv6, il devient maintenant une réalité pour se connecter de nombreux appareils plus petits comme les microprocesseurs, les agents intelligents artificiels, etc raccordement de ces appareils plus

petits favorise l'informatique omniprésente comme aujourd'hui plus de périphériques peuvent interagir et de partager l'information avec l'autre que jamais.

Depuis, l'informatique ubiquitaire cherche à connecter différents périphériques afin de partager l'information, il est donc absolument nécessaire pour un système à être considéré comme un système informatique ubiquitaire, de ne pas traiter une certaine technologie séparément plutôt toutes les technologies doivent être interconnectées et reliées à Internet. Aujourd'hui, il ya beaucoup de nouvelles applications de l'informatique omniprésente dans le monde informatique moderne. Par exemple, les appareils mobiles de télécharger automatiquement le dernier logiciel de rester à jour avec les dernières fonctionnalités telles que les applications sociales sur les téléphones intelligents. Un autre exemple pourrait être le lecteur de musique qui peut observer le choix de l'utilisateur de chansons et sur la base qui permet de télécharger les dernières chansons de telle sorte que l'utilisateur n'a pas a la recherche de contenus liés nouvelle. En outre, des projets de recherche tels que DIYSE essayons de développer un prototype de cuisine qui sera en mesure de télécharger les nouvelles recettes en fonction des goûts culinaires de l'utilisateur. Ces systèmes seront utiles dans notre vie quotidienne parce que les utilisateurs n'ont pas à passer leur temps à la recherche d'un contenu particulier et peuvent obtenir ce qu'ils désirent automatiquement par les applications intelligentes en cours d'exécution en arrière-plan.

Ces quelques exemples donnent un bon aperçu comment les systèmes informatiques ubiquitaires va ressembler dans l'avenir proche où ils ont atteint une certaine maturité et ce qui peut être leurs avantages pour l'utilisateur final. Ce genre d'idées ne se concentre pas seulement sur les performances de la technologie elle-même mais aussi sur la façon dont les différentes technologies peuvent bénéficier les humains dans leurs tâches de la vie quotidienne. Cela signifie l'interconnexion des appareils et leur utilisation à partir du point de vue humain plutôt que de simplement les performances des machines. Des systèmes

informatiques ubiquitaires donnera une nouvelle dimension à la vie humaine. Les humains pourront sauver leur temps qui a été précédemment utilisé dans la recherche, effectuer des tâches complexes, etc Avec ce concept, les humains seront capables d'obtenir l'information rapidement et sans trop d'effort. En outre, dans l'avenir de l'informatique ubiquitaire utilisation des systèmes sera dans presque tous les appareils et les êtres humains acceptent sa présence avec le confort, parce que les humains peuvent ensuite utiliser leur temps précieux pour mener à bien certaines tâches d'autres importants puisque le système informatique ubiquitaire sera travaillé dans les coulisses de fournir les résultats nécessaires et des adaptations en fonction de l'environnement utilisateur.

2.1.2 Informatique contexte, conscients Wikipedia (2005a) définit la sensibilité au contexte comme un terme d'informatique qui est utilisé pour signifier «appareils qui ont des informations sur les circonstances dans lesquelles elles opèrent, et peut réagir en conséquence." Cela signifie que les appareils ont un accès complet à l'information pertinente et peut s'adapter à l'environnement changeant en profitant d'informations contextuelles disponibles dans l'environnement, Chen & Kotz (2000)

Avant l'arrivée des applications contexte, conscients, les applications informatiques courantes ont émis des hypothèses sur le contexte de l'utilisateur parce qu'ils n'étaient pas conscients contexte, Lieberman et Selker (2000), mais maintenant, les applications peuvent détecter contexte actuel de l'utilisateur et d'ajuster leur comportement en conséquence. Lieberman et Selker (2000) affirment également que les applications contexte, conscients devrait se concentrer sur l'utilisateur et contexte d'application, car avec une telle approche, l'interaction avec l'utilisateur sera minime et les périphériques de surveiller automatiquement tout l'environnement qui entoure l'utilisateur sans nécessiter la moindre attention directe de l'utilisateur.

2.2 Contexte

Le contexte terme peut être ouverte à des interprétations diverses dans différents domaines d'application en fonction des applications informatiques qui utilisent le contexte. Selon Dey et al. (2000a), le contexte est "toute information qui peut être utilisée pour caractériser la situation des entités (à savoir si une personne, un lieu ou un objet) qui sont jugées pertinentes pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application eux-mêmes . "Cette définition est considérée comme un classique que des informations concernant les différentes entités peuvent être exprimées en tant que contexte. Selon Winograd (2001), le contexte »est un terme opérationnel: quelque chose est cadre en raison de la façon dont elle est utilisée dans l'interprétation, pas en raison de ses propriétés intrinsèques." Bien que, selon Coutaz et al. (2005), le contexte "n'est pas simplement l'état d'un environnement prédéfini avec un ensemble fixe de ressources d'interaction. Il fait partie d'un processus d'interaction avec un environnement en constante évolution composée de reconfigurable, migrateur, distribués, et des ressources multi-échelles. "

Comme indiqué précédemment, la définition la plus commune du contexte est "toute information qui peut être utilisée pour caractériser la situation des entités" [Dey, 2000]. Dans cette section, nous allons essayer de développer des données de contexte par rapport au contexte de la souligner comme un aspect plus orienté traitement du contexte. Ainsi, toute valeur détectée avec un supplément de méta-information, pour aider à l'interprétation d'une telle valeur, doivent être considérés comme des données contextuelles. La caractéristique essentielle est l'interprétation de la valeur détectée pour obtenir des données de contexte. Def. 1: ". Données de contexte est un élément d'information comprenant au moins une pièce de chaque valeur détectée correspondante, et des méta-informations à comprendre le sens de la valeur détectée" pièce telle de méta-informations peuvent être une identité du capteur qui est nécessaire pour comprendre une valeur, comme la température, tout dépend du niveau d'abstraction. L'approche commune dans l'informatique ubiquitaire est de fusionner unité de

mesure appropriée à la valeur détectée qui peut être facilement utilisé par des applications diverses contexte conscients.

Def. 2: «La méta-information ne peut être détectée et par conséquent doit être fournie de l'extérieur du système généralisé elle-même." Exemple le plus trivial de la méta-information est l'identité d'un objet qui fournit des informations concernant de domaine, la structure et le but de celui-ci, fourni par un utilisateur ou une autre application. On peut en déduire que les données de contexte peut être représenté par un modèle à étapes multiples dans lequel le niveau le plus bas, les valeurs captées et leur première méta-informations existent sans aucune relation existant entre eux. La valeur détectée et la méta-information sont fusionnés à un niveau plus élevé pour obtenir des données de contexte de base ou "contexte du premier ordre".

Def. 3: "valeur détectée et une méta-informations sont fusionnées pour obtenir des données de contexte appelé contexte de premier ordre en une seule étape." Contexte du premier ordre peut être considéré comme un ensemble de données plus fiables contexte comparativement à d'autres, la hausse des données de contexte ordre. La raison étant que tous les renseignements requis pour la génération et la fusion des données de contexte a été recueilli par le contexte de périphérique générant lui-même. Par conséquent, il est en mesure de donner une mesure de fiabilité élevé pour un tel contexte de premier ordre. Dans chacun des contextes ultérieurs d'ordre supérieur qui sont générés à des niveaux plus élevés en obtenant des valeurs détectées et méta-information exigent au moins un contexte de premier ordre. En outre, la génération du contexte d'ordre supérieur en fusionnant contexte du premier ordre et plus méta-information est un processus récursif.

Def. 4: "cadre supérieur commande est un ensemble de données de contexte obtenue par fusion au moins un contexte du premier ordre ou d'ordre supérieur différente cadre avec la valeur détectée avec, ou non, la méta-informations."

1. Introduction

1.1 Background

Context aware applications are the foundation for today's world of pervasive computing. Services and applications in a ubiquitous environment have to interact with a variety of mobile devices, communication networks, and wireless sensors along with the variability of their respective execution environment. Therefore, context aware applications should be conscious of their running context in order to react accordingly to multiple execution scenarios. Also, personalization of services depends upon context aware applications to achieve the vision of ubiquitous computing dreamt by Weiser (1991). In a ubiquitous computing environment, various heterogeneous applications and devices are involved, generating and consuming context information rapidly. Data and context information from sensor nodes, that are available in ad-hoc networks, need to be utilized by context aware services and applications. Mobile devices are also leaving and joining ad-hoc networks at a dynamic rate, making context information highly unpredictable in nature. Such an ever changing nature of context information makes context aware applications more complex to develop, maintain and comprehend.

Context is usually referred to the attributes in a continually dynamic computing environment in which various aspects like the user's location, social situation and interaction with other resources are alternating constantly. May be because of its more general approach, the definition provided by Dey and Abowd (1999) is most widely quoted in the research community: "Context is any information that can be used to characterise the situation of an entity." An entity can be defined as "a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves."

Before 1980's, there were already some researchers that have thought of ubiquitous computing, but it was in 1988 that the revolutionary idea of ubiquitous computing embedded in the human environment was presented by Mark Weiser. Previously, there were few researches that had touched upon the idea of ubiquitous computing. One such research was called Human-Computer Interaction (HCI). HCI was the closest to the idea of Weiser's ubiquitous computing. For example, HCI system improved qualities of learning and efficiency of system use. Furthermore, the gesture recognition of HCI is an important concept for ubiquitous computing. Similarly, there was the Computer-Supported Cooperative Work (CSCW), the online work participation system for people situated at multiple sites physically. These topics of HCI were related to the ubiquitous computing idea of Weiser (Brad, 1998).

Another research carried out in 1995, called *Things That Think* (Media Lab, 2011a), by MIT Media Lab studied smart designs for the interaction between home appliances and the users for the prospect of improving usability. Another related research that still is in use is called Radio-Frequency Identification (RFID) transponder, which was built for usability purposes and is now an essential part of any ubiquitous system scenario involving location of an object. RFID was initially developed in 1940's but perfected in 1973 (Landt, 2001). Another similar technology, the barcode, was developed in early fifties but was only available commercially in the 1980s.

A major step towards ubiquitous computing was the development of the first generation of smart phones which were able to support non-voice communication. It was adopted by the commercial users successfully because of the advantage of communicating from anywhere, with or without voice.

Berkeley Sensors and Actuator Center (BSAC, 2008) was established in 1986, and took the advantage of the sensors' and actuators' technologies coming out of the Xerox PARC. In 1990's, various technologies started to get interconnected as microcontrollers and transceivers

get smaller, becoming the main component of various technologies. In 1998, as a part of SmartDust project (SmartDust, 2008), the sensing technologies, microcontrollers and transceivers in ISM bands 2 were connected with the wireless sensor nodes, thus giving an interesting background for the development and research for today's wireless sensor networks such as TinyOS (TinyOS, 2011), Crossbow (Crossbow, 2008) and Dust (Dust, 2008). Furthermore, the development of Global Positioning System (GPS) kick started various location based applications, such as in scenarios where people use GPS to find their current location on the map during travel.

The idea of Weiser can also be attributed to science fiction novels and movies. In Philip K. Dick's 1969 novel Ubik, the writer wrote about an environment rich with different technologies helping humans in multiple ways. In the novel, the human everyday life was run by the news-reading machines, clean-up robots and talking doors. The novel provides a good reflection of the future of the ubiquitous computing. Fiction writers didn't have their own ideas much different than Weiser's. They didn't know anything about ubiquitous computing from a technical perspective, yet they were able to write such novels that overlapped the idea proposed by Weiser, for example using different home appliances to communicate with the user, find a location of the user, etc. All these fiction writings showed at that time how the human future could look like, and now it has started to happen in our lifetime.

1.2 Motivation

Every novel emerging technology has to face many challenges to achieve success and trust. Initially, the challenges are only in developing the prototype. Once the prototype works according to the required standards and the researchers think that now the project is about to reach a certain level of maturity which will benefit the community in the long term, then the upcoming challenges faced by the project to make the technology ubiquitous are known as

grand challenges. The definition of grand challenges was formulated by UK's Computer Research Committee (Grand Challenge, 2008). The grand challenges are:

(i) Backed by the scientific community

(ii) Ambition is far greater than achieved by a single research team

(iii) Challenge is directed at revolutionizing the project to industry ready products.

But the grand challenge opportunity for most projects comes rarely, as it is a huge task to bring the project to a certain maturity level so that it can then revolutionize the human lives. For examples, grand challenges can be the mapping the human genome (accomplished) or unifying the theories for the four types of forces in physics (under investigation).

The Ubiquitous Computing Grand Challenge (Chalmers, 2006) is one of currently nine such challenges, based on the theory and engineering of ubiquitous systems. On one hand, it strives to achieve Weiser's idea in this era and on the other hand, the engineering and the social issues which are focused on by different research groups. It also preserves "the theory required underpinning the design and analysis of ubiquitous systems which are intrinsically large-scale and complex". There are few important questions that are related to Ubiquitous Computing Grand Challenge like "How many computers will you be using, wearing, or have installed on your body, in 2020? How many other computers will they be talking to? What will they be saying about you, doing for you, or to you? (...) Shall we be able to manage such large-scale systems, or even understand them? How do people interact with them and how does this new pervasive technology affect society? How can non-computing people configure and control them? What tools are needed for design and analysis of these constantly adapting and evolving systems? What theories will help us to understand their behaviour?" (Chalmers, 2006). The answers to these questions are not straight forward. There is a need to observe every aspect of the Ubiquitous Computing Grand Challenge in order to overcome these

research problems. The questions can be analysed through three distinctive perspectives: experience, engineering and theoretical.

Experience – Study people's behaviour with the ubiquitous computing systems, and based on such study improve the system that should be more directed towards socio-technical perspective.

Engineering – Solutions to the challenges we faced are because of the dynamic nature of the ubiquitous system. Every system is dependent on efficient design, infrastructure and approach. These principles are also applicable to the ubiquitous computing systems. These systems should have efficient designs, and good network infrastructure.

Theoretical – Detailed analytical study is always important either before the system development or during the system performance. With proper research approach, new techniques, models and methods can be used within the system to make it more efficient and high performance that should meet the required standards of the customers.

Besides these perspectives, other goals should also be set that are more directed towards individual and social interaction. The proper design, use of techniques and models, fulfilling the requirements of individuals and societies and continuous research will remarkably enhance the growth of ubiquitous computing systems.

To define context dependency in ubiquitous systems, it's better to explain with a help of a small example. Most context aware applications use location context. The location context can be acquired from various sources. The common resources in context aware systems are the physical sensors, which sense the location of the person in the room e.g. Global Positioning System (GPS), and derived context e.g. if the personal computer of the person in his office is currently being use, it can be derived that the said person is his in office. So, the context aware application is dependent on one of these sources to acquire location context. Therefore, there exists context dependency between the context aware application looking for

location context and the source providing the location context. If, for some reason, the context source is not available, the context aware application can ask other context sources to provide the relevant context. So, finding context dependency and managing relevant context dependencies is important for the successful working of context aware application. If a context aware application is dependent on multiple context sources for acquiring various different contextual data, then a failure to acquire any one of those required context information can result in the failure of such a context aware application. So, it's important to reduce context dependence of context aware application on context sources and producers which are not reliable enough to provide consistent service as requested by the application.

The context acquisition process is a vital feature of any context aware system. Most approaches try to acquire the device context, e.g. the presence of a person in the room from the location sensor, instead of the "profile context". Profile context is the aggregated context from various context sources that gives an overall better picture of the subject, e.g. the profile context, that tells us that the user is busy, can be acquired from aggregation of location context, telling us that the user is in office, and usage context of his personal computer along with the busy phone line. Each device owned by a user is contacted individually for the purpose of acquiring data relevant to the user situation in common approaches but our approach provides aggregated context on one request in order to reduce context queries for the resource constrained devices. Context published by a device individually fails to capture the overall current picture of user's situation in comparison to context obtained from multiple sources that are aware of their mutual existence, and can provide better user's context, working in collaboration among them. User context, in our approach, is usually dependent upon two or more devices' sensed data, rather than just a single device context.

Despite their potential widespread applications, context-based applications and services are still in infancy because the context acquisition architecture is based, more or less, on query-

based silo framework in which a service or application directly asks for certain context from a context source individually. Such a framework is susceptible to numerous issues; most common of them are bottleneck performances and high probability of failure because of non-efficient resource management of the system. If the server-client approach is employed to realize a context aware service, it can be easily imagined that servers will soon be overloaded with queries, affecting their performance. The dynamic nature of information to be dealt with in a cloud computing environment is huge, and the numbers of queries increase with time as more users join the ad hoc network asking for constant updates from context sources for context types such as location.

Some simple context aware applications use sensed data only from a single source to make appropriate personalized decisions for the user. Most, though, utilize sensed data and processed context information from multiple sources to derive further high-level context information. Such abstract context requires multiple low-level device context which are aggregated and processed over many times to obtain an abstract-level context. To provide the user with most benefits and personalized services using the available computing resources in an ad-hoc networks, context aware applications need to disseminate both the accessed and derived context information in the network.

Derived context is inherently dependent upon sensed data or/and other processed context information. Context dependency may get inducted in a system due to various reasons such as error at the time of system deployment. But that issue aside, the more context dependencies an application has, the more complex the process becomes to develop and maintain such an application because a failure of one context dependency will render the context aware application useless, which are expected to be highly responsive to the user environment. Due to highly dynamic nature of mobile devices in an ad-hoc network, too many context dependencies can severely affect the performance of a context aware application. Reducing

context dependencies or providing alternative context sources is crucial for performance gains in context aware applications.

1.3 Contributions

The thesis contributes to the analysis, representation and reduction of context dependencies in a ubiquitous computing environment. For the analysis purpose, two approaches are used. One approach based on graph theory determines present context dependencies by using topological sort algorithm. This results in linear time complexity in terms of Big O. The other approach adopted to find the nature of context dependency is based on constraint networks. But until now, determining nature of dependencies among various variables in a constraint network was considered NP-complete. The thesis presented an approach which reduces the dependency analysis using constraint network to polynomial complexity, both in terms of space and time complexities.

The second contribution of this thesis is the representation model for context dependencies in a system. There hasn't been a context representation system that was developed with the sole purpose of representing context dependencies in general, let alone, the transitive dependencies in pervasive systems. The context representation model presented in this thesis is based on predicate logic and set theory. The model makes the process of representing context dependencies quite easy and intuitive. The insightful approach that is adopted in developing the context dependency model makes analysis of context dependencies straightforward to understand in terms of context types and context sources, alongside with the forethought of alternate context sources that can be used by a context dependent application.

In the third contribution of this thesis, an approach to reduce context dependencies is presented which aim to combine the idea of profile context and overlay networks together for the purpose of proposing an open framework for context acquisition which also address the

issue of context dependency. This approach enables the notion “Context is always available”, based on the feature that the user’s context is accessible even if he is offline or his devices are switched-off. This is possible because profile context distributes the latest context of a device in an overlay network through super nodes, and the context is made available even after the device has been turned-off or has left the overlay network. Thus, an application can always access last updated context from the overlay network, avoiding delay resultant of device being overwhelmed by context requests from multiple context consuming applications.

1.4 Organization

The thesis contribution is arranged in three main sections: finding, representing and reducing context dependencies. In the first part of the thesis, we present an efficient solution to find valid dependencies in a pervasive environment. Later, we present a context representation approach to represent these valid context dependencies and their alternatives. In the later part of the thesis, we aim to reduce context dependencies in an ad hoc environment by combining the idea of profile context and overlay network together for the purpose of proposing an open framework for context acquisition. This approach enables the notion “Context is always available”, based on the feature that the user’s “last” context in that environment is accessible even if he is offline or his devices are switched-off. This is possible because profile context keeps the latest context of the device in overlay network, and is available even after the device has been turned-off. Thus, an application can always access latest context, avoiding any delay resultant of context dependency on a context source which is no more accessible.

The chapters of the thesis are organized as follows: Chapter 2 gives basic background and definitions of context aware computing; Chapter 3 details literature review concerning the research problem; Chapter 4 presents approaches devised to determine context dependencies and their nature of dependence among different applications; Chapter 5 presents a context

representation model to depict these context dependencies; Chapter 6 introduces profile context approach in detail in the setting of overlay networks to reduce context dependencies; Chapter 7 uses results, based on simulations and mathematical modelling, to highlight the advantages of our context dependency reduction approach; and Chapter 8 concludes the thesis with focus on future work that is planned. In the next chapter, the background of the context aware computing is detailed.

2. Context in Ubiquitous Computing

2.1 Background Terms

Ubiquitous computing is not a single standard concept that everyone has to follow. It can change depending on the personal thinking, and therefore it has been called with many alternate names in literature. Some of the alternative ubiquitous computing names are "pervasive computing", "mobile distributed computing", "global computing", "the Internet of Things", "context aware computing", "ambient intelligence", "wearable computing", "tangible media", "physical computing", "everyware", "the disappearing computer", "calm technology" etc. Because of the different ways of thinking about ubiquitous computing today, the first idea of ubiquitous computing presented by Mark Weiser is not exactly similar to the established one today.

Greenfield's Everyware (2006) makes the point that referring to all these differently named emphases, "one gets reminded time and again of the parable of the six blind men describing an elephant". In the book's interpretation, the parable goes like this:

"Six wise elders of the village were asked to describe the true nature of the animal that had been brought before them; sadly, age and infirmity had reduced them all to a reliance on the faculty of touch. One sage, trying and failing to wrap his arms around the wrinkled circumference of the beast's massive leg, replied that it must surely be among the mightiest of trees. Another discerned a great turtle in the curving smoothness of a tusk, while yet another, encountering the elephant's sinuous, muscular trunk, thought he could hardly have been handling anything other than the king of snakes. None of the six, in fact could come anywhere close to agreement regarding what it was that they were experiencing."

Despite different versions of ubiquitous computing idea, the central idea is still the same as presented by Wesier. All these variations in the ubiquitous computing offer a good opportunity for the expansion of the research topics and their applications with different views

and goals. Because of the various views presented, there are different names describing ubiquitous computing but for the end users it doesn't really matters. As the functionalities of all the ubiquitous systems are almost the same, therefore the end users can't notice any particular difference in terms of definitions and for them all the ubiquitous system designs, names, functionalities are exactly the same.

This next section describes ubiquitous and context aware computing. It also covers the developments and challenges faced by context aware computing.

2.1.1 Ubiquitous Computing

Ubiquitous computing is commonly known as pervasive computing. The word ubiquitous is defined as "existing or being everywhere at the same time; constantly encountered; widespread."[1] The main concept of the ubiquitous computing is that different computing devices are interconnected with each other to perform specific tasks, e.g. monitoring office temperature when the user is in the office. The ubiquitous computing focuses its adaptation based on the user needs and requirements. These devices manage themselves automatically based on the changes in the requirements of the user. Mark Weiser, the pioneer of ubiquitous computing, said "*the most profound technologies are those that disappear, they weave themselves into the fabric of everyday life until they are indistinguishable from it",* Weiser (1991).

From the above definitions, following important characteristics can be inferred about ubiquitous computing:

• The computing devices are scattered everywhere around us, and they are interconnected to each other,

• They are autonomic and do not require continuous active attention from the user, and

• They become invisible by being seamlessly integrated into the environment.

[1] MerriamWebster Online Dictionary

Mark Weiser was major contributor to the idea of ubiquitous computing. He was the head scientist at Xerox Palo Alto Research Centre, and in the 1990s the Xerox PARC already had many contributions to the modern computing such as Ethernet and distributed computing, the software for the personal computer workstation, object-oriented programming, etc. Furthermore several new technologies like IPv6 etc. were developed by Xerox (1996). Weiser was innovation genius and visionary. He led the research team to contribute effectively to the innovations in the 21st Century. Weiser and his team are considered to be the first to explain the usefulness of computer technology from the human perspective.

Phase I – The *Mainframe* Era

The term *mainframe* "[recalls] the relationship people had with computers that were mostly run by experts behind closed doors. Anytime a computer is a scarce resource, and must be negotiated and shared with others, our relationship is that of the mainframe era," (Weiser, 1996) i.e. "from 1940 to about 1980."

Phase II – The *Personal Computer* Era

"In 1984 the number of people using personal computers surpassed the number of people using shared computers. The personal computing relationship is personal, even intimate, (...) and you interact directly and deeply with it. The personal computer is most analogous to the automobile – a special, relatively expensive item,that (...) requires considerable attention to operate." (Weiser, 1996)

Transition – The *Internet* and *Distributed Computing*

"[People] and their information have become interconnected. [The] Internet brings together elements of the mainframe era and the PC era. It is client-server computing on a massive scale, with web clients the PCs and web servers the mainframes." (Weiser, 1996)

Ubiquitous computing is not a new concept or a technology; it is an idea to bring all the current technologies to work together in the same environment. For example, few years back

the mobile phones were expensive and big in size but now they have evolved into thinner and smaller devices. This concept can also be applied to the computing. Now the data servers are big but in future they will also evolve into smaller and efficient resources like mobile phones, and therefore will be able to be integrated into different devices, like household appliances and office equipment, to interconnect them together and also with the internet. Such an era will be truly called ubiquitous computing when all the devices will be interconnected together.

Phase III – The *Ubiquitous Computing* Era

"[This] era will have (...) computers sharing each of us. Some of these (...) we may access in the course of a few minutes of Internet browsing. Others will be imbedded in walls, chairs, clothing, light switches, cars—in everything. [Ubicomp] is fundamentally characterized by the connection of things in the world with computation. This will take place at a many scales, including the microscopic." (Weiser, 1996) And, "activate the world. Provide hundreds of wireless computing devices per person per office, of all scales (...). This has required new work in operating systems, user interfaces, networks, wireless, displays, and many other areas. It is invisible, everywhere computing that does not live on a personal device of any sort, but is in the woodwork everywhere." (Weiser, 1991)

The technologies that are being used today also support the concept of ubiquitous computing. For example, IPv6 is designed to accommodate a lot more devices than IPv4 could. By extending the address space in IPv6, it now becomes a reality to connect many smaller devices like microprocessors, artificial intelligent agents, etc. Connecting such smaller devices promotes ubiquitous computing as now more devices can interact and share the information with each other than ever before.

Since, ubiquitous computing strives to connect different devices together to share the information therefore, it is absolute necessary for a system to be considered a ubiquitous

computing system, to not treat a certain technology separately rather all technologies must be interconnected and connected to the internet. Today, there are many new applications of ubiquitous computing in modern computing world. For example, mobile devices automatically download the latest software to remain update with latest features like social apps on smart phones. Another example could be the music player that can observe the user's choice of songs and based on that can download the latest songs so that the user doesn't has to search for new related content. Furthermore, research projects like DIYSE are trying to develop prototype kitchen that will be able to download the new recipes depending on the user's culinary taste. These systems will be helpful in our daily life because users don't have to spend their time on searching a particular content and can get what they desire automatically by smart applications running in the background.

These few examples provide a good overview how the ubiquitous computing systems will look like in the near future when they have reached a certain maturity and what can be their benefits to the end user. Such kind of ideas focuses not only on the performance of the technology itself but also on how different technologies can benefit humans in their daily life tasks. This means interconnection of devices and their use from the human point of view rather than just machines performance. Ubiquitous computing systems will give a new dimension to human life. Humans will be able to save their time that was previously being used in searching, doing complex tasks, etc. With this concept, humans will be able to get the information quickly and without much effort. Furthermore, in future the ubiquitous computing systems use will be in almost every device and the humans will accept its presence with comfort, because humans can then utilize their valuable time to carry out some other important tasks since ubiquitous computing system will be working behind the scenes to provide the required results and adaptations according to the user environment.

2.1.2 Context Aware Computing

Wikipedia (2005*a*) defines context awareness as a computer science term which is used to denote "devices that have information about the circumstances under which they operate, and can react accordingly." This means that the devices have complete access to the relevant information and can adjust to the changing environment by taking advantage of contextual information available in the environment, Chen & Kotz (2000)

Before the arrival of context aware applications, the common computer applications made assumptions about the user's context because they weren't context aware, Lieberman & Selker (2000), but now the applications can sense the user's actual context and adjust their behaviour accordingly. Lieberman & Selker (2000) also argue that the context aware applications should focus on the user and application context because with such an approach, the user interaction will be minimal and the devices will automatically monitor the entire environment surrounding the user without requiring any direct attention from the user.

2.2 Context

The term Context can be open to various interpretations in different application domains depending on the computer applications that utilize context. According to Dey et al. (2000a), context is "any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves." This definition is considered to be a conventional one as information regarding various entities can be expressed as context. According to Winograd (2001), context "is an operational term: something is context because of the way it is used in interpretation, not due to its inherent properties." While, according to Coutaz et al. (2005), context "is not simply the state of a predefined environment with a fixed set of interaction resources. It's part of a process of interacting with an ever-changing environment composed of reconfigurable, migratory, distributed, and multiscale resources."

As previously stated, the most common definition of context is “any information that can be used to characterize the situation of entities”[Dey, 2000]. In this section, we will try to elaborate on context data as compared to context to emphasize it as a more processing oriented aspect of context. So, any sensed value along with an additional meta-information, to help with the interpretation of such value, should be considered as context data. The vital feature is the interpretation of the sensed value to achieve context data.

Def. 1: “context data is a piece of information comprising at least one piece each of sensed value and corresponding meta-information to understand the meaning of sensed value.” Such piece of meta-information can be an identity of the sensor that is required to understand a value, such as temperature, all depending on the level of abstraction. The common approach in ubiquitous computing is to merge appropriate measurement unit with the sensed value which can be easily utilized by various context aware applications.

Def. 2: “Meta-information cannot be sensed and therefore must be provided from outside the pervasive system itself.” Most trivial example of such meta-information is identity of an object that provides information regarding domain, structure and purpose of it, provided by a user or some other application. It can be inferred that context data can be represented by a multiple stage model in which on the lowest level, the sensed values and their raw meta-information exist without any existing relationship among them. The sensed value and meta-information are merged at a higher level to obtain basic context data or “first order context”.

Def. 3: “sensed value and meta-information are merged together to obtain context data called First Order Context in one step.” First order context can be thought of as a more reliable context data comparatively to other, higher order context data. The reason being that all the information required for the generation and merging of context data has been collected by the context generating device itself. Therefore, it is in a position to give a high reliability measure for such a first order context. In each of the subsequent higher order contexts that are

generated at higher levels by obtaining sensed values and meta-information require at least one first order context. Furthermore, generation of higher order context by merging first order context and additional meta-information is a recursive process.

Def. 4: “Higher order context is a context data obtained by merging at least one first order context or different higher order context with sensed value along with, or not, its meta-information.”

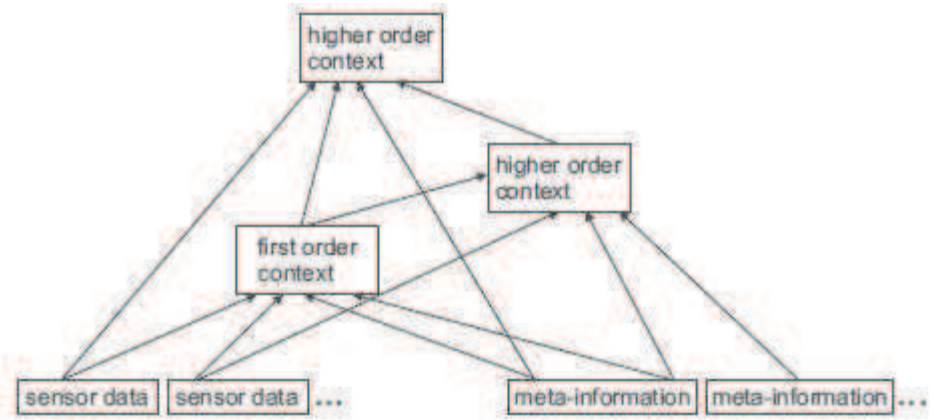

Figure 1 First order & higher order context

Figure 1 depicts an overview of context data generation steps. Every piece of context data has an associated type assigned to it. This associated type helps to identify that corresponding piece of context data as a result, produces a context class of the given context type.

Def. 5: “A context class gets induced by the context type associated with a particular piece of context data. This context class has all the attributes of a piece of context data but without any values assigned to those attributes.” In terms of object oriented design, a context class is a prototype which defines a concrete instance, an object, of a piece of context data.

Def. 6: “A context object or instance is a piece of context data belonging to a certain context class with values assigned to its attributes.” In a context aware application, two context objects belonging to the same context class should be distinguishable from each other by the values of some of their attributes. In pervasive systems, different entities interact with

each other by exchanging context data. An entity can either act as a provider or as a consumer of a piece of context data.

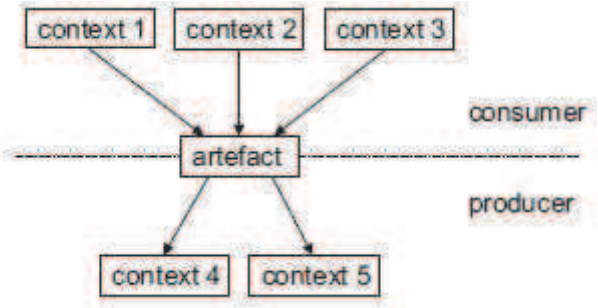

Figure 2 Context Producer and Context Consumer

Def. 7: "An entity that provides a piece of context data is called producer. An entity that uses a piece of context data is called consumer." Multiple entities can use the same piece of context data simultaneously. An entity can be a producer and a consumer at the same time, but not of the same piece of context data. Example can be the derivation of a higher order context from various pieces of context obtained from other entities and forwarding the higher order context data to its consumers. But an entity can only act as a consumer or a producer of a specific context data at any given time, not both. Otherwise, recursion occurs that can lead to deadlocks in the processing of context data.

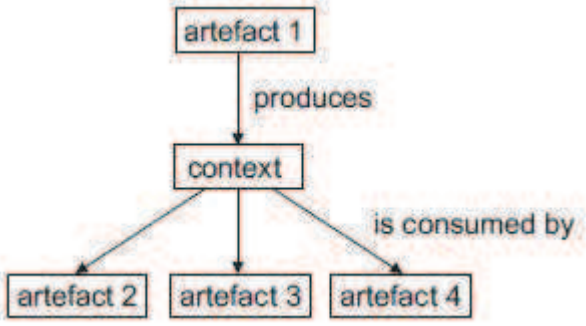

Figure 3Producer and consumer of specific context

Attributes of Context Data:

Context data have some general attributes that are available in the contextual representation even if the values of the context data are different. Understanding these attributes comprehensively will help the context aware applications by enabling them to better utilize these context data. The attributes of context data are the relevance of context data, validity of context data, reliability and history of context data. These four attributes can be classified into two groups: absolute and relative attributes. The value of absolute attributes can be evaluated independently, whereas the values of relative attributes need to be evaluated relative to the running state of the context aware application along with other previous versions of the same context data.

Relevance: The relevance of context data is dependent on factors like age and the distance of the source where the context was generated [Schmidt, 1999; Schmidt, 2002]. If a context data was generated a while ago, its relevance may be considered to be diminished relative to the current running state of the pervasive system. The reason being that the value being sensed at a sensor changes with time and context aware applications require latest sensed value to adapt their behaviour according to the prevalent user scenario. Therefore, after some time of being generated, the context data loses its benefit to the context consumer and is not relevant to the context aware applications anymore. The context data itself does not change over time, even though, its reliability has decreased. So, value of context data and its relevance attribute are independent of each other. One has to note that the history of context data is different from relevance of context data.

Reliability: In terms of trust, reliability is an important attribute because reliability of context aware applications is directly dependent upon the reliability of context data they operate on. It is essential to have a measure of reliability of context data so that context aware applications can judge the reliability of context data as their success depends on it. Dey (2002) introduced a way to handle ambiguous data by the process of mediation involving the

user. The user would rate the context data, but such an approach is only feasible for context data present at the application layer. User's input cannot be used for context data at lower layers. This approach lacked an applicable solution because the context aware application cannot wait for the user every time when there is new context data to operate on. This will take too much time and won't be context aware as such anymore due to regularly asking user for the input. Therefore, there must be some reliability metric to provide consumer of context data with added knowledge of assessing the context data without explicitly asking for the user input. Similarly, the derived context data can only be as reliable as the context data that was used to derive it. So, we can say that reliability of derived context data is dependent on the input context data's reliability. For this reason, higher order contexts are not more reliable than the lower order context that has been used to generate them. Reliability belongs to the group of absolute type attributes as the consumer of context data can directly use the reliability measure available from the source as a basis of its decision to utilize such context data or not.

History: the context data history attribute can be considered a dynamic attribute. If the latest context data is not recognized and utilized by the consumer, context aware application would most probably repeat the same action it took in the previous instance of time which won't be meeting the expectations of the user in terms of resource and time efficiency.

Validity: context data influences context aware applications, and therefore, need to be validated before it can be utilized. Evaluation in terms of validity of context data can only result in Boolean values: YES or NO. Validity attribute determines whether a piece of context data can be used for processing in a reasonable manner by context aware application or not.

2.3 Context Awareness

The term *Context awareness* refers to the situation when applications use context obtained from devices and sensors about the environment and the user, and then adapt their behaviour

accordingly. They can operate and adjust according to the changing environment. Generally, the computers are only used to provide information required by the users. They don't have any concern what type of information the user has requested. With the help of context awareness, the computer applications are able to know what the user's needs and priorities are. This all becomes possible by installing sensors, actuators and tiny computers that can gather the user's information and can create such environment that assist the user to retrieve information more easily, Nixon et al. (2002). Today the most commonly used applications of the context awareness are call forwarding, active badge systems, and office assistance, etc.

2.4 Context Categorization

It is easy to do context manipulation when there is homogeneous context information, but if the context information is heterogeneous then there is a need to simplify the context in order to manipulate it efficiently. For this purpose, context categorization is used through which the heterogeneous context can be categorized, and thus simplifying the context manipulation step.

There are two possible broad viewpoints regarding context categorization in the research literature:

• Conceptual viewpoint – who, where, what occurs, when, what can be used, what can be obtained, etc.

• Measurement viewpoint – what is the person's location or room temperature or network delay, etc?

2.4.1 Conceptual Categorization

The conceptual viewpoint is widely used and can be further categorized in to the following, Gwizdka (2000):

- Internal Context: the state of the user
- External context: the state of the environment, Petrelli et al. (2000)
- Material Context: the location, device and available infrastructure
- Social Context: social aspects and personal traits

Descriptions of contextual agents and their relations among them are provided in Table I to highlight the categorization of context.

Category	Semantics	Examples
User context	Who?	*User's Profile:* identifications, relation with others, to do lists, etc
Physical context	Where?	*The Physical Environment:* humidity, temperature, noise level, etc
Network context	Where?	*Network Environment:* connectivity, bandwidth, protocol, etc
Activity context	What occurs, when?	*What occurs, at what time:* enter, go out, etc
Device context	What can be used?	*The Profile and activities of Devices:* identifications, location, battery lifetime, etc
Service context	What can be obtained?	*The information on functions which system can provide:* file format, display, etc

Tableau 1 Conceptual Categorization of Context[2]

2.4.2 Measurement Categorization

Measurement context is about the observable values that are obtainable from sensors.

- Continuous Context

  In continuous context, the values are changing continuously and needs to be measured regularly to have an updated version of the context.

- Enumerative Context

  Here the values of context are a set of discrete values and defined in a list or set. They are based on set operations. A predicate calculus statement is used in describing context. Our approach of context modelling is also based on predicate logic. For more diverse

[2] Categorization and Modelling of Quality in Context Information by M.A. Razzaque, Simon Dobson and Paddy Nixon

information, these categories will assist the developers to efficiently manipulate the context information in the given environment.

- State Context

  In state context, there are a certain number of values that a context variable can take, e.g. the presence of the user in the meeting room: either the user is in the meeting room or is not in the meeting room, only two states in this example.

- Descriptive Context

  This is based on the description statement of the context and for this reason, it uses predicate calculus.

2.5 Characteristics of Context Information

To understand the nature of contextual information, it is imperative to realize the characteristics of the context information in pervasive computing systems. These are:

2.5.1 Temporal

There are two ways, as just discussed before, to categorize context information: static or dynamic. Static context describes the invariant aspects like date of birth, whereas dynamic context describes the aspects that are changing frequently and often. To obtain the dynamic context information sensors are used, whereas for acquiring static context information, users provide inputs to the system or it can be obtained from the user profile.

2.5.2 Imperfect

Context information is imperfect because most context information is dynamic, and this means that there is more probability of context information to contain errors as the sensed data could be inconsistent, incomplete, etc. These errors can result due to contradictory information or non-availability of complete information regarding the various contextual aspects.

Another reason for occurrence of such errors is processing and updating of data. In dynamic environment it is important to continuously update and process the information. Any delay can lead to the incorrect information being forwarded to context consumers that isn't relevant to the user environment anymore.

Besides information update and processing delay, it is quite possible that the users, and sensors, etc. have provided incorrect information. This can happen when the information is retrieved indirectly from different types of context information, like extracting a person's activity from location data.

2.5.3 Alternative Representations

Sensors are the vital component of pervasive systems. They provide the raw contextual information, but in doing so it is quite possible that some part of the data obtained from the sensors is not useful or reliable anymore, and this could lead to poor performance for a pervasive system. For example while locating the user, the sensors can provide the coordinates of the building instead of the room the user is present, and therefore, such information is not quite useful since the location of the user in terms of his presence in a certain room is still not precise enough. Thus, it is important to overcome such issues to obtain acceptable performance from the pervasive systems.

Furthermore, due to dynamic environment, it is quite possible that the requirements have also changed over time, and therefore, the context model must be able to allow and support multiple representations. This representation could be in different forms but must properly represent the characteristics of the context comprehensively.

2.6 Context Management

Managing contextual information is essential because for the applications to run smoothly, it is imperative that the retrieved information from the sensors and context producers must be meaningful and properly formatted. But that is not always the case. There is a possibility that

the information received from the sensors is in low-level, machine language format. Thus, making it impossible for the application to properly utilize that information in a timely manner without first converting it into a more accessible representation.

In order to overcome the delay in processing low-level or different formats of the information, a standard contextual model is essential. The model must provide standard specification necessary for correct assumptions within the applications, and could be used universally by the same set of applications or different ones.

Another important reason for using contextual modelling is the presence of application in heterogeneous environments. In such environments, heterogeneous devices use heterogeneous information sources and also share the information with each other. Therefore, to understand the contextual information and efficiently provide the requested context, all the devices must agree on certain protocol, and these protocols are provided by a common approach to contextual modelling.

Different contextual information may have different requirements for modelling (Chen & Kotz 2000). For example to locate the user, the location could be in latitude and longitude or detailed information like street number, building number, and floor and room number. Thus the same information can be retrieved in different formats, and therefore, modelling is required to properly structure and understand the information. Becker & Dürr (2005) describe the properties and requirements for modelling *location* as contextual information. Held, Buchholz & Schill (2002) elaborates the requirements required for comprehensive context representation that includes structured, interchangeable, composable/decomposable, uniform, extensible, and standardized. To properly model the context, a fact based modelling language was developed by McFadden, Henricksen & Indulska (2004) known as the Context Modelling Language (CML).

Other approaches for contextual modelling are key value pairs as the data structure (Salber, Dey & Abowd 1999, Schilit, Adams & Want 1994, Voelker & Bershad 1994, Maaß 1997), markup scheme models, graphical models (McFadden, Henricksen & Indulska 2004), Petri nets (Murthy & Krishnamurthy 2005), logic based models, ontology based models (Strang & LinnhoffPopien 2004), relational representation, and situation abstraction (Henricksen & Indulska 2004) to represent generic contextual information. Various approaches to modelling are based on some object-modelling technique like UML.

2.7 Context Adaptation

In context adaptation, the contextual information is accessed by the context aware consumer devices to use it for different applications. Context aware application adapts in two ways. First, when a new context in available, the application adjust its characteristics according to the context to fulfil the user requirement. Second, the application can notify the concerned devices and applications regarding the availability of the updated context.

Adaptability in context aware systems is the essence. A change in the user's surrounding environment is inevitable but a quick, appropriate response make context aware applications highly adaptive. To better understand the concept of adeptness, we must classify some parameters before a context aware system is judged for its performance.

Why there is a change: This parameter focuses on the change. The change in the behaviour of a context aware application is always attained to address user's requirement. This can be due to either that the user's requirements have changed or the previous state and the behaviour of the context aware application do not seems appropriate according to the user's expectations. It should also be taken into account that the requirements can be functional or non-functional. Functional requirements capture the behaviour and response of a context aware system. The non-functional requirements deals with the demands like performance, time to respond, reliability, etc.

What does or does not change: This parameter asks what part of the context aware system has changed to force it to adapt its behaviour.

When the change occurred: This parameter is presented to capture the moment in the lifetime of a context aware application when the change in behaviour occurred. The change does not have to occur at the runtime. The change can be dynamic or static.

Who manages the change: This parameter points out the mechanism to respond to a change in requirement. It can be context aware system or the user itself. This involves the tasks of monitoring the system to collect and evaluate relevant data, and then decide the outcome of the process to take appropriate action.

Context adaptation usually involves techniques from the field of Artificial Intelligence. The techniques commonly used are rule-based inference (Zhang 2004), expert systems (Kwon, Yoo & Suh 2005) and machine learning (Dey, Hamid, Beckmann, Li & Hsu 2004). Generally, it has been observed that rule-based inference and expert systems are less capable to adapt to the context information than machine learning systems.

2.8 Requirements & Challenges

In the previous sections, we discussed in detail about the characteristics of ubiquitous and context aware computing. In this section we try to describe the requirements and challenges faced by such systems. Some of the requirements are discussed below:

2.8.1 Ubiquitous Computing Requirments

• **Mobility.** To fulfil the requests of the user, it is imperative that the location of the user must be known and this can be done through the mobile devices. Such devices always remain with the user, and therefore, if the user changes his position, the pervasive system will automatically know the location and can adjust its behaviour accordingly. For the user location, it is essential that the device must support mobility so that the user is not aware of the technology by its absence (Satyanarayanan 2001).

• **Scalability**. In ubiquitous computing the devices are mostly for specific purposes and can only be used on a small scale. To improve scalability, the devices must be compatible in different environment without making any notable sacrifices. With such devices, more tasks can be handled in limited resources.

• **Invisibility**. The applications must be capable of sensing the needs of the user and can provide for it without user interaction with the system. Without any interaction, the context aware applications should perform properly and adjust their behaviour quickly to match the user's needs.

• **Minimum interaction**. The context aware applications must have friendly interfaces and should be smart enough so that user should feel comfortable interacting with them.

• **Resource consumption** (Want & Pering 2005). For efficient pervasive system, it is essential that the devices and applications must be resource efficient. This means that the applications must adopt strategies to discover and communicate contextual data using minimal resources, and also provide an acceptable level of performance at the same time.

• **Security and Privacy**. Main security and privacy concerns are confidentiality, integrity, accounting, availability, authentication, and security policies (Stajano 2002). For safety purposes a well-defined security procedure is essential. This procedure must always be followed by ensuring all the necessary steps to make the data secured properly. Other security mechanism could be some cryptographic technique.

2.8.2 Context Aware Computing Requirments

There are some challenges faced by context aware computing which are discussed in detail by Salber, Dey & Abowd (1999). These challenges are:

• Sensors are not like common devices. They are especially developed for specific purposes, and therefore, they have certain uses and requirements. Examples of common sensors include

GPS, biosensors, RFID, etc. but such sensors are expensive to install. Thus, the ease in availability and low cost of sensors is a challenge for context aware computing to overcome.

• The data acquired from the sensors could be in low level format or in a different format then the one the application can process. It is vital to enhance the representation of the contextual information so that it can be usefully employed by context aware applications.

• Heterogeneous environment always cause hurdles in proper management and sharing of contextual information because in such environment different kinds of devices and applications are interacting with each other, and the data acquired from the sensors is usually different from the data format requirements of the applications. So, the issue of compatibility in heterogeneous environment need to be addressed.

• Context aware computing system must be highly dynamic and sensitive. This means that it should have the ability to detect and adjust to the changing environment in a timely fashion to satisfy the user needs without any explicit intervention from the user.

• For context aware application, it is imperative to completely understand the user intent because only then it will be able to adapt to the user requirements.

The above challenges need to be resolved because, in future, the importance of pervasive systems will increase rapidly as sensors become cheaper to produce and deploy. To fulfil the requirements of the users successfully, the above challenges have to be dealt with first.

2.9 Context Dependency

In the real world, there are two types of relationships between people or devices; one type of relationship is obvious like the ownership of the devices, and then there is other one that is less obvious like dependency in terms of acquiring service from a certain entity. Therefore, the context information can be derived from more than one device or persons because any activity can be a part of other activities and it is essential to know all the details to obtain

context information. Thus, it is right to say that each activity of information is derived from some other activity and so on.

To further understand context dependency, consider the following scenario:

In the events section of her LinkedIn webpage, Alice finds out that one of her old friends, Bob, from school years will be in town next day to attend a conference. As she checks his LinkedIn profile, she thinks up of sending him an invite for a lunch. But she is unable to send him an email as her current account doesn't allow her such a luxury. A context-aware API running in Bob's LinkedIn profile suggests that his profile on another social networking site, Orkut, can be viewed. She likes the suggestion and visits his profile. Looking for his contact email, she is disappointed to find out that Bob has not made his email available to visitors who are not already added to his friend's list. So is the case with his scrapbook. If she just try to add him to her friend's list, it will be too late as who knows when Bob will be online before he lefts the town. As she was thinking "what to do now?" the context-aware API on Bob's LinkedIn profile shows that, surprisingly, Bob has just come online and is available on his mobile phone as he is using one of the mobile messenger clients. Alice sends an instant message on Bob's messenger through the in-browser chat window. Bob receives the message, and instantly replies back. Alice invites Bob for the lunch after the conference ends next day, but Bob is unsure whether he will be able to attend the lunch with her because of the packed schedule, instead he proposes to get together for afternoon tea to which she gladly agrees.

In the above scenario, the context aware API is dependent upon various sources to obtain availability of contact information of Bob. The context sources are his linkedin profile, his orkut account and its scrapbook, and the mobile messenger client. So the potential context dependency can exist among the context aware API and the three context sources. In the event of the failure of context acquisition from the first two context sources, the third source was able to provide the required context which indicates a valid context dependency.

Due to highly dynamic nature of mobile agents in an ad hoc environment, too much context dependency can severely affect the performance of a ubiquitous system. With the increasing number of participants in an ad-hoc network, context dependency induced issues need to be handled efficiently. If the problem is not properly addressed, interdependence on low quality context will increase, leading to poor decision making by context aware applications. There can be two basic kinds of context dependencies, as shown in the Fig 4. In the first dependency, a context of type A is being utilized by various applications to generate context of types B and C. Highlighting the direction of the dependency, B as dependent and A as the antecedent where B depends upon A, and a change in A implies a potential change in B. This definition is very similar to the one given by Booch (1998). The bidirectional dependency, as pointed out by Briand (1998), can be among more than two entities, N, where N > 2. In the second, there is interdependence of context between the participating applications. This cyclic context dependency can be a serious threat to smooth execution of a context aware system if it gets introduced unintentionally.

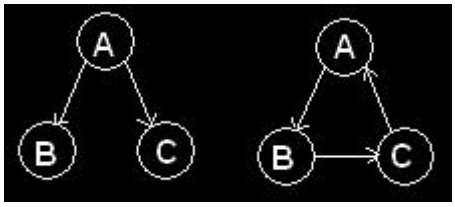

Figure 4 Types of Context Dependency

Most simple scenario to depict context dependency is the availability of the user on his mobile phone when he is in a meeting. In such a situation, user can be contacted on his mobile phone through text message only to avoid distraction of voice calls. The decision to drop voice calls is based on two contexts; user location, current time and calendar entry of the meeting in his online Google Calendar. The mobile application is dependent upon these three contextual data to make correct decision. If any one of the context is not acquired entirely or even acquired late, the usefulness of such mobile application is not valid anymore.

In cyclic dependency there is mutual interdependence of context between the participants. A simple example of cyclic dependency can be bandwidth adjustment for a certain multimedia data transfer over a wireless channel. The mobile application monitors the available bandwidth and then adjusts the quality of the video in transfer accordingly to achieve smooth streaming for the end user. The base station detects the video streaming requirements and adjusts the bandwidth, which in turn set in motion the whole process again. Such a cyclic context dependency can be serious threat to an effective execution of a context aware application. Any service that contacts another for context is acting as a client or more specifically, a context consumer, and the service that provides the contextual data is acting as a server or context producers in context aware application terminology. The developers must be careful to avoid cyclic context dependency among participating context consumers and producers. The chain of context requests introduced due to cyclic context dependency can continue indefinitely until all the involved applications exhaust their resources. The probability of circularity is particularly high when context services are designed independently, because no single developer can predict all possible interaction among services.

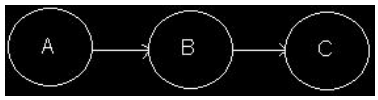

Figure 5 Transitive Dependency

In addition to two direct dependencies, Fig. 5 shows a transitive dependency between A and C. A transitive dependency exists when two entities are dependent through one or more than one intermediary nodes. Most transitive dependencies are generally not as significant as direct dependencies. If a system is designed properly, it is likely that a majority of changes will not have significant effect through transitive dependencies. In fact, UML (Unified Modelling Language) does not even acknowledge transitive dependencies as actual

dependencies in software systems as pointed out by Jackson (2004); and stated by Fowler (2001), if the dependency between two items is not direct, the items are presumed to be mutually independent. But we still have to analyze the effects of transitive dependencies in context aware systems more thoroughly.

Context dependency that change between entities quite frequently can affect the performance of a pervasive system as the search for an appropriate context source has to be initiated at each time when there is a change in the requirements of the dependent entity. If, let's assume, that entity X is dependent on entities Y and Z, but in the next iteration or time slot, X is dependent on entity A and A is in turn dependent on Y then we can assume that the context dependencies are unstable. When a context dependency change around frequently in such a manner, it becomes difficult for the pervasive system to maintain an overall stable state.

Our proposed approach in Chapter 4 tries to decompose complex and bidirectional context dependencies into a binary, unidirectional relationships as it make analysis easier where you have complex context dependencies broken down into various, simple unidirectional context dependencies. Since, a bidirectional context dependency can be represented with two unidirectional context dependencies between the same entities; we can assume the $2^{\binom{N}{2}}$ number of unidirectional context dependencies.

2.9.1 Context Dependency Attributes

An attempt at classifying dependencies, in general, has been made by Keller (2000) who listed six parameters to describe any given dependency. However, we consider that most of these attributes describe the entities themselves rather than elaborate on the relationship of dependence among the entities.

To better understand context dependency, we have tried to list some important attributes below. The attributes that we consider to better explain a context dependency are:

Impact: How can the entity's functionality get affected by its failure to obtain context from its context source? Few of all the possible values that can be assigned to this 'impact' attribute are: none, unreliable, performance degradation, non-dependable, total failure.

Need: This attributes highlights what kind of context is required by the dependent from its antecedent. This context dependency attribute will help to understand the context dependency relation between two entities in term of the context exchange. Few of the possible values can be the common context types like temperature, location, time, calendar entry, or an event.

Strength: The frequency of context provision from antecedent to dependent entity explains the strength attribute of the context dependency. Whether the context is provided every time the query is sent to the antecedent at a regular interval or the context is published to the dependent entity as an update, all these falls under the strength attribute of context dependency.

Stability: Stability is a measure of the "continuity of the dependency's vulnerability to compromise or failure (sensitivity) over time. One way of looking at stability is to ask the question: "When is the dependency fragile?" Possible values for this attribute are Extremely Stable, Infrequent, Periodic, Certain Defined Times only, etc." [Cox, 2001]

To our understanding, the above attributes are very much pertinent and applicable to context dependencies in pervasive systems.

2.10 Dependency Management

Lot of studies are conducted and published in the research domain of information integration in a multiple heterogeneous sources. The main purpose of integration is to provide user a unified view of the environment. Integration combines different data stored in systems and create a single outlook for all as presented by Batini (1986), Lenzerini (2002) and Rahm (2001). In integration there are two main categories of issues: semantic and architectural. Some of the issues are solving structural as well as semantic data conflicts, integrated vs. federated and multi-database architectures, and integration methodologies at DB design time

vs. on-line gathering and filtering of information in Batini (1992), Bergamaschi (2001), Chawathe (1994) and Ozsu (1991).

Context dependence of information and mobility can be described with different perspectives:

- Its use in the design of information sources suited to mobile environments
- Advantages offered in accessing information, particularly in the user's interest.

Information management methods using Distributed Hash Tables (DHTs) are proposed in numerous studies such as Ratnasamy(2001), Stoica(2001) and Zhao(2002). The DHTs can distribute the processing load by an efficient query routing. Techniques that construct a P2P network based on the topics of contents or communication times between peers have also been proposed by Bawa(2003) and Kwon(2002).

An important step in understanding effects of context dependency is to determine and model these context dependencies, and develop mechanisms to reduce them or at least provide alternative context sources that are closer to the dependent entities. Specifically, how it can be determined whether and how a change in context at its context source affects a dependent entity? What are the context types that form the basis of context dependency between the antecedent and dependent? And what nature of problems a pervasive system can face due to such context dependencies among entities? After monitoring the exchange or queries of context types, analysis of context dependencies can be performed dynamically to predict context dependency among different group of entities. The prediction can help the system to reserve the resources in advance, and to meet those future demands efficiently. In the next chapter, we present research works related to ubiquitous and context aware computing.

3. Literature Review

3.1 State of the Art

Today, the needs and demands of ubiquitous computing are growing rapidly and many users want to get a high performing efficient system that could prove to be useful. To fulfil this demand, researchers have speed up the development of the ubiquitous computing systems and are developing various prototypes to observe the performance in real life (Rogers, 2006). As new prototypes are being developed to achieve high performance and reliability, it shouldn't be neglected that today the customers are already using ubiquitous computing system but only in very narrow, specific scenarios, for example, *Octopus smart card* (Octopus, 2008), a debit RFID card that is used by about 17 million people in Hong Kong. It was developed in 1997 and because of its ease of use; it is now widely used for commercial and non-commercial purposes. Another advantage of such a smart card is that it is connectionless. In Hon Kong, around 95% of the adult population is using it, and many institutions and offices have adapted an interface specifically to utilize the Octopus smart card.

Another similar example is that of a Sony FeliCa (Sony, 2008). It contains a storage capacity of 64 KB compared to magnetic stripe card 125B. For near-field communication, its speed is 212 kbps in the range of 3-10 cm. With this speed a transaction will take 0.1s to complete. The data storage helps the providers to add new services because the storage capacity is in the form of operation system (directories, folders and files). If any provider wants to get access to the storage, unique access rights are given and only a specific directory or folder is allocated to the provider. No other provider can use that storage once it has been assigned to some other provider. If any commercial transaction is being made with this card, all the transaction data is

stored on the card and with the service provider. The data on the card is encrypted with two-way authentication and it has never been cracked and no data is lost yet.

The dream of ubiquitous computing is best described in Greenfield's Everyware (2006):

"A ubiquitous city or U-city is a city or region with ubiquitous information technology. All information systems are linked, and virtually everything is linked to an information system through technologies such as wireless networking and RFID tags. The concept has received most attention in South Korea, which is planning to build some 15 ubiquitous cities. The first U-City, Hwaseong-Dongtan U-City, has been partially completed and operated in 2007. It characterizes diverse U-Services that include U-Traffic, U-Parking, and U-Crime Prevention service. (...) A ubiquitous city is where all major information systems (residential, medical, business, governmental and the like) share data, and computers are built into the houses, streets and office buildings."

A Korean called Songdo City is known as U-city (O'Connell, 2005). The city was built from the ground up and uses the RFID technology to manage the most of the public services. In Songdo City, whenever a consumer throws a bottle in the recycle bin it is automatically credited with the RFID technology. Similarly, the pressure sensing floors for the aged people help to detect any fall and to send help quickly. Also, the cell phones that are used to record health problems can provide detail description of the user's health to the doctor. Furthermore, with a unique smart card, the user can easily enter into his home and or use it to access services at the subway and cinemas. Hence, till now, the Songdo City can be considered as the most favourable city for the use of ubiquitous computing and has the capacity to exceed expectations quickly in this technology.

From the above discussion, it is quite clear that in today's life ubiquitous systems are available but on small scales. The applications with main concept of ubiquitous computing are still missing from our lives. To proper implement all the aspects of ubiquitous computing, it is

necessary that there should be an appropriate design pattern. Furthermore, standard interfaces are required to interconnect devices with each other so that data sharing can be done smoothly without any hindrance. Meeting these challenges is not easy and will require time and resources to implement a real ubiquitous system.

Up till now, different examples have been stated to reflect the promising future of the ubiquitous system, but it will be better to analyse whether it is possible to achieve high standards of performance and efficiency. This can be accomplished by comparing the performance of the actual and registered ubiquitous systems.

It is really important that the hardware of the ubiquitous system should be as small as possible because to connect different devices together, hardware will be implanted in every device and if the size of the hardware is too big, the commercial use of the system will no longer be productive. Therefore, the size of the hardware should be around smaller scale and it is a tough task for the researchers. Furthermore, the system should consume low power. This means that all the interconnected devices should consume less power and can share more data with each other than they do now at the moment. Another challenge is to make the devices fast. The processors embedded into the devices should be fast enough to share the data, so that the devices should take less time. In essence, for ubiquitous system to be successful, small sized hardware, low power and fast processors should be of high importance.

For low power system, it is feasible to use wires because these are stable and take less power, but the main problem is that when a system starts to become complex, as more and more devices are interconnected with each other, then the wires will also start to grow and that make the system design below standard. The one solution to this problem is to use wireless communication. In this scenario, no wires will be used and all the devices will communicate wirelessly through radio transceivers. But in such situation, the radio transceivers consume more power. For example, in Telos wireless sensor node (Polastre, 2005), the active power

spent on the radio is 12 times more than the active power of microcontroller. Furthermore, if the processing speeds are taken into account between the radio transceiver and microcontroller (take again the Telos case, which sees a 250kbps radio transceiver versus a 16MHz microcontroller), the amount of energy consumed by the radio when receiving a single bit, is same as the energy consumed by the processor for 800 instruction cycles.

Based on various studies, the characteristics of the ubiquitous computing system are (Chalmers, 2006):

• *Fluidity*: long term structure evolution.

• *Partial autonomy*: the system will be self-adjustable. It means that the computing system will observe the environment and based on that will make the modification. More innovation and minimal developers' involvement will be needed. Furthermore the system components should guarantee long term reliability.

• *Trustworthiness*; all the devices are dependent on each other, and therefore no device or system should be allowed to affect the information provided by the sensors.

• *Scalability*; the computing system can be of varying size depending on the environment and need, for small projects small scale computing systems and vice versa.

Another important feature is the context that is directly related to the ubiquitous computing system. Usual context queries are where one is, who one is with, what one is doing and what resources are nearby. An important characteristic of ubiquitous computing system is its ability to adapt to the environment. This is known as context awareness. In context awareness the system *acquire* a measure of context and *adapt* to the context's current values by observing the entities currently involved in the environment. The measuring of context is accomplished through sensing of human activities and surroundings and based on these the system adapts itself. A common way to acquire context is service discovery: a process, in which entities provide the hardware or software resources required by an entity's application (e.g. the

nearest printer) are identified in the entity's surroundings. In ubiquitous computing, the service discovery protocols are designed in an ad-hoc fashion and follow logical topology. It means that the protocols should be a type of infrastructure like those built-in environments and also can run over the infrastructure like outdoor sites. This ad-hoc designing assist the protocols to replace static configuration by open systems, i.e. imposing few boundaries to system membership, instead allowing incoming entities a certain level of access.

When the system will start to take control of the situation in an environment, meaning self-configuration, then it is not feasible for developers to reconfigure the system, because the system has measured all the necessary factors in the environment and is now able to adapt according to the changing environment. When the system is in full control then it could also save the power utilization by updating the necessary software's, only using the power during the processing time, etc.

Besides these benefits one major concern that will affect the performance of ubiquitous system is security. Due to ad-hoc nature of system, the security threats can be higher. Scenarios presenting various types of security attacks from the Ubiquitous Computing Grand Challenge manifesto (Chalmers, 2006) are:

"A ubiquitous application may involve collaborations between ad hoc groups of entities. It may require migration or downloading of code, and may involve people moving and changing the system configuration. New encounters occur, and there are complex issues in knowing what entities to trust. Does a server trust an agent enough to allocate processing resource to it? Does a device trust a neighbour to send message packets for onward routing?

(The latter could be a 'denial of service attack,' aiming to deplete the device's battery.) Does a human using the UCS trust a host, a service, a device, or another human communicating and collaborating through the system? Based upon predefined trust, recommendations, risk evaluation and analysis of past interactions, an entity may derive new trust metrics and

authorization policies for what access it will permit to its resources, what services it should refrain from using, or what security mechanisms (...) to use."

It is also possible to mention the security risks with *theories* for ubiquitous computing. One theory, "is to extend existing models, such as process calculi, to accommodate space and mobility; promising candidates already exist, but difficulties of analysis still remain" (Chalmers, 2006). Another theory could be, 'awareness of context in a constantly changing context, one added challenge is to come up with models and analyses for the entities' context acquisition, adaptation and information flow.

With the help of these theories, the system's behaviour predication can be examined with more accuracy, system adaption can be analysed with minimum problems and a set of satisfying security policies can minimize the security risks. If all these can be achieved then it will form the scientific foundation for ubiquitous systems prototype design.

3.2 Dependency In Systems

Most of the research work carried out in context awareness involves either provision of frameworks for supporting abstract information in context aware systems, or modeling of context information for the relevant context queries. We review the context modeling efforts of these research directions in ubiquitous systems.

Most methodologies to service monitoring and analysis are concerned with the assessment of contracts between service provider and service consumer as evident by the works of Ameller (2008) and Flehmig (2006). Such approaches don't work if the services are composition of multiples services because composite services have vertical and horizontal dependencies that need to be taken under consideration. The methodology presented by Bodenstaff (2008) tries to monitor vertical dependencies in the system. The methodology is limited in its approach for a comprehensive analysis of dependencies among composed services because dependencies between two single services are not fully captured. The same

issue is with the COSMA approach of Ludwig (2008) that also monitors vertical dependencies among the services but lack an approach to handle horizontal dependencies.

A formal context modeling approach is proposed by Harter (1999). It is based on object modeling paradigm. The Entity-Relationship model is used as a basis for the language to construct a conceptual model of context. The resultant context information is then stored in relation database. The model presented by Gray (2001) is concerned with capturing meta-information from the context, and then describes features like quality, correctness, source, format, along with the transformation process it underwent for its current form from raw sensed data.

The context service presented by Ebling (2001) called OWL maintains and provides, on request, context information to its clients. Various context attributes like context history, scalability, quality, access rights, etc. are handled in OWL. These models lack formal basis to capture relevant context information like dependency.

Set theory is used by Stephen (2001) to define a context tuple of a certain size n, where n is the number of different context sources present in the device. The variable present in the tuple represent a value of certain context corresponding to a context source. The tuple contains an extra variable which represents the time when the tuple was created. Context information is schematically described by the set theory approach, but lacks any information about dependency relations.

In most of the research literature, dependency analysis is treated fundamentally as a static analysis problem to help better understand testing [De Lucia, 1996; Bates, 1993], debugging [Agrawal, 1993], and maintenance [Gallagher, 1991] of computer systems. Directed graph approach is proposed by Henricksen (2002) in which an entity is described by context information modeled in the form of a directed graph. The nodes are used to represent entity and attribute types, while the associations among them are arcs connecting the nodes. The

model is comparatively comprehensive as it can represent quality of context, and context dependency relations but lack accuracy in dependency representation.

A distributed algorithm, called Genetic Relation of Contexts, (GRC), analyses interdependence of context data in a decentralized environment in the work of Zimmer (2006). GRC tries to solve the problem of cyclic context dependencies with an approach of splitting and multiplication of context. Initial results are interesting, but the formal model for context dependencies is still lacking.

Regarding the research work on the issue of scalability of context aggregation and dissemination, Context Toolkit comes into mind by Dey (2000). It's a classic project cited in the state of the art that uses a distributed architecture for context distribution. Context Toolkit uses the concept of widget that wrap up a context producer, e.g. a sensor, and the sensed data can be acquired by querying the widget. Aggregators, working as a service, are made available to applications that can get the commonly asked contextual data from them.

By selection and chaining process, suitable components from a system repository can be used to construct a path from a data flow of contextual data from context sources to applications as presented by Hong (2001) and Kiciman (2000). Depending on the resource usage in the system, CANS, these components can be rearranged or even replaced by others if the system manager has installed them in the component repository Fu (2001). These approaches use pre-defined data flow paths among the applications and provide components to cater the needs of the applications.

Predicting dependencies in a context aware system is a challenge that still needs to be addressed. So far, there is not a single successful approach for predicting the dependency relation among various entities in context aware systems. Even in software engineering, a common approach is the assumption of dependencies among different software components but even such an approach is unrealistic [Eckhardt, 1985; Knight, 1986; Littlewood, 1989]

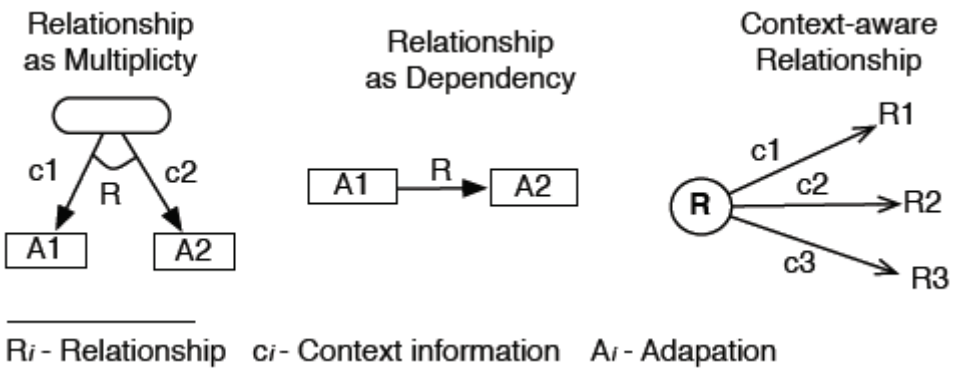

Figure 6 Context Dependency Relationships

Context dependency defines the existing and potential relationships between different entities. These dependencies can vary depending on the context of user. Rather than fixed dependencies, new dependencies can be selected and defined as the context changes. Desmet (2007) proposed an approach for context dependency modelling called Context Oriented Domain (CODA) diagram which is an extension to Feature Diagrams [Kang, 1990]. Figure 6 shows how various adaptations are selected when context C1 and C2 are present in the system. The context dependent adaptations are represented with rectangular boxes while context independent behaviour that spawns the adaptation is represented by boxes with rounded corners. When A1 and A2 adaptations are simultaneously applicable at a given time, the dependency shows the system semantics. The analysis results not in a singular, fixed dependency relationship R, but rather it is dependent on the context and can be R1, R2, …, RN respective to the C1, C2, …, CN context. CODA attempts at solving dependency issues at design level of the system by enhancing Feature Diagrams with context information and respective dependence relationships between them. CODA also supports the idea of mutual exclusion of context dependencies where two or more context adaptations cannot be valid at a given time. The major drawback of CODA is that since it's applicable at design level, the dependencies are supposed to be static in nature and CODA is not equipped to handle any change during system's active lifetime.

To assess the structure of a computer system, Stevens (1974) & Hutchens (1985) presented a common metric. The metrics pertains to a system in terms of data dependencies and functional relationships called bindings, and clustering methods were used for the evaluation of the extent of dependencies in a system. Selby (1991) studied data binding measures to understand system failures. The study correlated high dependency between system modules to increase number of defects.

The dependencies in a system that are not apparent are appropriately termed as "hidden dependencies" [Yu, 2001]. Another relevant concept is called "ripple effect" [Black, 2001]. Ripple effect is the propagation of a change through various dependencies in a system. In software engineering, a technique called program slicing is used to determine the software modules that can get influenced by a change in value in the system at a given point in time [Weiser, 1984]. Forward slice from a point shows which software modules that get influenced from a change at that particular point, and backward slice from the same point shows which software modules influence that particular module.

In [Goseva, 2001], there are three main classes of models for dependencies that can be found:

State-based Models: are described with the help of Markov Chains [Ross, 2000; Lyu, 1995]. Markov Chains are stochastic processes. At any given time, the current active dependency can be described by simple states while more complex states in the Markov Chains can be used to define all dependencies: active or passive. The Markov Chains are defined through transitional probabilities from one state to another state with rate λ:

$$P_{ij} = P(\text{going to state } j | \text{leaving state } i)$$

Markov Chains have exponentially distributed states which mean that arrival at a specific state only depends on the departure state and not the history of the previous states. To predict dependency in a complex system requires higher-order Markov Chains with more than one arrival rate. Some studies [Gokhale, 1998] pursue the path of discrete event simulation which

does provide an understanding of a complex system but less in comparison to analytical models.

Path-based Models: are used to determine the dependency in terms of possible execution path which are characterized by frequency of a path taken and its dependability of completing the execution path. Work in such models is carried out by Shooman (1976), Krishnamurthy (1997) and Yacoub (1999). These models apparently lack explicit handling of dependencies which was properly addressed by approaches of Hamlet (2001), Kuball (1999) and May (2002). Path based models are very close to classic reliability theory and this relation can very well be utilized to advance dependency analysis in complex systems.

Additive Models: attempt to predict as accurate as possible the failure rate of a system dependency by individually considering the failure rate of the involved components and entities rather than considering the structure of the software modules. In (Xie, 1995), the system failure rate λs(t) at a given time is obtained by summation of failure rate λit of individual entities comprising the system through

$$\lambda_S(t) = \sum_{i=1}^{n} \lambda_i(t)$$

The common observation regarding summation of individual failure rates of entities is that this results in a conservative prediction in terms of system's failure rate.

After studying various approaches, Gosova (2001) remarked that all such models assumed dependencies to be non-existent, and even those who took a realistic view, the problems that were addressed were narrow in solution and also the application of such results. However, there are certain studies that specifically addressed dependencies among software and system components. Calculating dependability of software components can be difficult even for simplest of cases. Littelwood (2000) considered assessing the dependence reliability of a software system with two components. The two components A and B were used to determine

dependency failure of the system given that PA = P(A Fails), PB = P(B Fails) and PAB = P(Both A and B Fail). PAB implicitly denoted the dependency of the system. The Bayesian approach was applied and when both PA and PB are assumed to be known, the "posterior" probability of the system failure due to dependency failure becomes higher than the "prior" probability. The reason being that if 1 – P(A) - P(B) + P(AB) gives the failure rate of the system when no component failed, and if this probability is large, it can also be due to P(AB) being large enough. Therefore, neither the previous approach of probability summations nor this approach is applicable to the analysis of dependencies and their failure in computer systems.

The idea of independent components developed and deployed in a system has been investigated by Knight (1986) whose results have shown that "assuming independence should be done with caution". The study showed that the failure rate of systems dependencies was much higher than was expected assuming independence nature of the involved entities. Further studies have elaborated on the same assumption [Eckhardt, 1985; Littlewood, 1989]. Eckhardt (1985) precisely defined the dependence and independence of software components with a score function, V(x), with x as an input,

$$V(x) = I\left(\begin{array}{l}\text{Randomly chosen component}\\ \text{fails on input } x\end{array}\right)$$

where

$$\{V_1(x) : x \in \Omega\}, \{V_2(x) : x \in \Omega\}, \ldots, \{V_n(x) : x \in \Omega\}$$

are random variables, independently chosen.

3.3 Context Modelling

To assist the growth of context aware applications by facilitating the processes of acquisition, management and dissemination of contextual data, an efficient and scalable context model is required.

Modelling of contextual information is highly significant in order to capture the following details for a pervasive system to work efficiently:

- relationship among different context types and the nature of such relationships
- to determine device capabilities and available resources
- for user profiling
- to determine application requirements

As contextual information is gathered, stored and used at various points in a pervasive system, it is essential that the representation model should be applicable through all the processes, and should be consistent.

3.4 Modeling Approaches

Here we present the most significant context modelling approaches. The classification of the modelling schemes has been adopted by analysing the basic approaches they followed. Of course, some of the context modelling schemes can be categorized into more than one category because of the multiple approaches they have adopted.

3.4.1 Key-Value Models

Key-Value model context attributes with specific values of contextual information. As one of the simplest model, key-value pairs the value of the context information e.g. room temperature is provided to an application as a variable. Key-value model is a common approach adopted by Schilit et al.(1994) and various other distributed frameworks (e.g. Capeus by Samumowitz (2001)). The service discovery manager uses a matching algorithm on the attributes given in a key-value structure and described by the services themselves e.g.

SLP, Jini, etc. by Strang (2003). The main advantage of key-value model is its simplicity and such a model is easy to maintain and update but the drawback can be the lack of support to enable context retrieval in an efficient manner.

3.4.2 Markup Scheme Models

Markup scheme models arrange contextual information in the form of hierarchical data structure that consists of tags to store attributes and their content. The tags can be recursively defined in terms of the content. Profiles are one of the common examples of markup scheme models that are mostly based on Standard Generic Markup Language (SGML), which has many offshoots, with XML as the most popular one. Most profiles are extension of the standards like User Agent Profile (UAProf) and the Composite Capabilities / Preferences Profile (CC/PP), that are based on XML and RDF. A profile is based on a two-level structure where each profile has a certain number of components, and each component has a certain number of attributes. The CC/PP specification doesn't define the vocabulary for the components and attributes which results in the vocabulary that isn't not rich enough.

The Comprehensive Structured Context Profiles (CSCP) by Held et al. (2002) is an example that doesn't have any pre-defined static hierarchy unlike the CC/PP because it has the ability to present profile information due the flexibility of RDF, and according to the position of attributes in the profile, the attributes are deduced contextually. This reduces the need to introduce attribute naming in the profile compared to CC/PP who requires such mechanism. Another drawback of CC/PP is the lack of any mechanism to change the default values. A flexible way to change the default values and merge them will help to produce more dynamic, complete profiles.

It is non-intuitive in CC/PP to have complex contextual information presented because of these constraints. This was also realised by an approach similar to the CSCP, called CC/PP Context Extension, by Indulska et al.(2003), which extended the basic vocabulary of UAProf

and CC/PP. The extension dealt with introducing contextual attributes like location and also certain types of dependencies and relations between the contextual data.

Apart from CC/PP approaches, there are various other markup schemes. *Context Configuration* of Capra et al.'s (2001) reflective middleware, the Centaurus Capability Markup Language (CCML) by Kagal (2001), ConteXtML by Ryan (1999) and the note-tags of the stick-e notes system by Brown (1997) are the few examples.

One important approach among these is Pervasive Profile Description Language (PPDL) by Chtcherbina (2003) that is based on XML and allows context dependencies when defined with relationship patterns. But the approach suffers because of the scalability issue as the language is not comprehensive enough to accommodate large number of contextual data. It also hasn't evolved as its design structure hasn't been made accessible to researchers from public domain.

3.4.3 Graphical Models

Unified Modeling Language (UML) is commonly used in software industry to model software application, especially the graphical component of UML, the UML Diagrams.

An extension to Object-Role Modeling (ORM) of Halpin (2001) is provided by Henricksen et al. (2003). The 'fact' is the basic modelling notion in ORM, and the modelling in any domain with the help of ORM requires the process of indentifying appropriate types of the facts. The extension provided by Henricksen (2003) to ORM allows facts to be categorized based on their types, along with their source. These facts can be categorized as static or dynamic, determined by the nature of the entities from which they are obtained from. If the fact types are categorized based on the source entities being dynamic, then the sources can be further categorized into derived or sensed types. The time stamp is also incorporated in the model to maintain the history of the facts. An interesting extension to ORM is the fact dependencies, trying to represent relationships between various facts, trying to map the effect

of change in one fact on another fact. The relationship was named “dependsON” relation. It has also been used to model context in the work of Bauer (2003) in which an air traffic management system’s contextual features are modelled with UML.

3.4.4 Object Oriented Models

The advantages of object oriented approach, reuse and information hiding, are the motivation common to various context models. Object oriented approach provides the tools to deal with the problems that arise due to the dynamic nature of the contextual information in a pervasive system. The interfaces are provided to have access to the information. Cues, by Schmidt (2001) developed as the result of TEA project (1998) and Schmidt (1999), are one of the prominent example of object oriented modelling in context aware systems. The cues act as an interface: they take a value of a sensor at a time and provide a corresponding value to that input, thus hiding the details. A similar approach is adopted by the GUIDE project, presented by Cheverst (1999). Encapsulation is utilized to hide the details of contextual data aggregation and fusion while maintaining scalability.

Another object oriented approach is used to represent contextual information, proposed by Bouzy and Cazenave (1997), in computer Go as they justified that the advantages of reuse and inheritance in object oriented approach are that “to define the smallest number of properties, functions and rules [..] in order to simplify knowledge representation in very complex domains and systems”.

3.4.5 Logic Based Models

Logic based context models have a formal approach to representation of contextual information. Contextual information is represented as facts and rules in a logic based model. A fact can be derived from existing facts in the system by the application of the pre-defined rules.

One of the first logic based context modelling approaches has been researched and published as Formalizing Context in early 1993 by McCarthy and his group at Stanford University. McCarthy introduced contexts as abstract mathematical entities with properties useful in artificial intelligence. He prevented, emphatically, to give a definition of what context is. Instead he tried to give a formalization recipe which allows for simple axioms for common sense phenomena, e.g. axioms for static blocks worlds situations, to be lifted to context involving fewer assumptions, e.g. contexts in which situations change. Thus lifting rules, which relate the truth in one context to the truth in another context, are an important part of the model itself. The basic relation in this approach is ist(c, p), which asserts that the it proposition p is true in the context c. This allows for formulas such as c0: ist(contextof("Sherlock Holmes stories"), "Holmes is a detective"), where c0 is considered to be an outer context. McCarthy's model already supports the concept of inheritance.

Another early representative of this kind of approach is the *Extended Situation Theory* by Akman and Surav (1997). As the name implies, it extends the *Situation Theory* which has been proposed by Barwise and Perry (1983). Barwise and Perry tried to cover model-theoretic semantics of natural language in a formal logic system. Akman and Surav extended this system to model the context with situation types which are ordinary situations, and thus, first-class objects of situation theory. The variety of different contexts is addressed in form of rules and presuppositions related to a particular point of view. They represent the facts related to a particular context with parameter-free expressions supported by the situation type which corresponds to the context.

A similar approach is the *Sensed Context Model* proposed by Gray and Salber (2001). They use first-order predicate logic as a formal representation of contextual propositions and relations.

Ranganathan et al., (2002) proposed a context model named called ConChat, based on Boolean algebra first-order predicate calculus. Conchat covers the various types of available contextual information and also supports operations like conjunction and disjunction of contextual data. First-order expressions can be created involving context, allowing to write different rules, and evaluate queries.

3.4.6 Ontology Based Models

Ontologies are based on the idea of specifying concepts and the relationships among them, mentioned by Grubber (1993) and Uschold (1996). They are used to describe information along with the relevant relations. One of the premiere approaches to model context with the help of ontlogies was proposed by Otzturk (1997). Their work was related to the analysis of psychological studies to determine differences between recognition and recall in different contextual settings. The study concluded the need of combining and normalizing the knowledge, belonging to diverse domains.

Aspect-Scale-Context Information (ASC) model by Strang (2003) is another approach based on ontology. It uses ontologies to describe core concepts along with the sub concepts and related facts as presented by Bruijn (2003). The knowledge is put through an evaluation process by ontology reasoners. The model was implemented through selective application of ontology languages which later formed the basis of Context Ontology Language (CoOL) from Strang (2003a, 2003b). Apart from being used in service interoperability in the web domain, it is also used to enable context awareness in different applications in a distributed service framework.

The approach called CONON by Wang (2004) is based on ontology capabilities of knowledge sharing and its reuse. It concentrates on context classification, representation and its reasoning with the help of inference engines. The strength of such models is based on structure level as observed by Strang (2004).

Another approach called CoBrA by Chen (2003) is based on ontologies. A set of ontologies concepts are provided by this approach to identify people, places and various other different objects according to their context. The approach provides runtime support for context aware applications as it uses a broker-based agent architecture.

3.5 Context Provisioning Architectures

The interconnection of everything in the society is mainly dependent on the availability of the context information. Valuable context information can enhance the reliability and performance of context aware systems. But to achieve such performance it is important to retrieve correct context information. The main question is how to retrieve such information?

Today large percentage of the researches conducted for provision of sensor information is mainly focused on middle-ware solutions. By using middleware solutions, the researches come across different challenges that are summarized by (Hadim, 2006). Some of the challenges include scalability, security and privacy, openness and ease of use. These challenges are still not being overcome by researchers. Researches like (Levis, 2002) were considered to be non-flexible with regards to heterogeneity. For heterogeneity, different approaches are developed like Mires, that is for subscribe interface to provide sensor information in direct support of context.

Some early researches like MobiLife Project (Floreen, 2005) focused on decoupling context information systems and wireless sensor networks by using abstraction for the sensors so that applications have single interface.

One of the approaches such as SenseWeb project (Kansal, 2007) is focused on shared sensing. This approach uses centralized server clusters for provisioning of data storage through web services and public services. With its use, the mashups deriving application sensor data from GeoDB can be possible. GeoDB is a centralized SQL based indexing database. But SenseWeb centralization means that it is scalable, and this characteristic raises issue whether SenseWeb

is capable enough that it can handle the demand required for large scale provisioning systems. This scalability also raises issues in database implementations. The issues like query response times with increase in data storage, have to be considered. These types of challenges have direct effect on the performance of a system (Bonnet, 2001) and it becomes difficult to solve the sensor data provisioning and querying problem through database solutions.

MobiScope (Abdelzahar, 2007) is an approach used for merging sensor information from multiple sources to support dependent applications. But to make this happen, some improvements were needed in the previous researches like CarTel project described by Hull (2006). CarTel project hasn't shown high performance with the sensor information as in real time, it performance only improved with opportunistic delivery of information between sensor sources and its centralized repositories. CarTel networks supports the applications based on sensor information but these applications couldn't be delivered based on present and changing information. On the other hand, Mobiscopes are connected to the internet and remains updated with the current sensor information. The information is access through the centralized repositories because the sources were distributed and heterogeneous. This distribution and heterogeneity also raises concern of scalability for the real time services and systems.

3.6 Distributed Approaches

For real-time access to the sensor information it is important that there should be a scalable sharing on the Internet of Things. For this, numerous researches have been conducted to support systems and technologies for scalable sharing and also the context aware applications should be supported to handle different sensor information models. With the evolution of Internet of Things the demand rise exponentially, thus to fulfil those demands dynamic yet robust context-centric architectures must be developed by integrating user-services relationships (Sundmaeker, 2010).

The increase in demand will surely require services to be organized in such manner that support reasoning and knowledge gathering. To achieve this, context information must be organized into different models. One such solution to organize the context information is proposed in Mobilife by Klemettinen (2007). But this solution is completely dependent on the DNS as web portals on the internet are consequently used. In this solution, DNS is used for locating these web portals and applications. The use of Mobilife also raises different issues like DNS availability due to DoS attacks and configuration errors. These issues are a big drawback for Mobilife and provide a valid reason to focus research towards Distributed Hash Table (DHT) alternatives (Pappas, 2006).

Solutions that are based on database distributions couldn't guarantee high performance because in order to maintain database among different wide area networks a reliable communication must be present all the time and in real time this isn't the case. Therefore, in heterogeneous mobile scenarios the solution based on distribution of database are not effective (Barbara, 1999). Another reason for low performance of such solutions is that during the support of real-time data manipulation the relational database are highly inefficient and because of that the sensor information can't be retrieved properly in time.

The SOFIA project's (Toninelli, 2009) main purpose is to create such spaces that can be used by the context aware systems. This is done by middleware solution based on RDF ontologies. The SOFIA project's performance can also be improved by filtering the context based information, through which the system can deliver the required services based on the current context information. All this is done through the brokers that are used to break the context information from ontologies and distribute this information to the end users. This project uses relational databases for ontology persistence. This means that in real time scenarios that performance will be decreased due to rational databases.

For the mobiles services, a project called SOCAM middleware (Gu, 2005) was developed. It provides different solutions for creating context-aware mobile services. This is done through the information that is organized into the models. But it is centralized and dependent on web service portals on the Internet and thus vulnerable to DNS and scalability related issues. Another project is Hydrogen Project (Hofer, 2003), implements multi-layered architecture between information collection, management and usage, and uses middle-ware based on an object oriented ontology. The advantage of this project is that is utilizes distributive approach, this means that different devices can exchange information but only at a closed range. The disadvantage is that only local location information can be shared and it doesn't provide any functionality for the distribution in regards to the provisioning of sensor information. Furthermore, the problems faced by mobile environment also place restrictions on storage and processing time of context information.

The approaches that are discussed above are not entirely perfect and may contain few drawbacks especially in the real-time environment. For a model to be successful, it must be expressive, scalable and dynamic and must not be dependent on the availability of web service portals over the internet.

3.7 Acquiring Context

For interconnected devices it is important that the all the devices must know the location of other devices so that proper communication route can be establish to retrieve the information. Proximity-based discovery is important in regards to discovering nearby sensors for context aware information (Hong, 2001). Since, context aware systems are not usually installed in static environment, therefore it is not necessary that all the sensors are connected in advance and the information can be retrieved by knowing the location of nearby sensors. Thus, proximity-based discovery has important part to play in real time context aware systems.

Hong further argued that with improved wireless communication the sensors can be more accessible through ad hoc manners.

Previous researches tried proximity estimation in seamless connectivity by utilizing sensors, actuators etc. AmbieSense (Lech, 2005) is one of such researches that use embedded context tags for proximity, seamless connectivity. These tags are embedded in public and private spaces, lounges, restaurants, etc. By embedded tags in such places, the user can interact with the tag to derive proximity to an artefact. This interaction helps to retrieve valuable context information for spaces that are not covered by GPS due to different factors. But the installation and location estimation of physical tag servers requires an additional layer of complexity that function between the user and the required service.

In order to use proximity to hear nearby sensors, it is important that there should be a strong ad hoc communication. Different projects are based on ad hoc communication that includes Smart Its project by Holmquist (2001) and Bardram (2003). In these projects the devices are embedded with sensors so that they should be located within context proximity. But these projects also require hardware implementation, and in addition these projects are unable to directly utilize the context information from other sources which give rise to scalability issue.

Ad hoc communication is an important feature, but to retrieve the context properly from the sources is also equally important and remains an integral part of certain projects like Senseweb (Kansal, 2007) and SENSEI (Presser, 2009). SENSEI project shows low performance in dynamic environment; this means that the discovery of nearby sensor requires explicit modelling of the user-sensor relationships. As these projects work well in static environment, this again raises the problem of scalability that is important to deal with in real time environment.

Till now proximity in regards to location is being discussed, but Schmidt (1999) argued that besides location, proximity should also be used as some other dimension. The context aware

applications shouldn't only be based on location but also examine the expressions of the context. Proximity as location is already in use by mobile applications and services therefore, there is a need to find new ways to express context proximity. The new dimensions of context proximity can be explored through modelling of context information by involving all possible dimensions. These could also include similarities among the devices.

In the context aware environment, if a person is moving his mobile device will continuously try to define the current location context and will try to forward it in response to queries from various services. During the move, the person will be exposed to many information points that will be used to inform about the current context information. All the information required by various applications is provided by the context aware system by analysing the current context of the person. In order to get the required information, it is important to discover information points. There are some researches like Holmquist (2001) and Bardram (2003) that provide a solution to the issue of discovering information points. In such solution, the devices are connected to the points around the city and also in contact with the person's mobile. But in order to implement such solution, additional installation of hardware is required. These solutions have certain advantages like overcoming the problem of GPS dead-spots, but for this user has to position him within spatial proximity of the device to initiate proximity sensing to the architecture. By positioning in the proximity a connection between the user and the devices will be established and then the access to the information points can be provided. When the number of users starts to increase, then it would become very difficult to synchronize all the users at the same time with this architecture. Thus, to create an efficient system it is important that the context can be easily retrieved in the heterogeneous environment and the interaction between the user and the architecture must be kept minimal. As stated above, the user has to be within the proximity of the sensors for information and this could be achieved by different location techniques, the most popular one GPS. But different

location techniques have different limitations and that should be considered for proximity. Schmidt (1999) and Salber (1999) offers context with location but there are other concepts through which the person can be connected to the system. One such concept is proposed by Presser (2009) which allows the person to connect to the system by static determination of the information points connected to "presentity". But the drawback of this proposal is that when there is a large gathering in the city, like a concert, the user will have no access to the ad hoc resources and it will only rely on the infrastructure, thus there is no option of modification or flexibility in this proposal. This means that (Presser, 2009) doesn't support dynamic environment in real time.

Schmohl (2009) proposed an approach that derives context proximity over multiple dimensions as a single value. Each dimension is mapped onto geographical map and the location coordinates are calculated by the Euclidean distance. Schmohl assume that all the dimensions are common on all presentities, and thus value can be derived accurately. Also in (Schmohl, 2009) all the calculations are done through Euclidean distance and the scales of the dimensions are ignored. In real time it becomes difficult to calculate distance that is not comparable to the entities outside of presentities, which have the same attributes on the same scales. Therefore, it is important to develop methodologies that are capable enough to establish localized information points that could answer various queries. The solutions should identify information resources within close proximity to a user's context and thus can provide context information application and services in the real time.

Databases are vital for storing information but only storing is not enough; data should be arranged in such a manner that the important and relevant information can be located easily. As with the evolution of the Internet of Things, the importance of databases is also rising, because, a lot of information will be stored in them, and proper ranking and finding the relevant information in time will become vital. In Internet of Things the information will not

be written in static documents rather the information could be modified after as soon as it is created. This means that when the information will be created the system will be capable enough to analyse it and at the same time can support the changes or share the experience with other devices or with a user based on some context. Some of the examples include sensors for temperature, humidity, lighting, internet connections and sensors for location, traffic and air quality.

To retrieve the relevant information it is important that the relevant sensors must be available and this is done through the availability of fixed information points. A user could be connected to SENSEI architecture (Presser, 2009) but he will be provided with the physical closest sensors in order to retrieve information like temperature value. Some approaches like (Kansal, 2007) can also provide the user with different sensors according to his context but these sensors will be centralized and thus their performance will be low in real time environment. This problem can be resolved by providing more sensors to the user but these sensors will be outside of the user domain because the sensors should be allocated based on the user context. For example, if the user is in the room the attached sensor should only be for temperature, lights, etc.

In context aware systems, multiple sensors are available to the applications so that they can acquire the relevant context information from these sensors. With multiple sensors user will be able to select the most suitable sensor for many applications like weather, temperature, location, etc. This sensor selection could be done manually or automatically. This could be based on the sensor ranking that could depend on the reputation and availability of the sensors. Some sensors will have higher ranking to which the users are connected continuously and are stable and accurate. The sensors that are temporarily placed in places like in buildings for temperature measure will not be available to the user right away (Presser, 2009) because such sensors will be first added to the profiles and then they are made available to the user for

context information and this is feasible in static environment but not in dynamic or real time environment. Thus, it is important to identify the changing environment because with multiple sensors, users are connected to different sensors and in such scenario sensor ranking plays a vital role for the users to extract context information (Hong, 2001).

To retrieve the information in less time, it is necessary that the system can find the sensors that are available at that specific time. Some approaches like (Elahi, 2009) are used to locate those sensors for information retrieval. As these approaches are more focus on querying for information retrieval, therefore they don't have any valuable information about the sensor itself. This means that the other users in the system are discarded and only the relevant user is accessed for information retrieval. Ranking of information is always advantageous and therefore is utilized in both centralized solutions and distributed solutions (Zhu, 2005).

One example of centralized solution is Google index, which has less than 10 billion of the estimated 550 billion pages, on the relatively static Internet (Zhu, 2005). But with centralized approach it will become difficult to rank the sensors in an Internet of Things environment because it will also raise the issue of scalability. To solve the problems of ranking and scalability, one solution is to increase the resources, but this will result in a huge investment in the network and will also make the network more complex. If, in future, 50 billion devices have to be connected together and with this solution it will be difficult to imagine the cost it will take to implement new resources and how much complex the system will become. Furthermore, the data computation will become extremely time consuming and will take lot of resources. To overcome these problems, it is better to use distributive architectures rather than centralized architectures.

In the next chapter, we present two approaches that attempt to determine context dependencies in pervasive systems using time efficient methods.

4 Finding Context Dependencies

4.1 Context Dependencies

When the user has to interact with numerous applications and devices present in a pervasive environment, the attention of the user becomes a rare commodity. It is unreasonable for a context aware application to expect that a user may step in to manage and configure in such an environment where applications and their interaction with the environment are continuously changing. Context aware applications must be responsive enough to the contextual changes around the user and themselves to not depend on user interference. They can compensate for the lack of user interference by being more proactive and adjusting their behaviour automatically in response to a change in user's context.

As context information is obtained and derived from various diverse resources, such as sensors providing temperature, it is fundamentally essential to collect sensed data efficiently and make an effective sense of such raw data. So, collection and processing of sensed data along with the efficient dissemination of contextual information to various context aware applications is important for a successful context aware pervasive system.

In the section below, we present an example scenario where there is a practical requirement to determine context dependency to provide time saving to the user.

4.2 Application Scenario:

The intention of this example scenario is to highlight the advantage of using the knowledge of context dependencies related to space, or to be more precise the position of current context within an indoor location, and the preference of the user. The foremost aim is to design a solution to a supermarket shopping trip in which benefits can be obtained from the location of the products available in the market, the position of the customers and their preferences along with their shopping lists to achieve a hassle-free and efficient shopping experience for the customers.

Knowing the position of the customers and the location of the various products on sale in the supermarket is important to draw an itinerary for the customer to follow that can reduce the time spent shopping for different products during sale season, and also help avoid the long waiting time in the queues at the cash points. Such an application can be used to guide the customers through shortest route and considering the position of other customers can also guide the customer through paths that are not congested in various zones of the market.

There are innumerable occasions when the customer going for shopping in a large supermarket has suffered from delays, exhaustion and frustration because of not knowing beforehand where the various products are placed in the supermarket. Not only the long queues at the cash registers are the cause for annoyance for the customers but also the congestion in shopping alleys between the shelves when the customers are in hurry and want to finish their shopping quickly. It is especially difficult for customers affected by agoraphobia who cannot stand crowded places in supermarkets. Such customers can be helped by using a context aware application that can guide them to avoid such scenarios by combining the information related to other customers' location and the shopping list of the affected customer to provide him a more accurate and personalized information to have a nice shopping trip and save him time too in the process.

The dependency analysis of this scenario setting is useful for various reasons. When a customer goes for shopping, he prefers to buy frozen food at the end of his trip. He won't buy frozen food at the very beginning as he wants to preserve the freshness and taste of the frozen food when he gets back home. So, here we have preference of dependency of buying frozen food at other stuff. If we are already done shopping other items on the list apart from frozen food, then we can proceed to buy frozen food at the mart. Therefore, frozen food is dependent on the condition of first buying the non-frozen stuff present in the customer's shopping list available on his PDA or smart phone. Similarly, buying fresh meat from the butcher's shop

needs to be also considered to define the dependencies. May be there is a long queue of customers at the butcher's counter and the customer could return later when there aren't many people waiting to be served. Also, most customers prefer to but meat at the end of their shopping trip due to the reason of taking the meat fresh to their homes.

The context dependency analysis in this supermarket shopping scenario can be affected by the following parameters:

- The number of customers currently present in the market for shopping, N. It is important to know the number of people current number of people present in the market on whole.

- The location of the customers. It is required for correct analysis to know the location of the customers in the supermarket. The customers can be waiting at the cash register, buying food at the frozen food mart, in the queue at the butcher's shop, etc. So after aggregating the positions of the customers, this location parameter can be used to determine the context dependencies for the provision of an optimized shopping route in that particular supermarket.

- Customer' preferences and the contents of his shopping list on his PDA or smart phone affect the context dependency analysis. What, when and from where to buy an item are the questions that context dependency analysis can answer in order to have a hassle-free shopping experience.

- The shopping list of other customers present in the supermarket, so that the context dependencies can be updated in real-time to reflect an optimised route for each of the customer according to his/her own shopping list and preferences. Also, the knowledge of what customer has already bought can definitely help to provide better solution and

dependency analysis. It can also help to determine expected congestion at certain aisles, or the long queues at the cash points.

- The number of items left at the shelves in the supermarket. If the product is on the customer's shopping list but there aren't many items left on the shelves, then the item will be given priority, meaning it will become independent of other items on the list and need to be purchased before other items can be bought.

The high-level algorithm for such a solution is shown below:

```
Loop (at every X time interval)
        Get positions of N customers;
        Determine possible congestions (using customers positions and remaining items on the
                                                                                shopping lists);
        if (new customer enters)
                Show optimised route(after determining context dependency analysis);
end of Loop;
```

The solution will always be estimating the customer's position, and the number of desired items still left at the shelves with a frequency of iteration that can be adjusted depending on the congestion in the supermarket.

And as shown in the algorithm, if a new customer enters the supermarket, the algorithms will check for possible congestion, and if so, the optimised solution will be executed so that it shows the new customer a non-congested path after analysing the context dependencies.

The initial point of the customer's personalized itinerary will always be the spot where the shopping cart is initially placed, and the path will change according to the congestion and the items that are left at the shelves. When he gets his item, that instance in time will be his initial point again, and then a new path will be calculated to his depending on the current number of customers present in the supermarket. And successively the process goes on until he gets all his items, and then the terminus point will be one of the several cash points with least number

of people in the waiting queue. At this point, the best path will be computed to the cash point with less expected waiting time using topological sort algorithm.

When the customer has finished buying all that he wanted then there are two approaches in this regard:

- The customer can delete each item in his shopping list on his smart phone application whenever he puts the item in his shopping cart.
- RFID tags are getting common in our daily lives and it can be easily imagined that in the near future, nearly all of the products will have some kind of RFID tags on them. That wide spread network of RFID tags can be used in this scenario, where a RFID tag reader can be placed in each shopping cart and so the system can know at any given moment what the customer has placed in his shopping cart. This can help to keep the shopping list updated on the smart phone without user interaction.

The above two are suitable solutions but the first one seems to be more user interactive so there is a possibility that the customer can make some mistake in deleting an item from his list, and also it appears that the second approach is the more context aware.

The possible outcomes from this scenario based on context dependency analysis are:

- Minimize the time the customer spends in the supermarket buying groceries. This is possible since the system have all the required context information to develop the path for the user for the supermarket trip. If the system has all the necessary context information available in time, then the shopping time can be greatly reduced compared to the same scenario without context dependency analysis.
- Personalization of the shopping service to preferences of any customer. The solution has the required customer information and can take benefit of this, offering

personalized offers according to the customer's interests on the smart phone application.

- To enhance the supermarket experience of the customer by presenting various information about the products, and guide the customer during the buying period through the supermarket to buy the required items present in the shopping list.
- The context aware application can also help the supermarket administration with the logistic process regarding the stocking of various products.
- For the comfort of the customers, more or less cash registers can be used depending on the number of customers waiting to be served in the supermarket. Knowing the information regarding the current customers and the congestion level, it can be decided spontaneously whether more or less cash registers need to be opened for servicing the customers.
- If the arrival of the customer at the cash register can be determined in advance through context aware approach before arrival at the cash register, the payment process can be swift, and the waiting times in the queue shorter.
- Enhance the information of the supermarket about the customer's actions and preferred items. The supermarket can have all the information related to the shopping: time required for each of the shopping trip, customers personalized paths, etc. Supermarket can utilize such information to adjust their service for the customers and deliver a better and an efficient shopping experience.

The scenario starts when the supermarket is opened in the morning and the scenario ends for all customers when the supermarket in the evening. Apart from this duration, the scenario is inoperative.

The shopping carts are placed at the entrance of the supermarket and when a customer arrives at the supermarket, he can either carry a basket or get a shopping cart. If the customer prefers to get a shopping cart, he has to detach it from the adjacent shopping cart which triggers the logging in to the main system of the supermarket. If the customer is not already registered in the supermarket system, he can register with a new username by providing some basic information about himself. This process is needed in order to be able to use the context aware system. When the customer login by a username and a password, the applications knows where the shopping carts are located at any given time and who is using each one of them.

Once the customer has introduced his information, the system synchronizes the shopping list from the customer's smart phone or PDA. After that the system will determine the congestion in the supermarket based on the customer's location.

Finally the context aware system begins to calculate all information about all the items shopping list and personalized paths to follow. It means that the context aware system only tries to localize the shopping carts when they are detached from the other shopping carts and the customer has logged in with his username and password.

The context aware system utilizes the shopping list to estimate the next item and the optimal path to access it. This way, the context aware system takes into account all those items and theirs location to determine the path that avoids congestion. Thus, makes the shopping experience stress free for the customers.

4.3 Determining Context Dependencies

Since pervasive systems are characterized by the dynamic nature of the participating agents, the context dependency order defined at the very start of an application may not be valid anymore after the initialization of the system. We believe that determining the context dependencies correctly can help to better understand the flow of context information. It can

also help to identify the most utilized context and the most critical context source which has no substitute in the existing system state.

Given a ubiquitous system, we would like to find a consistent system state that is similar to its previous version, but with more precisely identified context dependencies. To achieve this goal, we need to determine the currently valid context dependencies first with the help of graph theory, and reject the ones that are not applicable anymore.

To apply graph theory approach, we need to define some fundamental terms first. A directed graph G = (V, E) consists of a set V = {1, 2, … , |V|} of nodes and a set $E \subseteq V \times V$ of edges. A pair $(v, w) \in E$ is called an edge from v to w. We set n = |V| and e = |E|. A path from v to w, where $v, w \in V$ is a sequence $v_0, v_1, \ldots, v_k$ of nodes such that $v_0 = v$, $v_k = w$ and $\{ v_i, v_{i+1}\} \in E$ for $0 \leq i < k$: k is the length of the path. There is always a path of length zero from v to v. A cycle is a path of length greater than zero from v to v. A path is simple if $v_i \neq v_j$ for $0 \leq i < j < k$. If the path is simple then the cycle is simple too. The in-degree of a node v is the number of edges ending in v, $\text{indeg}_G (v) = | \{w; (w, v) \in E\}|$. Similarly, the out-degree of v is the number of edges starting in v, $\text{outdeg}_G (v) = | \{w; (w, v) \in E\}|$, as shown by Melhorn (1984).

Two methods for storing a graph are customary. In adjacency matrix, a graph G = (V, E) is represented by a |V| x |V| Boolean matrix $A_G = (a_{ij})_{1 \leq i, j \leq n}$ with

$$a_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{if } (i, j) \notin E \end{cases}$$

The storage requirement of this representation is clearly $\Theta (n^2)$.

In adjacency lists, a graph G = (V, E) is represented by n linear lists. The i-th list contains all nodes j with $(i,j) \in E$. The headers of the n lists are stored in an array. The storage requirement of this representation is O(n+e). The lists are not necessarily in sorted order, as shown by Melhorn (1984).

Since $0 \leq e \leq n^2$, we conclude that the adjacency list representation is often much smaller than the adjacency matrix representation and never much larger. Since most graphs which come up in applications are sparse, i.e. $e \square n^2$, therefore adjacency lists are preferred. It is a fact that the choice of the representation can have a drastic influence on the time complexity of graph algorithms.

To determine context dependencies of a given system, we use the topological sort[3] algorithm to traverse the network. If there are any cyclic context dependencies present in the system, the topological sort can point them out and describe an ordered list of nodes illustrating the flow of context information.

Topological Sort Algorithm

1) *Compute the in-degrees of all vertices of the given.*
2) *Find a vertex U with indegree 0 and store it in an ordered list*
   If there is no such vertex then there is a cycle
   and the vertices cannot be ordered. Stop.
3) *Remove U and all its edges (U,V) from the graph.*
4) *Update the indegrees of the remaining vertices.*
5) *Repeat steps 2 through 4 while there are vertices to be processed.*

The complexity of topological sort is O(|E| + |V|),

where,

|V| is the total number of vertices, and

|E| is the total number of edges.

The operations needed to compute the in-degrees depend upon the representation that is used to store the directed graph. If an adjacency list is used then it is O(|E|). In case of matrix being employed to store the graph, then the complexity is $O(|V|^2)$. If the directed graph is complete then $O(|V|^2)$, since |E|=|V|.

[3] http://en.wikipedia.org/wiki/Topological_sort

In the remaining part of the chapter, we introduce another approach to determine nature of the context dependencies using constraint networks. To help better explain the approach, we present a simple scenario in the next section.

4.4 Example Scenario: Meeting Room

We present here a rather simple scenario that is used later in the chapter to highlight our the advantage of our approach of constraint satisfaction problem. The scenario is concerned with meeting room situation. During the duration of the meeting, it is considered highly inappropriate to have someone's mobile phone ringing. In our example scenario, the user has a mobile phone with him at a meeting in his office. The context aware application determines that the user is in the meeting after considering location (i.e. the meeting room), time, and calendar entry about the meeting. If all these context readings indicate that the user is busy in a meeting at the moment, the context aware application turns on the mute mode so that no one gets disturbed by any incoming call. In addition to mute mode, the application also turns on the mobile vibrator to alert the user of any incoming calls. The messages received through SMS are handled in the same way as the incoming calls. Figure 7 explains the scenario with the help of a decision tree. When the user leaves the meeting, the change in the context is sensed by the context aware application and the mute mode is turned off along with the mobile vibrator.

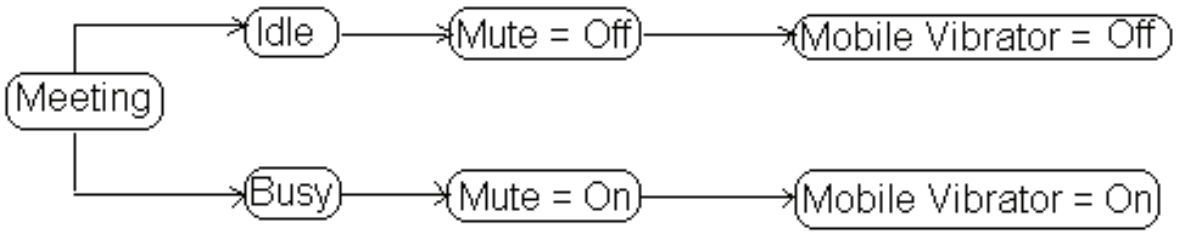

Figure 7 Meeting Scenario

4.5 Constraint Satisfaction Problem

In this section, we present the theoretical and mathematical introduction of our work. An introduction to constraint satisfaction problem (CSP) is presented, and then its formalization is specifically explained with the help of the above mentioned example scenario to demonstrate our algorithm. Later, the complexity of our designed algorithm to determine nature of context dependencies is derived.

4.5.1 Formal Introduction of CSPApproach

To capture design decisions and the assumptions related to those decisions, we have used CSP approach to serve as a model. The focal components that can be considered part of CSPs are a constraint network with a finite domain, and a dominance relation.

a) Constraint Network (CN)

A CN is a set of variables and constraints that are interrelated and define the valid values for the variables that satisfy the relevant constraints. A variable is used to represent each distinct piece of information in a system. The value of each of the variable is assigned from a given, finite domain. At any given instance of time, a subset of the domain of a variable comprises its set of possible, valid values. The system's view of the available choices for that variable is represented by such possible values. These possible values are always consistent with all the current constraints. Apart from possible values, each variable may have an assigned value at any given time which can be assigned by the user in the first place, or selected by the system from the set of possible values. The assigned value remains until it becomes inconsistent with the relevant constraints. Implicitly, a constraint defines the valid combinations of values for a given set of variables. A simple constraint defines valid permutations of values for a set of n variables. A 4-tuple is used to represent CN, (V, U, M, C). $V = \{v_1, v_2, ..., v_n\}$ is a set of n variables present in the system. U represents the universal set which contains all the possible valid values for the variables in the set V. The mapping from variables from the set V to valid

values from the set *U* is represented by *M*. Constraints upon the variables are represented by a finite set *C*.

Variables	**Domain Values**
Meeting	Busy, Idle
Mute	On, Off
Mobile Vibrator	On, Off

Constraints
Meeting = Busy => Mute = On
Meeting = Idle => Mute = Off
Mute = On => Mobile Vibrator = On
Mute = Off => Mobile Vibrator = Off

Tableau 2 Variables and Constraints

A CN that is presented in Table 2 shows the variables, valid values and constraints derived from the scenario. To specify the value of a variable when represented in a constraint, the term binding is used.

b) Dominance Relation(DR)

A DR relates dependency among variables to achieve design rules' annotation. From the presented scenario, we can extract three pairs of DR, i.e. (meeting, mute), (meeting, vibrator,) and (mute, vibrator) in which the two variables, mute and vibrator, are dependent on meeting, and as such, have no reverse effect on the meeting variable. The dependency among the variables can be formally defined as a relation $(u,v) \subseteq V \times V$, if *v* is dependent on *u*. Any change to *u* which results in CN's invalid state by negating any of the constraints, must force *v* to adopt a new value which is minimally different from its current one to restore the CN to a consistent, valid state by conforming to all the constraints. If *S* is the set of all solutions in a constraint network then a solution to a constraint network is a mapping of all variables to valid domain values $\forall v \in V \wedge s(v) \in M$ such that all the constraints are met accordingly.

c) System States

A finite number of states are derived from the CN and the DR. The states provide solutions to the CN, and the set aggregating such solutions is called *S*. If there is a solution s belonging to *S* in the design decision, with *v* the variable who can take different values, and *u* is one of the values in the solution that *v* can take, the transition function is represented as δ(*s*, *v*, *u*). Assigning value *u* to *v* must be handled in such a way that there is no violation in the initial valid state. If there is a violation when assigning a value *u* to *v*, then the values of the other variables in the initial state must also be altered to keep the system in a valid state. Updating the value of the secondary variables must follow the DR. If a variable *v'* must be changed to achieve a valid state after *v* has got a new value, it must be ensured that *v* dominates *v'*. If (*v', v*), then there is no need to update *v'* as it's the variable *v* that is the dependent on *v'*, and any change to *v* will be considered void if *v'* hasn't changed its value.

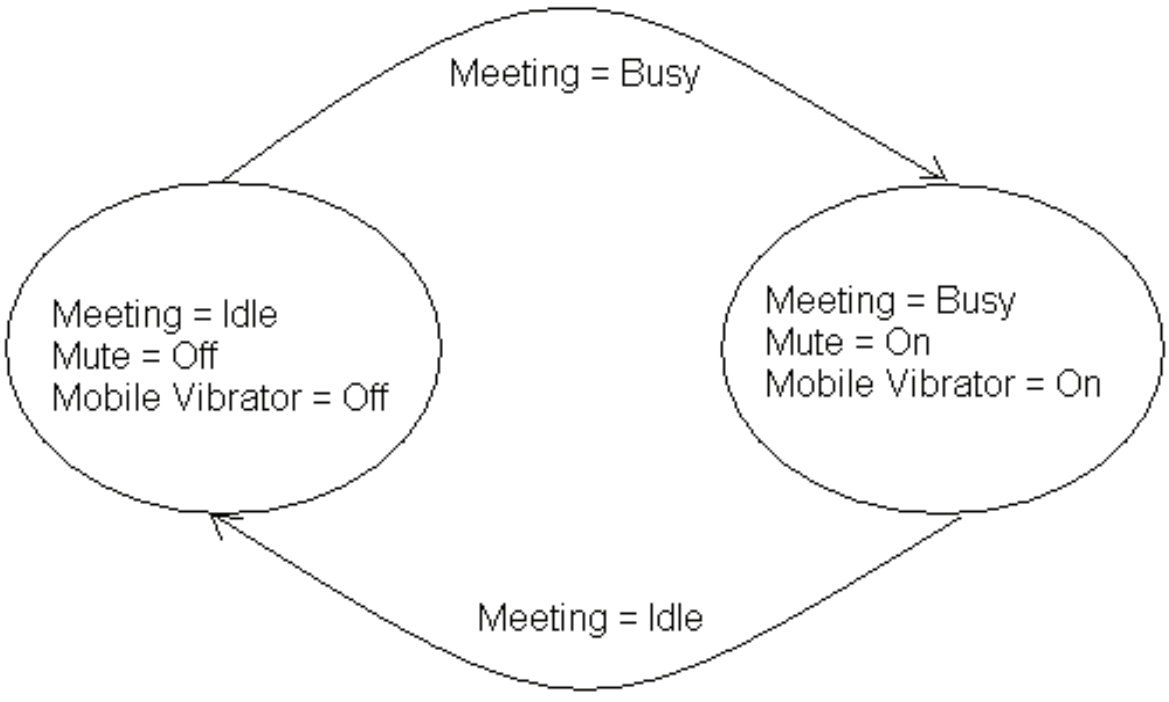

Figure 8 System States and Transitions

A system's states of meeting room scenario are shown in Figure 8. As the CN of this simple scenario has only two possible valid solutions that satisfy the constraints, there are only two valid states of the system. The system can be in either of the two states that we refer as busy and idle. In busy state, the variables 'meeting', 'mute', and 'mobile vibrator' have values of 'busy', 'on', and 'on', respectively. Alternatively, the state can be in idle state in which the variables 'meeting', 'mute', and 'mobile vibrator' have values of 'idle', 'off', and 'off', respectively. The valid transition between the two states is the resultant of minimal change. The variable 'meeting' is the dominant variable in the both of the DRs that we derived earlier. So, the transition from one state to another requires change in the variable 'meeting' only. Change in the value of variable 'meeting' requires the remaining variables to also adopt new values to satisfy the constraints, and hence, both variables, individually, are dependent on 'meeting'.

4.6 Determining Dependencies and NP-Completeness

According to Yokoo (1998), deriving dependency from a CN is NP-complete as CSP is NP-Complete problem. The constraint in CSP is true if the values assigned to all the system's variables satisfy the constraint. Therefore, finding a solution to CSP is equivalent to searching for an assignment of values to all variables such that all system's constraints are satisfied. In our approach, we convert the CSP into a dependency decision problem by using a given CN, and two given variables, *a* and *b*, to determine whether there exists a dependence (*a*,*b*) between the variables.

To proceed with our approach, a CN with two more variables $V' = V \cup \{a, b\}$, and two additional domain values for the added variables, $U' = U \cup \{true, false\}$ is considered. A corresponding constraint is also added to the CN, $a=true \Rightarrow b=true$. The variables in CN can have any values with the only restriction that the values should belong to the set of valid domain values. But here we have added an extra constrain on *a* and *b* to have only the Boolean

values of either *true* or *false*, and the constraints also ensures that *a* and *b* have the same value in all solutions, thus any change to *a* has a direct impact on the value of *b* to restore solution's validity. Since our approach is derived from CSP and the constraints of CN are subsets of the CSP constraint set, if there is no solution to the CSP case that can be obtained in polynomial time, then there is also no solution to the CN case, and therefore, it means determining dependency is non-polynomial, too. It is explicit that if there is a solution to the CSP case in the form of *s*, then there can be only two solutions to the constraint network case. One solution is when the value of both the variables, *a* and *b*, is *false*, the particular solution called s_0, and the other solution when the variables have the value of *true* with the solution labeled s_1. As there is only minimum difference between s_0 and s_1, any change to *a* causes *b* to update its domain value, it shows dependency of *b* on *a*, and if there is a solution to CSP case then *a* and *b* have dependence relation. But since CSP is NP-Complete according to Yokoo (1998), computing dependence relation is NP-Complete also.

4.7 Dependency Analysis and CSP

The CN can be successfully utilized to analyze dependencies on the conjecture that one entity is dependent on another one, and a change in an entity can have an effect on the dependent entity, which causes to alter the current state of the dependent entity. In such scenarios, usually, the prime interest is to determine the influence. This observation helps to abstract that a domain value of a variable in a constraint network can have only two possibilities: current state and changed state. The reduction to two states significantly facilitate in studying the modularity of pervasive systems, and the analysis of context dependencies thoroughly.

Essentially, in our approach, we consider a CN where each domain can provide two values to satisfy the variables, and every constraint involves two variables. Later, we will show that determining dependencies is not NP-Complete by presenting a polynomial time algorithm

which is based on CN with the CN having properties just mentioned above. The two domain values approach is the reason to the polynomial time solution to the computation of dependency. Any CSP can be altered to another form with only having binary constraints as mentioned by Bacchus (1998). Since the domain values in the CN can now have two states in our approach, it can be appropriately stated that one state is true and the other one is false, which makes the CN an instance of 2-SAT problems. Indeed, our approach is to treat this CN as a 2-SAT instance as 2-SAT can be solved in polynomial time as proven by Aspvall (1979). The easiest way to compute dependency relation is to search for all satisfying solutions and then get a construct and then obtain the remaining solutions in polynomial time as shown by Feder (1994). This seems rather tempting approach but soon, the state explosion problem surface up, and since in the worst case scenario, there are exponentially large number of solutions that makes this approach non workable for large domain space as predicted by Clarke (2001). The subsequent section shows that our algorithm is independent of the number of solutions, which help us to avoid the state explosion problem.

4.7.1 Dependency Analysis Algorithm

We continue with our earlier example scenario to explain our algorithm to determine context dependencies. The approach we adopt is based on a graph structure generally employ to solve conventional 2-CNF problems. This graph structure is called Implication Graph, and we utilize it to construct another graph structure, called Dependency Graph. The Dependency Graph is built to provide dependent variables' pairs.

d) Implication Graph

In the first part of our approach, we develop an implication graph, as done by Clarke (2001), which has two vertices for each of the variable in the constraint network as the domain values in our case can only have Boolean values. We build this graph so that we can model the constraints present in the constraint network. An implication graph is a non-symmetric directed

graph G (V, E) composed of vertex set V and directed edge set E. Each vertex in V represents the truth states (i.e. either true or false) of a Boolean literal, and each directed edge from vertex u to vertex v represents the implication "If the literal u is true then the literal v is also true"[4]. In Figure 9, the implication graph of the example scenario is shown where each constraint has two vertices for corresponding Boolean values.

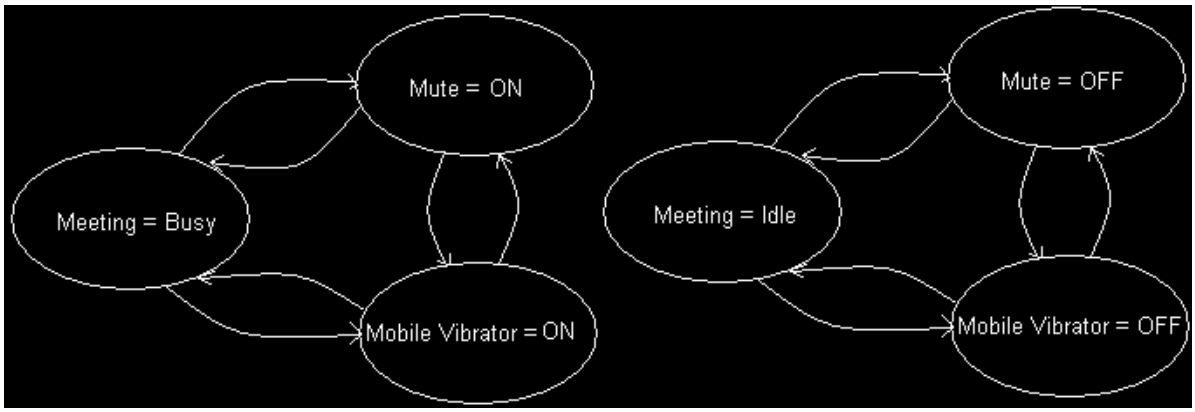

Figure 9 Implication Graph

e) Dependency Graph

In the second part of our approach, we develop a dependency graph using an implication graph. In the implication graph, we have two edges, in the first instance, between the two vertices, and two more edges for the other two corresponding vertices. We initiate the construction of the dependency graph by following the rule in which we keep the edges between the vertices if there is any dependency exist, otherwise if a vertices cannot influence the other vertices then the edges are removed between such two vertices in the implication graph. Formally, for two variables *a* and *b*, if (*a*, *b*) and (*b*, *a*) are not valid, we remove the edges (*a*, *b*), (*a'*, *b*), (*a*, *b'*) and (*a'*, *b'*) from the implication graph .We repeat the rule for each of the vertices, and at the end we have our dependency graph, as shown in Figure 10.

[4] http://en.wikipedia.org/wiki/Implication_graph

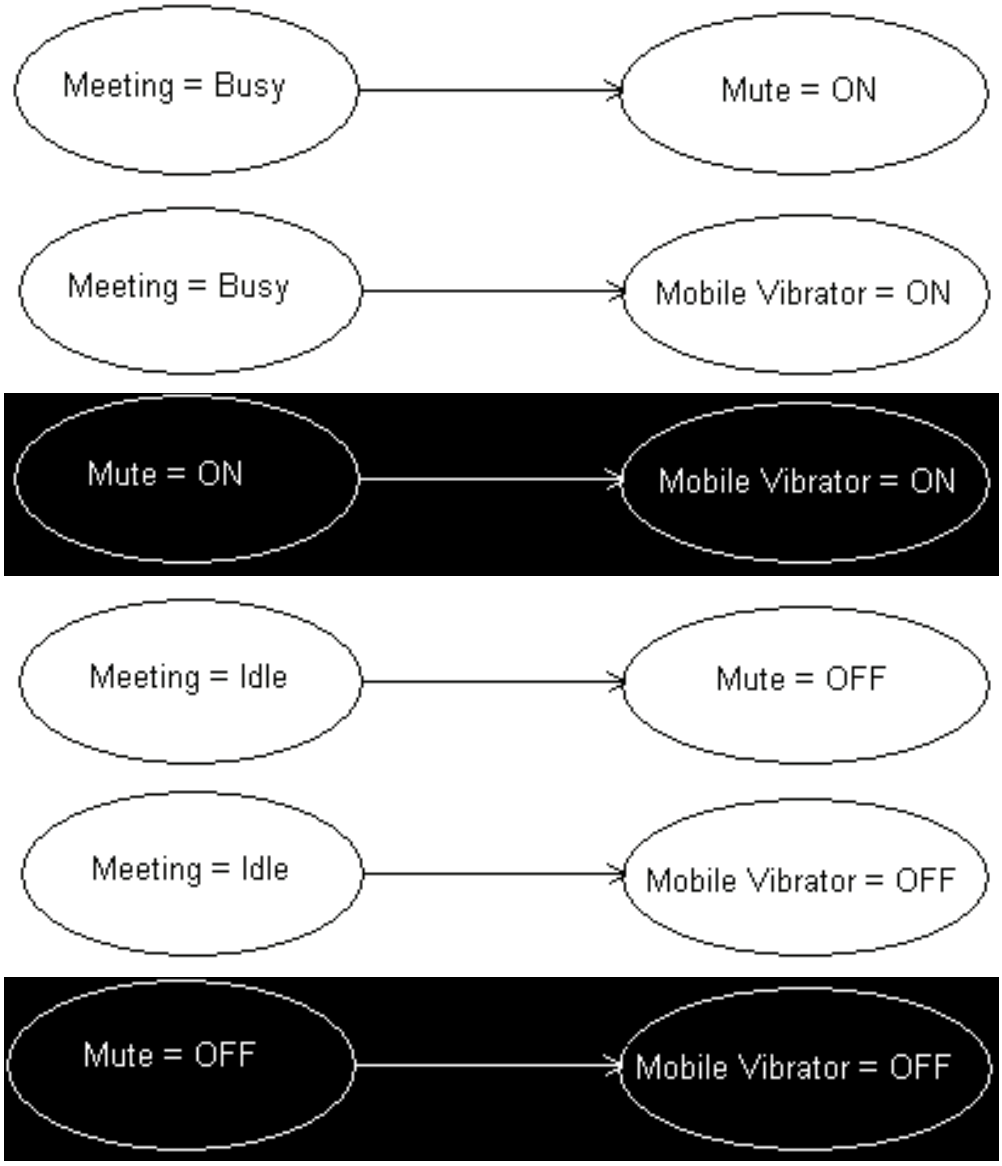

Figure 10 Dependency Graph

4.7.2 Complexity Analysis

For determining the complexity of our approach, we need to consider the running time complexity of the two main components. Let *n* be the number of variables involved in the CSP, and m be the number of constraints that need to be satisfied in the CSP.

1) Construction of Implication Graph – $O(n+m)$
2) Construction of Dependency Graph
   a) Selection of Dominant Edges – $O(m)$
   b) Removal of Invalid Edges – $O(m)$

So, the overall time complexity to determine dependencies is $O(n+m)$. The graphs constructed in our approach also use space corresponding to given variables and constraints, i.e. 2n nodes and 2m relation edges, resulting in both polynomial time and space complexity.

The chapter provides a dependency analysis approach based on constraint satisfaction problem for the context aware ubiquitous services. Context aware services are usually developed using context models and concepts that are informal, lack any clarity, mostly aimed at a particular application domain. The dependency analysis may reveal a lot about dependencies among ubiquitous services, and for each dependency some possible dependency reduction strategy may exist that need to be studied in the future. But the question is how to determine and manage all these dependencies, as new dependencies may get introduced due to reduction of a previously existing dependency. As a consequence, some dependency reduction approaches will introduce new dependencies, and some dependency reduction strategies will introduce new possibilities to resolve certain context dependencies. The issue of constraint solving to determine service dependency has got its complexity reduced from NP-complete to polynomial time complexity through our approach.

The future work that we envision regarding our approach involves enhancing the algorithm further by introducing dependency confirmation logic. The essential task will be to evaluate the pervasive systems with high coupling and direct dependencies in comparison to ones with low coupling and indirect dependencies.

A formal model is presented in the next chapter to model context dependency using predicate logic and set theory.

5 Modelling Context Dependencies

5.1 Context Dependency Representaion

The rapid technological development in wireless networking and the emergence of a variety of mobile embedded computing devices has resulted in the spread of the applications of computing from work and office to more dynamic environments in our lives. This has led to potentially phenomenal growth in the ubiquitous computing that uses small computing devices; interconnected and continuously communicating with each other, helping the users in carrying out tasks in their daily life. It is widely acknowledged that the success of pervasive computing requires a change in the design of such applications as mentioned by Norman (1998) and Henricksen (2001). The dynamic nature of the ubiquitous environment requires pervasive applications to be capable of reacting appropriately to change in context in short time, without user interaction. In order to meet such requirements, the pervasive systems need to be highly responsive to a change in context.

Modelling of context dependencies is an important step in analysis. Characterization, representation and followed by detailed analysis of context dependencies is essential to accurately model a pervasive system. It is important to identify context dependency among context-aware applications to determine the impact and extent of a change at a context source. What is required to achieve such insights is a formal characterization of context dependencies. Such formalization helps in a unified approach to context dependency analysis. In most of the related research literature, it is implied that modelling context dependencies can be simply achieved by a directed arc on a network graph where the context source and context consumer are the nodes of the graph. The approach in this chapter attempts to formalize the definition and characterization of a context dependency to better understand the effects of context dependencies in a pervasive system in a more unified way.

Much of the research literature takes the definition of a general dependency between modules for granted, and if any definitions are given, they vary widely among the few that we have encountered. Few consider dependencies as a first-order logic formulas as constraints in database terminology [Hall, 1997; Crestana-Jensen 2000; Thalheim, 1998]. Some studies have presented higher order logic, like Prost (2000), while few are based on probabilistic approach expressing dependencies as conditional probabilities among specific variables and only consider dependencies as a statistical problem [Briand, 1998; Levinson, 2000; Subrahamanian, 2000]. Still, most consider dependency a direct offshoot of client/server approach [Rumbaugh, 1998; Yu, 1996] while few consider dependency as functional such as [Keller, 2000] or as data dependency only [Papazoglou, 1997]. A detailed characterization of dependency along with its various "dimensions" is presented in [Keller, 2000] in addition to the work done in [Prost, 2000] that also took "type-based" approach to dependency analysis.

To understand context dependency, the thesis takes the Realist's view [Hayes, 1994] where the assumptions are that "a set can be a set of anything" and "the universe can be physical or abstract or any mixture". Allowing this perspective, an entity can be anything which is a member of this set and can be modelled. This entity can be an object, a concept, a system, an organization or anything. Another assumption is that the entities in such a set are not static; they can change their state or adapt new behaviour. At this moment, there is no rush to define a change; rather we just acknowledge that it exists. The change in an entity can be a part of its nature or is influenced by some other entity outside itself. It is this second kind of nature is under consideration in this thesis. Taking these assumptions a step further, it can be assumed an existence of a relationship among various entities exists. This relation implies a potential change. The change can propagate from one to one or one to many entities.

The current approaches to context modelling lack solutions regarding efficient context dependency modelling that make it difficult to develop capable context aware systems. The

main areas that we reckon for lacking various features in context aware systems are the context quality, diversity and dependency management. These have to be addressed to make context aware applications more resource efficient.

Before addressing the issues of context dependency, a formal analysis of dependencies is required to better understand the problem. Dependency analysis can help to choose sensible sets of loosely-coupled context dependent applications, with only absolutely required context dependencies or other context information. Context dependency analysis can help to determine exactly which of the derived contexts rely on a context source, *C*, along with the thorough nature of such dependencies. It must be determined which contexts are affected directly and indirectly, if there is any change in *C*. If *C* is not available in the overlay network, is there any other alternative context source, C_2, which can be utilized to provide the dependent context aware applications with relevant context information. The analysis can also help to determine the context dependency of C itself on other contexts, and how it affects the subsequent derived context if its derivation is delayed or even halted all together. The complete context dependency analysis of a complex ubiquitous system determines the context dependency hierarchy among the participating context aware applications and context sources. It facilitates the breaking down of large complex systems into smaller, loosely-coupled components whose complexity can be analyzed and managed more thoroughly.

5.2 Context Modelling Approach

Broadly, we can classify context modelling approaches in to two categories as presented by Christos (2007):

- Context Theoretic Modelling: In such an approach, the information regarding physical entities are fused together to represent context. The information regarding entities is constantly changing due to the dynamic nature of the ubiquitous environment the context aware systems operate in. There are two common ways to represent context in context

theoretic modelling approach. It can be situation centric, as the context can be derived from the events involving the subject entities, or it can be activities themselves that can be utilized to describe situational context.

- Context Conceptual Modelling: In such an approach, context information is described as concepts and the relations among different concepts are modelled to give a better view of the system and interactions in it. The conceptual modelling approach categorizes context based on the "prevalent characteristics". Context conceptual approach can also be further divided into two alternatives. In one of the two approaches, conceptual graphs can be employed to describe context information. Concepts are represented as vertices and the relationships among them are represented with the corresponding edges. The second approach is based on the sets of prepositions to describe context.

In the following section, we present a formal model for context dependency.

5.3 Dependency Abstract Model

Handling context dependency in ubiquitous systems is complex as it requires detailed knowledge of the distributed system. The complication involves reasoning, regarding approaches to context dependency issues, and algorithms that attempt to manipulate dependencies efficiently. In such situations, a mathematical approach to the problem is preferred for thorough analysis. Reducing context dependency to just Boolean expression is tempting because of its easier and simple approach, but such a model conceals the structure of the original problem. The proceeding section attempts to develop an alternate model sufficient enough to capture context dependency along with retaining the structure of the system.

5.3.1 Dependency Model Assumptions

Our dependency model is based on the following assumptions:

– Mobile devices. A description of the current state of each of the mobile devices' resources is known to the system so that such information can be utilized when considering

the solution to context dependency issue. Any slight change in the device resources may influence the suggested optimal solution.

– **Application and services.** Applications are considered to be composed of several dependent services. The state of services and their attributes need to be taken into consideration in order to achieve a complete model of the system. The dependency resolution is then attain by considering both the applications' and services' constraints.

– **Users.** Users may have preferences regarding the applications and services that need to be loaded on to their devices for certain usage. Preference levels can also be utilized for the selection of services or applications. Such user preference may influence the dependency resolution process.

5.4 Basic Model

The targeted entities represented in this basic model are context types, context data available from various sources, and the context dependencies among them. Context types are denoted by CT_1,…; context nodes can be represented by C_{N1},…; and context dependencies can be represented in the form $C_N \rightarrow \{ C_{N1}, ..., C_{Nn}\}$ which shows that a context node, C_N, is dependent upon certain context nodes $\{ C_{N1}, ..., C_{Nn}\}$ for gathering context information.

The state of an entire system can be represented as follows:

- C is the set of all the context types.
- S is the set of all context nodes present in the system.
- D is the set of all dependencies.

Since, context aware systems are based on limited resources, C, S, and D are all finite sets.

5.5 Dependency Model of Ubiquitous Systems

The precedent abstract model is now being expanded to model context dependencies in ubiquitous systems.

- C is the set of all the context types, e.g. location, time, etc.
- S is the universal set of all context nodes, C_{Ni}, present in the ubiquitous system. These context nodes can be of two types, either the node is generating context and is called a Context Source, C_S, or the node is using context and is called a Context Consumer, C_C. It should be noted that a context node can act as a context source and a context consumer at the same time but not for the same context type. The context source involved in producing a certain context type is indicated by C_S:CT_i. Similarly, when a context consumer is utilizing certain context type, it is denoted by C_C:CT_i. A context consumer is always utilizing at least one context type; otherwise it is not the part of a context aware ubiquitous system. When a certain context source is not being utilized in the ubiquitous system, it is indicated by C_{SU}.
- A context node, that is dependent upon some context information, can represent its preferred choice in terms of context type and its context source that it is utilizing. This representation is a function that maps the relationship between the context consumer and the context source. If a context node is using context type, CT_i, obtained from a certain context source, then this relation is represented as:

  $C_{Ni} = \{CT_i: C_{Si}\}$

- A set is formed by adding dependencies to it that are the result of a context node being dependent upon certain context types and their sources. For example, if a context node, C_N, declares a dependency on a context type, CT_1, which can be obtained from three different context sources, C_{S1}, C_{S2}, and C_{S3}, and C_{S3} is itself acting as a context consumer for context type, CT_2, then this set of dependency is,

  $D(C_N)=\{CT_1: C_{S1}\ ,\ CT_1: C_{S2}\ ,\ CT_1: C_{S3}\ ,\ C_{S3}:CT_2\}$

The context dependency super set of the ubiquitous system can be defined as,

$$D = \bigcup_{i=1}^{n} D(CN_i) \ \ \forall \ C_N \in S \ \ \wedge \ n = |S|$$

where D is an empty set to start with.

- A context aware system satisfies a dependency, D, for a context node , C_N, if in the current state:

  $C_N \rightarrow \{ C_{N1}, ..., C_{Nn}\} \ \wedge \ \forall \ C_{Ni} \in D(C_N)$

  The above expression means that the dependencies of a context node are satisfied if none of them is violated in the system, i.e.

  $C_N \neq \{CT: C_S\}$ if $C_S \notin D(C_N)$

5.5.1 Context Conflict

Effectiveness of context applications is largely reliant on sensors' reliability. There always exists some amount of doubt regarding consistency of context aware systems that are dependent upon device sensors for sensed inputs which are inherently uncertain or even incomplete sometimes due to unreliable readings of the sensors. In ubiquitous system, it is plausible to have conflicting contexts due to loss in data transfer, or erroneous sensor reading. If such a case occurs, the context aware application will have to decide which one of the context sources to trust. The trust preference can lead to complete screening out of the suspected data among the conflicting context sources.

Consider context source, C_{S1}, to have the same context type as another context source, C_{S2}, serving a common context consumer but there seems to exist disparity among their values when sensing or deriving the same attribute. The context application has to make a decision of preferring one of the context sources to cater its contextual data requirements. If C_{S1} is chosen, then C_{S1} along with C_{S2} have to be added to the dependency set for that context node. For example, if a contest node, C_N, declares a dependency on a context type, CT_1, which can be obtained from two different context sources, C_{S1}, and C_{S2}, and C_{S1} is preferred, while C_{S2} is

itself acting as a context consumer for context sources, C_{S3}, and C_{S4}; then this set of dependency is,

$$D(C_N)=\{CT_1: C_{S1}, CT_1:C_{S2}, C_{S2}: C_{S3}, C_{S2}: C_{S4}, CT_1: C_{S1} \wedge\neg C_{S2}\}$$

The set indicates that C_{S2} will not be utilized when C_{S1} is available, and the effectiveness of the context aware application is dependent upon this condition.

5.5.2 Context Acquisition

A formal approach is mathematically a sound approach to model context dependency, along with a way to determine valid context dependencies at run-time. To further extend the model; in this section we present a solution to show the frequency of context requests. If the frequency of a particular context-type's requests are greater than a certain numerical threshold in a given time period, we can say that the demand for that context-type is higher and it needs to be made promptly available in the system to ensure the successful working of the dependent context aware applications. In our model, we denote the frequency with symbols 'R', for rare requests, and 'O', for requests that are often communicated to the context source. The following dependency set shows that the requests for CT_1 are not very often, and the node that used to serve this request is rarely queried in this context:

$$D(C_N)=\{CT_1: C_{S1}, CT_1:C_{S2}, C_{S2}: C_{S3}, C_{S2}: C_{S4}, CT_1: C_{S1} \wedge\neg C_{S2}, CT_1: R\}$$

One another important aspect in any context aware system is the policy adopted for the distribution of contextual data. Many context aware services ask for a context from the source whenever there is a need for one. This process is like any other query-response approach, where the context consumer asks for a particular context from the context source according to its needs. The other way a context consumer can acquire context is that it get itself subscribed to the context changes. The context source sends the contextual data to the context consumers,

who have them subscribed to the context source, whenever there is an update in the context. For the subscription approach, the dependency set will be as follows:

$D(C_N)=\{CT_1: C_{S1}, CT_1:C_{S2}, C_{S2}: C_{S3}, C_{S2}: C_{S4}, CT_1: C_{S1} \wedge\neg C_{S2}, CT_1: R_{SUB}\}$

And for the query-response approach, the following dependency set is described:

$D(C_N)=\{CT_1: C_{S1}, CT_1:C_{S2}, C_{S2}: C_{S3}, C_{S2}: C_{S4}, CT_1: C_{S1} \wedge\neg C_{S2}, CT_1: R_{QUE}\}$

The model of context dependency management that we developed is based on the set theory and predicate logic. It can be categorized into context conceptual modelling. We faced hurdles when representing context dependencies as they can be static or dynamic, sensed or inferred. We acknowledge that our approach based on set theory and predicate logic cannot be compared to Entity-Relationship Model and the class diagrams of UML, which are more natural approaches to model systems, but our approach does succeed in representing dependencies more logically.

The forthcoming chapter of the thesis present approach to reduce context dependencies, with the help of profile context that we propose to reduce dependencies in an ad hoc overlay network based pervasive system.

6 Reducing Context Dependencies

6.1 Requirements of Context Dependent Access

A mobility oriented ubiquitous environment provides unified contextual information and format but also allows different context providers to get involved in the process. It is quite possible that the information format is completely different from the other context providers' data schema. The involvement of different context providers raises different problems like translation, compatibility and coherence and thus when the user switches from one provider to another these problems could cause delays and misconception, and hence must be addressed.

To overcome these problems the information management systems must allow users different queries like, whether the unified information is independent of the specific context provider and is it correct with respect to information schema.

To enhance the contextual information obtained from various sources, and then, make sense of all such information about a single entity or user is one of the more demanding requirements of any context model.

Because mobile communication is not ideal or free of errors therefore, its discontinuous nature enhances the issues of compatibility and coherence. It is possible that the context is provided by the same provider but still because of mobile communication environment the same information is interrupted. Another issue is of adeptness.

In comparison to static distributed computing environment like local area network, the ubiquitous computing is more vulnerable to problems discussed above and thus required demanding information for integration. It is because different services providers are offering their services and the context aware applications are able to access the context from these different sources. Thus the problem arises when the context is not accurately reached to the context aware applications when the user is at the overlap of two context aware services.

Therefore, to overcome this problem it is imperative that the application must feel that the context is retrieved from only one source even if in the background different sources are providing the context. Furthermore the problem of adeptness is more complex than in static environment. In ubiquitous computing different factors like device features, interest, and capability of integration are considered.

6.2 Context Dependent Access Design

Minimizing dependencies among different entities of a system is a well-recognized need. [Huynh, 2007]. In the Open Distributed Processing (ODP) reference model of Blair (1998), the idea of transparencies was presented which is a mechanism that allows hiding certain complications during the development phase of distributed applications and platforms. One example can be the location transparency, presented by Blair (1998) that hides the details regarding the process of finding the location of distributed objects. Logical names are used to find such objects rather than the physical addresses, giving more semblances to the design and development process.

We based our context dependency reduction approach on this similar idea by hiding the complexities of querying individual devices for context and presenting them as part of the profile context. The important aspects of our approach are:

- The context requirements must be specified in a formal context modelling language rather than a programming language to make context dependencies more visible for system's analysis.
- Our approach can help to gather context information from various heterogeneous devices that are part of the same ad hoc network.
- Our context dependency reduction approach can be utilized with an interface that allows the luxury of not knowing the exact physical details of each individual context source.

- Our context dependency management approach is responsible for interpreting the contextual dependencies to:
    - Initialize a context dependency process when a new context source is discovered by one of our previously mentioned approaches, select and create that dependency in our formal model.
    - During real time, keep looking for other sources of same contextual information to have the information about the alternative sources present in the system in case of failure of context acquisition from the current source.
    - Reducing context dependencies by aggregating device contexts into a profile context at one of the super nodes in an ad hoc network if the devices belong to the same user.

When a context source leaves the system and its context is not accessible anymore, or better quality context source becomes available, our approach tries to change the context dependency. When the context consumer doesn't require the context anymore, the dependency between the context source and consumer is removed from the system.

The super node can help to store the information regarding the dependency among the different agents in the ad hoc overlay network. The context retrieval can be either request based or subscription based. The implementation of request based is simple as the system has to service the context queries as they are executed, finding the relevant context information and forwarding it to the context consumer. For the subscription based approach, the dependency has to be stored in the super node of the overlay network. The mechanism is followed on the basis of application specified subscription condition. It can be either periodic updates or based on notification if there is change in the context information from the context source or the context data is updated.

When a device context dependency is discovered in the system, our approach retrieves the dependency and creates it as a component of profile context which serves to store that particular device context at one of the super nodes. This super node with profile context then act as a single point of contact for the applications to acquire the contextual information pertaining to that user who is the owner of such device. The super node can provide the context based on particular requests or as a periodic update, to the context consuming application or service.

6.3 Properties of Context Producer

Some of the properties of context producers that present challenges to our approach to reduce context dependencies are:

· Distributed: the context is provided by a multitude of various physically distributed context producers and the problem that they represent is how to discover relevant, alternative context producers and how to provide for the exchange of context information.

· Dynamic Availability: the context producers can physically enter or leave the system, common especially in ad hoc networks, at their will. This creates a problem for context consumers as they face uncertainty while asking for a particular context. Therefore, the presence and availability of any context producer is not guaranteed in ubiquitous computing.

· Dynamic quality of context: As context is obtained from different sources, so the quality of gathered context is dependent on the reliability and quality of sources that provided the contextual data. Therefore, improving the efficiency and accuracy of context aware applications require incorporating at least an aspect related to dynamic quality of the contextual data.

· Heterogeneous context models: Same type of context can be provided by various context producers but they might not have the same context model for storing and representing the contextual information.

We have developed our context dependency reduction approach considering peer-to-peer ad hoc networks. The plan is similar to the one in Capra (2001) which uses meta data for transfer of context information. XML can help to structure the contextual information easily and also making it easier for applications to access and process the contextual information. Easy access to specific fields is the main advantage of XML based formats. An example XML structure is shown. It shows the context information in regards to the scenario discussed earlier in one of the previous chapters. The profile context is prepared at one of the super nodes by aggregating device context in the format of XML.

```
<context type="location">
        <status value="meeting room"/>
</context>
<context type="busy">
        <status value=”yes”/>
</context>
<context type="text">
        <status value=”ON”/>
</context>
<context type="voice call">
        <status value=”OFF”/>
</context>
```

6.4 Context Acquisition

The context acquisition process is a vital feature of any context aware system. Most approaches try to acquire the device context instead of the “profile context”; each device owned by a user is contacted individually for the purpose of acquiring data relevant to the user situation. Context published by a device fails to capture the overall picture of user situation in comparison to context obtained from multiple sources that are aware of their

mutual existence, and can provide better user context working in collaboration among them. User context is usually dependent upon two or more devices' context.

Despite their potential widespread applications, context-based services are still in infancy because the context gathering architecture is based, more or less, on query-based silo framework as shown in Figure 11. Such a framework is susceptible to numerous issues; most common of them are bottleneck performances and high probability of failure because of single point of contact. If the server-client approach is employed to realize a context aware service, it can be easily imagined that servers will soon be overloaded with queries, affecting their performance. The dynamic nature of information to be handled in a ubiquitous computing environment is huge, and the numbers of queries increase with time as more users join the network.

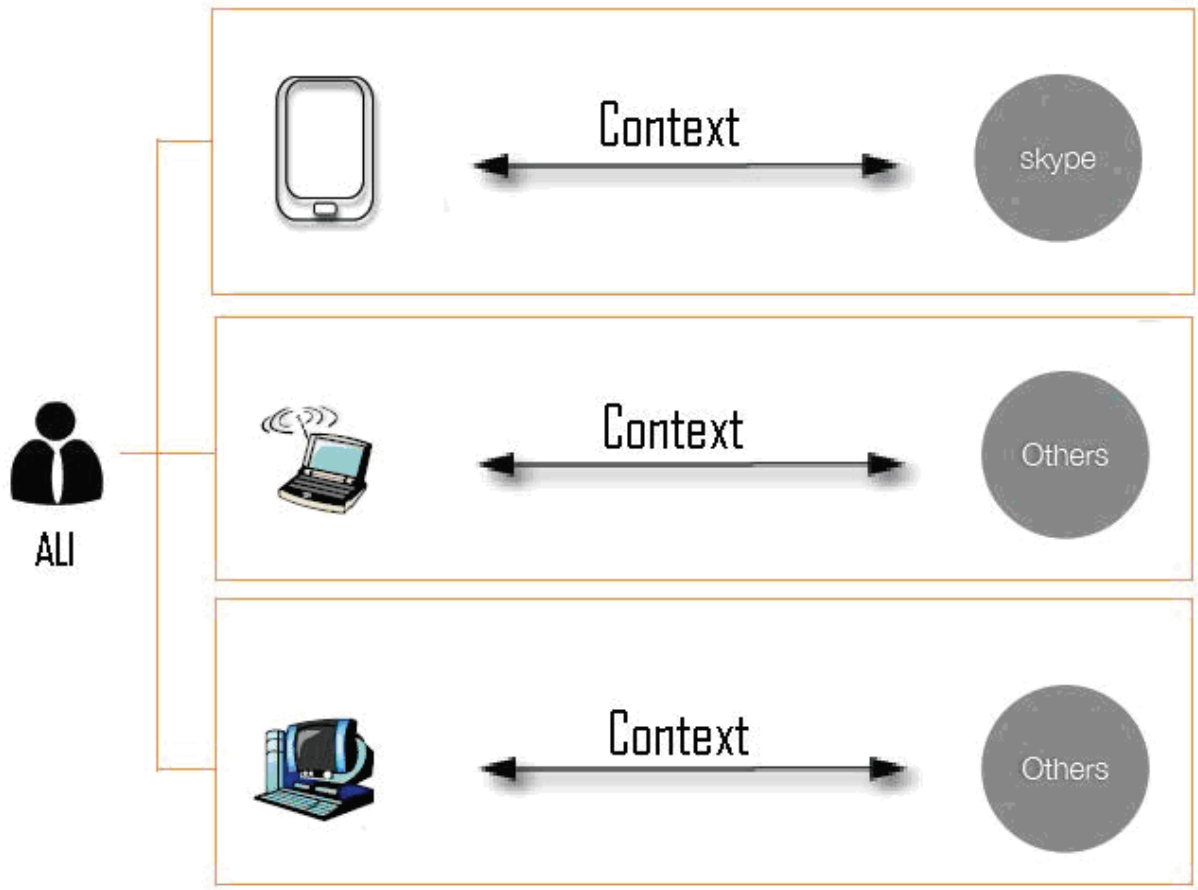

Figure 11 Query-based Silo Framework

Problem with such architecture is that if there are N applications asking for the context from the device, the device has to reply to N queries, which puts quite a lot of load on both the network and the resources of the device itself. Current context sharing applications provide user's context but such context is device-specific only, as it fails to provide the whole picture of the user situation. If an application needs to get the whole context of the user, it needs to query every device currently in use of the person.

In this chapter, we aim to combine the idea of profile context and overlay network together for the purpose of proposing an open framework for context acquisition which also address the issue of context dependency. This approach enables the notion "Context is always available", based on the feature that the user's context is accessible even if he is offline or his devices are switched-off. This is possible because profile context keeps the latest context of the device in overlay network, and is available even after the device has been turned-off. Thus, an application can always access latest context avoiding delay resultant of context dependency.

The design of our approach was driven by the following requirements:

- Adoption of a generic and flexible context model that could be dynamically deployed in an ad hoc environment.
- Improved performance, allowing fast and resource efficient access or dissemination of context information by reducing the intensive use of network and memory in resource constrained mobile devices.
- Support for context-related communication, facilitating the use of context aware services from different kinds of devices, operating systems and execution environments.

In our proposed approach, principally there are three components that interact to create, disseminate and consume context information in the form of profile context. These

components are context producer, context consumer and the context service. These components are part of an ad hoc network acting as entities and are represented by applications and devices that can be either mobile or not. The context provider is a network entity that is responsible for sensing context information of certain type. It is most of the times the generator of a raw, sensed data. The context consumer is an entity interested in a obtaining a certain type of context information. A context consumer is usually a context aware application that needs context data to perform its functions. The context service is accountable for aggregating, storing and disseminating profile context information in answer to the requests of context consumer.

The context service is assigned with certain responsibilities. The foremost responsibility of context service is to aggregate context from different devices and sources owned by the same user into a comprehensive profile context. Another responsibility of this context service is provision of asynchronous communication which means that it helps to deliver context information in the form of profile context to the context consumers. This context service is running on super node and it adopts a publish/subscribe approach to handle subscriptions of events based on context changes. The context service is also responsible for maintaining a distributed hash table to help store the profile context information in the overlay network and directs the requests to appropriate resource sufficient peer nodes in order to answer a query regarding a specific profile context pertaining to a certain user.

We follow an XML-based profile context model for context handling, instead of an ontology-based model, because, in our opinion, the ontology-based model requires resource hungry engines for processing ontology which hinder context management on resource-limited devices.

The deployment of such context service requires two main steps for the purpose of communicating context information. They are context modelling and then processing that

context model. The premier stage consists of modelling the context information into profile context using an XML-based approach. In an XML file, there are tags that identify user identity, device identity, characteristics, attributes, context history, quality of context attributes and the context information itself.

In the processing stage of the context model, the context service reads the XML file, and takes the following tasks:

- Validates the syntax of the XML file
- Updates the profile context for storing in the new context information using distributed hashing

When a developer of context aware application wants to utilize context information available in the ad hoc network, he makes the application reads the contents of the XML file with the help context model processing module to understand the semantics and meaning of the context information. Context dependencies among different context entities are also included in the XML file.

6.4.1 RDF-based Profile Context

A Resource Description Framework (RDF) is a knowledge representation language with resources for modelling context information. Thus, we use RDF for describing Profile Context. The RDF statement is usually represented as a triple, which contains a subject, a predicate and an object node as shown. We have designed the base structure of the triples, used in the Profile Context including the types of properties (representing a predicate node), the range of values belonging to each property and the resource types that has been given property called “domain” and “range” in the RDF Schema, respectively based on the set of templates.

6.4.2 Functions Requied in Profile Context

The common functions required in the profile context are described here:

Context Aggregation

The communication between the context generator and the context service running on the super node is based on the message exchange with an XML document on transport layer protocol like SOAP. Since RDF is intended for static descriptions, no procedure exists for dynamically updating triples in RDF. Thus, we therefore have to design a dynamic update module in the profile context aggregation process. The input context data is described by XML. An XML based input profile context is converted into an RDF based input profile context using the style reformat module. Then, the profile context is updated using the profile context update module. The style reformat module is described by eXtensible Stylesheet Language (XSL). In the profile context update module, the required functions here are:

- adding triples
- updating literal values
- unifying the same representation of triples.

We intend to implement this update module on semantic web framework like Jena.

Querying

RDF Data Query Language (RDQL) is an SQL like query language especially designed for RDF, which is designed in Jena. RDQL provides methods in which the application developers can write the declarative statements of what should be retrieved under certain given conditions. Thus, we can use RDQL to retrieve the target profile context or the values from the profile context.

Notification

In real life scenarios, multiple applications run simultaneously and can access the RDF statements in the profile context. Any modification in the profile context must be detected in order to detect the change in that particular context. Then, the context service provides access

to updated context information when it receives the notification from the profile context update module.

Privacy

Users are rightly concerned about privacy issues, which arise particularly in collecting and exchanging context profiles over the ad hoc network. Correct profile context must be exchanged with the context consumers in order to provide better context-aware applications. So, there is a balance that needs to be achieved between allowing personal context access and using context aware applications. In aggregating profile context from various sources, make sure that the communication channels are secure. And in distributing profile context, effective management is required to regarding the access rights to the profile context.

6.5 Contextual Events

Contextual events characterize notions about the surrounding environmental and circumstances that a context aware application is concerned with. A context event is defined in terms of the values of context attribute and the predicates.

Contextual events are the elementary components for constructing asynchronous notifications, and helping to various adaptations in context aware applications. Since context aware applications modify their behaviour in response to contextual changes that are usually identified by a contextual event. A simple example of a context variation is the drop in energy levels of a mobile device in an ad hoc network. The context aware application would be conceived to respond to such a change by subscribing to the corresponding contextual event, and thus avoiding the overhead of polling that it would have been needed to detect the change in context.

Query	Application Demands
Where is the user?	Get the current location of the user if present in the office building
How many people are present in the meeting?	Get the aggregate number of the people whose current location matches with the meeting room

Tableau 3 Examples of context query and the application demands

To efficiently access and disseminate context information, the context service differentiates between the static and dynamic parts of the context information. This approach helps to adjust context evaluation and its distribution according to the characteristics of the context information and the current execution environment. For each such part of context information, the context service chooses the most suitable strategy for ensuring the availability of context information.

Contextual Event	Response
User has entered the office	Turn on his desktop computer and play new received messages on the telephone
User is moving in the office building	Update the user location in the office database

Tableau 4 Examples of contextual events and the corresponding application responses

6.5.1 Static Context

To improve access performance based on the context information, the context service should deal dissimilar types of context data differently. The nature of the context information, obtained after processing the XML file, lets the context service select the most appropriate procedure to manage that certain context information. When a context service processes a context update, it assigns it a category for context access. The context service uses such

context access category to adjust its behaviour at the runtime. Since static context information that has an attribute which always has the same, constant value doesn't need to be provided multiple times to the context consumer, for example, the type of OS and its version that is running on a device. When handling such kind of context, the context service provides and updates this context attribute only the very first time when a context consumer requests such context. Management of context information belonging to local and different domains is another aspect of the task performed by the context service. Context information belonging to local domain is provided by a device in the same network that communicates its execution context like available memory. While on the other hand, context information belonging to a non-local, alien domain is provided by an external context provider. Therefore, access to non-local context information requires at least one hop in the ad hoc network. The context service stores local context information belonging to local domain in a local instance of the context repository i.e. peer node, which improves the performance for context information access times.

6.5.2 Dynamic Context

Besides handling of the static context information, the context service can also supports dynamic context. When a mobile device acting as a context consumer joins the overlay ad hoc network and registers itself at the context service, it can then describe an explicit contract for consuming certain context information from the context producers. These contracts are based on the context consumer's requirements regarding the details of the context information that is asked for. Freshness of the context information can be an example of such a requirement demanded from the context producers in the context information provision.

The approach is based on the peer-to-peer protocol used to exchange context between context producers and context consumers over the ad hoc overlay network. In detail, it is an XML-based application level approach that is aimed at providing dependable context information

among nodes that have joined the overlay network and also save resources on the resource constrained mobile devices by providing aggregated context in the form of profile context.
The approach uses a naming scheme based on the format of universal resource identifiers which identifies context information stored in the overlay network. The URIs easily enables unique identification of context information. Moreover, with these URIs it is also possible to relate to the available context information in a universal way which is both human and machine readable.

Our approach forms profile context storage in the ad hoc network that uses a Distributed Hash Table (DHT) to map URIs. The main advantage in using a DHT is that entries can be found in O(log(N)) time. The context service enables the exchange of context between context sources and context consumers with the use of the distributed lookup system. These context sources and context consumers are formed and combined in a single endpoint on the end devices which is a super node. The super node corresponds to a node in the DHT and in particular, the super node enables context aware applications to use the context service for distributing global context information between various entities in the ad hoc network and within certain time limits.

6.6 Overlay Network

One promising approach for realizing efficient context retrieval framework can be overlay network. An overlay network can efficiently manage network resources, and distribute both the processing load and the network load evenly in comparison to a server-client system. There have been many research efforts regarding information retrieval in an overlay network.

The context information of each device is acquired by the API on one of the super nodes in the overlay network. The super node helps to store the information in the peer nodes along with the context provided by the other devices own by the same user. The profile context is stored in the XML format. Applications requiring user context can enquire one of the super

nodes for the address of the peer node storing the XML file in the overlay network. The applications can also have them registered with a super node to receive the profile context whenever it is updated, or when a device context is added to the profile context. There can be other options apart from these two, e.g. the context is regularly provided to the application after an already agreed upon time interval. The storage and distribution of context data is handled by the APIs running on the super nodes. The list of subscribing applications, that want to have context information forwarded to them, is maintained by the API on the super node and notifications are sent to the relevant applications whenever the stored context is updated in the overlay network.

6.7 Profile Context Management

Profile context is the main idea of this chapter, and this section will highlight its management aspects, specifically how to manage profile context information in an overlay network. But first, we define profile context itself.

6.7.1 Profile Context

Almost all of the research literature has viewed gathering context as a server-client problem with context only being obtained from a single device, which can be appropriately called device context. The idea of “profile context” is being promoted in this thesis which can be rightly considered as a collection of context from various devices and sources whose context information is related to a common single user, as shown in Figure 12. Such a context engulfs all the context information efficiently from numerous resources, which can be then successfully utilized to determine current situation of a user precisely.

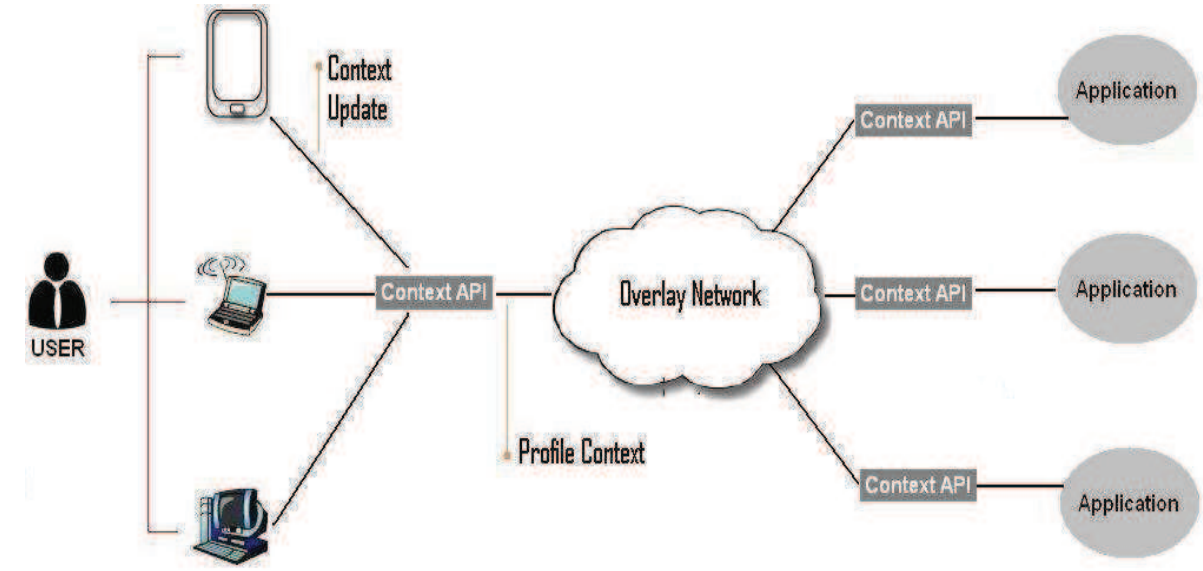

Figure 12 Profile Context Management Framework based on Overlay Network

As mentioned before, profile context rather device specific context is utilized which enables friends to see how they can reach the user. With the increasing dependency on Web 2.0, extension to include social networks to have access to profile context is a realistic option as depicted in Figure 13. Cloud computing based services are on the rise which can be one of the options in the future to replace overlay networks, eliminating scalability issues.

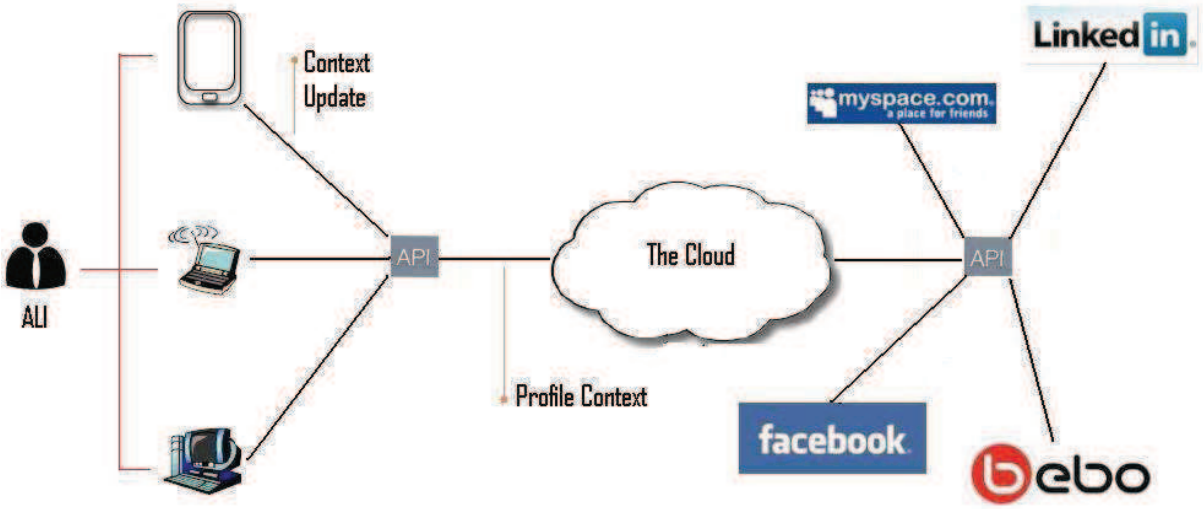

Figure 13 Profile Context in a Cloud

Profile context can be formally defined as a finite set, P_C, of all the devices' context belonging to the owner of the profile. If each of the device context is represented by D_{Ci}, and there are N devices providing the user context, then

$$P_C = \{D_{C1}, D_{C2}, D_{C3} \ldots D_{CN}\} \quad (1)$$

The context information present in whole of the overlay network can represent profile context of one or more than one users. The accumulated context, A_C, can be represented as a superset of all the profiles' context in the overlay network. If there are M users in the overlay network, then

$$A_C = \{P_{C1}, P_{C2}, P_{C3} \ldots P_{CM}\} \quad (2)$$

6.7.2 Context Update

For the purpose of better understanding the context update process, a sequence diagram in Fig. 14 is presented. There are two devices belonging to a user that act as the source of context information. Each device publishes its context whenever there is a change in the context. A super node in the overlay network running the context API is provided with the device context. The API then forward the context update to a peer node using appropriate hashing scheme. The peer node accepts and stores the context file. When another device belonging to the same user publishes its context to the context API, it is forwarded to the same peer node which stored the context of the user's last device. The peer node aggregates the context from the two devices into a single XML file. It should be noted that federated-identity provides the mapping between different profiles of the same person obtained from various devices.

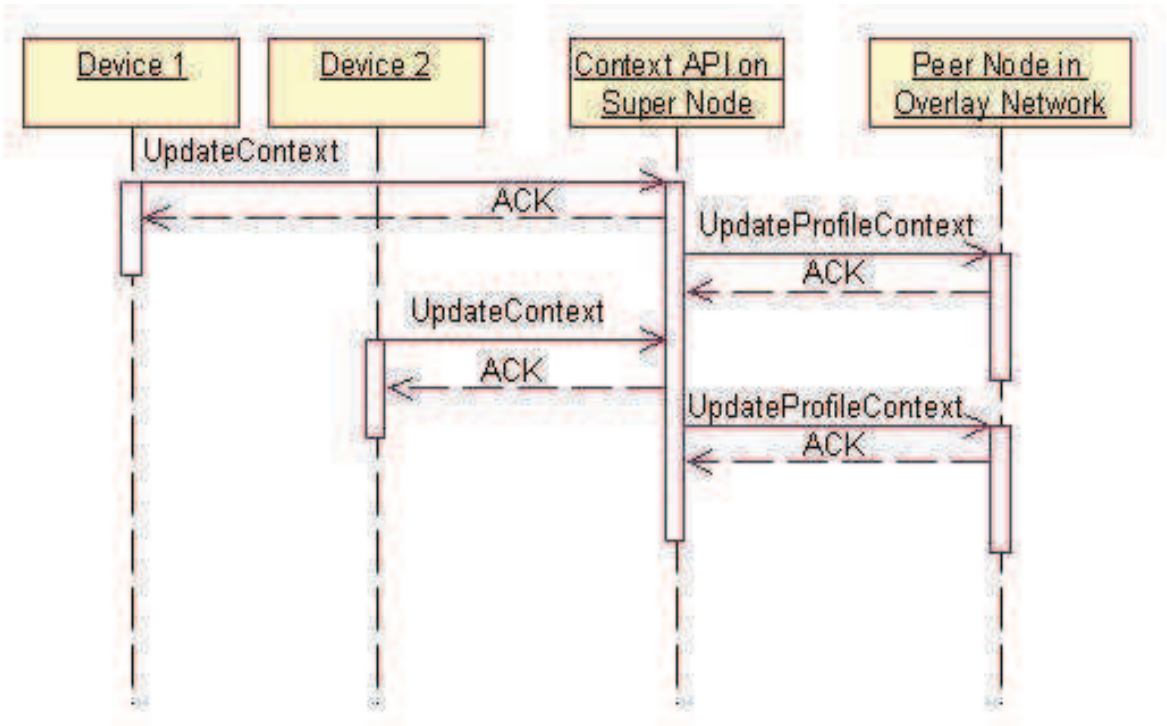

Figure 14 Sequence Diagram of Context Update

6.7.3 Context Retrieval

Context retrieval in an overlay network is quite simple in a sense that applications only need to send their query to one of the super nodes that take care of context retrieval for the respective applications. In the sequence diagram presented in Fig. 15, the application requests the context of the user from the super node running context API. By using appropriate hash function, the super node finds the address of the peer node storing the profile context. The peer node is informed of the context request from the particular application. The profile context of that particular user is then forwarded to the context requesting application.

There can be context dependent applications that have subscribed to receive context information when an update occurs. In this case, the super node takes the responsibility of multicast-like broadcast as it has more resources than a peer node, e.g. in terms of battery life, network bandwidth, etc.

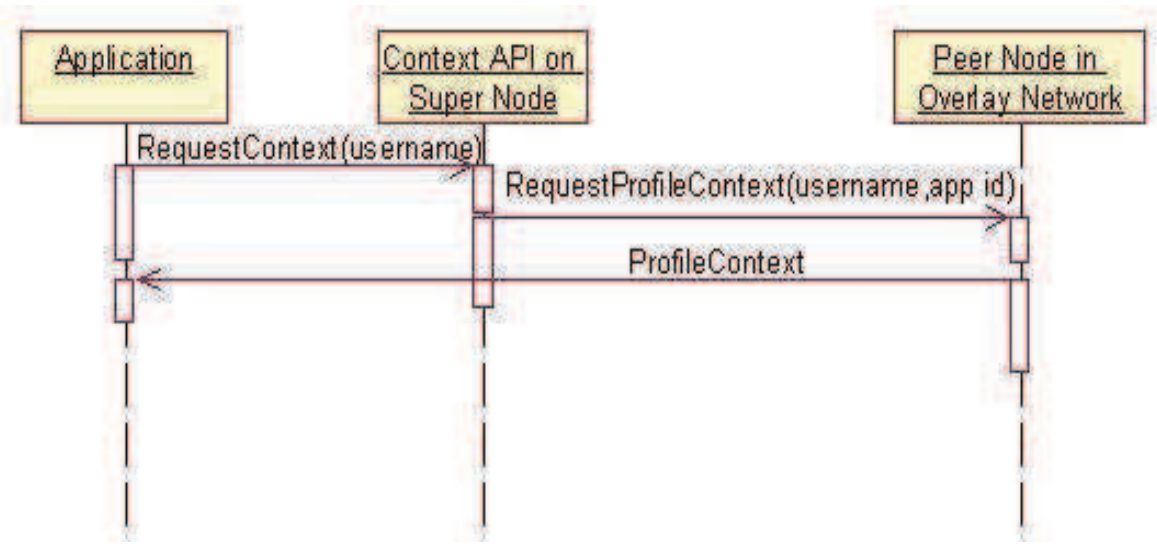

Figure 15 Sequence Diagram of Context Retrieval

6.8 Example Scenario

We present here an example scenario that can elaborate the advantage of using overlay network and profile context presented in our approach:

Bob is a medical doctor in the local hospital. He usually performs his duties in the emergency ward where all kinds of accident patients arrive in the need of first aid. As he is the doctor in charge of his team, he is required to be in contact with all of his team members. For this purpose, he keeps his pager on him all the time during his work shift. In case of arrival of new patient, he is paged on his pager. When Bob is busy with a patient, he sets the pager to busy so that no one can disturb him when he is providing first aid. For his identification, he is provided with an RFID readable badge. This badge can also be used to enter different rooms of the ward after authenticating at the door. This updates the location of the doctor in the central database. As the senior most doctor on the team, he is also required to sign medical reports and prescriptions on the digital pad installed in majority of the rooms. Whenever, there is a document that needs digital signatures of Bob, the context aware application looks at the profile context of Bob. The profile context consists of current status of Bob in terms of whether he is busy or not which is obtained from the pager status, and his current location that

is obtained from the last reading of his RFID readable identification badge. So, instead of looking for context information individually, the context of Bob is provided to the context aware application in the form of profile context. Thus, saving precious time of the doctors by not interrupting them when they are busy, and the resources that aren't much left on the resource constrained mobile devices like pager.

In the next chapter, we present the evaluation of our model. For the evaluation purpose, we did simulation and use mathematical modelling to formally analyse the results.

7 Evaluation

Ubiquitous systems have to deal with variability as context aware applications get deployed in different execution environment with varying computing platforms. Changing nature of the computing infrastructure and underlying communication, changes to the context source induced due to mobility and altering computing resources ask for context aware applications that are highly efficient at using resource constrained devices and responsive to the changes in context around the user. Due to the pervasiveness of ubiquitous systems and its demands to have adaptive behaviour, the performance analysis of context aware systems should be considered in conjunction with context dependencies.

7.1 Simulation Results And Formal Analysis

Apart from simulation results, formal analysis has also been undertaken using queuing theory models. In the simulation, the churn rate is assumed to be zero. As described earlier, the churn rate is the rate at which the mobile nodes join or leave an overlay network. If the churn rate is high, then the overlay network's performance drops considerably as the context information on the nodes, that are leaving the overlay network, needs to be copied or saved on to other peer nodes. This aspect is the one drawback in overlay network. The packet drop rate due to signal degradation is also set to zero, as the simulation was conducted to isolate the difference between device context approach and profile context approach, rather than the network communication. The simulation was developed in .NET programming environment. For the purpose of evaluation, we have concentrated on the number of queries in the system, each time when a service or application requires updated context. We have observed that our proposed framework shows better performance when the services or applications sent queries to all the devices owned by same user. If a query is aimed at only one of the devices, the performance gains in terms of network traffic remains negligible as the aggregated context factor in our approach does not come into play.

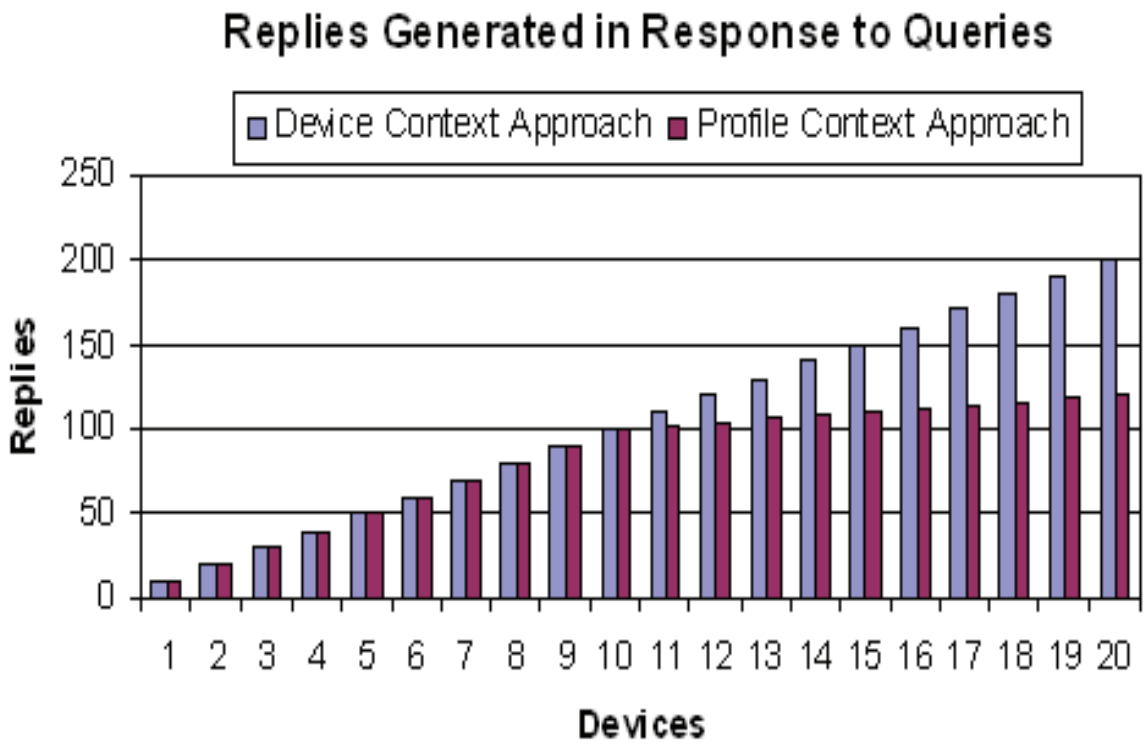

Figure 16 Comparison between Queries generated using Device Context and Profile Context

Performance in terms of traffic generated as the result of queries is plotted in Fig. 16. The number of users in the simulation was set at 10, and each user can simultaneously own up to 2 devices. Initially, when the overlay network is being populated by the introduction of users, the performance of overlay network using profile context remains almost the same as that of device context approach. The number of responses for profile context approach starts to divert from device context approach when the user starts to have more than one device in their use. This is due to the introduction of profile context consisting of multiple device contexts and replaces individual, separate device context. Performance in terms of traffic generated when a single user introduces more than one device in the network is plotted in Fig. 17. The number of devices in the simulation was gradually incremented up to 10 devices. The responses for profile context approach are significantly less than device context approach as the user has more than one device in his use which allows the aggregation of contextual data from multiple devices into a single profile context.

For further detailed performance comparison between the profile context approach and the device context approach, we use a queuing theory model presented in Baloch (2010), and also described in the Appendix, to present a more formal analysis. For the device context approach, total context nodes are split into two groups; one group with 5 super nodes shared among the two types of context requests, and the other part consists of context nodes with 2 separate context nodes each to service context requests related to context types I and II. The contextual data requested belongs to different two types, which means there are two devices acting as context sources, and each device can be accessed individually by further two participating nodes in the system beside the 5 super nodes.

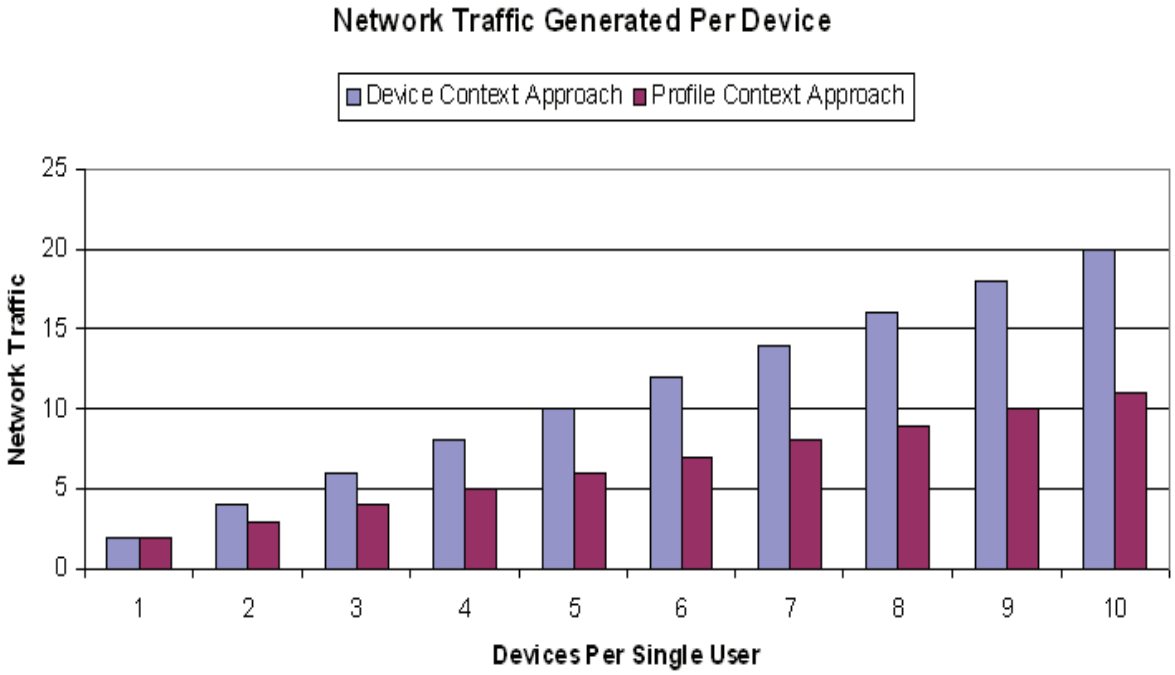

Figure 17 Comparison between Network Traffic generated for each device using Device Context and Profile Context

Profile context approach is evaluated by providing it with 5 super nodes shared among the context requests without any distinction based on their context types as the contexts are aggregated into profile context. There are also 4 peer nodes which are used to store profile context without any distinction of being a context source or not. So the total nodes are 9 in profile context approach, similar to the numbers in device context approach.

In our queuing model, Poisson arrivals for the two types of context request are assumed with parameters λ_1 and λ_2. The context requests' service times are exponentially distributed with means of $1/\mu_1$ and $1/\mu_2$ respectively. The buffer capacity at super nodes is limited to Q = 10. Fair queuing is implemented throughout the analysis.

Parameter	Value
λ_1	2.1-3.5
λ_2	2.5
μ_1	4.0
μ_2	4.7
Device Context Approach	9 nodes
Super Nodes	5
Separate Context Nodes	2 each
Profile Context Approach	9 nodes
Super Nodes	5
Shared Peer Nodes	4
Context Requests	6K

Tableau 5 Input Parameters

For the purpose of generating numerical results, the parameters listed in table 5 are used. These input values were used because it was thought that they depict the real time scenario more accurately. Initially, the simulation starts with medium context requests and then gradually has to cope with higher load on the network.

The stated comparison results exhibit that the comparative increase in context requests means queue length is less in the profile context approach than device context approach. The graph of mean queue length in Figure 18 of profile context approach has increasing trend in response to increased context requests traffic load. But in comparison to the device context

approach, the performance of the profile context approach is slightly better. This gain in performance is due to the introduction of accumulated context in the profile context approach, which is the main difference between the two schemes.

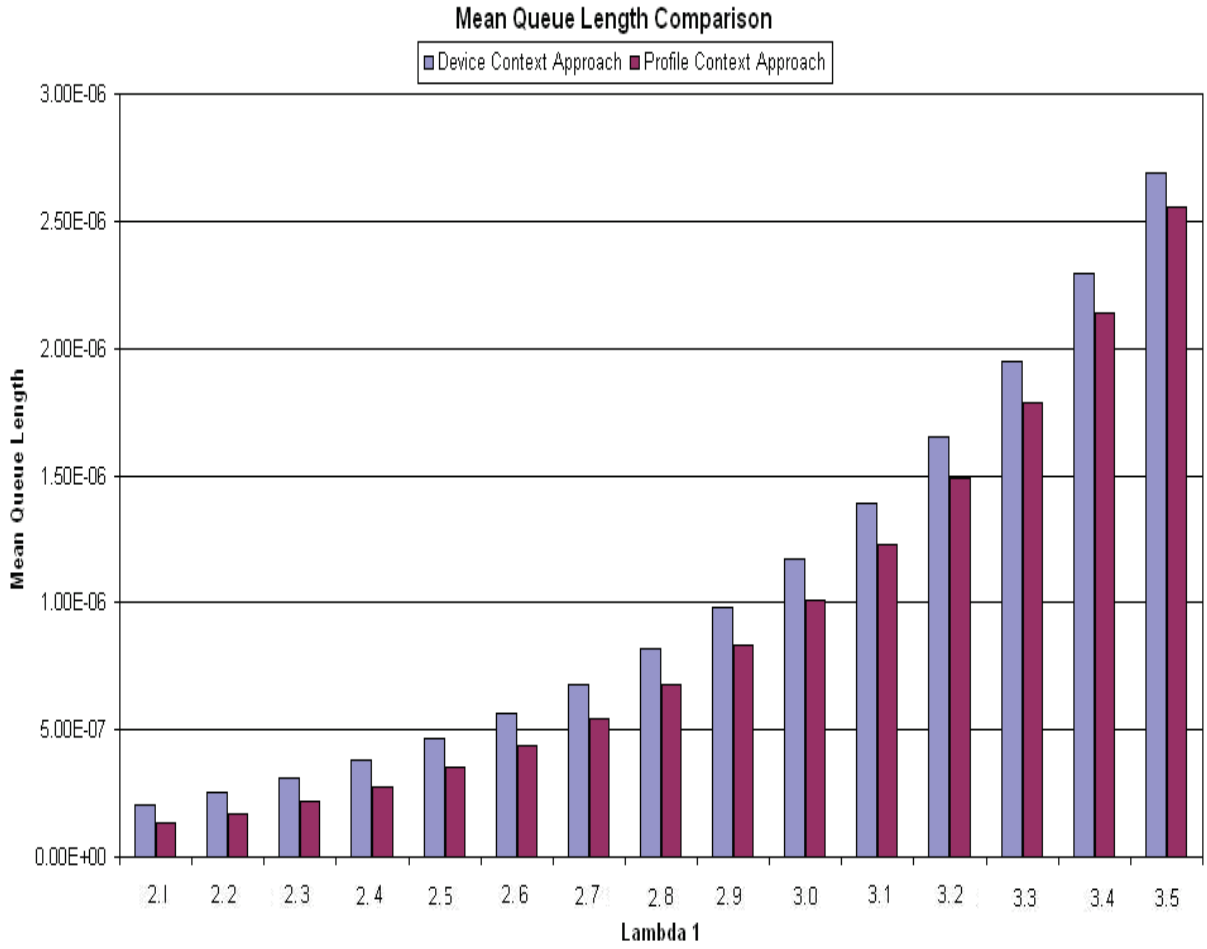

Figure 18 Mean Queue Length Comparison – Device and Profile Context Approaches

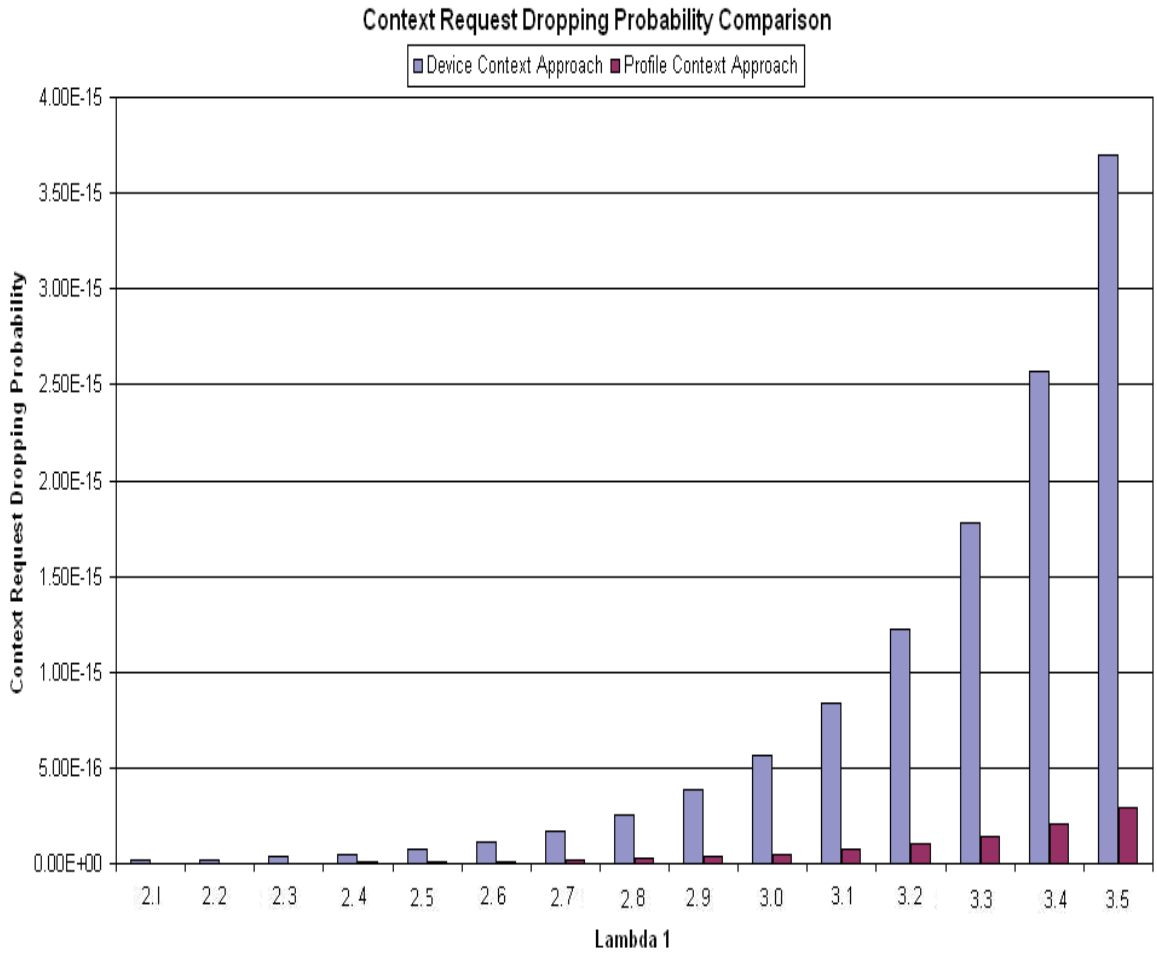

Figure 19 Context Request Dropping Probability Comparison – Device and Profile Context Approaches

Separate context nodes are reserved peer nodes which don't follow profile context approach and keep their respective context updates separate from other context types. This might result in increased context request blocking probability or context request dropping probability depending on the availability of these peer nodes in the network at the time of context request.

As it can be observed in Figure 19, the context request dropping probability of device context approach rises sharply due to rapid increase in context request load. Peer nodes can't serve any context request for which they don't have the required context type, even if context request has to be dropped or blocked due to timeout. Device context approach is easier to implement but has the risk of underutilizing the network capacity, and increasing the blocking probability of new context request generated in the network.

The results indicate that queue discipline also plays major role. The queue discipline in this study is FIFO, first-in first-out, where no context request is given priority. If the performance of the profile context approach has to be further improved for a certain types of context request, then a prioritized queuing approach should be adopted.

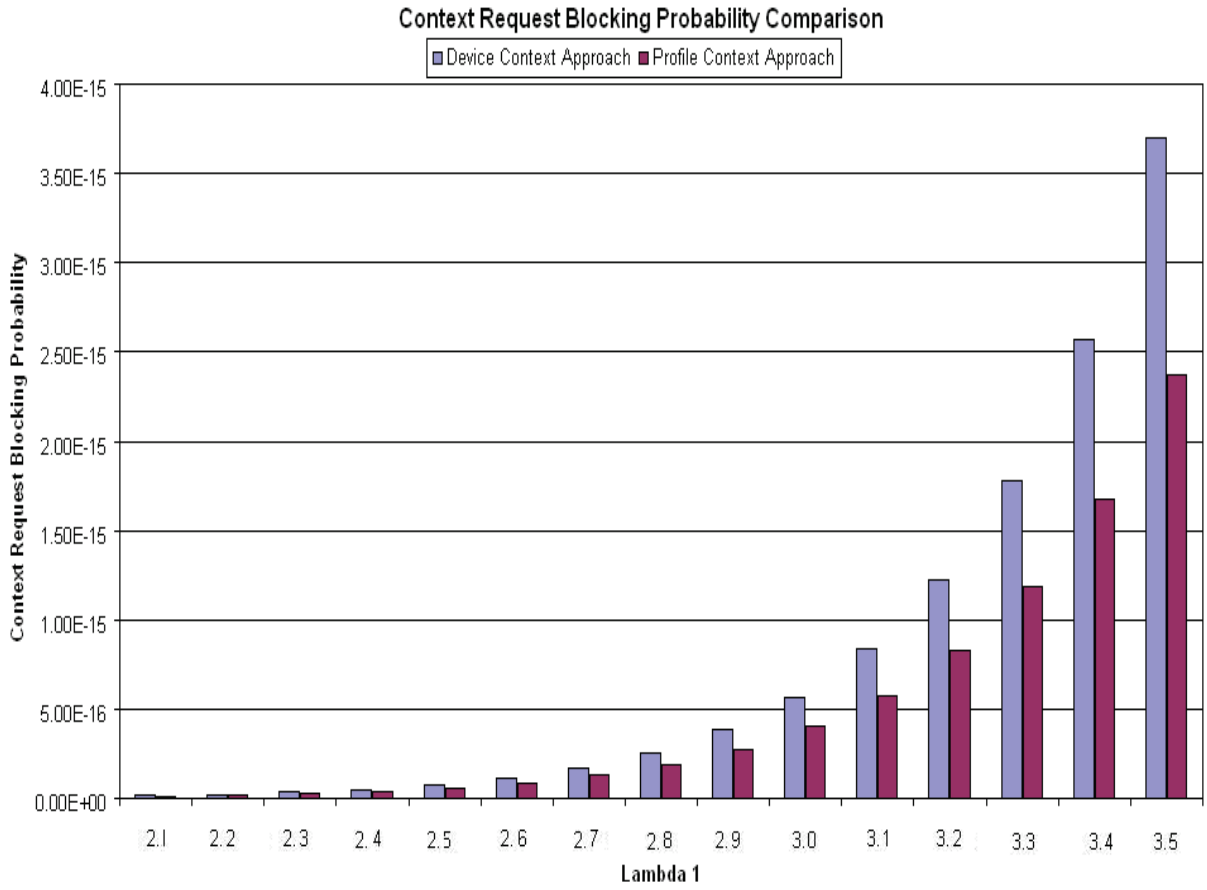

Figure 20 Context Request Blocking Probability – Device and Profile Context Approaches

In the profile context approach, the context request blocking probability slightly suffers to compensate for context request dropping probability. Such an approach, coupled with a queue, results in low context request dropping probability for the prioritized context requests, and a low context request blocking probability for the remaining context requests. Figure 20 exhibits the performance of device context approach and profile context approach in regard to context request blocking probability.

The context node utilization comparison is shown in Figure 21. The utilization of context nodes is much more in profile context approach than in comparison to device context approach because the context nodes in profile context nodes accumulate more than one context types which allow different context requests to be fulfilled whenever there is a context node available despite it being a different context type provider.

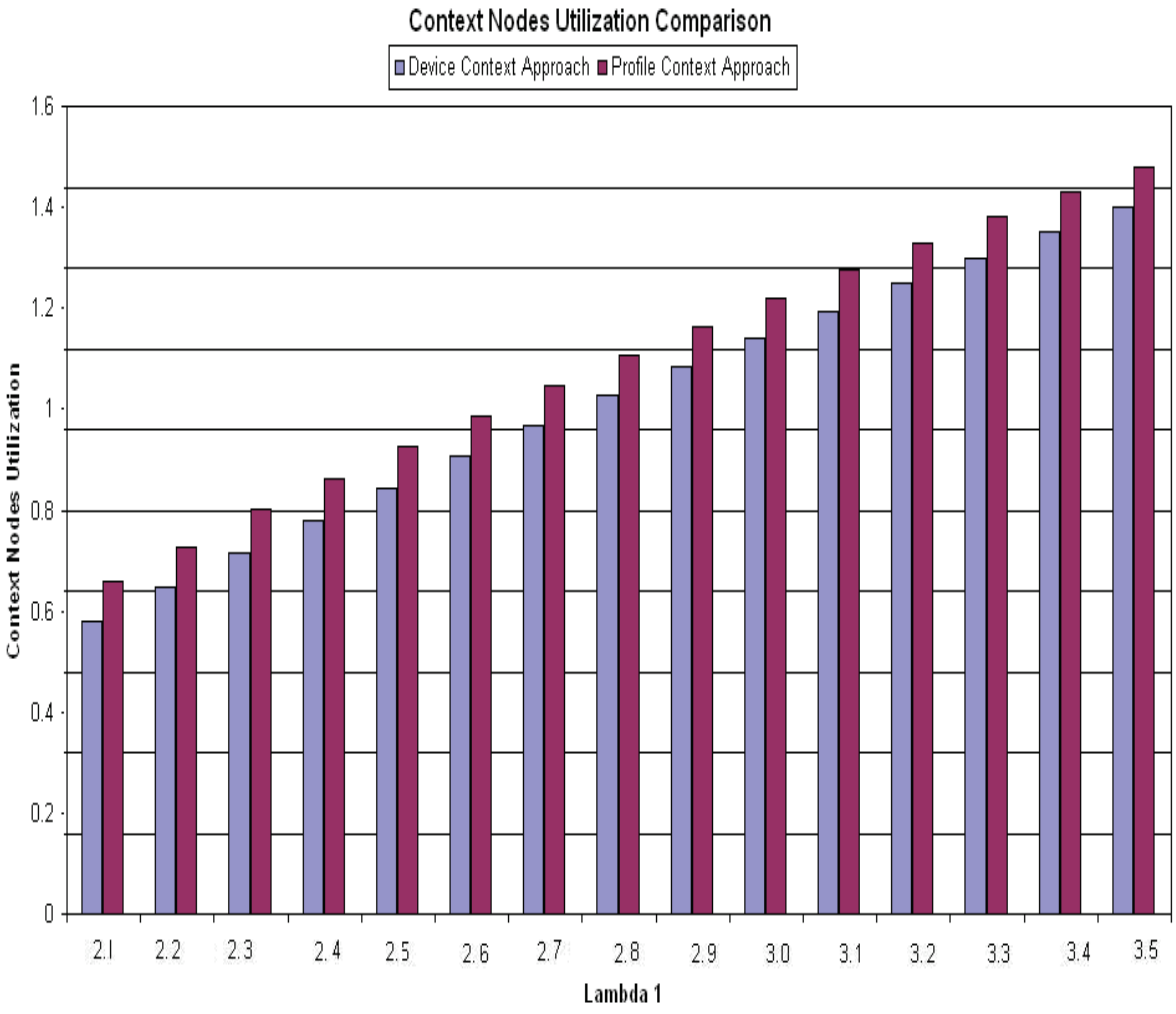

Figure 21 Context Nodes Utilization Comparison – Device and Profile Context Approaches

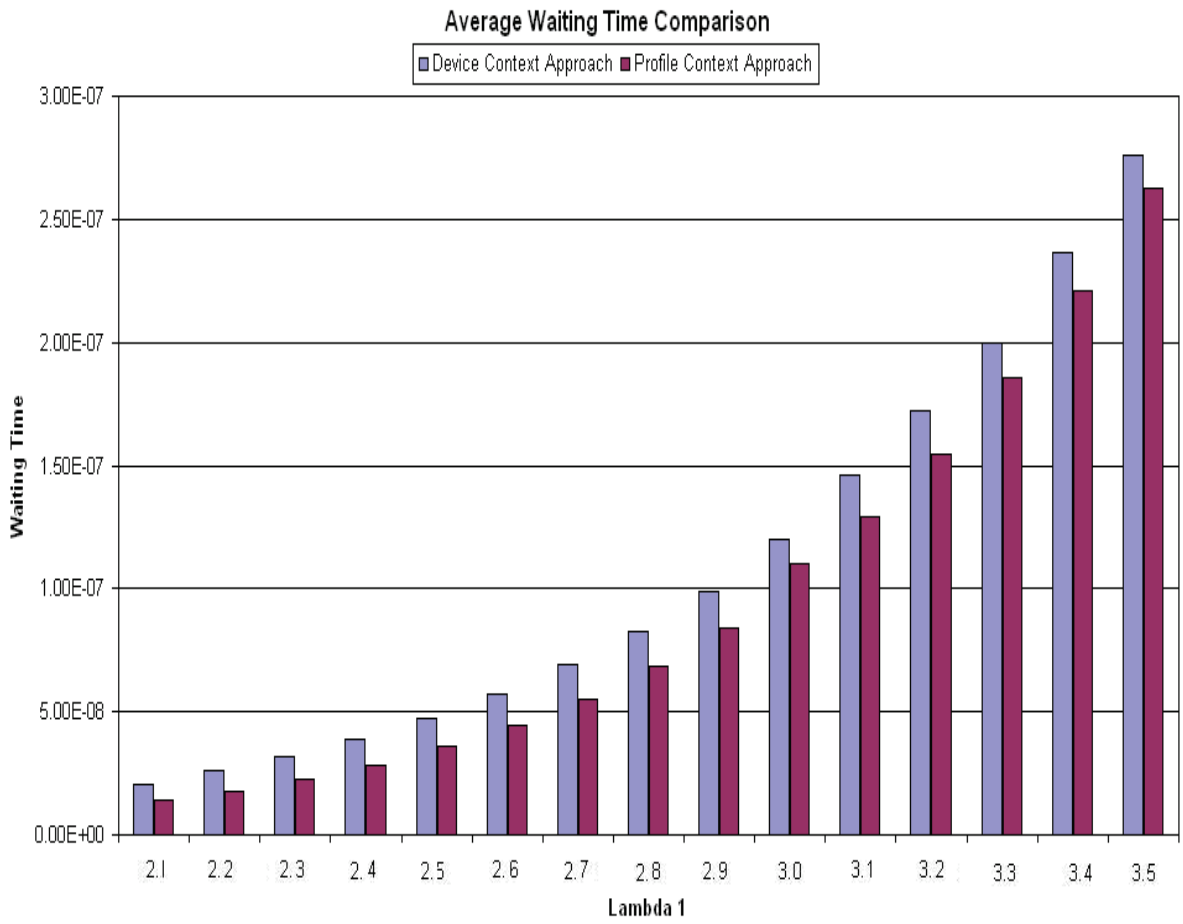

Figure 22 Average Waiting Time Comparison – Device and Profile Context Approaches

The context requests average waiting time comparison is shown in Figure 22. The average waiting time of a context request is less in profile context approach than comparatively to waiting times in device context approach because, as mentioned earlier, the context nodes accumulate more than one context types which allow different context requests to be fulfilled whenever there is a context node available despite it being a different context type provider. So, the context request can get served by any context node in profile context approach while the context request has to wait for a specific context node to be free to acquire the particular context.

7.2 Lessons Learnt

The analysis of the mathematical model and simulation results show that profile context approach exceeds the performance of device context approach. If we consider the mean queue length comparison, the advantage of profile context is due to the multiple requests being answered in the same reply back to the context requesting application as aggregated context instead of device context. The provision of aggregated context in reply to context request prevent the application to ask for device context individually, thus reducing the number of requests, which in turns reduce the number of requests that have to be queued up to get processed at one of the super nodes in ad hoc overlay network. Similarly, in context request blocking probability comparison, the context requests don't have to wait long in the queue in profile context approach, and as the requests get serviced quickly, the new context requests on arrival at one of the super node always get a place in the queue without facing the prospect of drop out, more common in device context approach. The average waiting time for context requests is slightly lower in profile context as comparison to device context as there are multiple nodes that can provide the profile context whereas only a single node can provide the device context, thus increasing the average waiting time for the requests in device context approach. The context node utilization comparison shows that the utilization of context nodes is more in profile context approach than device context approach as the context nodes in profile context nodes accumulate more than one context types which allow different context requests to be fulfilled. So, the overall analysis of the results clearly highlights the advantages of profile context in terms of various parameters over device context.

In the final chapter, we discuss the various challenges that we still need to address. We present the future direction and the steps that we intend to take in the coming months, and finally conclude the thesis.

8 Conclusion

8.1 Discussion

The proposed context management framework is intended as an application of the concept of profile context, and its acquisition and distribution in an overlay ad hoc network. As can be imagined, the concept currently lacks solutions to various obstacles that are to be faced in a practical implementation. One of the most prominent features of any context management system is the provision of adequate user privacy and security of user's data, and we concede that our approach lack such assurances at the moment. The question of how to make sure that only the relevant information from the profile context is disclosed to the service or application remains to be addressed. If an application is only authorized to access a single device context, there should be some mechanism to ensure that only that information is disclosed to the application, not the whole context which comprises information from more than one device.

The issue of scalability is another hurdle that needs to be dealt with. The information overflow due to high churn rate can clog up the performance of the overlay network. In my opinion, the proposed framework is more than capable to handle context information with in a considerable community of users. Stress testing of SCOPE still needs to be done to determine its failure point in terms of the number of users in the overlay network. The increasing wireless network bandwidth availability and the idea of cloud computing can combine to provide a practical solution to the problem of scalability in the near future. Convergence of flexibility of Web 2.0 and the reliability of network infrastructure can result in an interesting future for context management approaches in terms of cloud computing.

The distributed nature of overlay networks provide improved robustness in terms of failures by making the availability of context information possible over multiple peer nodes. It provides the advantage of low latency and high resilience in regards to fault tolerance which is an obvious advantage.

The goal is to propose an open platform for context management in a cloud environment, and overlay networks have the advantage that permits both application users and developers to design and implement their own context aware services. The end nodes can always communicate with one another through overlay network, so robustness in overlay networks is another attractive feature for the proposed system. The high connectivity of end nodes allows effective sharing of context information in a distributed environment which usually lacks centralized authority.

The proposed framework, as mentioned earlier, is a workable solution that already shows with the help of results that it can provide sufficient gains in terms of reducing the additional network load commonly associated with other client/server approaches. Further detailed study is essentially required to assess the viability of the system in a more common, real life situation with the help of a working prototype.

8.2 Future Work

For the near future, we intend to propose a workable solution to the privacy issue. One approach can be directed in a way that an unauthorized application cannot access the context information, through the introduction of authentication dialogue between the owner of the context and one of the super nodes. This is a very crude solution, as it results in extra information flow that we are trying to reduce in the first place. Introducing some privacy features in XML can be a feasible solution.

The context acquisition system proposed in this thesis is to be maintained on a P2P service platform called SCOPE that is being developed and improved by the RS2M team at Institute Telecom SudParis. Its aim is to provide a distributed infrastructure with unified API that enables development of miscellaneous P2P services on ad-hoc networks. The main motivation is to provide an efficient overlay for P2P services over ad-hoc networks presented by Mani (2009). The services framework is based on Bamboo open-sources which provide PASTRY

DHT services, as discussed by Mani (2007). Bamboo with its public name OpenDHT has been deployed on Planetlab. Bamboo has also been developed to handle high churn as mentioned by Rhea (2004). Churn rate means the continuous process of node arrival and departure in an ad hoc network. It also uses proximity neighbour selection, according to Rhea (2005).

Further testing in real environment is on our agenda, and I hope that the approach will be able to perform at least as better as the current results have indicated. The evaluation will be done with data-rich services on low wireless bandwidth, as is usually the case in ad hoc networks. Enhancement of formal model of context dependency will also be aimed in near future.

The future work that we envision in this dimension involves further refinement of the current model to better understand the effects of direct and indirect context dependencies. The important work will be to evaluate the systems with high data coupling and direct dependencies in comparison to low data coupling and indirect dependencies. Not only there are data dependencies in a pervasive system, but we do need to consider enhancements to the model that can represent process dependencies as well. Concerning user preferences, the future model also needs to be flexible enough to incorporate context dependencies that are not performance based only.

Further, we plan enhancing the algorithm further by introducing dependency confirmation logic. We further intend to undertake intensive prototype implementation to provide more validation of our work.

8.3 Conclusion

The thesis provides a dependency analysis approach based on constraint satisfaction problem for the context aware applications and services. Context aware services are usually developed using context models and concepts that are informal, lack any clarity, mostly aimed

at a particular application domain. The dependency analysis may reveal a lot about dependencies among such services, and for each dependency some possible dependency reduction strategy may exist that need to be utilized. But the question is how to determine and manage all these dependencies, as new dependencies may get introduced due to reduction of a previously existing dependency. As a consequence, some dependency reduction approaches will introduce new dependencies, and some dependency reduction strategies will introduce new possibilities to resolve certain context dependencies. The issue of constraint solving to determine service dependency has got its complexity reduced from NP-complete to polynomial time complexity through our approach. Further, the thesis provides a formal model for the context dependent ubiquitous systems.

In the last part, the thesis presents a context management framework that is based on the idea of leveraging overlay network to efficiently manage profile context, making it available at all times in the system, and in turn address the issue of context dependency. The approach utilises overlay network paradigm in an innovative and unique way to optimize the distribution and aggregation of users' context information. Until now, context management systems have presented various solutions that are mostly based on query-based centralised architecture, but in this approach the user context is stored and distributed using an overlay network. Availability of context due to overlay network reduces the risk of performance bottlenecks introduced due to context dependency as in the approach, multiple context sources are replicated in the environment reducing the dependency on a single context resource. The simulation results and queuing model analysis have shown that the proposed context management approach performs better in terms of different criteria. The proposed context management framework is particularly aimed at devices and applications in a pervasive environment where lack of resources restricts optimal performance, and the results have supported the approach in this regard.

The work presented in the thesis is important in terms of finding solutions to problems that hinder the widespread adoption of pervasive systems by main-stream application developers. Finding the dependencies can be resource exhausting procedure and the ad hoc nature of the mobile network make the procedure expensive in terms of resources and time for mobile devices. The two approaches presented in this thesis aim to reduce the time and complexity of finding dependencies among different mobile devices and applications.

So far, there hasn't been a formal model that was developed with the sole purpose of handling context dependencies in general, and specially, the transitive dependencies in ubiquitous computing. The formal model, based on predicate logic and set theory, presented in the thesis is makes the process of formalizing context dependencies quite easy. The intuitive approach that is adopted in the model makes analysis of context dependencies easy to achieve in terms of context types and context sources, along with the provision of alternate context sources that can be used by a context consuming application.

The idea of Profile Context can be potentially the next big idea that social networks and cloud computing applications may adopt in the near future. Asking for context from each, individual device is not efficient in terms of network traffic, as shown by the analysis of various parameters in the evaluation part. Asking for device context also results in wastage of resources for mobile devices. The Profile Context gives comprehensive context about the user rather than each individual context. This might help with the faster development of context aware applications in the near future as the context consuming application won't have to process the device context to understand the context of the user.

Appendix: Mathematical Model

Dedicated servers help to improve the probability of successful service provision by reserving a fixed or dynamically adjustable number of servers exclusively for specific type of requests. N servers from the total of C servers are reserved for that certain type of service request. The rest (C-N) servers are jointly shared among all types of service requests. If certain requests of a context type are not many and the other context requests are increasing, the fixed server assignment scheme can hurt the efficiency of the system. To combat this inefficiency, some algorithms work on the idea of dynamic server allocation schemes. It requires determining the optimum number of dedicated servers based on the knowledge of the service request pattern of the system, and estimation of certain other factors.

In the partial sharing scheme, the servers are divided into distinct groups with each group corresponding to a particular service request, along with the pool of shared servers.

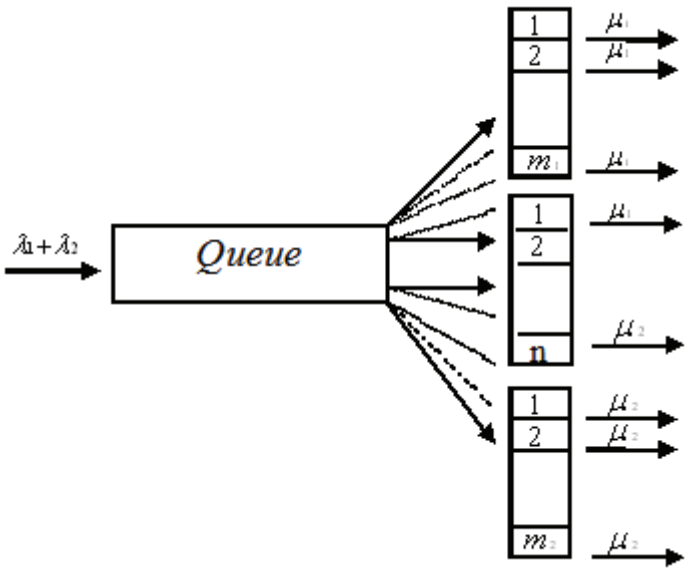

Figure 23 System Model – Partial Sharing Scheme

Only guarded servers in a scheme is inefficient in use of resources if the predicted service requests for a particular context type is greater than the actual service request demand. It has been shown that a partial sharing scheme, comprising of both shared and guard servers,

provides better performance than both complete sharing and complete partitioning of servers, over a range of offered loads in various service environments.

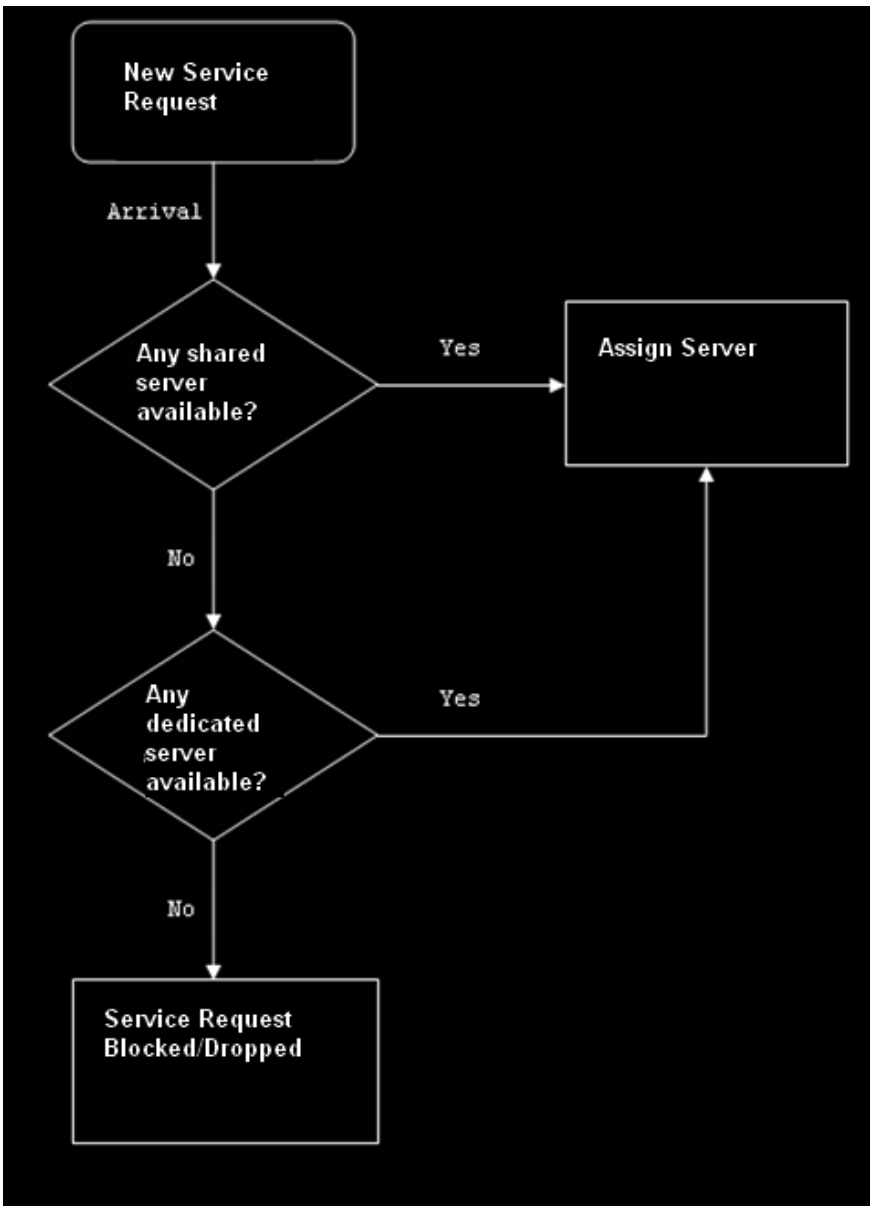

Figure 24 Service Processing Flow Diagram – Partial Server Sharing Scheme

Markov Model & its Solution:

The Markov model employed in this section is based on the combination of shared servers and dedicated servers. There is some percentage of total servers which is reserved for servicing requests belonging to specific types of classes.

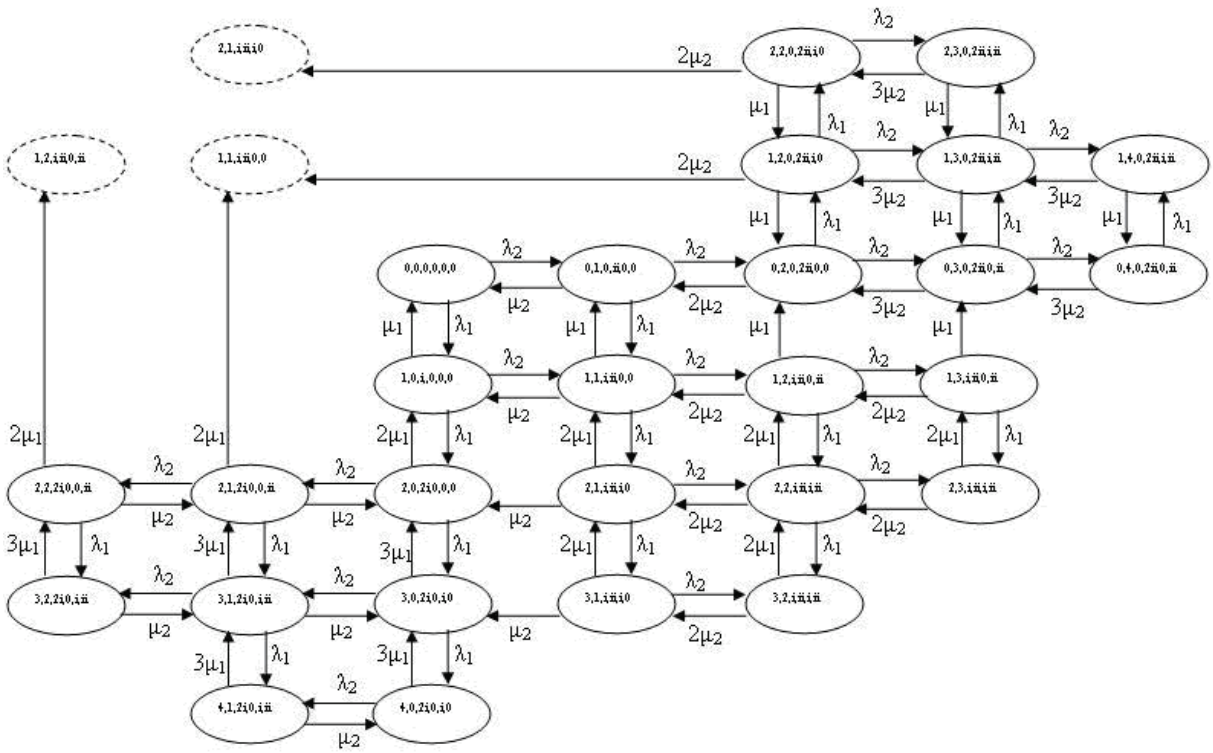

Figure 25 Markov Chain of 2 Shared Servers, 1 Guard Server Each, Queue Size 1 – Complete Sharing Scheme

Each class has to compete for a server from a specific number of shared servers available in the system. If all the shared servers are busy, then an idle server from reserved pool for that class is provided to meet the server request.

This model is general and can be applied for any queue size and number of classes and their respective servers. The particular example considered has queue size, Q = 1, and the number of shared servers for among the two types of classes is 2, i.e. S_T = 2. The number of dedicated servers for each class in this model is set to one, i.e. TG_i = 1 and TG_{ii} = 1. Service requests from both are assumed to be Poisson processes, with rate λ_1 and λ_2, respectively. The server holding time is assumed to be exponentially distributed for both classes; $1/\mu_1$ for class 1 arrivals and $1/\mu_2$ for class 2 arrivals. A generalized Markov model can be described by a two dimensional Markov Chain with state (n_i,n_{ii},S_i,S_{ii},G_i,G_{ii}), where n_i and n_{ii} are the total numbers of service requests present in the system belonging to different classes. While S_i and S_{ii} are the

numbers of servers occupied by requests from each of the class. Similarly, G_i and G_{ii} represents the number of guarded servers which are being used by their respective classes.

The approach for this particular Markov model assumes that the request arrival processes for the two classes are completely independent Poisson processes, and there is no correlation among these two processes.

Considering the approach of the two Markov models discussed before, it can be safely assumed that between any two adjacent states there is complete "flow in, flow out" equilibrium without any effect on the other remaining neighboring states.

For the state occupancy probabilities $P(n_i,n_{ii},S_i,S_{ii},G_i,G_{ii})$, where $n_i + n_{ii} < S_T + G_i + G_{ii} + Q + 1$, the general equilibrium equation can be written as

$$P(n_i,n_{ii},S_i,S_{ii},G_i,G_{ii}) = \frac{P(0,0,0,0,0,0)\,(\lambda_i)^{n_i}\,(\lambda_{ii})^{n_{ii}}\,M_i\,M_{ii}}{(G_i+S_i)!\,(G_i+S_i)^{n_i-S_i-G_i}\,(G_{ii}+S_{ii})!\,(G_{ii}+S_{ii})^{n_{ii}-S_{ii}-G_{ii}}}$$

Where,

$M_i = \frac{1}{(\mu_i)^{n_i}}$ if $G_i + S_i > 0$, else 1 and

$M_{ii} = \frac{1}{(\mu_{ii})^{n_{ii}}}$ if $G_{ii} + S_{ii} > 0$, else 1

$(G_i+S_i)^{n_i-S_i-G_i}$ is 1 if $G_i+S_i = 0$

$(G_{ii}+S_{ii})^{n_{ii}-S_{ii}-G_{ii}}$ is 1 if $G_{ii}+S_{ii} = 0$

Using the general rule that the sum of all probabilities is 1, i.e. $\sum_{i=0}^{C} P(i) = 1$, $P(0,0,0,0,0,0)$ can be expressed as,

$$P(0,0,0,0,0,0) = \left[1 + \sum_{i=1}^{C}\left(\frac{(\lambda_i)^{n_i}\,(\lambda_{ii})^{n_{ii}}\,M_i\,M_{ii}}{(G_i+S_i)!\,(G_i+S_i)^{n_i-S_i-G_i}\,(G_{ii}+S_{ii})!\,(G_{ii}+S_{ii})^{n_{ii}-S_{ii}-G_{ii}}}\right)\right]^{-1}$$

where $C < S_T + G_i + G_{ii} + Q + 1$.

Proof by Induction

The Proposition is:

$$P(n_i, n_{ii}, S_i, S_{ii}, G_i, G_{ii}) = \frac{P(0,0,0,0,0,0)\,(\lambda_i)^{n_i}(\lambda_{ii})^{n_{ii}}}{(\mu_i)^{n_i}(G_i+S_i)!\,(G_i+S_i)^{n_i-S_i-G_i}(\mu_{ii})^{n_{ii}}(G_{ii}+S_{ii})!\,(G_{ii}+S_{ii})^{n_{ii}-S_{ii}-G_{ii}}}$$

is true for all n where $n_i + n_{ii} < S_T + G_i + G_{ii} + Q + 1$

The Base Case is:

The base case for n = 0, can easily be proved.

$$P(0,0,0,0,0,0) = \frac{P(0,0,0,0,0,0)\,(\lambda_i)^0(\lambda_{ii})^0}{(\mu_i)^0(0)!\,(\mu_{ii})^0(0)!}$$

$$\rightarrow P(0,0,0,0,0,0) = \frac{P(0,0,0,0,0,0)\;\; 1 \times 1}{1 \times 1 \times 1 \times 1}$$

$$\rightarrow \;\; P(0,0,0,0,0,0) \;\; = \;\; P(0,0,0,0,0,0)$$

Inductive Step is:

The inductive part is for states $P(k_1+1,k_2+1,S_i,S_{ii},G_i,G_{ii})$. The inductive parts for the remaining states, $P(k_1+1,k_2,S_i,S_{ii},G_i,G_{ii})$ and $P(k_1,k_2+1,S_i,S_{ii},G_i,G_{ii})$, can be proved similarly to Guarded Server Scheme. Assuming that the derivation is true for n = k.

$$P(k_i, k_{ii}, S_i, S_{ii}, G_i, G_{ii}) = \frac{P(0,0,0,0,0,0)\,(\lambda_i)^{k_i}(\lambda_{ii})^{k_{ii}}}{(\mu_i)^{k_i}(G_i+S_i)!\,(G_i+S_i)^{k_i-S_i-G_i}(\mu_{ii})^{k_{ii}}(G_{ii}+S_{ii})!\,(G_{ii}+S_{ii})^{k_{ii}-S_{ii}-G_{ii}}}$$

Now, the proof will be devised for n = k + 1.

$$P(k_i+1, k_{ii}+1, S_i+1, S_{ii}+1, G_i+1, G_{ii}+1) =$$

$$\frac{P(0,0,0,0,0,0)\,(\lambda_i)^{k_i}\,(\lambda_{ii})^{k_{ii}}}{(\mu_i)^{k_i}\,(G_i+S_i)!\,(G_i+S_i)^{k_i-S_i-G_i}\,(\mu_{ii})^{k_{ii}}\,(G_{ii}+S_{ii})!\,(G_{ii}+S_{ii})^{k_{ii}-S_{ii}-G_{ii}}} \times$$

$$\frac{(\lambda_i)\,(\lambda_{ii})}{(\mu_i)\,(G_i+S_i+1)\,(G_i+S_i)\,(\mu_{ii})\,(G_{ii}+S_{ii}+1)\,(G_{ii}+S_{ii})}$$

$$\rightarrow P(k_i+1, k_{ii}+1, S_i+1, S_{ii}+1, G_i+1, G_{ii}+1) =$$

$$\frac{P(0,0,0,0,0,0)\,(\lambda_i)^{k_i+1}\,(\lambda_{ii})^{k_{ii}+1}}{(\mu_i)^{k_i+1}\,(G_i+S_i+1)!\,(G_i+S_i)^{k_i-S_i-G_i+1}\,(\mu_{ii})^{k_{ii}+1}\,(G_{ii}+S_{ii}+1)!\,(G_{ii}+S_{ii})^{k_{ii}-S_{ii}-G_{ii}+1}}$$

Hence proved that the derivation for this Markov model is true for all n where $n_i + n_{ii} < S_T + G_i + G_{ii} + Q + 1$.

References:

[Abdelzaher, 2007] Abdelzaher, T., Anokwa, Y., Boda, P., Burke, J., Estrin, D., Guibas, L., Kansal, A., Madden, S. and Reich, J. "Mobiscopes for Human Spaces," in *IEEE Pervasive Computing*, 6(2):20–29, 2007.

[Agrawal, 1993] Agrawal, H., Demillo, R. and Spaord, E. "Debugging with Dynamic Slicing and backtracking," in *Software Practice and Experience*, vol.23, No.6, pp.589-616, 1993.

[Akman, 1997] Akman, V., and Surav, M. "The use of situation theory in context modelling," in *Computational Intelligence* 13, 3 (1997), 427–438, 1997.

[Ameller, 2008] Ameller, D. and Franch, X. "Service level agreement monitor (salmon)," in *Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008)*, pp. 224–227, Washington, DC, USA. 2008.

[Anagnostopoulos, 2007] Anagnostopoulos, C., Tsounis, A., and Hadjiefthymiades, S. "Context awareness in mobile computing environments," in *Wireless Personal Communications*, 42(3):445-464, 2007

[Aspvall, 1979] Aspvall, B., Plass, M., Tarjan, R. "A linear-time algorithm for testing the truth of certain quantified boolean formulas," in Information Processing Letters, 8 (3): 121–123, 1979.

[Bacchus, 1998] Bacchus, F. and van Beek, P. "On the conversion between non-binary and binary constraint satisfaction problems," In *Proc. of 15th AAAI Conference on Artificial Intelligence*, 1998.

[Bardram, 2003] Bardram, J. and Kjær, R. "Context-Aware User Authentication Supporting Proximity-Based Login in Pervasive Computing," in *UbiComp: Ubiquitous Computing*, pp. 107–123, 2003.

[Baloch, 2010] Baloch, R.A., Awan, I. and Min, G. "A mathematical model for wireless channel allocation and handoff schemes," in *Special Issue on Performance Modelling and Evaluation of Telecommunication Systems, Telecommunication Systems Journal*, 2010.

[Barbara, 1999] Barbara, D. "Mobile computing and databases-a survey," in IEEE Transactions on Knowledge and Data Engineering, 11(1):108–117, 1999.

[Barwise, 1983] Barwise, J., and Perry, J. "Situations and Attitudes," MIT Press, 1983

[Bates, 1993] Bates, S., and Horwitz, S. "Incremental Program Testing Using Program Dependence Graphs," in Conf. Record of the 20th Annual ACM SIGPLAN-SIGACT, Sym. of principles of Programming Languages, pp.384-396, ACM Press, 1993.

[Batini, 1986] Batini, C., Lenzerini, M. and Navathe, S. B. "A comparative analysis of methodologies for database schema integration," ACM Computer Survey, 18(4):323.364, 1986.

[Batini, 1992] Batini, C., Ceri, S. and Navathe, S. “Conceptual database design: an Entity-relationship approach,” Benjamin-Cummings Publishing Co. Inc., 1992.

[Bauer, 2003] Bauer, J. “Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic,” Diplomarbeit, March 2003.

[Bawa, 2003] Bawa, M., Manku, G. and Raghavan, P. “SETS: Search Enhanced by Topic Segmentation,” in *Proc. 26th Annual ACM Conference on Research and Development in Information Retrieval*, pp. 306–313, 2003.

[Becker, 2005] Becker, C. and Dürr, F. “On location models for ubiquitous computing,” in Personal and Ubiquitous Computing, 9: 2031, 2005.

[Bergamaschi, 2001] Bergamaschi, S., Castano, S., Vincini, M. and Beneventano, D. “Semantic integration of heterogeneous information sources,” in *Data Knowledge Engineering*, 36(3):215.249, 2001.

[Black, 2001] Black, S. “Computing ripple effect for software maintenance,” in *Journal of Software Maintenance and Evolution: Research and Practice*, 13:263–279, 2001.

[Blair, 1998] Blair, G. and Stefani, J. “Open Distributed Processing and Multimedia,” Addison-Wesley, 1998.

[Bodenstaff, 2008] Bodenstaff, L., Wombacher, A., Reichert, M., and Jaeger, M. C. "Monitoring dependencies for SLAs: The mode4sla approach," in IEEE SCC (1). IEEE Computer Society, 2008.

[Bonnet, 2001] Bonnet, P., Gehrke, J., and Seshadri, P. "Towards sensor database systems," in Mobile Data Management, pages 3–14, 2001.

[Booch, 1998] Booch, G., Jacobson, I., and Rumbaugh, J. "The Unified Modeling Language User Guide," Addison-Wesley, 1998.

[Bouzy, 1997] Bouzy, B., and Cazenave, T. "Using the Object Oriented Paradigm to Model Context in Computer Go," in *Proceedings of Context'97*, Rio, Brazil, 1997.

[Myers, 1998] Myers, B. "A Brief History of Human Computer Interaction Technology," in *ACM Interactions*, 5(2):44–54, 1998.

[Brown, 1997] Brown, P. J., Bovey, J. D., and Chen, X. "Context-aware Applications: from the Laboratory to the Marketplace," in IEEE Personal Communications 4, 5, 58–64, 1997.

[Briand, 1998] Briand, L. C., Wust, J., and Lounis, H. "Using Coupling Measurement for Impact Analysis in Object Oriented Systems," at *IEEE International Conference on Software Maintenance (ICSM98)*, Bethesda, MD, 1998.

[Capra, 2001] Capra, L., Emmerich, W. and Mascolo, C. "Reflective Middleware Solutions for Context-Aware Application ," in *3rd International Conference on Meta-level*

Architectures and Separation of Crosscutting Concerns (Reflection 01), LNCS, September 2001.

[Chalmers, 2006] Chalmers, D., Chalmers, M., Crowcroft, J., Kwiatkowska, M. Milner, R., O'Neill, E., Rodden, T., Sassone, V., and Sloman, M. "Ubiquitous Computing Grand Challenge: Manifesto," http://www-dse.doc.ic.ac.uk/Projects/UbiNet/GC/Manifesto/manifesto.pdf; version 4, 23-2-06; accessed September 10, 2011.

[Chawathe, 1994] Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. and Widom, J. "The TSIMMIS project: Integration of heterogeneous information sources," in *16th Meeting of the Information Processing Society of Japan*, pp. 7.18, Tokyo, Japan, 1994.

[Chen, 2000] Chen, G. & Kotz, D. "A Survey of Context Aware Mobile Computing Research," Dartmouth Computer Science Technical Report TR2000381, 2000.

[Chen, 2003] Chen, H., Finin, T., and Joshi, A. "Using OWL in a Pervasive Computing Broker," in *Proceedings of Workshop on Ontologies in Open Agent Systems (AAMAS 2003)*, 2003.

[Chen, 1976] Chen, P. "The entity-relationship model - toward a unified view of data," in ACM Transaction on Database Systems 1, 1, 9–36, March, 1976.

[Cheverst, 1999] Cheverst, K., Mitchell, K., and Davies, N. “Design of an object model for a context sensitive tourist GUIDE,” in Computers and Graphics 23, 6 (1999), 883–891, 1999.

[Chetcherbina, 2003] Chetcherbina, E., and Franz, M. “Peer-to-peer coordination framework (p2pc): Enabler of mobile ad-hoc networking for medicine, business, and entertainment,” in *Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet* (SSGRR2003w), L’Aquila, Italy, 2003.

[Clarke, 2001] Clarke, E., Grumberg, O., Jha, S., Lu, Y., and Veith, H. “Progress on the state explosion problem in model checking,” in Informatics, pp. 176–194, 2001.

[Coutaz, 2005] Coutaz, J., Crowley, J., Dobson, S., and Garlan, D. “Context is key,” in Communications of the ACM, 48(3), 2005.

[Cox, 2001] Cox, L., Delugach, H., Skipper, D. “Dependency analysis using conceptual graphs,” in *Proceedings of the 9th International Conference on Conceptual Structures*, ICCS, 2001.

[Crestana-Jensen, 2000] Crestana-Jensen, V. M. and Lee, A. J. “Consistent Schema Version Removal: An Optimization Technique for Object Oriented Views,” in *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, pp. 261-280, 2000.

[Crossbow, 2008] Crossbow Technology. Crossbow, 2008. http://www.xbow.com/; accessed October 20, 2011.

[De Bruijn, 2003] De Bruijn, J. « Using Ontologies – Enabling Knowledge Sharing and Reuse on the Semantic Web," Technical Report DERI-2003-10-29, Digital Enterprise Research Institute (DERI), Austria, 2003.

[De Lucia, 1996] De Lucia, A., Fasolino, A. R., and Munro, M. "Understanding function behaviors through program slicing," in *Proceedings of the Fourth Workshop on Program Comprehension*, Berlin, Germany, 1996.

[Desmet, 2007] Desmet, B., Vallejos, J., Costanza, P., and Meuter, W. D. "Context-oriented domain analysis," Lecture Notes in Computer Science, 2007.

[Dey, 1999] Dey, A. & Abowd, G. "Towards a Better Understanding of Context and Context-Awareness," in *CHI '00: Workshop on the What, Who, Where, When, and How of Context-Awareness*, Georgia Institute of Technology, Atlanta, GA, USA, 1999.

[Dey, 2000] Dey, A. K. "Providing Architectural Support for Building Context-Aware Applications," PhD thesis, College of Computing, Georgia Institute of Technology, 2000.

[Dey, 2004] Dey, Anind K., Raffay Hamid, Chris Beckmann, Ian Li & Daniel Hsu. "CAPpella: Programming by Demonstration of Context Aware Applications," in CHI Letters, 2004.

[Dey, 2002] Dey, A. K., Mankoff, J., Abowd, G. D. and Carter, S. "Distributed Mediation of Ambiguous Context in Aware Environments," in *Proceedings of the 15th Annual Symposium on User Interface Software and Technology (UIST)*, Paris, France, pp. 121–130, 2002.

[Dick, 1969] Dick, P. *Ubik*. Gollancz S.F., 1969.

[Dust, 2008] Dust Networks. Dust Networks : Embedded Wireless Sensor Networking for Monitoring and Control, 2008. http://www.dustnetworks.com/index.shtml; accessed October 21, 2011.

[Ebling, 2001] Ebling, M., Hunt. G., and Lei, H., "Issues for Context Services for Pervasive Computing," in Workshop Middleware for Mobile Computing, Heidelberg, Germany, 2001.

[Eckhardt, 1985] Eckhardt, D. E. and Lee, L. D. "A theoretical basis for the analysis of redundant software subject to coincident errors," NASA tech. Memo 86369, 1985.

[Elahi, 2009] Elahi, B., Romer, K., Ostermaier, B., Fahrmair, M. and Kellerer, W. "Sensor ranking: A primitive for efficient content-based sensor search," in *Proceedings of International Conference on Information Processing in Sensor Networks*, IPSN 2009, pages 217–228, 2009.

[TEA, 1998] Esprit project 26900. Technology for enabled awareness (tea), 1998.

[Feder, 1994] Feder, T., "Network flow and 2-satisfiability," In *Algorithmica*, vol. 11, no. 3, pp. 291–319, 1994.

[Flehmig, 2006] Flehmig, M., Troeger, P., and Saar, A. “Design and integration of SLA monitoring and negotiation capabilities,” in Adaptive Services Grid Deliverable D5.II-7, 2006.

[Floreen, 2005] Floreen, P., Przybilski, M., Nurmi, P., Koolwaaij, J., Tarlano, A., Wagner, M., Luther, M., Bataille, F., Boussard, M., Mrohs, B. and Others. “Towards a context management framework for MobiLife,” in *Proc. 14th IST Mobile & Wireless Summit*, 2005.

[Fowler, 2001] Fowler, M. “Reducing coupling,” in IEEE Software, 18(4), 102-104, 2001.

[Fu, 2001] Fu, X., Shi, W., Akkerman, A. and Karamcheti. V. “CANS: Composable, adaptive network services infrastructure,” in USITS 2001, San Francisco, California, 2001.

[Gallagher, 1991] Gallagher, K. B., and Lyle, J. R. “Using Program Slicing in Software Maintenance,” in IEEE Transaction on Software Engineering, vol.17, No.8, pp.751-761, 1991.

[Gokhale, 1998] Gokhale, S., Lyu, M. and Trivedi, K. “Reliability simulation of component-based software systems,” in *Proceedings of the Ninth International Symposium on Software Reliability Engineering* (ISSRE’98), 192–201, 1998.

[Goseva-Popstojanova, 2001] Goseva-Popstojanova, K. and Trivedi, K. S. “How different architecture based software reliability models are related?” in *Performance Evaluation* 45(2-3), 179–204, 2001.

[Gray, 2001] Gray, P., Salber, D., "Modelling and using sensed context in the design of interactive applications," in 8th IFIP Conference on Engineering for Human-Computer Interaction, Toronto, 2001.

[Greenfield, 2006] Greenfield, A. "Everyware: The Dawning Age of Ubiquitous Computing," New Riders Publishing, 2006.

[Gruber, 1993] Gruber, T. G. "A translation approach to portable ontologies," in Knowledge Acquisition 5, 2, 199–220, 1993.

[Gu, 2005] Gu, T., Pung, H.K. and Zhang, D.Q. "A service-oriented middleware for building context-aware services," in *Journal of Network and Computer Applications*, 28(1):1–18, 2005.

[Gwizdka, 2000] Gwizdka, J. "What's in the Context," in CHI2000 Workshop, 2000.

[Hadim, 2006] Hadim, S. and Mohamed, N. "Middleware: Middleware challenges and approaches for wireless sensor networks," IEEE Distributed Systems Online,7(3):1–1, 2006.

[Hall, 1997] Hall, R. S., Heimbigner, D., and Wolf, A. L. "Software deployment languages and schema," Dept. of Computer Science, University of Colorado CU-SERL-203-97, 1997.

[Halpin, 2001] Halpin, T. A. "Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design," Morgan Kaufman Publishers, San Francisco, 2001.

[Hamlet, 2001] Hamlet, D., Mason, D., and Woit, D. “Theory of software reliability based on components,” in International Conference on Software Engineering 23, 361–370, 2001.

[Harter, 1999] Harter, A., Steggles, P., Ward, A., and Webster, P., “The Anatomy of a Context-Aware Application,” in *Proceedings of Conference on Mobile Computing and Networking (MOBICOM)*, 1999.

[Hayes, 1994]Hayes, P. “Aristotelian and Platonic Views of Knowledge Representations,” in Second International Conference on Conceptual Structures (ICCS94), College Park, MD, 1994.

[Held, 2002] Held, A., Buchholz, S., & Schill, A. “Modeling of Context Information for Pervasive Computing Applications,” in Sixth World Multiconference on Systemics, Cybernetics and Informatics, SCI2002, Orlando, 2002.

[Henricksen, 2001] Henricksen, K., Indulska, J., Rakotonirainy, A. “Infrastructure for pervasive computing: Challenges,” in Informatik 2001: Workshop on Pervasive Computing, Vienna, 2001.

[Henricksen, 2002] Henricksen, K., Indulska, J., Rakotonirainy, A. “Modeling Context Information in Pervasive Computing Systems,” in *Proceedings of Pervasive*, Zurich, 2002.

[Henricksen, 2003] Henricksen, K., Indulska, J., Rakotonirainy, A. “Generating Context Management Infrastructure from High-Level Context Models,” in *Proceedings of the 4th*

International Conference on Mobile Data Management, (MDM2003), Melbourne, Australia, pp. 1–6, 2003.

[Henricksen, 2004] Henricksen, K. and Indulska, J. "A Software Engineering Framework for ContextAware Pervasive Computing," in 2nd IEEE Conference on Pervasive Computing and Communications (PerCom), Orlando, 2004.

[Hofer, 2003] Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., Altmann, J. and Retschitzegger, W. "Context-awareness on mobile devices - the hydrogen approach," in *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, vol. 9, 2003.

[Holmquist, 2001] Holmquist, L., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., and Gellersen, H.W. "Smart-its friends: A technique for users to easily establish connections between smart artefacts," in *Proceedings of Ubicomp: Ubiquitous Computing*, pp. 116–122, 2001.

[Hong, 2001] Hong, J. I. and Landay, J. A. "An infrastructure approach to context-aware computing," in Human-Computer Interaction, 16(2&3), 2001.

[Hull, 2006] Hull, B., Bychkovsky, V., Zhang, Y., Chen, K., Goraczko, M., Miu, A., Shih, E., Balakrishnan, H. and Madden, S. "CarTel: a distributed mobile sensor computing system," in *Proceedings of the 4th international conference on Embedded networked sensor systems,* pages 125–138, 2006.

[Hutchens, 1985] Hutchens, D.H., and Basili, V.R. “System Structure Analysis: Clustering with Data Bindings,” in IEEE Trans. Software Eng., vol. 11, no. 8, pp. 749-757, 1985.

[Huynh, 2007] Huynh, S., & Cai, Y. “An evolutionary approach to software modularity analysis,” in *Proceedings of the First International Workshop on Assessment of Contemporary Modularization Techniques*, Minneapolis, Minnesota, USA, 2007.

[Indulska, 2003] Indulska, J., Robinsona, R., Rakotonirainy, A., and Henricksen, K. “Experiences in using cc/pp in context-aware systems,” in *Proceedings of the 4th International Conference on Mobile Data Management,* (MDM2003), Melbourne, Australia, 2003.

[Jackson, 2004] Jackson, D. “Module dependences in software design,” in Radical Innovations of Software and Systems Engineering in the Future: 9th International Workshop, RISSEF 2002, Venice, Italy, pp. 198-203, 2004.

[Kagal, 2001] Kagal, L., Korolev, V., Chen, H., Joshi, A., and Finin, T. “Centaurus: A framework for Intelligent Services in a Mobile Environment,” in *Proc. Intl. Workshop on Smart Appliances and Wearable Computing, Workshop at 21st IEEE Intl. Conf. Distributed Computing Systems*, Phoenix, 2001.

[Kang, 1990] Kang, K., Cohen, S., Hess, J., Nowak, W., and Peterson, S. “Feature-Oriented Domain Analysis (FODA) Feasibility Study,” 1990.

[Kansal, 2007] Kansal, A., Nath, S., Liu, J. and Zhao, F. “SenseWeb: An Infrastructure for Shared Sensing,” in IEEE Multimedia, 14(4):8–13, 2007.

[Keller, 2000] Keller, A., Blumenthal, U., and Kar, G. “Classification and Computation of Dependencies for Distributed Management,” in *Proceedings of the Fifth International Conference on Computers and Communications,* (ISCC 2000), 2000.

[Kiciman, 2000] Kiciman, E. and Fox, A. “Using dynamic mediation to integrate COTS entities in a ubiquitous computing environment,” in HUC 2000, Bristol, UK, 2000.

[Klemettinen, 2007] Klemettinen, M. “Enabling technologies for mobile services: the MobiLife book,” Wiley, 2007.

[Knight, 1986] Knight, J. C. and N. G. Leveson “An experimental evaluation of the assumption of independence in multiversion programming,” in IEEE Transactions on Software Engineering 12(1), 96–109, 1986.

[Krishnamurthy, 1997] Krishnamurthy, S. and Mathur, A. “On the estimation of reliability of a software system using reliabilities of its components,” in *Proceedings of the 8th International Symposium on Software Reliability Engineering,* (ISSRE’97), 146–155, 1997.

[Kuball, 1999] Kuball, S., May, J. and Hughes, G. “Building a system failure rate estimator by identifying component failure rates,” in *Proceedings of the 10th International Symposium on Software Reliability Engineering,* (ISSRE), 32–41, 1999.

[Kwon, 2002] Kwon, M. and Farmy, S., “Topology-Aware Overlay Networks for Group Communication,” in *Proc. 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 127–136, 2002.

[Kwon, 2005] Kwon, Ohbyung, Keedong Yoo and Euiho Suh. “ ubiES: An Intelligent Expert System for Proactive Services Deploying Ubiquitous Computing Technologies,” in 38th Hawaii International Conference on System Sciences, 2005.

[Landt, 2001] Landt, J. “Shrouds of Time. The history of RFID,” an AIM (Association for Automatic Identification and Data Capture Technologies) Publication, 2001.

[Lech, 2005] Lech, T. and Wienhofen, L. “ AmbieAgents: a scalable infrastructure for mobile and context-aware information services,” in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pp. 625–631, 2005.

[Lenzerini, 2002] Lenzerini, M. “Data integration: a theoretical perspective,” in *Proceedings of the 21st ACM IGMODSIGACT-SIGART Symposium on Principles of Database Systems*, pp. 233.246, ACM Press, 2002.

[Levinson, 2000] Levinson, R. and Goodwin, A. R. “Explorations in Scientific Thinking: a Systems Theoretic Approach: Chapter 2,” Scientific Thinking: a Systems Theoretic Approach. Santa Cruz, CA, p. 65, 2000.

[Levis, 2002] Levis, P. and Culler, D. “Mate: A tiny virtual machine for sensor networks,” in ACM SIGARCH Computer Architecture News, 30(5):85–95, 2002.

[Li, 2003] Li, J., Loo, B., Hellerstein, J., Kaashoek, M., Karger, D. and Morris, R. "On the feasibility of peer-to-peer web indexing and search," in Peer-to-Peer Systems II, pages 207–215, 2003.

[Lieberman, 2000] Lieberman, H. and Selker, T. "Out of context: Computer Systems that adapt to, and learn from, context," in *IBM Systems Journal*, vol. 39 No.3/4, pp. 116, 2000.

[Littlewood, 1989] Littlewood, B. and D. R. Miller "Conceptual modeling of coincident failures in multiversion software," in IEEE Transactions on Software Engineering 15(12), 1596–1614, 1989.

[Littlewood, 2000] Littlewood, B., Popov, P. and Strigini, L. "Assessing the reliability of diverse fault-tolerant systems," in *Proceedings of the INucE International Conference on Control and Instrumentation in Nuclear Installations*, 2000.

[Ludwig, 2008] Ludwig, A. and Franczyk, B. "Cosma an approach for managing slas in composite services," in Bouguettaya, A., Krueger, I., and Margaria, T., editors, ICSOC 2008, number 5364 in LNCS, page 626632. Springer-Verlag Berlin Heidelberg, 2008.

[Lyu, 1995] Lyu, M. R. (Ed.) "Handbook of Software Reliability Engineering," IEEE Computer Society Press, 1995.

[Maab, 1997] Maaß, Henning. “Location aware mobile applications based on directory services,” in Third Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 23-33, 1997.

[Mani, 2007] Mani, M., Seah, W., Crespi, N. “Efficient P2P service control overlay construction to support IP Telephony services over ad hoc networks,” in *Proceeding of IEEE MASS*, 2007.

[Mani, 2009] Mani, M., NGYUEN, A.M., Crespi, N. “What’s Up: P2P Spontaneous Social Networking” in *Proceeding of PERCOM 2009, IEEE International Conference on Pervasive Computing and Communications*, Galveston TX, USA, 2009.

[May, 2002] May, J. “Component-based software reliability analysis,” Technical report, Department of Computer Science, University of Bristol, 2002.

[McCarthy, 1993] McCarthy, J. “Notes on formalizing contexts,” in *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence,* San Mateo, California, R. Bajcsy, Ed., Morgan Kaufmann, pp. 555–560, 1993.

[McFadden, 2004] McFadden, T. and Indulska, J. “Context Aware Environments for Independent Living,” in 3rd National Conference of Emerging Researchers in Ageing (ERA), Brisbane, Australia, 2004.

[Melhorn, 1984] Melhorn, K. Graph Algorithms and NP-Completeness in Data Structures and Algorithms, Vol. II, Springer, 1984.

[Media Lab, 2011a] MIT Media Lab: Things That Think Consortium. Things That Think, 2011. http://ttt.media.mit.edu/; accessed October 12, 2011.

[Murthy, 2005] Murthy, V. and Krishnamurthy, E. "Contextual Information Management Using Contract Based Workflow," in 2nd Conference on Computing Frontiers, CF'05, Ischia, Italy, 2005.

[Nixon, 2002] Nixon, P. Wang, F., Terzis, S. and Dobson, S. "Engineering context-aware systems," in *Proceedings of the International Workshop on Engineering Context-Aware Object-Oriented Systems and Environments*, 2002.

[Norman, 1998] Norman, D. "The Invisible Computer," MIT Press, Cambridge, Massachusetts, 1998.

[O'Connell, 2005] O'Connell, P. "Korea's High-Tech Utopia, Where Everything Is Observed,". http://www.nytimes.com/2005/10/05/technology/techspecial/05oconnell.html; accessed September 28, 2011.

[Octopus, 2008] Octopus Cards Limited. (2008). Octopus. http://www.octopuscards.com/enindex.jsp; accessed September 23, 2011.

[Otzturk, 1997] Otzturk, P., and Aamodt, A. "Towards a model of context for case-based diagnostic problem solving," in *Proceedings of the Interdisciplinary Conference on Modeling and Using Context*, Rio de Janeiro, Brazil, pp. 198–208, 1997.

[Ozsu, 1991] Ozsu, M. and Valduriez, P. “Principles of distributed database systems,” Prentice-Hall, Inc, 1991.

[Papazoglou, 1997] Papazoglou, M., Delis, A., Bouguettaya, A., and Haghjoo, M. “Class Library Support for Workflow Environments and Applications,” in IEEE Transations on Computer, vol. 46, pp. 673-686, 1997.

[Pappas, 2006] Pappas, V., Massey, D. and Terzis, A. “A comparative study of the DNS design with DHT-based alternatives,” in *Proceedings of 25th IEEE International Conference on Computer Communications*, pages 1–13, 2006.

[Petrelli, 2000] Petrelli, D., Not, E., Strapparava, C., Stock, O., and Zancanaro, M. “Modeling Context is Like Taking Pictures,” in CHI2000 Workshop, 2000.

[Polastre, 2005] Polastre, J., Szewczyk, R. and Culler, D. “Telos: Enabling Ultra-Low Power Wireless Research,” in *Proceedings of Fourth International Symposium on Information Processing in Sensor Networks(IPSN),* pp. 364–369, 2005.

[Presser, 2009] Presser, M., Barnaghi, P.M., Eurich, M. and Villalonga, C. “The SENSEI project: integrating the physical world with the digital world of the network of the future,” in Communications Magazine, IEEE, 47(4):1–4, 2009.

[Prost, 2000] Prost, F. “A Static Calculus of Dependencies for the l-Cube,” in 15 th Annual IEEE Symposium on Logic in Computer Science (LICS'00), Santa Barbara, CA, 2000.

[Rahm, 2001] Rahm, E. and Bernstein, P. “A survey of approaches to automatic schema matching,” in *The VLDB Journal*, 10(4):334. 350, 2001.

[Ranganathan, 2002] Ranganathan, A., Campbell, R., Ravi, A. and Mahajan, A. “ConChat: A Context-Aware Chat Program,” in IEEE Pervasive Computing. Vol.1,Iss.3, p51 –57, 2002.

[Ratnasamy, 2001] Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S. “A Scalable Content-Addressable Network,” in *Proc. ACM SIGCOMM 2001 Conference*, pp. 161-172, 2001.

[Rhea, 2004] Rhea, S., Geels, D., Roscoe, T. and Kubiatowicz, J., “Handling Churn in a DHT,” in *Proceedings of USENIX*, 2004.

[Rhea, 2005] Rhea, S., Godfrey, B., Karp, B., Kubiatowicz, J., Ratnasamy, S., Shenker, S., Stoica, I. and Yu. H. “OpenDHT: A public DHT service and its uses,” in *Proceedings of SIGCOMM*, 2005.

[Rogers, 2006] Rogers, Y. “Moving on from Weiser’s Vision of Calm Computing: Engaging UbiComp Experiences,” In *Ubicomp*, pages 404–421, 2006.

[Ross, 2000] Ross, S. M. “Introduction to Probability Models,” 9th ed.. Academic Press, 2000.

[Rumbaugh, 1998] Rumbaugh, J., Jacobson, I., and Booch, G. “The Unified Modeling Language Reference Manual,” Addison-Wesley, 1998.

[Ryan, 1999] Ryan, N. “ConteXtML: Exchanging Contextual Information between a Mobile Client and the FieldNote Server,” Available at: http://www.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html, 1999.

[Salber, 1999] Salber, D., Dey, A. & Abowd, G. “The Context Toolkit: Aiding the Development of Context Enabled Applications,” CHI’99, Pittsburgh, PA, USA, 1999.

[Samulowitz, 2001] Samulowitz, M., Michahelles, F., and Linnhoff-Popien, C. “Capeus: An architecture for context-aware selection and execution of services,” In New Developments in Distributed Applications and Interoperable Systems, Krakow, Poland, Kluwer Academic Publishers, pp. 23–39, 2001.

[Satyanarayanan, 2001] Satyanarayanan, M. “Pervasive Computing: Vision and Challenges,” IEEE Personal Communications, 2001.

[Selby, 1991] Selby, R.W. and Basili, V.R. “Analyzing Error-Prone System Structure,” in IEEE Trans. Software Eng., vol. 17, no. 2, pp. 141-152, 1991.

[Schilit, 1994] Schilit, B. N., Adams, N. L., and Want, R. “Context-aware computing applications,” in IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, US, 1994.

[Schmidt, 2002] Schmidt, A. “Ubiquitous Computing – Computing in Context,” Ph.D. dissertation, Lancaster University, 2002.

[Schmidt, 1999] Schmidt, A., Beigl, M., and Gellersen, H. “There is more to context than location,” In Computers and Graphics 23, 6, 893–901, 1999.

[Schmidt, 2001] Schmidt, A., and Laerhoven, K. “How to Build Smart Appliances,” in IEEE Personal Communications, 2001.

[Schmohl, 2009] Schmohl, R. and Baumgarten, U. “The Contextual Map-A Context Model for Detecting Affinity between Contexts,” in Mobile Wireless Middleware, Operating Systems, and Applications, pp. 171–184, 2009.

[Shooman, 1976] Shooman, M. L. “Structural models for software reliability prediction,” in *Proceedings of the 2nd International Conference on Software Engineering*, 268–280, 1976.

[Sony, 2008] Sony Corporation. (2008). Sony Global – FeliCa. http://www.sony.net/ Products/felica/abt/dvs.html; accessed August 23, 2011.

[Stajano, 2002] Stajano, Frank. “Security for Ubiquitous Computing,” Wiley, 2002.

[Stephen, 2001] Stephen S. Yau, F. Karim. “Context-Sensitive Middleware for Real-time Software in Ubiquitous Computing Environments,” in *Proceedings of 4th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pp.163 –170, 2001.

[Stevens, 1974] Stevens, W.P., Myers, G.J., and Constantine, L.L. "Structure Design," in *IBM Systems Journal*, vol. 13, pp. 231-256, 1974.

[Stoica, 2001] Stoica, I., Morris, R., Karger, D., Kaashoek, M.F. and Balakrishnan, H., "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," In *Proc. ACM SIGCOMM 2001 Conference*, pp. 149–160, 2001.

[Strang, 2003] Strang, T. "Service Interoperability in Ubiquitous Computing Environments," PhD thesis, Ludwig-Maximilians-University Munich, 2003.

[Strang, 2003a] Strang, T., Linnhoff-Popien, C., and Frank, K. "Applications of a Context Ontology Language," in *Proceedings of International Conference on Software, Telecommunications and Computer Networks (SoftCom2003),* (Split/Croatia, Venice/Italy, Ancona/Italy, Dubrovnik/Croatia), D. Begusic and N. Rozic, Eds., Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, Croatia, pp. 14–18, 2003.

[Strang, 2003b] Strang, T., Linnhoff-Popien, C., and Frank, K. "CoOL: A Context Ontology Language to enable Contextual Interoperability," in *LNCS 2893:Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003),* (Paris/France), J.-B. Stefani, I. Dameure, and D. Hagimont, Eds., vol. 2893 of Lecture Notes in Computer Science (LNCS), Springer Verlag, pp. 236–247, 2003.

[Strang, 2004] Strang, T. and Linnhoff-Popien, C. “A Context Modeling Survey,” in First International Workshop on Advanced Context Modelling, Reasoning and Management, Ubicomp, 2004.

[Subrahamanian, 2000] Subrahamanian, V. S., Bonatti, P., Dix, J., Eiter, T. and Ozcan, F. “Heterogeneous Agent Systems,” 1st ed. Cambridge, Mass: MIT Press, 2000.

[Sundmaeker, 2010] Sundmaeker, H., Guillemin, P., Friess, P. and Woelffle, S. “Vision and Challenges for Realising the Internet of Things,” Cluster of European Research Projects on the Internet of Things (CERP-IoT), 2010.

[Thalheim, 1998] Thalheim, B. “Entity-Relationship Modeling: Foundations of Database Technology,” Springer-Verlag, 1998.

[TinyOS] The Open TinyOS Community. TinyOS. http://www.tinyos.net/;accessed May 18, 2011.

[Toninelli, 2009] Toninelli, A., Pantsar-Syvaniemi, S., Bellavista, P. and Ovaska, E.”Supporting context awareness in smart environments,” ACM Press, New York, USA, 2009.

[BSAC, 2008] University of California at Davis University of California at Berkeley. The Berkeley Sensor and Actuator Center (BSAC), 2008. http://wwwbsac.eecs.berkeley.edu/; accessed October 19, 2011.

[SmartDust, 2008] University of California at Berkeley: Robotics Lab. SmartDust. Autonomous sensing and communication in a cubic millimeter, 2008. http://robotics.eecs.berkeley.edu/ pister/SmartDust/; accessed October 19, 2011.

[Grand Challenge, 2008]UK Computing Research Committee. Grand Challenges in Computing Research, 2008. http://www.ukcrc.org.uk/grand challenges/ index.cfm; accessed September 10, 2011.

[Uschold, 1996] Uschold, M., and Gruninger, M. "Ontologies: Principles, methods, and applications," in Knowledge Engineering Review 11, 2, 93–155, 1996.

[Voelker, 1994] Voelker, Geoffrey M. and Bershad, Brian N. "Mobisaic: An information system for a mobile wireless computing environment," in IEEE Workshop on Mobile Computing Systems and Applications, pp. 185190, 1994.

[W3C, 2011a] W3C. Composite Capabilities / Preferences Profile (CC/PP). http://www.w3.org/Mo- bile/CCPP.

[Wang, 2004] Wang, X. H., Zhang, D. Q., Gu, T., and Pung, H. K., "Ontology based Context Modeling and Reasoning using OWL," in *Proceedings of CNDS*, 2004.

[Want, 2005] Want, R. and Pering, T. "System Challenges for Ubiquitous & Pervasive Computing," in ICSE'05, 2005.

[WAPFORUM, 2011a] WAPFORUM. User Agent Profile (UAProf). http://www.wapforum.org.

[Weiser, 1984] Weiser, M. “Program slicing,” in IEEE Trans. Software Eng., 10(4):352–357, 1984.

[Weiser, 1991] Weiser, M. “The computer for the 21st century,” in Scientific American, 265 (3):66–75, 1991.

[Weiser, 1996] Weiser, M. and Brown, J. “The Coming Age of Calm Technology,” 1996. http://www.ubiq.com/ hypertext/weiser/acmfuture2endnote.htm; accessed September 12, 2011.

[Wikipedia, 2011a] Wikipedia. (2005). Context awareness. http://en.wikipedia.org/wiki/Context_awareness. 22 JULY, 2011

[Winograd, 2001] Winograd, T. “Architecture for Context,” in Human Computer Interaction, Vol. 16,pp401-419, 2001.

[Xerox, 1996] Xerox Palo Alto Research Center Press Release. “Xerox Names Computing Pioneer As Chief Technologist For Palo Alto Research Center,” http://www.ubiq.com/weiser/weiserannc.htm; accessed September 12, 2011.

[Xie, 1995] Xie, M. and Wohlin, C. “An additive reliability model for the analysis of modular software failure data,” in *Proceedings of the 6th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 188–194, 1995.

[Yacoub, 1999] Yacoub, S., Cukic, B. and Ammar, H. "Scenario-based reliability of analysis of component-based software," in *Proceedings of the 10th International Symposium on Software Reliability Engineering (ISSRE'99)*, 22–31, 1999.

[Yokoo, 1998] Yokoo, M., Durfee, E. H., Ishida, T. and Kuwabara, K., "The Distributed constraint satisfaction problem: formalization and algorithms," in IEEE Transactions on Knowledge and Data Engineering 10(5), 673-685, 1998.

[Yu, 1996] Yu, E. S. K., Mylopoulos, J., and Lespérance, Y. "AI Models for Business Process Reengineering," in IEEE Expert, vol. 11, pp. 16-23, 1996.

[Yu, 2001] Yu, Z., and Rajlich, V. "Hidden dependencies in program comprehension and change propagation," in *Proceedings of the 9th International Workshop on Program Comprehension*, Washington, DC, USA, IEEE Computer Society, 2001.

[Zhang, 2004] Zhang, Tao. "An Architecture for Building Customizable ContextAware Applications by EndUsers," in Pervasive 2004 Doctoral Colloquium, Austria, 2004.

[Zhao, 2002] Zhao, B.Y., Kubiatowicz, J.D. and Joseph, A.D., "Tapestry: A Fault-tolerant Wide-area Application Infrastructure," in ACM SIGCOMM Computer Communication Review, vol. 32, No. 1, pp. 81, 2002.

[Zhu, 2005] Zhu, Y., Ye, S. and Li, X. "Distributed PageRank computation based on iterative aggregation-disaggregation methods," in *Proceedings of the 14th ACM International*

Conference on Information and Knowledge Management, pp. 578–585, New York, USA, 2005.

[Zimmer, 2004] Zimmer, T. "Towards a Better Understanding of Context Attributes," in PerCom Workshop, pp. 23-27, 2004.

[Zimmer, 2006] Zimmer, T., "Quality of Context: Handling Context Dependencies," in *Proceedings of 2nd International Workshop on Personalized Context Modeling and Management for UbiComp Applications*, California, USA, 2006.