Supporting Persistent Social Groups Using Context-Aware Ephemeral Device Group Service

Bin Wang, John Bodily, and Sandeep K. S. Gupta *

Department of Computer Science and Engineering, Ira A. Fulton School of Engineering, Arizona State University, Tempe, AZ 85287

Abstract

In this paper, we analyze the role of the social group in a Ubiquitous Computing (Ubicomp) environment as a source of contextual information. A model is presented to address the social group member's perceptions of how devices in a Ubicomp environment should aid them in collaboration. A goal of the model is to minimize user distraction while ensuring that the user has the final control over all the major decisions, such as group session continuity and formation of new groups. Based on the model a distributed context-aware group membership management scheme, called CAEG, is developed. CAEG has been prototyped as a middleware service on a platform consisting of wireless personal digital assistants and laptops. A sample application, group chat, was implemented which uses CAEG middleware service. This application was a part of Smart Classroom application suite. Usability and experimental tests were performed in the context of Smart Classroom. Lessons learned from these tests were used to enhance the model and establish the feasibility of the proposed approach.

Key words: Group Management, Ubiquitous Computing, Context-Aware

* Corresponding author. Telephone:+1-480-965-3806; Fax:+1-480-965-2751. A prelimi-

1 Introduction

A Ubiquitous Computing (Ubicomp) environment contains devices that are embedded in the environment, and a set of transient wearable, and handheld devices that are being carried by users inside the environment. Devices are connected over wireless networks, and may possibly be connected to a fixed network infrastructure such as the Internet. Devices in such environments have serious resource constraints, and they form numerous webs of short-range low-power mobile ad-hoc networks to exchange information. A principal goal of a Ubicomp environment is to provide computational support to help and mediate the users activities in a natural, and non-intrusive way [2,3]. This requires designs to be user-centric, and not simply device-centric. Most research has focused on supporting users on an individual basis in Ubicomp environments. It is common for users to be members of some social group. Thus, in order for interactions between the user and the computing environment to be seamless, support must exist for the collaborative efforts of groups as well.

1.1 Social Group

The definition of a social group that is used for this paper is defined in [4] as follows: “A number of individuals, defined by formal or informal criteria of membership, who share a feeling of unity or are bound together in relatively stable patterns of interaction”. This definition stresses two properties of the social group; the first is that people in a social group must feel some sense of membership. Second, social groups have a set of predictable collaborative interactions that group members

nary version of this paper appeared in Percom 2004 [1].

engage in with one another.

In order for a person to be a member of a group there must be consensus between group members to determine their inclusion. Membership is not decided solely at the individual level. The group, as a whole must recognize it. For example, in a software engineering class students are often assigned to groups. In these groups there is usually a provision made for the group to decide whether or not a student will persist as a member of the group. Thus, a student must identify themselves as a member of the team, but also the team must continually recognize the student as a member.

An interaction indicates some collaborative activity that requires one or more users to perform a set of actions toward the completion of a common goal. Because a group is formed for a specific purpose, the collaborative interactions become predictable. For example, a team of software engineering students will have a predictable set of collaborative tasks as they work on gathering requirements, designing, testing and evaluating software. The students will also have regular meetings to discuss progress, and plan for future milestones. It is from the concept of membership, and stable interactions in a social group that we derive our concept of “persistence” in the social group.

In general when a person is identified as a member of a group they’re considered a member of that group for a considerable amount of time. We refer to this as the “lifetime” of that person’s group membership. Some memberships such as being a family member have a lifetime that starts when one is born and ends one dies. The membership lifetime in other groups may last only a few months, such as a school project group. However, when compared to the time it takes for a user to complete an individual session on a computer, the lifetime of a social group membership is

generally much longer than the time taken to complete the computing session. So membership in a group is a source of persistent state data that cuts across computing session boundaries. From this point of view, social groups act as a source of persistent contextual information that should be leveraged by computing applications in order to seamlessly support users in a Ubicomp environment.

1.2 Social Context

Both location and identity contribute to the characterization of a situation that a user experiences but they are not the only sources of context present in an environment. A broader spectrum of contexts has been discussed in regards to context-aware computing in [5]. This paper is more concerned with social contexts:

Definition 1 Social Context: *Information relevant to the characterization of a situation that influences the interactions of one user with one or more other users.*

Social mores and norms are an example of social contexts. These influence the proper use of technology in a given environment. For example, when at a movie theater it is impolite to allow a cellular telephone to ring during a showing. No law states that your phone cannot ring during a movie, it is just a social norm to consider it rude.

A large part of multi-user software system design revolves around determining the functional scope of the system by capturing the social contexts that govern how users are allowed to use the system as a tool. When building such a system the intended usage of the system is often examined using use-cases. Then based on certain social roles that are designated to a user by a governing authority (generally a business) functionality is restricted to a subset of the total system functionality

and controlled using access rules.

It was established in [6] that in order for software to be unobtrusive to the user in a Ubicomp environment that it must become context-aware. As illustrated by the previous section, social contexts have played a large part in traditional system design. However, filtering the content and functionality available to a user through a set of persistent static access rules and roles does not make a system context-aware. In order for a system to be context-aware it must interface with both the user and the environment to dynamically sense the context and adapt it's functionality in order to best serve the user for a given situation. Social contexts are intangible in that you cannot build a sensor to register readings for a social more. Is there a way to leverage other contexts to characterize social contexts and detect them? An example follows that will be used to illustrate the challenges associated with overcoming this goal.

Example 1: Consider a group of students taking a software engineering course that are involved in regular meetings to peer review reports. Assume that each user carries a PDA that runs software allowing the users to participate in some computer mediated collaborative review process. In order for this to be ubiquitous the software should support this in environments with either fixed network infrastructure or in ad-hoc environments. Now, for a document peer review we must assume that the software supports the roles of reviewer, recorder, author, and moderator. When all members are in attendance at the designated location, at the correct time the application should start the peer review process. At some point during the review if the author and a reviewer choose to leave to review the raw data for the results presented in the report they should be able to continue working with one-another independent of the other group members. Similarly when they return to the peer review they should be integrated into the group as a whole again seamlessly. Then

once the review process is over the software should end the review process and send the meeting minutes from the review to everyone. □

1.3 Challenges of Supporting Groups in Ubicomp Environments

The main challenge illustrated by Example 1.2 is the seamless support of the group's interactions once membership has been established. The social context we are interested in detecting is the fact that a peer review is in progress. As we cannot sense the actual event of "Peer Review" the only option is to infer the occurrence of the event somehow using other contexts. The following challenges are specific to finding a solution for this problem:

(1) **Determining when an interaction is going to occur -** During the lifetime of a user's group membership several collaborative group interactions will occur which will require communication links to be established between devices owned by the group. These interactions occur based on some social context. A social context must first be represented in a format that can be used to perform computation in order to detect the occurrence of the collaborative interaction that this context denotes. Furthermore, computation should be supported in a repeatable manner with minimal demands on the end-user.

(2) **Determining when an interaction is going to end -** Most interactions will have an end, which is also denoted by some social context that needs to be detected and used to do processing specific to supporting the end of a group interaction. Similar to detecting the beginning of an interaction, we also need a way to represent the social contexts that characterize the end of a group interaction. Again, this should be supported with minimal demands on the end-user.

(3) **Supporting interactions in environments with varying network capabilities -** For a truly ubiquitous solution, support offered to the group should not be dependent on the networking capabilities of the environment. A solution should be able to function in both fixed and ad-hoc network environments. In order to provide a solution that is well suited to varying network capabilities a distributed solution is preferred. The solution must also allow a user or subgroup to leave a group due to becoming disconnected from the other members, and then rejoin once a connection can be re-established.

The rest of the paper is organized as follows. In Section 2, we describe our system model and a group model for Ubicomp environments, which focuses on demonstrating and presenting the relationship between two key groups: user group and device group. Section 3 describes a distributed Context-Aware Ephemeral Group (CAEG) Membership Management service. Section 4 presents a sample Group Chat application to demonstrate how the group model can be combined with CAEG to support discussion between users over ad hoc networks. In Section 5, we present the testbed which we have used to implement the CAEG service and the Group Chat application. Then some related work is discussed and used as the basis for comparison in Section 6. Finally we give some concluding remarks in Section 7.

2 System and Group Model

2.1 System Model

Our system model is designed around four primary assumptions. First we assume that each user will carry one or more Ubicomp devices that can communicate over a wireless network. Second, we assume that the users collaborate by using distributed

peer-to-peer application software over a mobile ad hoc network with provisions for multi-hop message routing. Three, we assume that there is no Byzantine (or malicious) failures in the system. This assumption considerably simplifies our design. However, this restricts the application of our system to those scenarios where devices (users) trust each other. Four, we assume that the system clocks at individual devices are synchronized with each other within the tolerance desired by the application. Some applications may have very stringent synchronization requirement while others may not require any synchronization at all. The problem of how to achieve synchronization in a distributed system is a well studied problem and we do not address these issues here in this paper.

2.2 Group Model

The system must manage the user's view of the social group in terms of membership criteria and collaborative interactions in order to support the user's perception of his/her social groups in a manner that minimizes human distraction [7]. We assume that consensus must somehow be achieved to decide on assessing the candidacy of a member for a group, and validating their membership in the group. As discussed in Section 1 this is a non-trivial task, and it is necessary to include the user in decisions about group membership. Similarly, discovering a social group to join is a non-trivial task, and may require user intervention. Some groups may openly publish their existence and allow new users to join freely. Other groups may wish to keep their existence a secret and protect the names of their members. The social factors involved in selection of membership for groups is very complex. Feedback from the user must be obtained, and full system automation is not desirable.

In order for the user to perceive that the system shares their notions about a so-

cial group, the system should provide unobtrusive support for the general routines (such as devices setup and group member discovery) that the group carries out. Furthermore, support for these routines should last for the entire lifetime of the user's membership with the group, and in order to reduce distraction should minimize effort needed by the end-user. As in traditional system design, limitations of the systems functional scope can be controlled by tailoring functionality to the roles that are present in a social group. We now present a model that adheres to the above mentioned criteria.

Based on the challenges discussed in Section 1 we have developed a conceptual model that primarily addresses the need to support group membership and routines with minimum system set-up. The fundamental goal of our approach is to support the social group with minimal cognitive effort expended on actual system operation, and to make users perceive that a device is a group collaboration tool. First the definition of a device context is presented. Next, the conceptual group model will be described. Finally a description of how the group model addresses Challenges 1 and 2 from Section 1 is provided.

Definition 2 Device Context: *Any detectable and relevant attribute of a device, its interaction with external devices, and/or its surrounding environment at an instant of time.*

For example, device context includes location of device, available memory of device, etc.

A description of our approach to developing a group model is now given. The basic premise of our model revolves around two different conceptual groups: User Group and Device Group.

User Group: The User Group is a social group that the end-user interacts with as discussed in Section 1. Our model is based on the fact that user groups are expected to have stable patterns of interactions. This paper refers to these regular and repetitive collaborative interactions as a user group session:

Definition 3 *The time period for which the members of a user group are actively participating in interactions with one another is known as the* **User Group Session**.

In Example 1.2, a user-group session consists of a report peer review. The report review happens at a regular and reoccurring time, and for approximately the same duration each time.

The second group in our model is known as a device group. According to our system model we assume that an end-user will be carrying one or several computing devices that will be used to collaboratively interact with the devices of other group members. Logically a user may view these devices as a group that forms whenever the group needs to collaborate. This is referred to as a device group and is defined as follows:

Device Group: A device group is one or more devices that form a complete unit in composition with the purpose of collectively supporting some computational tasks on behalf of the user-group. The duration for which the device-group forms is determined by the user group session. However, not every device is a candidate for each device group. This implies that there must be a method for selecting the devices which become members of the device-group. The Device Context is used in this case to facilitate member selection for a device group, and in Example 1.2 we see that the Device Context can be used to allow only the devices owned by the team members to take part in the group. By giving the device group functionality to

select the correct device members based on social context the end-user is given the perception that the device-group supports their understanding of social situations.

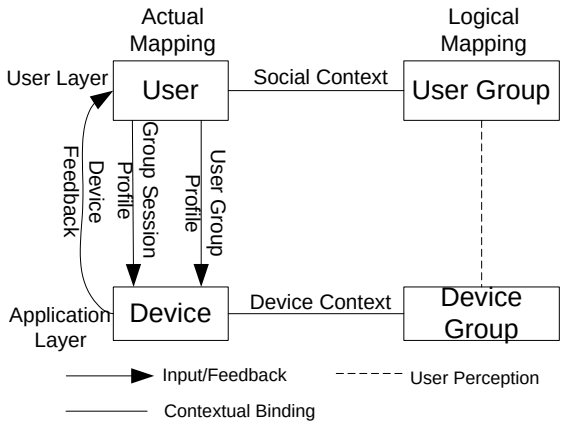

Fig. 1. Multi-layer group model in Ubicomp environment.

Fig. 1 shows the overall relationship between the User Group, and Device Group. Notice that in this case the membership of a user in a user group is based on social contexts, whereas membership in a device group is characterized by the device context. The dotted line indicates that the user will develop the perception that the device group as a whole is interpreting the social context in order to support the user group. In order to provide this perception to the user each device must support device group formation based on the device contexts used to characterize the social contexts that decide when a group collaborative effort begins and ends. As shown in Fig. 1 our model uses profiles to accomplish this. A profile is nothing more than a file that resides on a device that is used to specify values of context. The values in the profile are used to trigger the formation of a device group, or provide informative social context information to an application in order to mediate and support both group interactions, and functionality specific to the role(s) of the end-user. The model uses the following two profiles as shown in Fig. 1:

User Group Profile (UGP): This profile provides a representation of social contexts that are relevant to a user group, and are used by applications at a device that

support collaboration to facilitate the specific actions required for a user's role or preferences. The User Group Profile includes a unique user identifier, a group identifier, the user's preferences, a group purpose and the role of the user in the group. In Example 1.2, each user will have a specialized function in the group such as reviewer, recorder and author.

Group Session Profile (GSP): The Group Session Profile is used to characterize some specifics that can be used to represent the contexts which define the beginning and ending of a collaborative interaction between group members. The profile should at least include the time period of the group session, the location where the group session will occur. In some cases be an "anywhere" location, and the interval of the group session. In Example 1.2, the group session profile is used to define the time period and location of a meeting for reviewing reports.

Device Feedback: The goal of our model is to minimize user distraction, but removing user participation completely may result in undesired inflexibilities. So Device Feedback is incorporated in the model. This provides the flexibility to users in device group formation and dispersion. For example, during the group meeting in Example 1.2, students find that they need to extend the group meeting for one hour. When the scheduled time is over, devices notify corresponding users about the finished time of group meeting; hence causing user distraction. However, this distraction is tolerable compared to the frustration a user may experience if she is unable to finish her task in a satisfactory manner. In this case, the feedback can be used to get from users new contextual criteria for dissolving the group session e.g. the user may extend the group session duration for one hour by changing GSP. This ensures that users are the decision-makers on whether the device group should be disbanded or not, instead of the device.

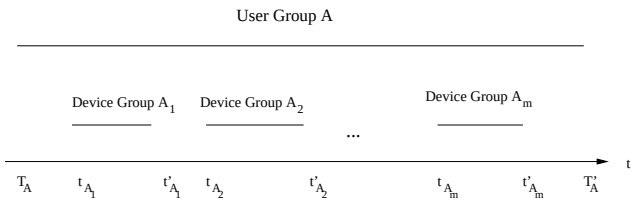

Fig. 2. Duration of groups in Ubicomp environment.

Now that all of the components of the model have been introduced we can analyze the model from the perspective of how it uses context in two ways. First to provide support for user group routines, and second to support the user's social roles and preferences.

Based on the group model we can see that the user has a perception of identity which is tied to the group. In particular this identity has a lifetime associated with it. In Example 1.2, the lifetime of the user's membership with the group lasts from the time it is assigned by the instructor to the time that the semester ends. During this time the group will have several routine user group sessions that need to be supported by device group formation. Context information represented in the Group Session Profile (GSP) is used by the device to form the device group that supports a user group session.

Fig. 2 shows a graphical representation of times that the groups from the group model are being formed, and dispersed as viewed by a group member over time. If a user is a member of the User Group A during the time (T_A, T'_A) shown in Fig. 2, they will participate in m routinely held user group sessions during the times (t_{A_1},t'_{A_1}), (t_{A_2},t'_{A_2}), $\ldots$, (t_{A_m},t'_{A_m}). Each device group A_i is shown in the figure as being active from the time it is formed, to the time it is dispersed, during an interval (t_{A_i},t'_{A_i}). Thus the context representation from the Group Session Profile after initial creation will be used m times to form a device group for each of the m user group sessions. The Context-Aware Ephemeral Group Membership

Management described in Section 3 is used to form the device group.

We now examine how the model uses social contexts represented in the User Group Profile (UGP) to support the user group. As previously stated, the UGP is an input used by the devices to support collaboration between members in the user group. More specifically in Example 1.2 we assumed the existence of some distributed peer-to-peer application software to support document peer reviews, the UGP would be used as input for that application. Based on the user's preferences and through some pre-defined roles that are supported by the application the software will directly support the user's expectations of what interactions they should be allowed to perform with the group. This provides the user with a perception that the applications understand their needs and purpose in the group. In Example 1.2, a possible role that could be assigned to a user in the team should be moderator, and hence in the UGP for that person they will be described as having the social context of moderator. The document peer review application should use the information in the UGP to provide the appropriate functionality to the user.

A device group is a collaborative tool used to support a user group. Social contexts determine the inclusion of a user in a group, and the types of collaborative activities that a group performs. UGP is an abstraction of the social context that are relevant to a user group. The GSP specifies device contexts that describe when a social context indicates a collaborative interaction between group members will begin and end. The UGP contains information about social contexts relative to an individual member of the group such as their role in the group. By using the GSP to construct a group membership management service, and the UGP to customize application functionality the end-user is given the perspective that the device group interprets social contexts, and supports their user group. In this paper, we assume that all of the collaborative interactions in a group can be abstracted into a GSP and that con-

texts included in the GSP can be detected by devices. For example, after an exciting game of football, if the fans of the winning team decide to celebrate together after the game, their group celebration is a user group session. However, the beginning and ending of this user group session can not be abstracted into a GSP. Handling such spontaneous user group sessions is out of the scope of this paper.

3 Context-Aware Ephemeral Group Membership Management

In this section we study how to form device groups over ad hoc networks by using context to characterize the event associated with GSP in a distributed manner. Because device groups in a Ubicomp environment are ephemeral, they should be facilitated by a non-deterministic process group[8]. A group is defined to be deterministic if each member should receive and act on a request. Deterministic process groups in general require strong co-ordination and synchronization among the group members. As opposed to this, a Non-deterministic process group is used to implement distributed applications that do not require strong consistency between peers, and hence the overhead expended to manage the group will be more minimalist. Based on Challenge 3, we design CAEG membership management in a distributed manner, which is similar to [9,10].

As we discussed in Section 2.2, the User Group Profile describes the function of the group and the role of the user in the user group. The UGP is a tuple UGP = {GID, GroupPurpose, UserRole} where the fields are defined as follows:

- **GID:** *the Group Identifier is used to uniquely identify a Group on each device.*
- **GroupPurpose:** *the GroupPurpose describes the function of the group, for example "report peer review" or "discussion"*

- **UserRole:** *a User Role describes the role of the user in the group, such as "recorder" or "author".*

The Group Session Profile (GSP) also introduced in Section 2.2 describes the session of a user group. For the sake of brevity, we assume that there are only two kinds of context (location and time) in GSP, although it would be easy to incorporate other kinds of context in GSP. GSP = {GID, Location, StartTime, FinishTime}.

- *StartTime, FinishTime: When the CAEG starts and finishes.*
- *Location: The geographical area in which group members are located.*

Using the GSP, a device group will be formed and dissolved. Actions at each device in a device group are based on the content of the Group Session Profiles at that device. For example, a user belongs to User Group A in Fig. 2. User Group A has m device group sessions. And the user uses device d at device group session A_1. Then the lifetime of d's membership in A_1 is based on the duration for which d's device context matches with the GSP for A_1. At time t_1 $(t_{A_1} \leq t_1 < t'_{A_1})$, if d's device context matches the GSP, d is able to become a member of device group A_1. Later at time t_2 $(t_{A_1} < t_2 \leq t'_{A_1})$, d's device context may no longer match the GSP resulting in d's deletion from A_1. And at time t'_{A_1}, the device group A_1 will disperse.

Assuming every device has already obtained a valid User Group Profile, a device group is then needed to support the user group session. The function of CAEG Membership Management Service is to automate the formation and disbanding of device groups for user group sessions by using the following steps:

1) **GSP Initiation and Distribution:** The group initiator defines the GSP and distributes it to the anticipated group members.

2) **GSP Registration:** Once the group session profile is transmitted and received, an application running on the device is notified about the arrival of the new GSP. The application will then be used to decide whether or not to accept the GSP. If the GSP is rejected, it will be discarded. Otherwise, if the GSP is accepted the application will register the GSP with the CAEG Membership Management Service. A GSP also contains a membership list that consists of the group member's identifiers. Initially, the GSP's membership list is Null. A GSP with a non-null member list is called an *active* GSP, and several active GSPs can exist on a single device to support various device groups simultaneously.

3) **Device Group Activation and Member Addition/Deletion:** A device gets data for the contexts that are relevant to the fields of a GSP from sensors. The data from the sensors is then used as input to the CAEG Membership Management Service. The context values that are captured by the CAEG Membership Management Service are known as the *CAEG Context*, where the *CAEG Context* = $\{$*time, location*$\}$. The CAEG Membership Management Service monitors the *CAEG Context* and takes the following context-dependent actions:

   If the *CAEG Context* C matches the contexts specified in the Group Session Profile gsp, i.e. $gsp.StartTime \leq C.time \leq gsp.FinishTime$ and $C.location$ is currently $gsp.location$, it adds the device identifier to the GSP's membership list.

   If the CAEG Context no longer matches the Group Session Profile, the device identifier will be deleted from the GSP's membership list.

4) **Group View Maintenance:** A group view maintenance protocol for devices to exchange a *Group View (GV)* with each other in mobile ad hoc network has been developed. To deal with fault tolerance issues in group view maintenance, our protocol guarantees view accuracy and view completeness (The

group view correctly expresses the group membership in all kinds of scenarios of network connection). The interested reader should refer to Appendix A for a detailed description of our group view maintenance protocol. A device's Group View includes the GID from the GSP and the GSP's current group membership list [1] . Using this protocol each device will know the Group Membership List of all devices in its vicinity, and according to those Group Views a device will update its own group view.

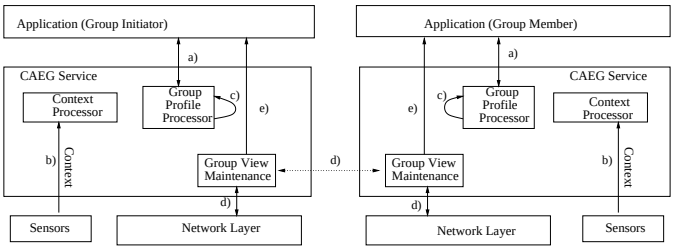

Fig. 3. Architectural overview of CAEG membership management.

Fig. 3 shows the architecture of CAEG Membership Management. There are three modules in CAEG Membership Management Service: Context Processor, Group Profile (UGP and GSP) Processor, and Group View Maintenance.

a) The Group Profile Processor stores the UGPs and GSPs. Using Group Profile Processor, application can change to the UGPs and GSPs.
b) The Context Processor abstracts the context that it obtains from sensors;
c) The Context Processor passes the *CAEG context* to the Group Profile Processor; By comparing the *CAEG context* with the GSP, the Group Profile Processor determines whether or not to activate/deactivate the group session profile.
d) After a GSP is activated, the Group View Maintenance protocol will exchange the group view with other CAEG services running on remote peer devices.

[1] A list of currently active and connected group members from the device point of view is the group view of the device group [11]. So a single device's group view need not be a global group view.

e) If the current group view changes, the CAEG Membership Management Service will notify applications about the new group view.

To provide flexibility to users, CAEG Membership Management Service allows users to dynamically make changes to GSP. In Example 1.2, if the meeting of reviewing report is postponed by an hour, users and/or applications can increase *StartTime* and *FinishTime* of the GSP at their individual devices by one hour. This will ensure that the device group formation will be also be postponed for an hour.

4 Group Chat

This section will demonstrate how the User Group concept can be used to implement an actual application. The application in this case is a chat client named GroupChat for use over ad hoc networks. Group collaboration is supported by letting users communicate, and exchange files with one-another in settings where infrastructure support is unavailable. Lack of infrastructure support necessitates the need to design the application in a peer-to-peer manner with no reliance on a central server. The chat application we discuss uses the CAEG service, and the UGP to establish groups for the purpose of communication, and file sharing.

Users are organized for GroupChat into social groups of users, which wish to communicate with each other. The groups used in GroupChat are analogous to a buddy list, or a chat channel. A user is a member of a social group if they have a valid group session profile, and user group profile on their device for that group. Two basic roles exist in the GroupChat application; a role of “chat moderator” who initiates the group, and the role of “chat user”. The basic interactions the application associates to users within the group with the role “chat user” are one to many

communication, one to one communication, and file transfer. Additionally, an interaction must exist to view a user profile for each user in the group. Users with the role “chat moderator” should have access to all of the interactions provided to the “chat user” role plus the ability to perform “mute”, and “unmute” interactions. The GroupChat application contains four major components as described below, and depicted in Fig. 4:

ChatGroup Manager: The ChatGroup manager updates the application data needed to characterize the state of activated groups. It uses the User Group Profile to determine which interactions the end-user should be allowed to perform on a per group basis, and Group Session to determine the user group sessions. All text sent to, or received from a group is buffered here for retrieval by other GroupChat components.

GroupUpdate Processor: This component is responsible for registering the application with the CAEG service. It also processes notifications sent from the CAEG about changes to the the current set of active user groups, and to changes in the membership views for those groups. The results of processed notifications are used to update the ChatGroup Manager’s state.

ChatMessage Processor: The ChatMessage processor interacts with the ChatGroup manager and the CAEG service to send, and receive messages, or files destined for a remote-host in a group. Messages still needing to be sent are extracted from the ChatGroup Manager and passed to the CAEG service, and vice-versa.

Application GUI: The application GUI provides the user with the functionality to carry out the role-based interactions that they are allowed to execute. It must also display to the end-user the current set of activated groups, the membership views for those groups, as well as text messages, and the status of file transfers.

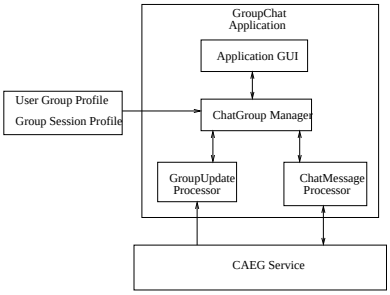

Fig. 4. Architecture of the GroupChat application.

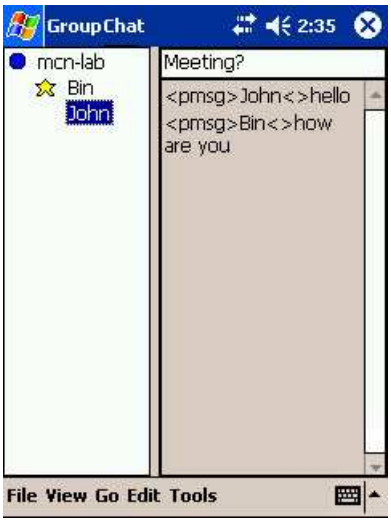

Fig. 5. Screen-shot of the prototype GroupChat application.

The GroupChat application has been developed as a sample application in our research testbed that is discussed in the next section. Fig. 5 shows a screen-shot of the prototype GroupChat application.

5 Prototype of CAEG Service and GroupChat Application Implementation

To evaluate the CAEG service, we are currently implementing a Smart Classroom [12]. The test bed will facilitate different activities leading to efficient teaching and collaboration in a classroom. Smart Classroom monitors the classroom context (location of students or instructor, noise, light, mobility) and uses the context to trigger communication activity among students and instructor to facilitate

collaborative learning [12]. Also to deal with heterogeneous environment issues, a Reconfigurable Context-Sensitive Middleware (RCSM) has been developed [13].

5.1 Test Bed Configuration

In our current design, each node in the test bed has the configuration shown in [13]. The configuration consists of the following components: Casio E-200 PDAs, Trenz Electronic USB-compatible Xilinx Spartan II FPGA boards, noise, light, and motion sensors and location-tracking components. Each PDA uses an Intel Strong Arm 1110 with 206 MHz clock speed CPU. Each PDA had Flash ROM and RAM of 32 MB and 64 MB respectively. Each PDA was also equipped with a D-LINK Air DCF-660W Compact Flash 802.11b adapter. These adapters were configured in ad hoc network configurations. As such, no infrastructure, such as an access point, was necessary to provide the communication support.

We have used Microsoft Embedded Visual Studio to develop the CAEG service, and GroupChat application for the Windows CE operating system. A screen-shot of the chat application as captured from the PDA is shown in Fig. 5. The size of the executable programs for the prototype of the CAEG service, and GroupChat application is 35KB and 84KB, respectively.

5.2 Performance Results of CAEG Membership Management Service

We use two performance metrics to denote the latency of CAEG Membership Management: The delay of activating a GSP and the delay of triggering the group view maintenance protocol. The delay of triggering the GSP is the time period from when the CAEG Membership Management obtains raw context data to the time when the

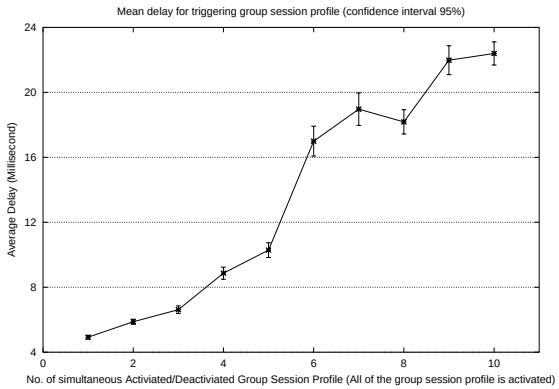

Fig. 6. Mean delay for triggering GSP (All groups are triggered).

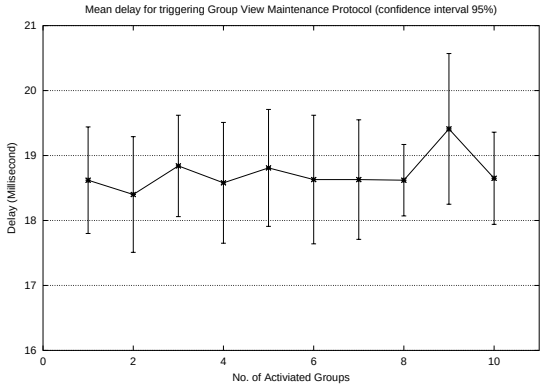

Fig. 7. Mean delay for triggering group view maintenance protocol.

corresponding GSPs are activated/deactivated. The delay of triggering the group view maintenance protocol is the time period from when a GSP is activated to the time when the first group view is sent out.

Fig. 6 illustrates the delay of triggering the GSP. In Fig. 6, we notice the correlation between the number of GSPs being maintained by the CAEG Membership Management, and an increase in delay. Fig. 7 shows the delay of triggering the group view maintenance protocol. Notice that the delay remains stable even if the number of activated group profiles increases. Based on an observed overall end-to-end delay of approximately 1 second in our testbed, we can see that the additional overhead introduced by CAEG Membership Management produces an imperceivable amount of delay.

From the experimentation that was conducted with the test-bed software and the CAEG service, we realized the importance of factoring resource optimizations that encompass the whole, rather than just the parts, into design. It is especially important when applications must run on resource-constrained devices such as PDAs used in our experiment. With no infrastructure support to share the burden of facilitating communication or computation the importance of factoring energy-efficiency and fairness into designs becomes very apparent. For example when implementing the CAEG group view maintenance protocol the design could have been carried out for a leader based protocol where the leader node once elected beacons requests to neighboring nodes to send in a message to indicate their presence, and then once a certain time interval elapses the leader node would send the view out to the other nodes that it received replies from. While this solution is valid, it does not consider how quickly the leader will lose power, and hence when it must be shutdown the group view will be lost.

We deployed the CAEG service and chat application in Software Engineering Project I (CSE 461) and Software Engineering Project II (CSE 462) courses at Arizona State University in Fall 2003 and Spring 2004. All of the students believe that the CAEG service and chat application are very useful and practical. In our previous group model [1], we did not consider the device feedbacks to users. So most of the negative comments from students are the user's participation to the group formation - system flexibility. According to these comments, we add the device feedbacks to users in our group model and CAEG implementation. So users can dynamically change the duration of the group session. Another popular comment from the students was about the formation of device group based on user's interest.

For example, some students at a pub may want to form a chat group for the ongoing NBA game. The solution is adding an “interest” field in UGP and GSP. So device group is formed according to user interest instead of predefinition. Facilitating the collaboration of “interest”-based device group needs further study.

6 Related Work

Few studies have been done for group communication in a Ubicomp environment. Prakash and Baldoni give a simple architecture for group communication in mobile systems [14]. In their architecture, a proximity layer below a group membership layer is introduced. They use a three-round group construction protocol to construct a group. The core idea is to create a group of all the nodes that are within a given distance D from the group initiator. They use a flooding algorithm to discover the nodes located in the area. However, there is no support for disconnection and context processing in the architecture, and a node can only be a member of a single group. Killijian et al. in [10] describe a location-aware group, which is similar to [14]. The group membership of mobile devices in this scheme depends on the location of the devices, but they do not make provisions for ad hoc networking, or any contexts except location. Roman et al. proposed a mobile group, which is defined by a safe distance [15]. Only the nodes that are within the safe distance of its nearest neighbor can be considered as a group member. This notion of a group is very restrictive and does not deal with the general understanding of group communication; however, it is useful to implement disconnection awareness and network partition anticipation.

Briesemeister et al. [9] presents a localized group membership service that maintains the group membership status of neighbor nodes (Neighbor nodes are the nodes

within a node's maximum transmission range) and proves the correctness of their implementations. Both Babaoglu et al. [16] and Fekete et al. [17] study partitionable group communication services to support continuing operations of "partition-aware" applications without blocking in multiple concurrent partitions.

Context-aware applications adapt their behavior in order to better meet the needs of the end-user in their current environment, while trying to make these adaptations as unobtrusive as possible. Applications such as the CyberGuide [18] from Georgia Tech use the context of location to provide relevant and useful information to tourist about nearby attractions, or displays inside a museum. Portable Help Desk (PHD) [7] developed at Carnegie Mellon University uses both spatial and temporal context to provide several services to students when they are on campus. This includes map services that display nearby available resources which can be computing and non-computing related, such as nearby printers, and events on campus. PHD can also be used to locate other individuals on campus whom may be a friend or a class team member. Context-awareness has been blended with instant messenger and chat applications in the past by the Hubbub [19] project at AT&T, and by the ConChat [20] application developed at University of Illinois at Urbana-Champaign. The Hubbub project uses the context of sound, to foster opportunistic interactions with users on the buddy list by playing a unique sound for each user when they become available. ConChat uses context-awareness to aid parties in determining the overall context in which an on-line chat is occurring, by displaying the context present in the room where a remote user is chatting from. Neither Hubbub, or ConChat support messaging without support from a centralized server.

7 Conclusions

In this paper we have presented a model for supporting social groups in a Ubicomp environment. First, we explored the properties of the social group, and defined what is meant by a “Social Context”. From this we determined that the key areas needing support for Social Groups in a Ubicomp environment are: support for the persistent membership of a user in a social group, and support for automating the collaborative routines that take place between group members.

Next, a conceptual model was constructed which embodies the relationship between a social group, and the computing devices used to support those groups. The model also illustrates how to use a device that is context-aware to form a device group based on contexts such as location, and time in order to support group collaboration, and provide the user with the perception that the device group is interpreting social contexts to support them. Third, we described a distributed CAEG membership management scheme that facilitates the formation of the context-aware ephemeral device groups discussed by our conceptual model.

Then a description of a sample application that demonstrated the overall concepts of our approach was given. We also discussed the details of our testbed, the primary focus of our usability testing, and the experimental results that were produced by that testbed. We also discussed the lessons learned from conducting our experimentation and discussed some related work.

8 Acknowledgments

This work is supported by National Science Foundation (NSF) grant ANI-0123980. We thank the anonymous reviewers and Crystal West for their helpful comments for improving the quality of this paper.

References

[1] B. Wang, J. Bodily, S. K. S. Gupta, Supporting persistent social groups in ubiquitous computing environments using context-aware ephemeral group service, in: Proceedings of 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom-04), Orlando, FL, 2004, pp. 287–296.

[2] M. Weiser, The computer of the twenty-first century, Scientific American 265 (3) (1991) 66–75.

[3] S. K. S. Gupta, W. C. Lee, A. Purakayastha, P. K. Srimani, An overview of pervasive computing, IEEE Personal Communications 8 (4) (2001) 8–9.

[4] G. Marshall, et al., The Concise Oxford Dictionary of Sociology, Oxford University Press, Great Britain, 1994.

[5] A. Schmidt, M. Beigl, H. W. Gellersen, There is more to context than location, Computers and Graphics 23 (6) (1999) 893–901.
URL `citeseer.nj.nec.com/schmidt98there.html`

[6] M. Satyanarayanan, Pervasive computing: Vision and challenges, IEEE Personal Communications (2001) 10–17.
URL `citeseer.nj.nec.com/satyanarayanan01pervasive.html`

[7] D. Garlan, D. P. Siewiorek, A. Smailagic, P. Steenkiste, Project aura: Toward distraction-free pervasive computing, IEEE Pervasive Computing 1 (2) (2002) 22–31.

[8] L. Liang, S. Chanson, G. W. Neufeld, Process groups and group communication:classification and requirements, IEEE Computer 23 (2) (1990) 56–68.

[9] L. Briesemeister, G. Hommel, Localized group membership service for ad hoc networks, in: International Workshop on Ad Hoc Networking (IWAHN), Vancouver, British Columbia, Canada, 2002, pp. 94–100.
URL citeseer.nj.nec.com/briesemeister02localized.html

[10] M.-O. Killijian, R. Cunningham, R. Meier, L. Mazare, V. Cahill, Towards group communication for mobile participants, in: 1st ACM Workshop on Principles of Mobile Computing (POMC), Newport, Rohade Island, 2001.

[11] G. Chockler, I. Keidar, R. Vitenberg, Group communication specifications: a comprehensive study, ACM Computing Surveys 33 (4) (2001) 427–469.
URL citeseer.nj.nec.com/chockler01group.html

[12] S. S. Yau, S. K. S. Gupta, F. Karim, S. I. Ahamed, Y. Wang, B. Wang, Smart classroom: Enhancing collaborative learning using pervasive computing technology, in: ASEE 2003 Annual Conference and Exposition, Nashville, Tennesee, 2003.

[13] S. S. Yau, F. Karim, Y. Wang, B. Wang, S. K. Gupta, Reconfigurable context-sensitive middleware for pervasive computing, IEEE Pervasive Computing 1 (3) (2002) 33–40.

[14] R. Prakash, R. Baldoni, Architecture for group communication in mobile systems, in: 17th IEEE Symposium on Reliable Distributed Systems (SRDS-17), West Lafayette, IN, 1998, pp. 235–242.

[15] G. C. Roman, Q. Huang, A. Hazemi, On maintaining group membership data in ad hoc networks, Technical Report wucs-00-26.

[16] O. Babaoglu, R. Davoli, A. Montresor, Group communication in partitionable systems: Specification and algorithms, IEEE Transcations on Software Engineering 27 (4) (2001) 308–336.

[17] A. Fekete, N. Lynch, A. Shvartsma, Specifying and using a partitionable group communication service, ACM Transactions on Computer Systems (TOCS) 19 (2) (2001) 171–216.

[18] G. D. Abowd, C. G. Atkenson, J. Hong, S. Long, R. Kooper, M. Pinkerton, Cyberguide: a mobile context-aware tour guide, ACM Wireless Networks 3 (5) (1997) 421–433.

[19] E. Isaacs, A. Walendowski, D. Ranganathan, Mobile instant messaging through hubbub, Communications of the ACM 45 (9) (2002) 68–72.

[20] A. Ranganathan, R. H. Campbell, A. Ravi, A. Mahajan, Conchat: A context-aware chat program, IEEE Pervasive Computing 1 (3) (2002) 51–57.

[21] R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, A. Schade, Deapspace - transient ad hoc networking of pervasive devices, Computer Networks 35 (4) (2001) 411–428.

A CAEG View Maintenance Protocol and Performance Analysis

A.1 Description of CAEG View Maintenance Protocol

Because we envision the formation of device groups will be necessary in locations with no infrastructure support, the CAEG should be able to form a device group over an ad hoc network. This means however that some devices may not have a large transmission range, but devices in a CAEG should build a timely picture of the network in its area called a “global Group View(GV)”. For example, given that there are four devices belonging to the same group, and the GSP of each device has already been activated, how can every device know the existence of the other three? Ideally each device would make a reliable broadcast to the others, and construct a

current global GV for each device. Aside from the complexity introduced by the need to support reliable broadcast, under normal conditions this approach carries with the risk that a broadcasting device cannot know which devices are present to receive its advertisement. A similar problem is introduced when that group view set changes between beacons. Also it will introduce a big overhead to maintain the group view. This problem is similar to the problem of service discovery in mobile ad hoc networks.

To maintain the group view efficiently, we propose a solution similar to service discovery by Hermann et al. in [21]. The idea is that each device should local broadcast, beacon, its entire GV. If all listening devices incorporate new members found in each beacon into their own views, then occasionally missing a beacon will not result in serious problems for any particular device, because the next beacon by any device that did not miss the first one will repeat the updated information as reflected in its current group view. Also, as each device changes the contents of the global group view before repeating it, the group view undergoes a constant slow change. In this way, group view is not only shared but also timely. The protocol also allows for device groups to partition and reform as the complete device group at a later time, in this situation whichever nodes are in a sub-group will still be able to communicate until that sub-group unites with any other sub-groups to reconstruct the whole group. The following is a general outline of device behavior for group view maintenance that is performed by the CAEG service, the Pseudocode for which is outlined in Fig. A.1 and A.2.

(1) Each node maintains a list of group member descriptors, where a group member descriptor is represented by a time-to-live, and a unique identifier for the group member. The list of group member descriptors is the group view of the node.

```
GroupViewMaintenance(A) / / A is device i's current group view
    tout := t_small
    Loop(forever) {
        B := read group view beacon from Network layer
        forevery a ∈ A {
            if (a.expiry ≤ 0) then {
                if (a.id = i.id) then {
                    a.expiry = 2t_big
                    tout := t_small
                }
                else
                    delete a from A
            }
        }
        if (timed out tout) then        ⇐ α
            beacon(A)
        else
            RecevieView(A,B)
    }
```

Fig. A.1. Pseudocode of Group View Maintenance Protocol at device i

(2) Each node advertises its group view by a beacon mechanism.

(3) The Group View includes the GID and a list of identifiers which includes an entry for each of the connected nodes in the group.

ReceiveView(A,B) / / B is device i's received group view

tout := t_{small} $\Leftarrow \gamma$

if $B = A$ **then**

tout := t_{big} $\Leftarrow \gamma$

else

forevery $b \in B$ {

A := $A \bigcup \{b\}$ $\Leftarrow \beta$

if $b \in A$ **then**

Update b.expiry with the received time-to-live value $\Leftarrow \theta$

}

Fig. A.2. Pseudocode of Group View Maintenance Protocol at device i

(4) The beacon mechanism works as follows:

Beacons are scheduled to occur at a fixed time interval t_{small} or t_{big}. And beacon message just local broadcast one hop neighbors. $\Leftarrow \alpha$

(5) Each node processes a newly received Group View in following ways:

• It combines the received list of group member descriptions with its own group view. $\Leftarrow \beta$

• It updates the time-to-live values of the group member descriptors. $\Leftarrow \theta$

• If a node discovers that its group view is not same as the received group view, it will set the beacon interval to t_{small}. Otherwise set the beacon interval to t_{big}. $\Leftarrow \gamma$

Fig. A.1 and A.2 show the Pseudocode of the Group View maintenance protocol. Intuitively, because a device's own time-to-live should be larger than its beacon interval, i.e. $t_{small} < 2t_{big}$. Otherwise, its own identifier will time out too soon at

the other nodes in the group and the beacon will not catch up, causing periods of disconnection between nodes.

A.2 Analysis of Group View Maintenance Protocol

Briesemeister et al. [9] present a group view exchanging protocol that obtains a group view by hearing beacons from one-hop neighboring nodes. In their protocol, every node sends out a beacon which includes group identifier and the node's identifier. Their protocol is only suitable for a single hop mobile ad hoc network. In the following discussion, we refer to their protocol as SViewP (Single hop group View Protocol). Our group view maintenance protocol is not only suitable for single hop but also multi-hop mobile ad hoc networks. We call our group view maintenance protocol the CAEGViewP (CAEG View Protocol). Compared with SViewP, CAEGViewP is more efficient: $1)$ The total number of beacons is fewer because a device keeps silent if one of its one-hop neighbor beacons the same group view; $2)$ Since total number of beacons is reduced, the possibility of beacon collision is also decreased; $3)$ Because each beacon is a globe GV of the node, the header overhead of beacon is dramatically reduced compared with simple broadcast. Now we will analyze the performance of these two protocols in terms of number of beacons and latency.

For comparison simplicity, we assume that the network is single hop and that a beacon can be received reliably by all nodes. We denote the lifetime of a group session by T_G. And we use N to denote the number of nodes in the network that are available to be group members.

For SViewP, the total number of beacons sent out by a node during the lifetime of

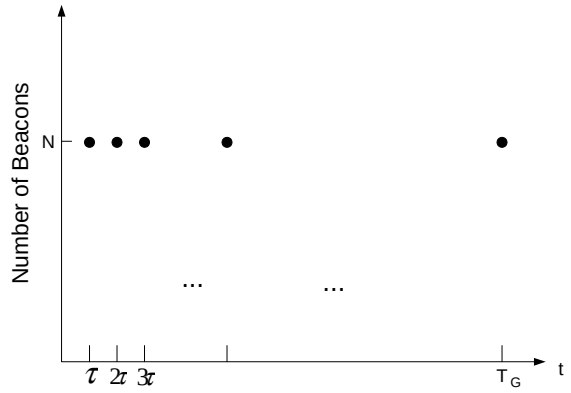

Fig. A.3. SViewP: No. of Beacons at all the nodes for SViewP

a group session is $\frac{T_G}{\tau}$ (if SViewP uses τ as beacon interval). So the total number of group view messages sent by all the nodes in the group is $N\frac{T_G}{\tau}$ as shown in Fig. A.3.

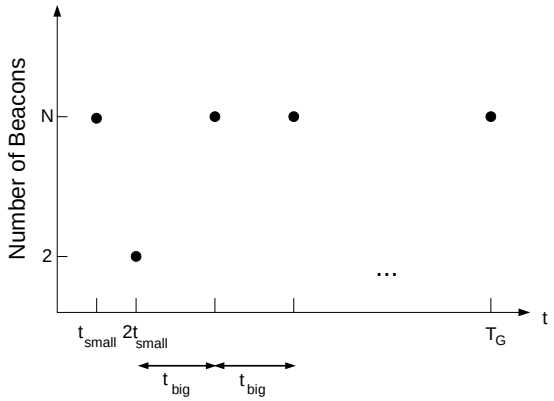

Fig. A.4. CAEGViewP - Best Case, No. of Beacons at all the nodes.

For CAEGViewP, we assume all of the node's group session profiles have been activated at the beginning of the group session. We also assume that none of the nodes in the group will deactivate their group session profiles until the end of group session. The number of beacons that need to be sent out varies for CAEGViewP, because every node must make a decision to send out the group view depending upon the most recently received group view. Initially, every node beacons its own

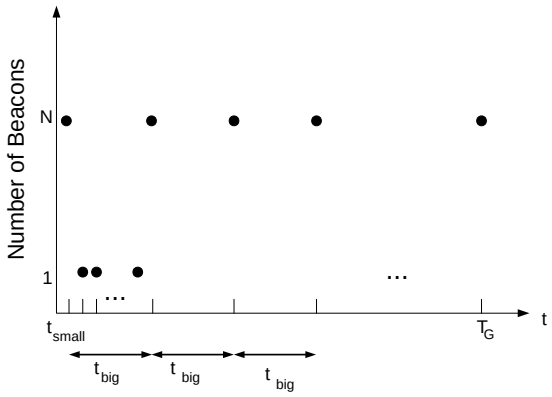

Fig. A.5. CAEGViewP - Worst Case, No. of Beacons at all the nodes.

identifier as the current group view in the first beacon interval, t_{small}. In the best case scenario, two nodes, say i and j, beacon their group views, GV_i and GV_j, faster than others at Time $2t_{small}$. GV_i and GV_j include all of the N nodes identifiers in the network, because they are the second beacons of nodes i and j. After Time $2t_{small}$, every node finds that its neighbor has the same as group view as its own. So every node revise their beacon interval to t_{big}. We denote the current network state by **Group View Stable** state which means that every node has the correct globe GV and changes its current beacon interval is t_{big}. So the total number of beacons sent by all of the nodes is $N + 2 + N\frac{T_G - 2t_{small}}{t_{big}}$ in the best-case scenario as shown in Fig. A.4. In the worst case scenario, a node, say i, beacons GV_i faster than other nodes at Time $2t_{small}$. Similarly to the best case scenario, GV_i includes all of the N nodes identifiers in the network. So except node i, all of the nodes revise their beacon intervals to t_{big}, It means that node i will not receive the other nodes beacon until Time $t_{small}+t_{big}$. After Time $t_{small}+t_{big}$, by hearing other node's beacon, node i knows its neighbor has the same group view. Then node i increases its beacon interval to t_{big}. So the total number of beacons in the worst case scenarios is $N + \frac{t_{big}}{t_{small}} - 1 + N\frac{T_G - t_{big} - t_{small}}{t_{big}}$ as shown in Fig. A.5. Consequently, the total number of beacons is between $N + 2 + N\frac{T_G - 2t_{small}}{t_{big}}$ and $N + \frac{t_{big}}{t_{small}} - 1 + N\frac{T_G - t_{big} - t_{small}}{t_{big}}$.

So if beacon interval of CAEGViewP is equal to the small beacon interval of SViewP, then total number of beacons in CAEGViewP is smaller than in SViewP, i.e. if $t_{small} = \tau$, then $N + \frac{t_{big}}{t_{small}} - 1 + N\frac{T_G - t_{big} - t_{small}}{t_{big}} < N\frac{T_G}{\tau}$. Both CAEGViewP and SViewP need at least a delay of t_{small} to obtain the correct global GV at all the nodes.

So if beacon interval of CAEGViewP is equal to the big beacon interval of SViewP, then total number of beacons in CAEGViewP is larger than in SViewP, i.e. if $t_{big} = \tau$, $N + 2 + N\frac{T_G - 2t_{small}}{t_{big}} > N\frac{T_G}{\tau}$, when $\frac{t_{small}}{t_{big}} < \frac{1}{2}(1 - \frac{2}{N})$. CAEGviewP still only needs at most t_{big} delay to obtain the correct global GV at all the nodes. However, SViewP needs at least t_{big} delay to obtain the correct global GV at all the nodes.

Now let us consider the scenario that a new node joins a the group which is in Group View Stable state. What is the number of additional beacons in CAEGViewP to reach Group View Stable state again? Considering a N nodes group that is in the Group View Stable state, a new node k joins the group. If node k beacons GV_k in the first t_{small} interval, under the best case scenario, both nodes i and j (two nodes of the N nodes) hear GV_k and beacon GV_i and GV_j faster than the other nodes in the second t_{small} interval. GV_i and GV_j include identifiers of node k and all N nodes. Then all of the $N+1$ nodes will enter the Group View Stable state with three more beacons (GV_k, GV_i and GV_j). In the worse case scenario, all of the N nodes hear GV_k and beacon their GVs in the second t_{small} interval. After $N + 1$ beacons (assume that $\frac{t_{big}}{t_{small}} < N$), all of the nodes enters Group View Stable state. So the additional number of messages needed to convert a group back into the Group View Stable state after a new node joins is between 3 and N+1 messages.