Evaluation of an Approach for Distributed Cooperative Reasoning of Global Context States

Alexandre Skyrme
Department of Informatics
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
askyrme@inf.puc-rio.br

Markus Endler
Department of Informatics
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
askyrme@inf.puc-rio.br

***Abstract*—In many occasions of our daily lives, we are willing to spontaneously interact or collaborate with nearby people for sharing ideas, chatting, saving time/money, or helping each other. For that purpose, it is often necessary to identify and reason about shared context situations that are based on distributed sources of local contexts. So far, most of the work that investigates mechanisms to support spontaneous discovery and interaction among mobile users has not thoroughly explored means of automatic detection of common *Global Context States* (GCS).**

In this paper, we discuss a distributed reasoning approach and algorithm that determines a distributed Global Context State among potentially interacting agents. We also evaluate the complexity of the algorithm — through simulation — and identify how the convergence of the algorithm is influenced by users' mobility patterns, the requested minimum number of contributing agents required to conclude a reasoning process and the volatility of each agent's local context.

***Keywords*-reasoning; distributed; context; cooperative;**

I. Introduction

Collaboration is an essential part of our daily lives. People collaborate with each other oftenly and for a number of different reasons, that range from socializing to working cooperatively to achieve a common goal. As mobile devices enjoy increasing popularity, new opportunities arise for pervasive collaboration, especially among people that do not necessarily know each other previously [1]. However, pervasive ad hoc (or spontaneous) collaboration requires means of distributed cooperative reasoning between agents to enable real-time matchmaking among an open group of people [2].

A distributed cooperative reasoning system is typically comprised of independent agents with some means of communication [3]. Each agent runs a reasoning engine, and is capable of inferring or checking a new piece of information that is not explicitly available at its local knowledge base. Communication architectures frequently used in such reasoning systems include blackboard or message passing systems. Blackboard systems rely on a shared data structured (called a blackboard) where an agent can post information, as well as read and act on information posted by other agents. Message passing systems, on the other hand, rely on agents sending messages to other agents and receiving messages from other agents as part of a reasoning process.

Another common distinction in distributed cooperative and rule-based reasoning is the choice between *data partitioning* and *rule partitioning*. In data partitioning, data is distributed among agents and no single agent knows all information available globally; whereas all rules are applied to each data subset. In rule partitioning, rules are split, meaning that each agent only checks parts of the rules against its local data. However, in order to check the original rule (the conjunction of the parts) on the global state of all data available, agents have to exchange data among each other. In this work, we focus on distributed rule-based reasoning with rule partitioning, and how to accomplish it in message passing systems.

An important issue of reasoning in pervasive collaboration applications is the determination of a system's *Global Context State* (GCS) [4], that is simply the combination of all participating agents' *local contexts* at a same instant of time. Concrete examples of such a Global Context State shared among a group of mobile users are: **quality of connectivity -** all user devices are connected through a high-speed wireless connection, enabling them to use a collaboration app with high communication demands; or **location -** all users are located within less then 200 meters from a common meeting place, so that they may gather in a few minutes; or else, **phone settings -** all users have their device set to normal ring-tone, implying that all members of the group are mutually available for some consultation or chatting.

The determination of the Global Context State has several applications in pervasive collaboration. For example, consider the following scenario: a conference attendee is interested in meeting with other nearby attendees who are idle (e.g. waiting in the conference hall) and who share similar interests or expertise. In this scenario, distributed reasoning could be used to compare the local context state of all attendees (i.e. current location, interest/s and expertise) in order to identify which attendees, if any, match the

978-0-7695-4890-6/12 $26.00 © 2012 IEEE
DOI 10.1109/SBSC.2012.8

meeting criteria set by the seeking attendee. This would require the reasoners at the device of each attendee (further called, *Peer Reasoners*) to exchange their local context state information (e.g. current locations) and cooperatively select which attendees match the criteria. We should further consider the existence of an additional reasoner, the *Ambient Reasoner*, which would act as a communication/coordination hub for the reasoning process and also provide public information about the ambient-specific context, such as the conference program, the layout of the rooms, the room-specific planned activities, etc. This "ambient context state", e.g. the current activity in each room, could also be relevant for the matching: for example, only attendees located in the conference hall, the registration desk, or in the restaurant should be considered for the matching.

The above examples of matching criteria can be defined by a Description Logic (DL) rule that would better be split in parts, to be evaluated independently by each of the Peer Reasoners (at the attendee's devices) on one side, and the Ambient Reasoner, on the other side. The former would handle rule parts/predicates more closely related to the attendee's context data and preferences (e.g. current location, if available or busy, affiliation, interests, etc.); the latter would process the parts of the rule that refer to public information (e.g. place-specific activities) and which contain predicates matching issues, such as a precise definition of user co-location or proximity.

In this paper we describe an algorithm used for cooperative detection of distributed Global Context States, that is the basis for distributed cooperative and rule-based reasoning. The main contributions of this paper are to define the concept of rule-based Global Context States (GCS), as well as some required properties and premises, to discuss an algorithm to determine a distributed GCS among potentially interacting agents and to simulate the execution of the proposed algorithm. The paper is structured as follows: in Section II we discuss some related work on distributed (context) reasoning for pervasive applications. In Section III we define the concept of rule-based Global Context State, list some premises and expected properties of possible solutions, as well as present an overview of the algorithm. In Section IV we explain how we simulated the algorithm's execution and present some results of the simulation. Finally, in Section V we present some concluding remarks concerning our approach and future work.

II. Related Work

The present work is largely based on the work by Viterbo and Endler [5], [6], which explores distributed rule-based reasoning (with rule partitioning). It proposes a simple two-tier model comprised of a user/client side and an ambient side and a protocol that must be executed by both sides to converge towards the evaluation of a partitioned rule. It also formalizes the notion of (decentralized) cooperative reasoning, discusses the necessary stability conditions required for convergence. In this work, we extend the aforementioned two-tier approach to work in with an open set of nodes contributing to the reasoning process, i.e. detection of the subset of nodes whose local context satisfies the rule's antecedent.

In [7], Padovitz, Loke and Zaslavsky propose and formalize an approach that enables individual nodes to reason about common context situations by considering distributed information. However, instead of presenting a specific approach for collaborative reasoning, their work's main focus is on context model transformations that allow nodes to obtain a merged perspective of the common context. Since our work does not consider heterogeneous context models, one can see their work as complimentary to ours, since it proposes a solution for merging different context model visions.

In [8], Gu, Pung and Zhang present a peer-to-peer system, where peers are organized according to an ontology based semantic network, to support distributed reasoning for collaborative context-aware applications. The algorithm we present in this paper works in a similar fashion to the aforementioned push mode; it allows for continuous monitoring of context changes in a distributed environment. In our algorithm, however, we're not concerned with intermediate context states or with informing the user of intermediate context updates, but rather with verifying if a certain Global Context State holds while a user is interested in it. Also, in our work we expect all participating nodes to be homogeneous, or to be able to provide the same type of information, while the system presented by Gu, Pung and Zhang could support heterogeneous peers, which have different sensing capabilities, and then have a context interpreter combine the data in order to infer a high-level context. Lastly, their system is fully distributed and thus does not rely on a central entity such as the Ambient Reasoner which is necessary in our algorithm; it is worth noticing, though, that the prototype used to evaluate their system used standard desktop computers to run context producers and interpreters, which suggests these roles may not be suitable for execution in mobile devices.

Distributed Reasoning Architecture for a Galaxy of Ontologies (DRAGO) is a distributed reasoning system implemented as a peer-to-peer architecture, in which every peer registers a set of ontologies and mappings [9]. In DRAGO, the reasoning operations are implemented using local reasoning over each registered ontology and by coordinating with other peers when local ontologies are semantically connected with the ontologies registered in other peers. The reasoning with multiple ontologies is performed by a combination of local reasoning operations, internally executed in each peer for each distinct ontology.

P2P-DR [10] is a system for distributed reasoning focused on Ambient Intelligence (AmI), which uses a peer-

to-peer model and accounts for potential conflicts which might happen during the reasoning process. Each node holds independent (local) information, expressed as rules, and is also able to exchange information with neighbor nodes, by means of *bridging rules*. Potential conflicts which may arise from global consolidation of local information are dealt with by considering bridging rules as defeasible (can be overridden) and defining trust levels between nodes in order to settle disputes between conflicting rules. The authors point out context data can be inconsistent, for instance due to imprecise or faulty sensors, or become ambiguous, when conflicting data is reported by different sources. They highlight that ambient environments host nodes that are heterogeneous and dynamic in nature and thus might not be able to communicate directly or even be aware of all nearby nodes.

A peer-to-peer inference system (P2PIS [11]) is a network of peer theories. Each peer has a finite set of propositional formulas and can be semantically related by sharing variables with other peers. A shared variable between two peers is in the intersection of the vocabularies of the two peers. Not all the variables in common in the vocabularies of two peers have to be shared by them. Besides, two peers may not be aware of all the variables that they have in common but only of some of them. In a P2PIS, no peer has the knowledge of the global P2PIS theory. P2PIS distributed algorithm splits clauses if they share variables of several peers. Each piece of a split clause is then transmitted to the corresponding theory to find its consequences. The consequences that are found for each piece of split clause must then be re-composed to get the consequences of the clause that had been split.

DRAGO, P2P-DR and P2PIS propose distributed reasoning solutions considering data distributed over different elements in an AmI system. The main concern of DRAGO is to reason in distributed environments overcoming the barrier of the heterogeneous knowledge representation that independent entities in a AmI system are very likely to employ. DRAGO relies on predefined mappings to align different ontologies. In a similar way, P2P-DR and P2PIS are peer-to-peer frameworks in which peers can communicate with a subset of the other available peers to import the knowledge necessary to answer queries based on mappings that define how their local knowledge relates to their peers' knowledge. In such way, P2P-DR and P2PIS are capable of performing inference to answer queries that check if a rule is true or false, in which the knowledge, i.e., set of literals that represent context information, is fully distributed in a peer-to-peer system. Nevertheless, P2P-DR and P2PIS are not capable of answering queries with variables. Moreover, DRAGO, P2P-DR and P2PIS are also limited by the fact that in practical implementations of AmI it is not feasible to build in advance mappings of all possible pairs of different ontologies that may be needed. On the other hand, our approach is not a fully decentralized peer-to-peer system, as it relies on the Ambient Reasoner for mediation and coordination. Finally, unlike DRAGO and P2PIS, in our work we do not handle heterogeneous ontologies.

III. Distributed Reasoning of Global Context States

In distributed context-aware applications, pieces of context information may be available to different agents in the system, and it may be necessary to check if a global condition referring to the all, or some, of the agent's local context states is satisfied. Such global condition is commonly expressed as the conjunction of antecedent predicates, R_i, in a Description Logics (DL) rule of the form $R_1 \wedge R_2 \wedge ... \wedge R_k \Rightarrow C$. In such rule, the consequent C is usually an action, and the predicates R_i describe relations between concrete context facts/data at one or more agents, and may also contain free variables (denoted by, $?v, ?w$, etc.) which range over context facts/data of certain type. When a rule like the above is evaluated, these free variables are bound to sets of concrete context facts of any of the agents. For example, if an agent's local context state has only facts *sentMsg(addr1,Hi)*, and *sentMsg(addr2,Hallo)*, then predicate *sentMsg(*$?a, ?msg$*)* would cause the binding of the pair of variables $(?a, ?msg)$ to the set {<*addr1, Hi*>, <*addr2, Hello*>}. The result of evaluating a DL rule with free variables $?v_i$, (i= 1 ... N), is therefore a set of tuples T = $< c_1, ..., c_N >$, where c_i is a concrete context fact, somewhere, that is bound to the rule's free variable $?v_i$, such that the tuple T satisfies the rule's antecedent $R_1 \wedge R_2 \wedge ... \wedge R_k$. Note that if the global context state does not satisfy the rule's antecedent, the set of tuples is empty.

Since the local context state of each agent changes spontaneously, and independently of the other agents' context state, the global condition may be satisfied only for a limited amount of time. Thus, the main objective of *distributed reasoning over Global Context States* is to detect if the Global Context State matches the rule's antecedent part $R_1 \wedge R_2 \wedge ... \wedge R_k$, and if this is the case, to execute the action of the rule's consequent C. For this, the reasoners on the different agents must run a distributed algorithm enabling them to exchange partial reasoning results (the bindings of free variables with their local context facts) until some tuple (i.e. the combination of all partial results) satisfies the rule's antecedent.

In order to illustrate the above concepts, let's look at a very simple example: consider three nodes, n1, n2 and n3, which repeatedly throw individual dice ($d1$, $d2$ and $d3$) asynchronously, i.e. at random instants. Making an association with the aforementioned, the current die number plays the role of the node i context state (which here would be a single, unary tuple (di)). So, a rule to identify the global state in which $d3$'s number is odd, and is the sum of the other two die numbers, would be:

$$\text{even(?d1)} \wedge \text{odd(?d2)} \wedge \text{odd(?d3)} \wedge \text{equals(?d3,?d1+?d2)} \Rightarrow \text{OddSumObtained}$$

In this case, the only tuple set that satisfies this rule is T= {<2, 1, 3>, <4, 1, 5>, <2, 3, 5>}, where the elements of each tuple are T_n1, T_n2, and T_n3, respectively.

In this particular case, we can see that the above rule can be split in a way that each of the reasoners becomes responsible for one of the predicates even/odd() - to be checked whenever the node throws a die - and one reasoner will evaluate predicate equals(). Moreover, the reasoners have to exchange their data whenever some of them detects that a new die throw - a change of its local context state - satisfies its local predicate. In this specific example, only the reasoner with predicate equals() also has to take into account the latest data received from the other two reasoners. However, there are other examples of rules where any partitioning and assignment of predicates to the nodes, causes nodes to share more than just one free variable, so that data does not flow only towards one node, as in the example, but that all nodes have to exchange partial results of their predicate evaluations with some, or all, the other nodes.

However, we must precisely define in which case a Global Context State (GCS) is *considered to occur*. Since the local contexts that constitute a GCS can change in arbitrary and system unknowably ways, we must consider the occurrence of a GCS in relation to real time. This is quite different than the concept of a distributed system's global state [12], [13], which is determined only by program-produced local events at the agents and by events associated to their interactions, and hence there is no concern about time. On the other hand, in the case of GCS, we have to consider that the agents have a notion of time. In particular, all agents must record the run of real time by periodic events, a clock tick, Δt. This assumption does not require the agents to have the same clock tick counter, nor to have their clock tick at the same real time. It just enforces that their Δt do not drift from each other. With this, it is possible to define:

Definition 1: A GCS has occurred **iff** its constituent local states have overlapped during at least $2 * \Delta t$, from the perspective of each agents contributing with a constituent local state.

This definition essentially says that only global states which remain stable for a minimum period of time are *de facto* considered, while a quick overlap of constituent local states should be ignored. The $2 * \Delta t$ limit is required due to the fact that agents may not have their clock ticks synchronized, being incremented exactly at the same moment. In the following section we will show that this $2 * \Delta t$ is a function of the system model parameters.

Based on the definition, we can now discuss the general required properties of any distributed solution for reasoning over Global Context State, and present an overview of the proposed algorithm.

A. Required Properties and Premises

Any solution/algorithm for distributed rule-based reasoning over a Global Context State (GCS) must have the following properties:

Convergence:
if the GCS satisfying the antecedents of a rule R remains stable for a sufficiently long period of time, then eventually the algorithm will evaluate that the rule has been satisfied.

Safety:
if the algorithm detects that the antecedents of rule R have been satisfied, then the corresponding GCS has actually occurred at some moment during the processing of the algorithm.

The convergence property leaves open the possibility that the solution never converges, if the GCS defined by the rule stays valid only for short times (yet longer than $2 * \Delta t$). In such case, due to the required exchange of messages between the reasoners, it may be impossible for them to collectively grasp this short-lived Global Context State. However, it is worth noting that the convergence property does not state that the GCS will be valid at the moment when the algorithm detects it, but that it detects that the GCS has in fact occurred in the past. This detection delay may be inevitable due to the processing and message transmission latencies.

The safety property, on the other hand, requires the solution to be correct in that it never detects a GCS that has never occurred in the sense stated in Definition 1. Since this definition is based on the notion of time, it requires a synchronous model for a distributed system, i.e., a model where the maximum processing time and message transmission latency are constant and well-known. Moreover, we make other assumptions about the distributed system, summarized as follows:

- Each agent is capable of doing all local processing related to one incoming request (i.e. handle any incoming or outgoing messages plus evaluate any DL predicates in regard to its local context state in less than λ time units;
- Agents do not fail, and stationary agents can be found and are always reachable by any other agent. However, the total number of mobile agents is variable;
- Message delivery is reliable and follows FIFO ordering;
- All agents have a unique ID and communication address/endpoint;
- The clocks of all agents do not drift from each other;
- The timer period at each agent (Δt) is much larger than the communication and processing latencies (i.e., $\delta + \lambda \ll \Delta t$);
- Each reasoner on a agent checks its local context state periodically and does this in an atomic way and in a negligible amount of time.
- The periodic check of local context state is stateful,

meaning that it does not just grasp the momentaneous state of local context resources/sensors, but also registers any change of context state that might have occurred since the previous check, even if this change was very quick.

Without this last assumption about statefulness of context probing, it would be impossible to ensure the safety property, i.e., that a Global Context State that did not actually occur - e.g. the overlap was less than $2 * \Delta t$ - would not be detected by the agents. In other words, we need the testimonies of very quick local context changes to invalidate the occurrence of associated Global Context States.

B. Overview of the Algorithm

As previously mentioned, we assume that the system has a single stationary agent which is associated with a place/location (e.g. the conference site), and which will be the communication mediator and coordinator of the reasoning process. This agent will execute the Ambient Reasoner, and for the cooperative reasoning, all mobile agents will interact only with this Ambient Reasoner. From this point on, from our algorithm's perspective, we will refer to mobile agents simply as *peers* or by the name of the corresponding role they play in the algorithm.

The algorithm essentially works as follows: whenever an application on a a Requesting Peer (ReqP) needs to evaluate a Global Context State, it submits the corresponding DL rule to the middleware at this peer[1] . The rule is then partially evaluated at ReqP, taking into account its current local context state, and then forwarded to the Ambient Reasoner with the partial results (i.e. a set of tuples denoted by T_I) produced by ReqP. When this message is received by the Ambient Reasoner, it will also partially evaluate some parts of the rule based on its local context state and the partial results received from ReqP. This results in yet another new set of possible partial results (denoted by T_A), but where some of the rule's free variables (related to the other peers' context facts) are still unbound, i.e. undefined. Then, the Ambient Reasoner broadcasts the rule and the partial results (T_I and T_A) to all other participating peers, requesting them to reply whenever their local context states, together with the partial results T_I and T_A, satisfy the rule antecedents.

Figure 1 shows a diagram which represents the main architectural components used in the algorithm.

IV. Simulation

A. Sinalgo

In order to simulate the proposed algorithm's execution we used Sinalgo [14], a free open-source Java framework for validating and testing network algorithms in mobile networks. Sinalgo allows for quick prototyping of network algorithms in Java, easy extensibility and customization, two and three dimensional network graphs, plus synchronous and asynchronous simulations. It includes a network graph visualization utility which can be used to visually inspect an algorithm's execution steps.

[1]We assume that split of the rule is pre-defined by the application and sent as part of the submission.

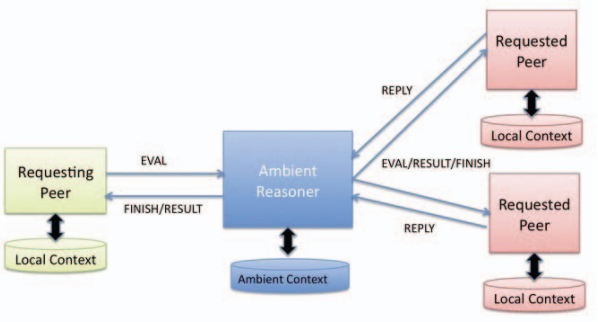

Figure 1: Architectural diagram of the components used in our algorithm.

B. Scenario and Variables

We propose a general simulation scenario similar to the one mentioned in Section I: during a small conference, an attendee is interested in engaging in a group discussion with nearby attendees who share a common interest. For that purpose, the attendee must rely on a reasoning process to assess the interests of other attendees, to determine if there are enough attendees nearby that share a common interest in order to have a group discussion and, in case so, to find out who are these attendees. The attendee could also benefit from knowing if there are any available rooms in order to host the group discussion, although room unavailability will not stop the discussion from happening if there are enough attendees with a common interest nearby.

We assume all attendees are potentially interested in group discussions and, thus, are willing to share their own interests and current locations to support the reasoning process. We also assume attendees can move within the conference grounds, in order to attend sessions in different rooms, and that their interests change as a result of attending sessions about different topics. This effectively means that both attendees positions and interests change throughout time. Room availability to host group discussions also changes throughout time, as conference sessions have different time schedules. In addition, since it is a small conference, we take for granted that the organizers are able to provide reliable connectivity access with a single central communication hub.

In this scenario, the attendee who is interested in finding nearby attendees with a common interest plays the role of the Requesting Peer (ReqP), as it wants to evaluate a Global Context State; the other attendees play the role of Participating Peers (ParP), as they contribute to the

reasoning process with partial results; and finally, the central communication hub provided by the conference organizers plays the role of the Ambient Reasoner, since it is used to mediate and coordinate the reasoning process. According to our algorithm, we assume that for all reasoning purposes, attendees only interact with the central communication hub, and not directly between themselves. Table I summarizes the mapping between the entities involved in the scenario, in our algorithm and in Sinalgo.

Scenario	Algorithm	Sinalgo
Attendee interested in promoting group discussions	Requesting Peer	Node (PeerNode)
Other attendees	Participating Peer	Node (PeerNode)
Central comm. hub	Ambient Reasoner	Node (AmbientReasoner)

Table I: Mapping between the proposed scenario, our algorithm and Sinalgo entities.

The Global Context State looked for in this scenario can be expressed by the following Description Logic rule, which is cooperatively evaluated among the Ambient Reasoner, the Requesting Peer and the Participating Peers:

$$\mathsf{InCenter(?ReqP, ?Region)} \wedge \mathsf{CurrentInterest\ (?ReqP, ?I)} \wedge \mathsf{InsideRegion(?P, ?Region)} \wedge \mathsf{CurrentInterest\ (?P, ?J)} \wedge \mathsf{Equals(?I, ?J)} \wedge \mathsf{Available(?Room)} \Rightarrow \mathsf{NotifyAll(?ReqP, ?P)}$$

In this rule, variables ?ReqP and ?P bind, respectively, to the Requesting Peer and any Participating Peers that are nearby, while ?Region describes the dynamic perimeter region around ?ReqP (as it moves), ?I and ?J bind to the current interest of the Requesting and Participating Peers, respectively, and ?Room binds to the name/number of the rooms that are available at the moment. As rule's predicates are self-explanatory, one could imagine that InCenter() and CurrentInterest() — its 1st occurrence in the rule — are evaluated at the Requesting Peer, while each of the Participating Peers evaluates InsideRegion() and CurrentInterest() — 2nd occurrence — , and Equals() and Available() would be evaluated at the Ambient Reasoner. It is worth mentioning that the above rule is just one of several possible rules that describe the Global Context State. Other, perhaps more detailed, rules could be used instead of this one.

Both types of attendees (Requesting Peer and Participating Peers) and the central communication hub (Ambient Reasoner) are implemented in Sinalgo as standard simulation nodes, each with its own class. Initial node deployment relies on a circle pattern (Circle distribution model) with the central communication hub in the center and attendees around it. Network edges, Sinalgo's abstraction of communication links between nodes that are within communication range, are created only between attendees and the central communication hub. Since we consider the conference's wireless network to be reliable, we do not use interference, we use reliable message delivery (ReliableDelivery reliability model) and constant transmission time for messages (ConstantTime message transmission model in Sinalgo) equal to 1 round.

Attendees' mobility is implemented according to two different models. The first model is Sinalgo's standard RandomWayPoint, which moves each node to a randomly selected waypoint and then waits at that point for a certain amount of time, before choosing another waypoint and repeating the same procedure. Both the movement's speed and the waiting time are determined according to standard Sinalgo distributions (Gaussian and Poisson, respectively). The second model, which we implemented based on the previous model, is called *Random Waypoint with Fixed Meeting Points*. It works in a similar fashion to the standard RandomWayPoint model, however, it allows for up to two fixed meeting points to be configured. When it selects a new waypoint, it has a configurable probability of choosing one of the meeting points instead of a random waypoint. Consequently, it is possible to use this model's configuration in order to increase the likehood of attendees with similar interests getting near each other. The central communication hub is placed at the center of the conference grounds and does not move (NoMobility). All the models used in the simulation are summarized in table II.

Model Type	Model(s) Used
Mobility (Attendees)	RandomWayPoint Random Waypoint w/ Meeting Pts
Mobility (Comm. Hub)	NoMobility
Connectivity	StaticConnectivity
Distribution	Circle
Interference	(not used)
Reliability	ReliableDelivery
Transmission	ConstantTime (= 1)

Table II: Summary of Sinalgo models used in the simulation.

We assume that each attendee switches between three different interests: A, B or C. Interests are chosen randomly and last for a random period of time, measured in simulation rounds, after which they may change. The minimum and maximum durations for interests are configured by custom parameters in Sinalgo's standard configuration file; by default they are set to 25 and 40, respectively. It is also possible to configure the timeout (Δt), measured in simulation rounds, before each entity participating in the reasoning process will need to reevaluate its own context and act accordingly as predicted by the algorithm.

For our simulation we used Sinalgo's standard deployment field, which is 1000x1000, in order to simulate the conference grounds. We consider that an attendee is nearby another attendee when the distance between them is less than

250. In this manner, a group discussion can only happen when enough attendees that share a common interest have gathered within a 250 radius of the attendee who is interested in promoting the discussion. The amount of attendees needed to start a group discussion can be configured by means of establishing the *minimum response bag size*. Moreover, since we are not concerned with short lived Global Context States which might not be detected by the algorithm, we require that attendees must maintain a common interest and stay within range for at least twice the timeout value ($2 * \Delta t$).

We used Sinalgo's standard CustomGlobal class in order to implement a global monitor which detects when the simulation is over, maintains some statistics and tracks the detection delay between reasoning occurs and the simulation finishes. In order to check if the algorithm has finished its execution, it checks at the end of each round if all peers have been notified of the end of the reasoning process. It also keeps track of all Participating Peers by inspecting them at every round; it checks, for each of them, if its interest matched the interest of the Requesting Peer when it was evaluated and if its distance to the Requesting Peer was within the acceptable range when it was evaluated. Doing so allows it to keep track of how long Participating Peers who share an interest with the Requesting Peer and are near it have stayed that way; thus it can determine, externally to the algorithm, when reasoning should become stable.

In figure 2 we present a screenshot of the standard Sinalgo deployment field already loaded with a central communication hub, identified by the blue square in the middle and 10 attendees, identified by the person icon. The circle around attendee 2 identifies it as the Requesting Peer and determines the range of attendees considered to be nearby. In figure 3 we present another screenshot of the simulation, this time after reasoning has finished. The Requesting Peer is identified by the green color, while the nearby peers who share a common interest are identified by the blue color.

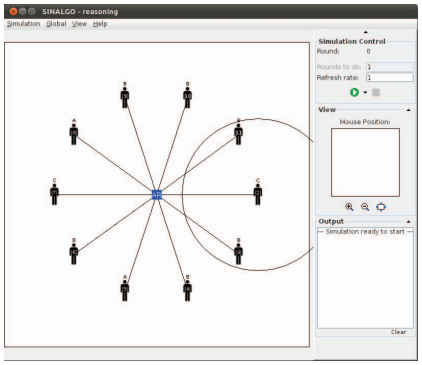

Figure 2: Sinalgo screenshot of reasoning about to start.

We have run simulations in three settings. In each setting we work with a fixed number of attendees (20) excluding the one who originates the reasoning process (Requesting Peer)

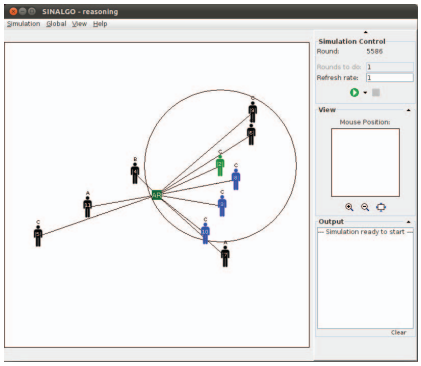

Figure 3: Sinalgo screenshot of finished reasoning.

and we measure the amount of time needed for reasoning to stabilize, the total number of point-to-point messages exchanged and the detection delay (the time between reasoning has stabilized and the algorithm finishes its execution). In the simulation, timeouts (Δt) occur synchronously every 10 rounds; this effectively means that every 10 rounds each participating entity will reevaluate its context and act accordingly. A brief description of each setting follows.

In the first setting, we vary the minimum response bag size based on percentages, in steps of 5% and up to 35%, of the number of attendees (corresponding to response bag sizes from 1 to 7) and the mobility model. We experiment both with the standard RandomWayPoint mobility model, varying the movement's speed, and with the custom Random Waypoint with Fixed Meeting Points model, varying the amount of fixed meeting points available at the conference grounds, but always with a 50% chance of choosing a fixed meeting point instead of a random waypoint. In this setting we are interested in observing how the mobility models, as well as the relation between the number of attendees and the minimum response bag size, influence the simulation.

In the second setting, we vary the range (minimum and maximum values) that defines how long a local context (interest) lasts based on the amount of time required for a Global Context State to be considered stable by the algorithm ($2 * \Delta t$, where $\Delta t = 10$). In this setting we are interested in observing how much local context volatility influences the reasoning process, in particular in terms of the time taken to finish reasoning and the amount of messages exchanged. We use both the Random Waypoint with Fixed Meeting Points mobility model, with one and two meeting points, and the standard RandomWayPoint model.

In the third setting, we vary the probability of attendees heading to a meeting point, from 25% to 75%. In this setting we are interested in observing how much the probability of attendees heading to meeting points and the existence of meeting points influence the reasoning process. We stick to the Random Waypoint with Fixed Meeting Points mobility model, with both one and two meeting points.

C. Results

In this section we present the results for the simulation settings. Each variation of a setting was executed 10 times and the numbers shown in this section correspond to the rounded averages of the executions. We present, for each setting, the simulation duration (measured in Sinalgo rounds) and the number of messages exchanged.

When accounting for messages exchanged during the algorithm's execution, we separate point-to-point messages (i.e., messages sent directly from specific peers to the Ambient Reasoner or in the opposite direction) and broadcast messages (i.e., messages sent from the Ambient Reasoner to all peers). We choose not to present the total number of broadcast messages, as that number is rather deterministic: the Ambient Reasoner only does a broadcast once it receives an EVAL message from the Requesting Peer; the Requesting Peer only sends an EVAL message once it timeouts; thus, we expect broadcasts to occur in regular timeout intervals, which means the total number of broadcasts can be estimated by dividing the simulation duration by the timeout. Indeed, our results showed that assumption to be true.

We also choose not to present the detection delays observed during the simulations. The detection delay is the number of rounds between the moment that the Global Context State becomes stable (i.e., there are enough replies with count $\geq 2 * \Delta t$) and the algorithm finishes executing. Our results showed that it is a constant value (3 rounds), which is the time needed for the Participating Peers to send their last messages to the Ambient Reasoner and for the Ambient Reasoner to send the results to the contributing peers.

1) Simulation Setting 1: Figure 4 presents the simulation results for the first setting, where we vary the minimum response bag size and the mobility model. It shows some of the algorithm's characteristics.

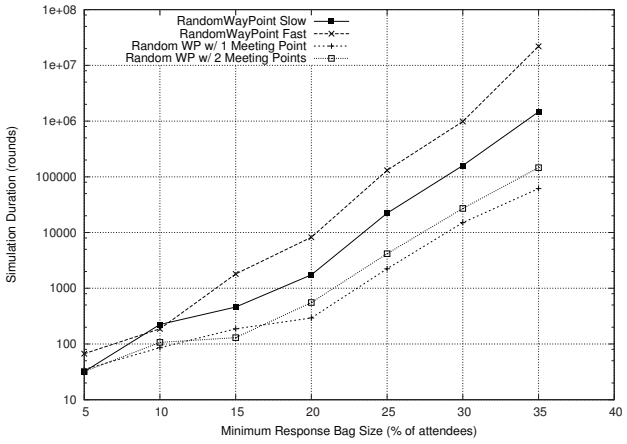

Figure 4: Minimum response bag size versus duration of the simulation for setting 1.

First, and perhaps more obvious, it is clear that as we increase the minimum response bag size, the longer the reasoning process lasts. This is expected, as a greater minimum response bag size demands that more attendees have a similar interest and are near each other. Figure 4 depicts how the minimum response bag size influences the duration of the simulation. It shows that as we linearly increase the minimum response bag size, the duration of the simulation increases exponentially. This implies not only that the minimum response bag size is a key factor for the performance of the algorithm, but also that the algorithm is better suited for reasoning among small groups of peers.

Concerning the mobility models, the results show that the faster the attendees move, the longer the reasoning process lasts, as it is more difficult to get the required number of attendees with a similar interest in the same area. Moreover, the results show that the introduction of meeting points help to promote colocation of attendees, increasing the likehood of attendees with similar interests getting near each other and thus reducing the duration of the reasoning process. This can be observed in figure 4, which shows that having a single meeting point results in a shorter reasoning process than having two meeting points, as attendees tend to gather at a single location and thus the probability that attendees with similar interests will get near each other increases.

2) Simulation Setting 2: Figure 5 depicts how local context volatility influences the duration of the simulation. Consistently with the first setting's results, the number of exchanged point-to-point messages also increases as the duration of the simulation increases. Probably the most interesting pattern which can be observed is that less volatile (longer lasting) local contexts result in shorter reasoning processes than more volatile local contexts. This is most likely due to the fact that the more stable the local contexts are, the more time (and thus the greater the chance) there is that attendees with similar interests will get near each other at some point in time.

3) Simulation Setting 3: Figure 6 presents the simulation results for the third setting. The results for this setting clearly show that adding meeting points to the simulation result in shorter reasoning processes. Moreover, it shows that adding a single meeting point shortens the reasoning process more than adding two meeting points, which is also consistent with the previous settings. This is especially true as we increase the probability of attendees heading to meeting points.

V. Conclusion

In this paper, we have proposed a distributed algorithm for detecting Global Context States (GCS) among an arbitrary set of mobile peers, presented its applicability for cooperative reasoning for pervasive collaborative applications, and have conducted some simulation experiments to evaluate

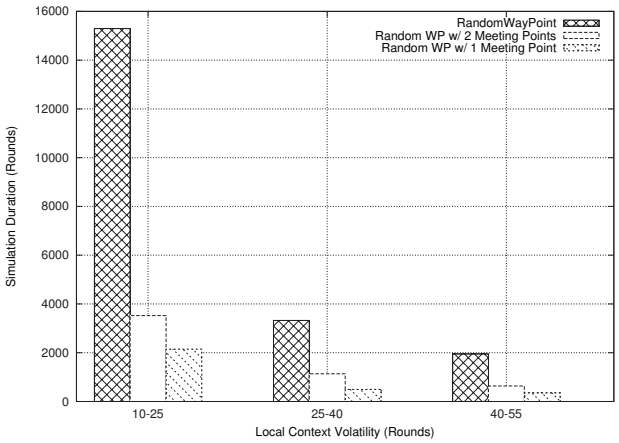

Figure 5: Local context volatility versus duration of the simulation for setting 2.

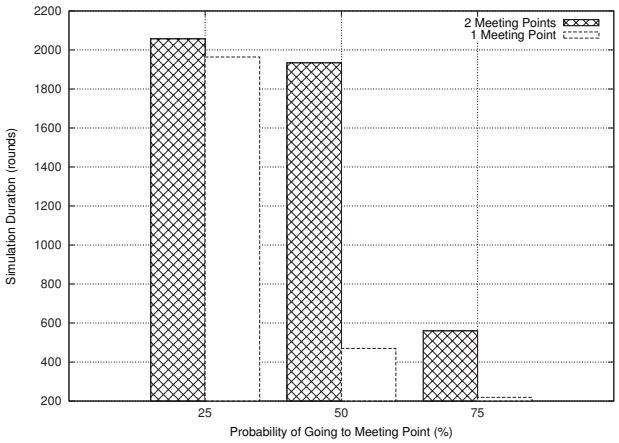

Figure 6: Probability of going to a meeting point versus duration of the simulation, for one and two meeting points.

the performance of the algorithm in different mobility and context volatility scenarios.

The obtained results show that the convergence time of the algorithm grows exponentially with the percentage of peers that are required to contribute to the Global Context State (i.e. the relative size of *response bag*), but is not so much sensitive to context volatility. Moreover, the results show that with more regular mobility patterns, e.g. with some fixed "meeting points", the convergence time drops significantly, and that the communication complexity is proportional to the convergence time, making it feasible for such scenarios with less mobility entropy. We have also identified that the detection delay is very small and almost constant, suggesting that Global Context States will be informed timely to users. Collectively, theses results suggest that the approach can be applied in practice in situations where the percentage of contributing peers is less than 35% of the total number of peers, when there exists some user clustering points, such as meeting points or coffee tables, and the Global Context State is defined by a few context variables, which do not change very frequently.

A limitation of our algorithm is the fact that it requires a careful calibration of the timer period (i.e. Δt), which is inherently dependent on the application domain. However, once the minimum period of stability of a Global Context State to be inferred is determined, then our algorithm does a fairly good job in detecting it. Another point of criticism may be the premise that all peers must adopt exactly the same timer periodicity. However, if we assume that each peer is expected to run the same client program so as to have the ability to perform the decentralized reasoning, then this client would of course use a common timer periodicity. Also, in regard to the assumption that local clocks don't drift from each other, we believe that current processor clock technology is already capable of guaranteeing this property for periods of time which exceed, by large, the time scale of the expected time of use of our system.

It is worth noting also that in our current implementation of the algorithm (used in the simulations) the peers and the Ambient Reasoner perform only the detection of the specific Global Context State of the small conference scenario (Section IV-B). Hence, our implementation does not yet support general purpose reasoning of DL rules – expressing Global Context States — nor the rule splitting and distribution process among the peers. Hence, as part of future research we plan to introduce DL rule processing engines at the peers, and evaluate the performance of the complete reasoning algorithm.

A. Future work

This work is just a first step towards a decentralized reasoning approach, and we envisage several possible lines of future work, both as improvements of the algorithm, as well as in regard to the reasoning approach, as a whole.

Regarding the algorithm, in our simulations we only evaluated its convergence using two simple local context variables: location and a small set of (three) interests. It would be interesting, though, to evaluate the convergence on scenarios where the global context depends on more local context variables, possibly with different volatilities. Still concerning the algorithm's convergence, we showed that the minimum response bag size appears to be central to that matter, however we did not delve into trying to refine and optimize the convergence time, which could result in some improvement to the algorithm. Also, we believe some of the algorithm's premisses could be weakened in order to allow for further testing and adjustments to the algorithm. Examples of weakened premisses include different evaluation timeouts at each peer and local clocks with a small drift. Ultimately, these adjustments could result in an asynchronous algorithm.

Regarding the reasoning approach, we believe it could be interesting to define a fully decentralized approach. In this approach, the Ambient Reasoner could become a role. Thus, instead of relying on a single pre-determined peer, any peer with sufficient resources and low mobility could assume the Ambient Reasoner role throughout the reasoning process.

REFERENCES

[1] R. Aldunate, M. Nussbaum, and R. González, "An agent-based middleware for supporting spontaneous collaboration among co-located, mobile, and not necessarily known people," in *Proceedings of the Workshop on "Ad hoc Communications and Collaboration in Ubiquitous Computing Environments" – ACM Conference on Computer Supported Cooperative Work (CSCW)*, November 2002.

[2] A. Padovitz, S. W. Loke, and A. B. Zaslavsky, "Multiple-agent perspectives in reasoning about situations for context-aware pervasive computing systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 38, no. 4, pp. 729–742, 2008.

[3] E. Rich and K. Knight, *Artificial Intelligence*. Tata McGraw Hill Education Private Limited, 2010.

[4] F. Amigoni, N. Gatti, C. Pinciroli, and M. Roveri, "What planner for ambient intelligence applications?" *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 35, no. 1, pp. 7–21, Jan 2005.

[5] J. Viterbo and M. Endler, "Decentralized reasoning in ambient intelligence," *Software Engineering Workshop (SEW), 2009 33rd Annual IEEE*, vol. 0, pp. 115–124, 2009.

[6] J. V. Filho, "Decentralized reasoning in ambient intelligence," Ph.D. dissertation, Pontifical Catholic University of Rio de Janeiro (PUC-Rio), 2009.

[7] A. Padovitz, S. W. Loke, and A. B. Zaslavsky, "Multiple-agent perspectives in reasoning about situations for context-aware pervasive computing systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 38, no. 4, pp. 729–742, 2008.

[8] T. Gu, H. K. Pung, and D. Zhang, "Peer-to-Peer Context Reasoning in Pervasive Computing Environments," in *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 406–411.

[9] T. A. Serafini, L., "Drago: Distributed reasoning architecture for the semantic web," in *Proceedings of ESWC 2005*, ser. LNCS, no. 3532. Springer, 2005.

[10] A. Bikakis and G. Antoniou, "Distributed Reasoning with Conflicts in an Ambient Peer-to-Peer Setting," in *Constructing Ambient Intelligence*, ser. Communications in Computer and Information Science, M. Mühlhäuser, A. Ferscha, and E. Aitenbichler, Eds. Springer, 2008, vol. 11, pp. 24–33.

[11] P. Chatalic, G. H. Nguyen, and M. C. Rousset, "Reasoning with inconsistencies in propositional peer-to-peer inference systems," in *Proceeding of the 17th European Conference on Artificial Intelligence (ECAI)*. IOS Press, 2006, pp. 352–356.

[12] K. Marzullo and G. Neiger, "Detection of global state predicates," in *Proc. 5th Intl. Workshop on Dist. Algorithms (WDAG)*. Springer, 1991, pp. 254–272.

[13] R. Schwarz and F. Mattern, "Detecting causal relationships in distributed computations: In search of the holy grail," *Distributed Computing*, pp. 149–174, 1994.

[14] "Sinalgo – simulator for network algorithms," Website, ETH Zurich – ITET – TIK – Distributed Computing Group. [Online]. Available: http://disco.ethz.ch/projects/sinalgo