With A Little Help From A Friend: Applying Overhearing To Teamwork

Silvia Rossi[1,2]
[1]Istituto di Cibernetica, CNR
Napoli – Italy
silvia.rossi@dit.unitn.it

Paolo Busetta[2]
[2]ITC-Irst
Trento – Italy
busetta@itc.it

Abstract

Software agents must have some degree of autonomy in order to be able to adjust to changing and sometimes unpredictable situations due to communication problems. Introducing so-called *overhearers* for monitoring team activities and helping recovery from failures seems to be a very promising approach. In this paper, we claim that having a "global" representation of the interaction protocol can be useful to monitor team communication. Moreover, using this global description of group interactions, overhearers can monitor activities without requiring a priori, precise knowledge of which and how many agents are involved. We show how, given a global description of a protocol in terms of involved roles rather than agents, it is possible for an overhearer to monitor its evolution and detect, and recover from, certain types of failures. We evaluated an implementation of the overhearer monitoring a group of agents executing a Contract-Net Protocol.

1 Introduction

In our increasingly complex and networked world, software agents must have some degree of autonomy in order to be able to adjust to changing and sometimes unpredictable situations. We assume, by principle, that a software agent cannot be really autonomous without understanding its "surrounding" context. This necessarily implies that the agent must be able to "oversee" and "overhear" what is happening in its environment, extract what is useful and filter out what is not, without a-priori restricting its view or deciding what should be part of the context it has to know and what not. Moreover, in large distributed system like the Internet, communication problems and unpredictable hosts can cause agents to die or messages to be delayed or lost. The same happens when agents are used for distributed ubiquitous applications with communication infrastructures (e.g., Wi-Fi or Bluetooth) inherently unreliable. So, the increasing use of the multi-agent paradigm in networking and ubiquitous computing calls for increasing robustness, for instance when failures occur during coordination activities or teamwork; in these cases, agents should be able to recognize the disagreements they caused and somehow recover, without necessarily replicating messages.

Handling failures increases the burden of agent developers; they have to anticipate and program the agents for all the possible exceptions not just according to their own possible failures but also according to the failures of any other agent involved in a team. Inherently robust distributed computational models exists (e.g., nested transactions [Busetta *et al.*, 2003]) but they are not suitable to all types of problems and environments. Moreover, some agents may be computationally limited; consider, for instance, wearable wireless computers with limited power.

In the current literature on multi-agent systems, the role of overhearing to monitor distributed systems is gaining attention, as demonstrated by some recent works, e.g. [Gutnik and Kaminka, 2004; Nair *et al.*, 2004; Legras and Tessier, 2003; Kaminka *et al.*, 2002]. Introducing so called *overhearer* agents for monitoring team activities and helping recovery from failures seems to be a very promising approach, but anyway it sets challenges to the agent community. Overhearers may detect certain message losses [Busetta *et al.*, 2001; Rossi and Busetta, 2004; Gutnik and Kaminka, 2004], and act in order to pro-actively provide suggestions, influence behaviors and recover from failures. But, in order to do that, the overhearer must have a model of the group interactions [Rossi and Busetta, 2004] (for example of the protocols of communication) or a public specification of each agent's behaviors [Busetta *et al.*, 2001].

To prove whether the overhearing principle can lead to a practical software development methodology, we have been exploring group communication. This means that every message is received by many partners simultaneously, some of which may be unknown to the sender, as an alternative to traditional direct point-to-point interactions among pairs. Group communication is a means for providing multi-point communication by organizing the receivers (and possibly the senders) of a message in groups [Rossi *et al.*, 2005]. Group communication is common in collaborative applications of all kinds, such as server replication, clustering, grid computing, distributed transactions and database replication, distance learning, drawing on a shared white-board, video conferences, application sharing, distributed interactive simulations, on-line games and financial markets. It is also worth noticing that, in a number of foreseeable cases – such as in-

telligent buildings filled with people carrying wearable and mobile computers of all kinds –, the number of agents can be fairly large, unknown and continuously changing.

We claim that having a "global" representation of the protocol (i.e., not from the perspective of a single agent but of all the participants as a whole) can be useful to monitor complex team activities. Moreover, we aim at a global description of group interactions (i.e. group communication protocols) without requiring precise knowledge of which and how many agents are involved. The Joint Intention Theory [Levesque *et al.*, 1990] and the landmark based approach [Kumar *et al.*, 2002] support a high level formalization of this kind that is logically well founded. Once that this is given, we can think of dealing with failures and message losses at a group level. In this sense we talk of *group coherence* in contrast with *single agent coherence* with respect to a protocol.

In this paper, we show how, given a global description of a protocol using JIT in terms of involved *roles* rather than agents, it is possible for an overhearer to monitor its evolution and detect, and recover from, certain types of failures (Section 2). We propose a computational model for an agent acting as an overhearer (Section 3). We demonstrate its use in the case of agents coordinating via the Contract Net Protocol (Section 4); for practical purposes, the overhearer is a specialized member of the team, but in principle nothing prevents any other from overhearing, too. We conclude with a short review of related works (Section 5) and draw some conclusions and directions for future works (Section 6).

2 Group Protocols vs. Pairwise Protocols

The increasing usage of multi-agents systems in complex applications commonly leads to situations where conversations and interactions involve more than two agents. However, most of the research on communication protocols and infrastructure is still focused in modeling multiparty communication as conversations between two parties at a time. Artificial agents should be able to communicate with groups as well as individuals, where group communication is not only intended as the possibility to address messages to many individual addresses but also as the possibility to address messages to groups without knowing which are the potential recipients [Kumar *et al.*, 2000]. A typical scenario is broadcast communication, but current mainstream agent communication languages and Web services lack adequate support for broadcasting; publish/subscribe services move in that direction, but imply the presence of specialized middleware, require additional operations such as registration to a server, and are typically implemented as pair-wise communication (event producer to server, server to each event consumer), thus partially missing the advantages of specialized group communication protocols. For example, groupware applications such as webcasting often use IP multicast to transmit data to all group members using minimum resources. Efficiency is achieved because the message needs to be transmitted once. In such situations, updating mutual beliefs (e.g. about a change of state of a peer and its consequences on all others) by exchanging pair-wise communicative acts requires $\mathcal{O}(n^2)$ messages. By contrast, with messages addressed to groups we would have just $\mathcal{O}(n)$ messages.

As highlighted in [Mazouzi *et al.*, 2002], common formalisms take into account sequential processes only and are limited in dealing with concurrency of interactions. We would like to have a way of representing global description of the group interaction (i.e. group communication protocol) without having the knowledge of which and how many are the agents involved into the interaction. The only knowledge that we want to express is about the main roles involved into the interaction [Rossi and Busetta, 2004]. In the representation of conversation protocols by [Kumar *et al.*, 2002], one way of specifying protocols is to specify just a partially ordered set of states (called *Landmarks*) instead of state transitions. These states can be intended as the subgoals that the group has to achieve in order to reach the global goal, and can be characterized by a conjunction of Joint Intention Theory [Levesque *et al.*, 1990] formulas.

Semantics of the FIPA communicative acts imposes the precondition that the sender has certain beliefs about the mental state of the (well known) addressee. Consequently, there is no way to send messages to unknown agents. The group extension of many of the communicative acts made by [Kumar *et al.*, 2000; Rossi *et al.*, 2005] is sufficient to model one-to-many communicative interactions. Problems arise when trying to describe many-to-one interactions (unknown numbers of agents perform different communicative acts), or the even worse scenario of many-to-many interactions. The main difficulty is that we do not know beforehand the order in which the different senders perform their respective communicative acts, so we cannot specify the various communicative acts as a joint action expression. However, in our landmark-based protocol formalization, we have that there are no dependencies among the messages exchanged for going from a landmark to another. Roughly speaking, this is to say that to the transitions, in our landmark representation, correspond sets of independents messages, or that we have a partial order (a dependence rule) on sets of mutually independent actions. So, in the landmark representation we have that:

- For going from the current state to the next state, the group can exchange n messages from m different agents. For example, in the Contract-Net Protocol (CNP) (as formalized by Kumar et al., see [Kumar *et al.*, 2002]), each member of the group can send his own bid and all these messages yields to only one state transition.
- For going from the current landmark to the next landmark, the agents can exchange different type of messages. For example in the CNP the transition to the final state, where the job cannot be done or the escape condition is true, can be performed by exchanging different messages. In other words, different actions can lead to the same result. Another example is in our representation of the CNP (see Sec. 4) where in one of the transition the bidder can send an "ACCEPT" plus some "REJECTs".
- In computing the next expected state, one single message can be compatible with more than one next state. This means that, when dealing with a flexible and fault tolerant protocol, actions can be ambiguous. For exam-

ple in our representation of the CNP a single "REJECT" message may not be sufficient to understand which landmark will be the next one.

2.1 Global Coherence vs. Local Coherence

As we said above, researchers have mainly concentrated to modeling dialogue from the perspective of a single agent involved in a two-party dialogue at the time; a similar thing happens in software engineering concerning agent behaviors and recovering from failures. An agent (or in general a system) is considered to be closed in nature, and hence researchers have mainly focused in achieving local coherence and managing failures from a single agent perspective. This means that the intended functionality is designed by looking only at local behavior, i.e. to each agent and its interaction with others taken individually.

However, looking at group behavior, agents that are coherent from their own perspective and their own protocol may be incoherent with the behavior of their group as a whole when observed from the outside. This can happen, for example, if an agent loses a message or just for reasons of timing and resource availability. With the increasing number and complexity of multi-agent applications in sensitive domains, such as assistance to disabilities and accident prevention in smart homes, it becomes increasingly important that the entire system exhibits a "global coherence". This is different from the notion of emergent behavior, because we attribute intentionality to agents. In other words, while an emergent behavior is in the eye of the beholder (i.e., attributed by a third party to the group of agents it is observing), we want that each agent attempts to reach a local state (corresponding to achieving a goal or accomplishing some task) which is coherent with the global state (including goals and intentions) of the group it is part of, and adapts when the latter changes.

The notion of global, as opposed to local, coherence focuses on the idea that there exist certain structures of relations between entities in an environment. These structures can evolve over time and can emerge as a consequence of interactions among the single agents. Ensuring that a multi-agent system (MAS) exhibits coherent collective behavior is a challenge, because most MAS's lack of a systematic way to maintain an updated global perspective on their evolution. In this global perspective, coherence is a property of a MAS that could be measured as the ability of the system (as a whole) to recover from local failures. To this end, what is needed is a shift of focus from the local perspective to a more general, global one. We need a way to monitor the global behavior and the global coherence of a multi-agent system in order to be able to recover a single agent's local failure in the context of its group activity. Our claim is that it is not possible to make recovery on complex interactions without having such global perspective. In other words, only by recognizing the group's collective behavior we can understand which agents are out of sync with the group and re-align them with the global state.

3 The Overhearer Computational Model

In our approach, we assume that an overhearer wants to recognize social roles in order to pro-actively provide suggestions, influence behaviors [Conte and Dignum, 2001] and help recovering from failures. The general computational model we propose for an overhearer is composed of three main concurrent processes (see Fig. 1), and it is a direct extension of the computational model presented in [Rossi and Busetta, 2004]. In this previous work, we designed a simple language that allows the definition of the interaction knowledge meant as two types of rules. The first type associates a message with the current state of the group and the intended effects of such message, and the second associates a message not only with a current state of the group but also with the social roles of the sender and intended receivers.

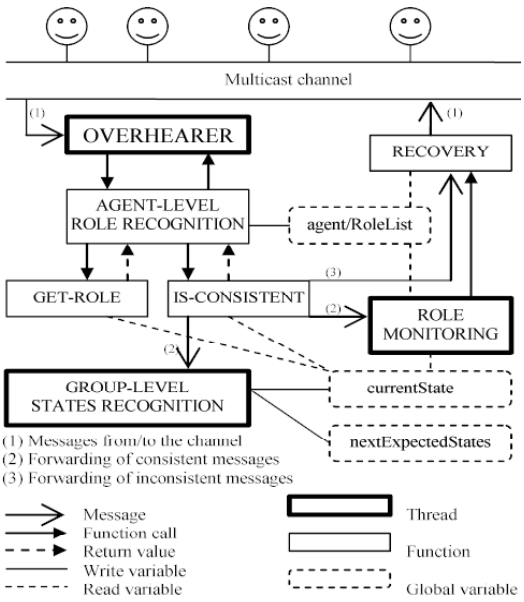


Figure 1: Computational model for the Overhearer.

We distinguish a main `overhearing` module from a module dedicated to the recognition of the current state (landmark) of the group and the possible next states through the application of the message rules. Goals of `overhearing` are: (1) to handle a list of the agents within the organization; (2) to assign to each agent and to modify, during the interaction, a particular social role. This process is achieved by using two external functions: `is-consistent` and `get-role`. The first function checks the global consistency of a message with the current state of the organization and with the possible next states. If the message is consistent, it is forwarded to the `group-level` module. The second function recognizes the role played by the agent by applying social rules. Both functions need to know the current state of the organization. This value (as well as the possible next states) is maintained in a global variable, so that it can be read by these functions and written by `group-level` (we assume that any problem of concurrency is handled by the implementation).

In order to recover from message loss or delay, we added a process for monitoring the behavior of the team of agent. The `role monitoring` process receives from `overhearer` only the messages that are consistent from the global point

Algorithm 1 MONITORING

▷ *AGENTS* $\equiv \{(A_i, srole) | A_i \in$ *Agent-identifier, srole* $\in$ *SRoles*$\}$*: known agent/social-role couples*
▷ s_i *= Current state of the organization*
▷ $W_l = \{A_i\}$ *waiting list for the current landmark*
▷ *bufferOfMessages* = $\{ m_i | m_i \in messages \}$*: Buffer of consistent messages*

1 **repeat**
2 $s_j = s_i$
3 $W_l = \emptyset$
4 **repeat**
$(A_j, r_j) =$ FIRST-OF($\{(A_i, srole)\}$)
if EXIST-S-RULE(r_j, s_j)
$W_l = W_l + (A_j)$
5 **until** $\{(A_i, srole)\} = \emptyset$.
▷ *Create the waiting list for the current Landmark from the Agent/Role list according to the SocialRoles Rules*
▷ *Let's notice that if the waiting list is empty we don't have information for making monitoring activities or we are at the beginning of the protocol*
6 **repeat**
m_i = DEQUEUE-MESSAGE;
A_i = GET-AGENT($\{m_i\}$);
REMOVE(A_i, W_l);
▷ *Remove the agent from the waiting list (if it is present)*
until TIMEOUT $\vee$ $W_l = \emptyset$ $\vee$ $s_j \neq s_i$
7 **if** $W_l \neq \emptyset$
NEW-RECOVERY(W_l)
8 **until** STOP-EXECUTION.

Figure 2: The Monitoring Process.

of view. Roughly speaking, the monitoring process consists of two main activities (see Fig. 2). The first is to create a "waiting list" of the agents that are expected to send a message in the current state of the group. This list is created according to the list of agents and roles created by the `agent-level` recognition process. The second activity consists in updating such list when the monitor receives a coherent message. Every time there is a change of state (as recognized by `group-level`) or a timeout expires, the monitor checks the waiting list and, if this is not empty, it sends the list of agents to the `recovery` module.

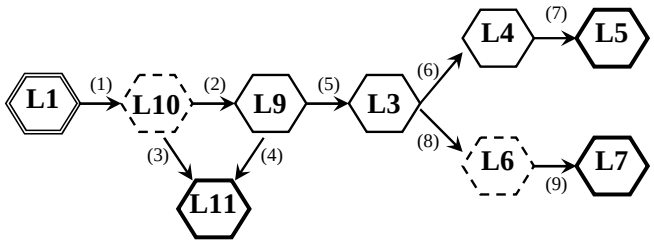
`Recovery` has the only function of sending error notifications to the agents. When an agent receives an error notification, it has to resend the last message to the specified recipient. Note that the agents involved in the interaction do not have to store all the messages they send; the only ones they are required to store are the messages they sent during the last transition of landmark. This is motivated by the fact that, at each landmark, the overhearer checks the global consistency of the state of the agents, and so the possible inconsistencies are evaluated and corrected at each transition. The recovery mechanism is applied in two cases: 1) the overheard message is inconsistent with the current landmark; and, 2) the monitoring process was expecting a message from an agent x, but it received nothing. However, suspecting a possible inconsistency or a crash is not the same as actually detecting a crash; the suspected agent may have sent the message but only the overhearer has lost it. In a real network, it is impossible to achieve any deterministic decision about the reason for not overhearing a message. Moreover, intervening if the overhearer was the only agent losing a message would increase the agent communication for no real reason. Given these considerations, the overhearer decides whether to intervene or not according to the social role of the agent involved. This means that, if the agent has an important role for the accomplishment of the protocol, the overhearer intervenes immediately, otherwise it waits to catch any inconsistencies until the following message; agents suspected of inconsistency are added to the waiting list.

4 An Example: Monitoring on the Contract Net Protocol

The purpose of this experiment was to evaluate how the knowledge about the roles influences the performance of the monitoring process of multi-agent systems when there are message losses. To do so, we set up a first experimentation made on a group of agents that use a group version of the well known Contract Net Protocol (CNP), used to allocate a task for execution to a member of a group of agents. The "call for proposal" (CFP) (Fig. 3(1)) is clearly a case of group communication – there are multiple intended recipients, so the intended recipient is a group, and moreover, the sender may not know which agents will respond so the intended actors are unknown. By contrast, during the PROPOSAL stage (Fig. 3(2)), we have many-to-one communication, i.e. each agent in the group can send its own proposal to the contractor agent, but we do not know in which order the agents will send their messages. Moreover, more than one proposal can be accepted. This fact may be motivated by the need for having reliable information after the computation is done, so, in order to deal with a single point of failure, it is preferable to have more than one machine performing the same task.

To recognize or to model simple patterns of interaction (i.e. natural policies or protocols), one of the most obvious ways to proceed is, as we see from the current literature, to model them as Finite States Machines. A finite state machine is an abstract machine consisting of a set of states, a set of input events and a state transition function. Recently, Kumar et al. [Kumar *et al.*, 2002] and Chopra et Singh [Chopra and Singh, 2004] pointed out that a state machine representation of a protocol has many limitations. The main criticism is that, by labelling arcs with communicative actions, these actions are fixed, so in open multi-agent systems, where agents are autonomous and heterogeneous, such agents cannot handle exceptions and opportunities. Also, there can be several actions (communicative and non-communicative) that can lead to the same result [Kumar *et al.*, 2002]. It is easy to see that this kind of representation leads to a huge number of possible states, because the number of the possible interactions grows exponentially with the number of group members. Finally, we can be also in a situation where we do not exactly know who such members are and so we cannot describe the states of the group as a product of their finite state machines.

In the representation of conversation protocols made by [Kumar *et al.*, 2002], instead of specifying the state transitions, the authors specify just a partially ordered set of states, called landmarks. We started from the landmark representation of the CNP [Kumar *et al.*, 2002] and we made a few extensions in order to obtain a group protocol (see Fig 3). As



(1) $(CFP\ \alpha\ \beta\ \gamma\ a\ \varphi\ )$

(2) $[\forall x\ \gamma_{propose}(x) \supset (PROPOSE\ x\ \beta\ \alpha\ a\ \phi)] \vee timeout$

(3) $timeout$

(4) $[\forall x\ \gamma_{propose}(x) \supset (REJECT\ \alpha\ \beta\ x\ \phi)] \vee$
$[(INFORM\ \alpha\ \beta\ \gamma_{propose}\ \neg\phi)$

(5) $[\forall x\ \gamma_{accept}(x) \supset (ACCEPT\ \alpha\ \beta\ x)] \wedge$
$[\forall x\ \gamma_{reject}(x) \supset (REJECT\ \alpha\ \beta\ x)]$

(6) $\forall x\ \gamma_{accept}(x) \supset (INFORM\ x\ \beta\ \alpha\ (DONE\ x\ a))$

(7) $(INFORM\ \alpha\ \beta\ \gamma_{accept}\ (BEL\ \alpha\ (DONE\ \gamma_{accept}\ a)))$

(8) $[\forall x\ \gamma_{accept}(x) \supset (INFORM\ x\ \beta\ \alpha\ \neg\varphi)] \vee$
$[\forall x\ \gamma_{accept}(x) \supset (INFORM\ x\ \beta\ \alpha\ \Box\neg(DONE\ x\ a))] \vee$
$[(CANCEL\ \alpha\ \beta\ \gamma_{accept}\ a\ \varphi)] \vee$
$[(INFORM\ \alpha\ \beta\ \gamma_{accept}\ \neg\varphi)] \vee$
$[(INFORM\ \alpha\ \beta\ \gamma_{accept}\ \Box\neg(DONE\ \gamma_{accept}\ a))$

(9) $[(INFORM\ \alpha\ \beta\ \gamma_{accept}\ (BEL\ \alpha\ \neg\varphi))] \vee$
$[(INFORM\ \alpha\ \beta\ \gamma_{accept}$
$(BEL\ \alpha\ \Box\neg(DONE\ \gamma_{accept}\ a)))] \vee$
$[\forall x\ \gamma_{accept}(x) \supset (INFORM\ x\ \beta\ \alpha\ (BEL\ x\ \neg\varphi))] \vee$
$[\forall x\ \gamma_{accept}(x) \supset (INFORM\ x\ \beta\ \alpha$
$(BEL\ x\ \Box\neg(DONE\ \gamma_{accept}\ a))]$

Figure 3: Landmark representation of the CNP.

previously mentioned, the landmark approach is a formalism within the framework of Joint Intention Theory (JIT), where conversation protocols are regarded as having an associated goal that the agents are meant to achieve. A landmark is characterized by the propositions that are true in the state represented by that landmark.

In this paper, for the sake of simplicity, we do not present the complete formalization of the CNP using the group landmark extension, although we provide a graphical representation of the state (see Fig 3) and some of the possible communicative acts in order to make a transition from a state to another state. The semantics of these communicative acts is a direct extension of the semantic presented in [Kumar *et al.*, 2002], as made in [Rossi *et al.*, 2005]. The terms α, β, and γ in the communicative acts can represent either groups or individuals. If $a(\alpha, \beta, \gamma)$ is a communicative act, α is the entity performing the act, β is the recipient (including the overhearers) of the request message, and γ is the intended actor [Kumar *et al.*, 2000; Rossi *et al.*, 2005]. A group is defined by a characteristic function such as the membership property. This can be captured by a predicate consisting of a free variable that ranges over individuals. With Greek letters we will represent groups' names, and we use the same symbol in a functional notation to denote the associated membership predicate. For example, γ is a group having the membership predicate $\gamma(x)$ where x is a free variable. The group property for the group $\gamma_{propose}$ specifies the agents who chose to propose from a rational choice between proposing or not. The group $\gamma_{propose}$ is also specified by noting that it consists of those agents who performed an PROPOSE within the specified timeout period after the original CFP was performed. In either case, the group predicate is evaluated retrospectively, i.e. by looking backwards from a future point in time to determine which agents "proposed". The group γ_{accept} is specified by noting that it consists of those agents who performed a PROPOSE within the specified timeout period after the original CFP was performed, and whose proposal is accepted. Similarly, the group property for γ_{reject} specifies the agents whose proposal was rejected after the original PROPOSE was performed. In this case, the groups γ_{accept} and γ_{reject} are dynamically created by α (who sent the CFP) when it sends REJECTs and ACCEPTs. Note that, in order to establish a joint commitment or to discharge a precedent commitment, the entity α has to send to each agent that made a PROPOSAL either a REJECT or an ACCEPT ($\gamma_{propose} \equiv \gamma_{accept} \cup \gamma_{reject}$).

This kind of formalization allows us to reason about group of agents without knowing a priori who and how many are such members. In this sense it is impossible to model our protocol by legal combination of individual agent states. Indeed, let us stress again that we do not know how many agents are involved, and so how many states. To further enforce this point, let us take state L3 as an example. L3 is reached by sending a number of ACCEPTs and a number of REJECTs from the contractor to all the agent that made a PROPOSE; since the number of accepted agents (which correspond a particular internal state of the contractor) and the number of the rejected agents (which correspond a different internal state) are chosen dynamically, we cannot imagine to have the representation of the global state just as a cartesian product of the agent internal states.

4.1 Experimental Results

We performed experiments on monitoring a group of agents' execution of the Contract-Net Protocol, implemented in Java. We use a multicast communication infrastructure LoudVoice [Busetta *et al.*, 2002] to support the communication needs of the agents. LoudVoice uses the fast but inherently unreliable IP multicast and XML for message encoding. LoudVoice is language-independent: we currently have a Java implementation of the API and a beta version of the C# porting that runs both on handheld devices and PCs.

In the first experiment we considered the case where only the overhearer loses messages. If the number of lost messages (n) is less than the number of messages required to make a transition from a landmark to one of its successors (m), in our approach the overhearer can follow the behavior of the group; the computational complexity of the problem is linear and depends of the number of messages and social rules. We can demonstrate that the computational complexity of our algorithms, when the number of lost sequential messages is more than the one required for making a transition,

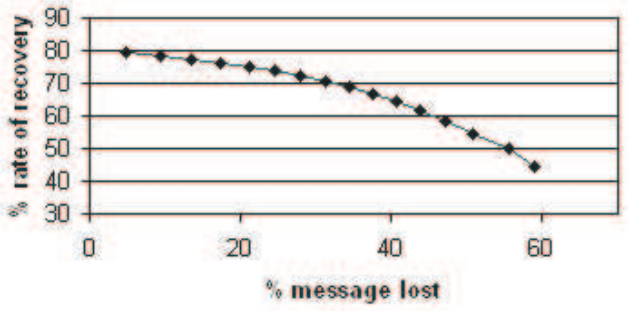


Figure 4: Overhearing recovery rate in varying the percentage of lost messages.

is equivalent of the complexity obtained using a final state machine representation.

In the second experiment, we evaluated the performance of our monitor process. To this end, we assumed that the overhearer does not lose any message. We ran our experiments on a local network, and so we added a random error generator within each agent in the group and we modified the error rate from 10% of messages lost up to 50%. We considered only errors in the process of sending a message; this means that when an error occurs all the members of the group do not receive such message. Of course in a real network it may happen that a message is received only by part of the group, but we left this experimentation to future works. Let us just observe that a message loss by a subgroup composed of l agents is equivalent to l message losses in pairwise communication; the same considerations made above about the limitations of modeling group states as cartesian product of individual states apply here. Getting back to our experiment, our group was composed by 20 agent and we randomly generated the number of agent to be accepted and the number of agent to be rejected. For each rate of error, we performed 50 runs and we measured the total numbers of messages sent, lost, and recovered. From the analysis of our results, we had that the recovery rate (the number of recovered messages divided by the number of lost messages) decreases with the increment of the percentage of lost messages. In particular, starting with a performance of 80% at the rate of 10% of lost messages, we arrive at 50% when the rate of message lost is 60% (Fig. 4).

This type of behavior in the performance of the monitor process depends on the fact that the overhearer does not know who and how many are the members of the group. In fact, a strong impact on the overall performance of the monitor process is due to the loss of the first message of an agent. If the overhearer loses the first message (i.e., the PROPOSE), it cannot recognize the role of that agent. Moreover, the agent will be not included in the interaction by the group (they also lost the message) and the overhearer will have no chance to recover from the failure. By contrast, if we consider the performance of the monitoring process on the subgroup of agents that starts correctly the interaction, the overhearer can recognize and recovery from every failures. This limitation feeds back into the criteria for group protocol design when overhearing is adopted: the first message sent by an agent should be duplicated or somehow recovery mechanisms must be built into the protocol itself to avoid that an agent is simply kept out of the group. For instance, in a practical case (the policy negotiation protocol described in [Busetta *et al.*, 2004]), the protocol is inherently redundant. Its goal is to reach an agreement on a set of objects known by everybody (policies, in this instance), and this is obtained by having each agent repeating multiple times (at least twice for the protocol to work in absence of message loss) what it thinks being the set of common objects and which agents they refer to. Notwithstanding this redundancy, the total number of messages exchanged is well below what would have been required by pairwise communication.

5 Related Works

The work of [Paurobally *et al.*, 2003] also analyzes the problem of consistency for a group that follows a protocol. In particular, this work is focused on ensuring the consistency of mutual beliefs among a group of agents when the communication is faulty. As in our approach, the state of the interaction is represented by the propositions believed by all the agents. The main difference between this work and ours is in the way of achieving such consistency. While in the former coherence is achieved by a synchronization protocol (which consists of repeating messages) and the detection of communication failures is provided by the lower communication layer of the infrastructure, in ours the detection of incoherence is kept at the agent reasoning level by the overhearer. Moreover, mutual beliefs among agents are achieved by defaults and so we do not need the repetition of messages except in the case when we have the detection of an inconsistency. In this sense, Paurobally's approach is more related to the work of [Busetta *et al.*, 2004] where the protocols for the group of agents are derived from a high level representation of landmarks, and the consistency of the group beliefs is achieved by periodically sending some particular messages (called "reminders") whose only purpose is, indeed, stating what an agent thinks is the groups' perspective. Finally, in the protocol formalization made in [Paurobally *et al.*, 2004], the members of the group have to be known a priori, and communication is still single agent to single agent.

Kaminka and Tambe in [Kaminka and Tambe, 2000] analyzed the concept of socially-attentive monitoring for teamwork. They focus on detecting failures in the social relationships that have to hold among agents in a team. Even if the work of Kaminka and Tambe is mainly concerned with plan recognition and not with communication protocols, it is related to ours because social relationships which should hold among the team members enable the generation of expected ideal behaviors of the agents, and can help in selecting the agent to be monitored according to their roles. The notion of consistency among the beliefs of the agents involved in the teamwork is also present.

6 Discussion and Future Works

We have briefly presented a way to describe interaction protocols that enables overhearing for monitoring the progress of the protocols themselves when the number of the agents involved is not known a priori. Protocol representations of the

interaction from the perspective of single agent (i.e. "local" perspectives) differ from "global" perspectives. In the local perspective, the common approach is to define a finite state machine for each agent and then, if we want a global view of the process, we need to find a way to interconnect such machines in one global, large machine. Assuming that such formalization is actually possible for large groups, we foresee that its representation leads to a huge number of possible states, because the number of the possible interactions grows exponentially with the number of group members. Finally, we can be also in a situation where we do not exactly know how many the members are and so we cannot describe the states of the group as a product of their finite state machines.

We described monitoring for systems in which we do not have direct access to an agent's internal state, but we can overhear all the messages exchanged through a multicast channel. For this reason, our protocols are modeled as joint states of the interaction by means of a landmark-based approach and some simple extensions to Joint Intention Theory formulas. In actual applications, the recognition of social roles enables the overhearer to detect some faults of agents due to message losses. The key novelty in this paper consists in achieving group monitoring while the number and the members of the group are unknown a priori thanks to the dynamic recognition of social roles. The use of group communication allows us to model different types of teams as in [Rossi *et al.*, 2005] but is still an open issue for agent communication languages. Finally, the use of JIT and of the landmark formalization allow us to represent interaction protocols in a compact and logically well founded way, since we can represent communication among groups and, by using groups, the roles involved within the interaction from a global point of view.

The problem of monitoring multi-agent systems when the number of agent increases will unavoidably lead to a high computation complexity. So, implementations of monitoring processes can require significant computation and communication overhead, which prevents them for being effective as the number of agents is scaled up or the number of failures increases. In order or overcome these problems, some issues regarding selection mechanisms have to be faced. If we are able to dynamically associate roles to agents, and so we are able to make some predictions about the behaviors of the latter, we expect to be able to make a selection of the events expected from specific agents and of the events to analyze on behalf of their roles. For example, in some state transitions a message loss by the overhearer (which may well be voluntary, i.e. when the agent is overrun by the amount of communication and has to discard input) has no effect on the ability of tracking the group state. Moreover, we argue – and will attempt to demonstrate in future work – that the overhearer, once that the social roles are recognized, can focus on monitoring only particular agents whose roles are deemed critical for the development of the interaction.

References

[Busetta *et al.*, 2001] P. Busetta, L. Serafini, D. Singh, and F. Zini. Extending Multi-Agent Cooperation by Overhearing. In *Proceedings of the Sixth International Conference on Cooperative Information Systems (CoopIS 2001)*, Trento, Italy, 2001.

[Busetta *et al.*, 2002] P. Busetta, A. Doná, and M. Nori. Channeled multicast for group communications. In *Proceedings of the first international joint conf. on Autonomous agents and multiagent systems*, pages 1280–1287. ACM Press, 2002.

[Busetta *et al.*, 2003] P. Busetta, J. Bailey, and R. Kotagiri. A Reliable Computational Model For BDI Agents. In *Proceedings of the Workshop on Safe Agents at AAMAS 2003*. ACM Press, July 2003.

[Busetta *et al.*, 2004] P. Busetta, M. Merzi, S. Rossi, and F. Legras. Intra-Role Coordination Using Group Communication: A Preliminary Report. In *Advances in Agent Communication*, volume LNAI 2922, pages 231–253. Springer, July 2004.

[Chopra and Singh, 2004] Amit K. Chopra and Munindar P. Singh. Nonmonotonic Commitment Machines. In *Advances in Agent Communication*, volume LNAI 2922. Springer, July 2004.

[Conte and Dignum, 2001] R. Conte and F. Dignum. From social monitoring to normative influence. *Journal of Artificial Societies and Social Simulation*, 4(2), 2001.

[Gutnik and Kaminka, 2004] G. Gutnik and G. Kaminka. Towards a Formal Approach to Overhearing: Algorothms for Conversation Identification. In *Priceedings of the Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2004)*, NewYork, USA, July 2004.

[Kaminka and Tambe, 2000] G. Kaminka and M. Tambe. Robust agent teams via socially-attentive monitoring. *Journal of Artificial Intelligence Research*, 12:105–147, 2000.

[Kaminka *et al.*, 2002] Gal. A. Kaminka, D. Pynadath, and M. Tambe. Monitoring Teams by Overhearing: A Multi-Agent Plan-Recognition Approach. *Journal of Artificial Intelligence Research*, 17:83–135, 2002.

[Kumar *et al.*, 2000] S. Kumar, M. J. Huber, D. McGee, P. R. Cohen, and H. J. Levesque. Semantics of agent communication languages for group interaction. In *Proceedings of the 17 th International Confernce on Artificial Intelligence*, pages 42–47, Austin, Texas, 2000.

[Kumar *et al.*, 2002] S. Kumar, M.J. Huber, P.R. Cohen, and D.R. McGee. Toward a formalism for conversation protocols using joint intention theory. *Computational Intelligence*, 18(2):174–174, 2002.

[Legras and Tessier, 2003] F. Legras and C. Tessier. Lotto: group formation by overhearing in large teams. In *Proceedings of the II international joint conference on Autonomous agents and multiagent systems*, pages 425–432. ACM Press, 2003.

[Levesque *et al.*, 1990] H.J. Levesque, P.R. Cohen, and J.H.T. Nunes. On acting together. In *Proceedings of the 8th National Conference on Artificial Intelligence, AAAI*, pages 94–99, 1990.

[Mazouzi *et al.*, 2002] H. Mazouzi, A.E.F. Seghrouchni, and S. Haddad. Open protocol design for complex interactions in multi-agent systems. In *Proceedings of the first intern. joint conf. on Autonomous agents and multiagent systems*, pages 517–526, 2002.

[Nair *et al.*, 2004] R. Nair, M. Tambe, S. Marsella, and T. Raines. Automated assistants for analyzing team behaviors. *Autonomous Agents and Multi-Agent Systems*, 8(1):69–111, 2004.

[Paurobally *et al.*, 2003] Shamimabi Paurobally, Jim Cunningham, and Nicholas R. Jennings. Ensuring consistency in the joint beliefs of interacting agents. In *Proceedings of the conference on Autonomous agents and multiagent systems AAMAS03*, pages 662–669, 2003.

[Paurobally *et al.*, 2004] Shamimabi Paurobally, Jim Cunningham, and Nicolas R. Jennings. Verifying the contract net protocol: A case study in interaction protocol and agent communication language semantics. In *Proceeding 2nd International Workshop on Logic and Communication in Multi-Agent Systems*, 2004.

[Rossi and Busetta, 2004] S. Rossi and P. Busetta. Towards monitoring of group interactions and social roles via overhearing. In *Proceedings of Cooperative Information Agents VIII (CIA 2004)*, LNCS 3191, pages 47–61, Erfurt, Germany, 2004.

[Rossi *et al.*, 2005] S. Rossi, S. Kumar, and P.R. Cohen. Distributive and collective readings in group protocols. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI*, page (forthcoming), 2005.