Reliable group communication and institutional actions in a multi-agent trading scenario

Stephen Cranefield
Department of Information Science
University of Otago
PO Box 56, Dunedin, New Zealand
scranefield@infoscience.otago.ac.nz

Abstract

The use of asynchronous communication is traditionally seen to be an important element of an agent's autonomy. This paper argues that groups of agents within a society need the ability to choose forms of communication with stronger guarantees for particular interactions, and in particular, focuses on the use of reliable group communication. An example electronic trading scenario—the game of Pit—is presented, and it is shown how a formal institution for a particular critical phase of Pit can be built on top of the semantics for totally ordered and virtually synchronous multicasting.

1. Introduction

One of the most common criteria used in definitions of the term "agent" is a requirement for the entity under consideration to be autonomous [22]. This criterion has led researchers in the field of multi-agent systems to focus on asynchronous modes of communication such as KQML [9] or FIPA ACL [10] messaging, because handling a synchronous message (such as a remote method call in a distributed object-oriented system) requires the recipient to devote some of its computational resources to handling incoming requests at a time dictated by the initiator, and this implies a loss of autonomy.

However, it has also been recognised by the MAS community that in order for agents to form societies in which agents can collaborate or provide services to each other, there must be some standards or agreements within the societies on ontologies, interaction protocols and notions of commitment and trust. Adopting a common ontology and interaction protocol can be seen as a choice made by an agent (or agent designer) to forgo some of its autonomy in order to gain the benefits of existing as a functional part of a community.

In this paper it is argued that there is a need for communications infrastructure in multi-agent systems to provide a range of communication mechanisms, including those with stronger guarantees than are provided by asynchronous messaging. Providing a range of communication mechanisms will give agents (or protocol and institution designers) a range of options that can be used for specific interactions, just as a human manager may opt to call a face-to-face meeting or make a phone call when reaching a rapid consensus is desired. These communication mechanisms should include the ability to multicast messages to groups of agents, not only for efficient distribution of information to multiple parties, but also to enable commitments made between agents to be publicly observed and therefore more likely to be honoured (e.g. consider the wedding vows exchanged between bride and groom in the presence of witnesses).

The structure of the paper is as follows. Section 2 discusses previous work on group communication in multi-agent systems and gives a brief overview of reliable group communication mechanisms. Section 3 describes the card game Pit and presents an interaction protocol for a phase of Pit using a reliable group multicasting primitive. This is followed in Section 4 by an analysis of this scenario using a formal model of commitments and institutional action developed by Verdicchio and Colombetti. A summary of their formalism is presented and it is used to model reliable group communication and the Pit scenario. It is shown how an "agree to trade" communication can be given an institutional meaning as making a commitment to change to a trading state (and hence a different protocol) when certain future circumstances arise. Finally, some related work is discussed in Section 5, and Section 6 concludes the paper with some limitations of and future research questions for this work.

2. Reliable group communication

In distributed systems it is useful to be able to send the same information to a set of distributed processes with a single command. This is referred to as broadcasting, or, when distinct groups of recipients can be specified, multicasting. When all processes are running on a network that supports a multicast network protocol, such as IP Multicast, there can be sizable performance gains over replicating unicast messages to multiple recipients. However, at the application level, and for agents in particular, there can be benefits from having a multicast communication primitive available, even if the underlying implementation must simulate this by sending multiple unicast messages. Kumar et al. [12] argue that communication addressed to groups is a common feature of human society, and so it is important that agent communication languages and their underlying semantics support group communication. Busetta et al. [2] discuss the use of channeled multicasting for agent communication where messages can be addressed to channels that have a name, a theme (a list of terms from an application-specific taxonomy) and an IP multicast group address. Their architecture allows agents to "tune in" to channels of interest to receive messages that were addressed to channel names or themes, rather than to individually named agents. This architecture then allows specialist agents (when requested) to "overhear" conversations and provide additional information to the participants when appropriate, and it can also be used for monitoring the behaviour of agents.

This paper extends the above work by considering the use of *reliable* group communication channels for agent communication. Researchers into data replication, failure detection and failure recovery in distributed systems have developed protocols that provide applications with multicast primitives having various reliability guarantees [1]. Depending on the application, desirable properties may include *FIFO ordering* (messages are guaranteed to be received in the order that each process sent them), *causal ordering* (if B sends a message after receiving a message from A, all recipients of the two messages see them in this order), *agreed ordering* (all recipients see all messages in the same order), *safety* (agreed ordering with the additional guarantee of atomic delivery: a message is either delivered to all members or, in the case of any members having failed, none) and *virtual synchrony* (all processes observe the same events in the same order, including processes joining or leaving a group)[1]. The JGroups library [11] combines these ideas with a channel-based architecture. Java applications can create a channel that is connected to a named group—all agents with channels connected to that group are automatically members of the group. The channel constructor is passed a symbolic description of the stack of protocols that are to be used in the channel's implementation, and these can be chosen so that particular properties such as safety and virtual synchrony are assured.

When humans interact in society they enter into discussions and negotiations of various degrees of importance. We select the appropriate communication channels and protocols used for interaction with other agents depending on the degree of loss we might face if other agents fail to uphold their obligations or the norms of society. For example, we need to be more careful when negotiating a house sale than when negotiating a choice of movie to watch with friends. For everyday life we use simple lightweight communication channels, but for signicant interactions we are prepared to trade-off simplicity and efficiency for reliability and safety. Artificial agents will also need a choice of communication mechanisms to use for different types of scenario. The following section discusses an example scenario where a reliable group communication channel provides an agent with guarantees about another agent's state that that would be highly complex (or impossible) to achieve using forms of communication with weaker guarantees.

3. Example scenario: the Pit game

Pit [16] is a card game dating from 1904 that simulates commodity trading in the American Corn Exchange of that era. It is notable for the way in which players trade cards publicly in a concurrent and asynchronous manner. In the simplest form of the game there is a deck of cards, each representing a unit of a tradeable commodity such as barley, with nine identical cards for each commodity. The game proceeds in repeated rounds of card trading until one player wins by reaching a score of 500 points. A round begins with each player being randomly dealt nine cards and ends when a player manages to "corner the market" on a single commodity by possessing all nine cards of that type, and is the first to announce this fact. The score earned for this ranges from 60 to 100 points, depending on the commodity. Once dealing is completed, trading takes place by players concurrently and vocally advertising the number of cards they wish to trade (which must be at least two, and the cards must be of the same commodity). When a trading partner is identified, the trading partners exchange the agreed number of cards, consider their new hand and either announce the achievement of a "corner" or select some new cards to trade and return to the advertising phase. At all times, a player can only see the face of his or her own cards—the other players' cards can not be distinguished from each other as their backs have a uniform appearance.

As an application involving concurrent activity, competitive behaviour and rules designed to ensure fair play, Pit is a good testbed application for investigating issues of elec-

1 This list follows the terminology of Dolev and Malki [6].

tronic agent communication and institutional rules and actions. It has the flavour of a realistic e-commerce scenario while also having a simple and well defined economy in which the players seek to profit through trade. Previous work has investigated the use of MAS techniques to implement an extended version of Pit, designed to be scalable to large numbers of players [18] and the game theory of a simplified version of Pit [19]. This paper is based on the original version of Pit (or, at least, an electronic version that is intended to be as close as possible to the original) and, in particular, investigates one state transition in the game.

In the physical game, players advertise the number of cards they wish to trade by holding those cards up in the air (with only the backs visible to other players) while shouting, for example, "Two! Two! Two!". At the same time they look around the room to locate and make eye contact with another player seeking to trade the same number of cards. It then becomes clear if those players wish to complete the trade: the players focus their attention on each other and the cards are physically exchanged. If at any time one party decides not to trade or their attention is drawn to another, more favoured, trading partner, this is immediately apparent. This feature of the game is difficult to reproduce in an online version where players are in different locations and communicate via messages. While safe trading transactions could be achieved by introducing trusted agents or co-opting other players to to act as notaries that manage the transaction, our preference is to preserve the peer-to-peer nature of the physical version of Pit. The approach taken here is to analyse the game in terms of commitments made by players, and to investigate mechanisms that allow these commitments to be understood and observed.

When players are advertising their desire to trade, they have no commitments to others (apart from having to obey the rules of the game)—they are free to make and break eye contact until they locate another player with whom they are happy to trade and who appears to want to trade with them. At this point there is an important transition in the system. A player beginning a card exchange with another player must focus on that operation and is therefore forgoing the chance to actively seek a possibly preferred trading partner (e.g. one with a score that is further from the winning margin) and to attract his or her attention. In an electronic version of the game, a card exchange transaction may involve executing a relatively complex protocol, so each player would like to have confidence that the other party is committed to the transaction before beginning the exchange. In fact, to have complete certainty that it is safe to begin the transaction, each party A must know that the other party B is committed, that B knows that A is committed, that B knows that A knows that B knows that A is committed, ad infinitum—in other words, the trading partners must achieve common knowledge of their mutual commitment [8]. Another way to view this is that the two players must make a joint action to change their institutional state to one in which they must complete the trade with one another.

It has been proven that it is impossible to achieve common knowledge in a distributed system using asynchronous messages on an unreliable network [8]. However, various approximations of common knowledge are possible, e.g. making use of a "publication" multicasting primitive allows (logically) "time-stamped common knowledge" to be achieved [14]. Although we do not take an epistemological viewpoint of distributed systems in this paper, we follow the above-cited research by relying on the properties of a reliable group communication primitive that is similar to this notion of publication but does not expose logical timestamps to the application layer. In particular, we assume the following operations (supported by the Java JGroups library [11]) are available to agents:

Join a named group
: The agent connects to a virtually synchronous channel associated with the group.

Send a message to all members of a named group
: A message is sent using a channel to all members of a group (including the sender).

Leave a named group
: An agent can leave a group by disconnecting from the associated channel, or (depending on the channel properties) this event may be considered by other group members to have implicitly occurred if the agent's channel fails to respond to pings (in JGroups these are optionally sent as part of the channel's protocol stack).

In addition, whenever an agent joins or leaves the group the agent receives a message from the channel containing an updated "view" of the group membership.

These operations are applied to the Pit scenario as follows. Each player follows the same protocol, which depends on the player's state. For much of the game, and particularly when an agent is in the *Advertising* state, normal asynchronous FIPA-style messaging is used. However, this paper addresses one particular state transition in which reliable multicasting has a valuable role to play: the transition from state *SeekingPartner*(P) (meaning that the player is seeking to establish a trading session with another player P) to state *TradingWith*(P) (meaning that the player has agreed with player P to begin trading their advertised cards). A player enters state *SeekingPartner*(P) after identifying another player that is advertising the same number of cards as itself and with whom it wishes to trade. It then begins the protocol that is shown in Figure 1 as a sequence diagram in an extended version[2] of UML 2.0 [15].

2 The extensions are: (i) The identification of the state of a lifeline, writ-

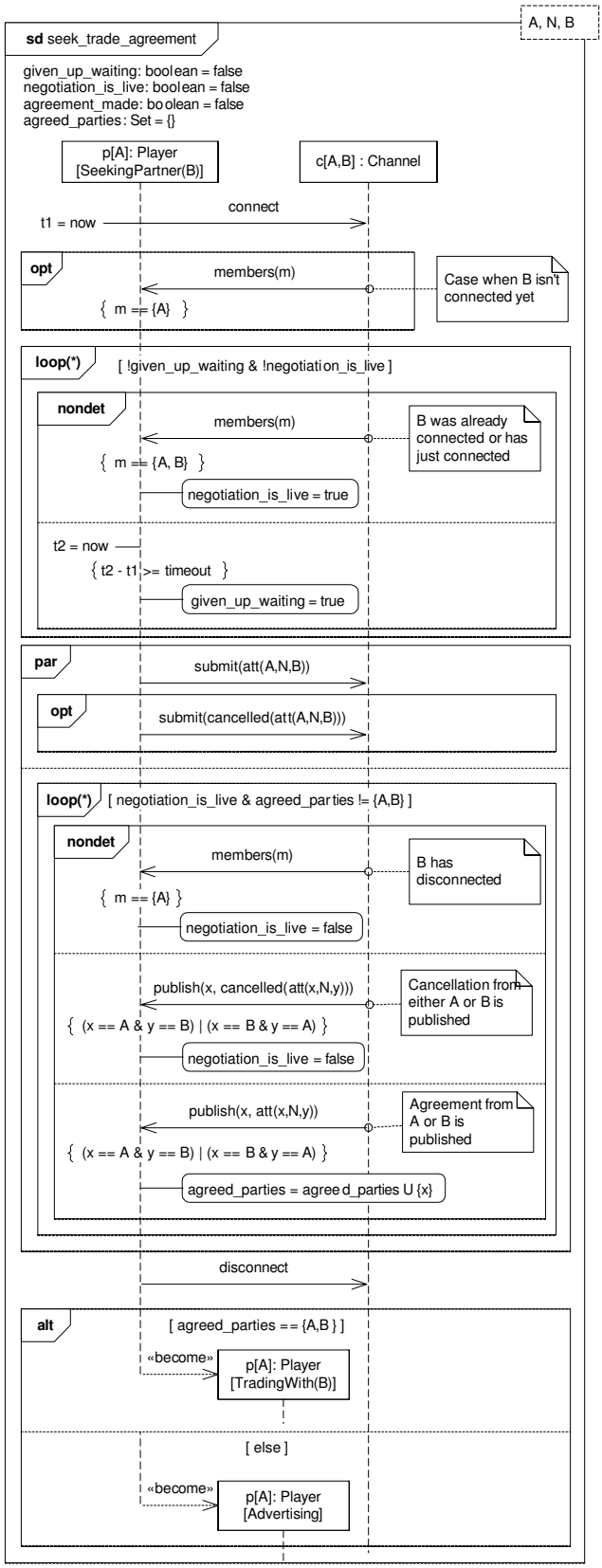

Figure 1: The "seek trade agreement" protocol

A brief summary of the notation is as follows. The operators in the boxed regions are: `opt`, meaning an optional section; `loop`—the "(*)" means there is no lower or upper limit to the number of iterations; `par`, meaning parallel composition by arbitrary interleaving of the interactions in the subsections of the box; `alt`, meaning a guarded choice of the interactions in the subregions, with the guard written in square brackets; and `nondet`, meaning a nondeterministic choice that is not under the control of the agent[3].

The protocol assumes that for each pair of players there is a group, and that the agent infrastructure only allows those players to join that group. In practice these groups will only exist if they are needed and associated channels are created. A player A attempts to "catch the eye" of another player by connecting to a channel for the group $\langle A, B\rangle$ (A could also send a standard asynchronous unicast message of invitation to the other player, but this is not depicted due to space constraints).

Upon connection, the channel sends the current list of group members to A, and this is repeated when the membership changes, e.g. if A was the only member initially and then B joins. If B joins the group before a timeout period, A submits the statement *att*(A, N, B), meaning "A agrees to trade N cards with B", to the channel. For the two players to know that each other is committed to trading, they must each have submitted matching *att* statements to the group. If neither player has crashed or dropped off the network, these messages will eventually be published by the channel and received by each player. However, if a player wishes to withdraw its agreement before receiving the agreement of the other, it can submit a *cancelled(att(...))* message to its channel. This is where the virtually synchronous property of the channels come into play: all *member* messages (e.g. if a player disconnects from their channel) and *att* and *cancelled* publications will be received in the same order by both players. A cancellation is only deemed valid if it is received before the other player's agreement arrives, and virtual synchrony guarantees that both players agree on the validity of any cancellation publications.

Finally, if player A has observed its own and player B's *att* statements without observing any cancellations or a notification of B leaving the group in between, then it changes to state *TradingWith*(B) and begins to follow a separate trading protocol[4]. However, it is important to note that vir-

ten in square brackets under the lifeline name and role, and a notation (appearing at the bottom of the diagram) for a state change action; (ii) the introduction of the `nondet` interaction type described in the main text above.

3 This requires a branching time semantics whereas UML 2.0 sequence diagrams have trace-based semantics, but we don't address this issue here.

4 A discussion of possible trading protocols is beyond the scope of this paper.

tual synchrony only guarantees *logically* simultaneous receipt of messages, not real-time synchrony. The trading protocol must therefore be designed with an initial phase that verifies that both agents have reached the *TradingWith*$(\ldots)$ state, just as the illustrated protocol begins by waiting until both players have joined the group.

If the protocol in Figure 1 ends without both players agreeing, player A will return to the (previous) *Advertising* state and its associated protocol.

The use of virtually synchronous group multicasting ensures that the two players have a consistent view of the outcome of the negotiation, and therefore they each know that it is safe to change state and begin the card trading protocol. However, this assumes that each player believes that the other is correctly following the protocol. In an open system, such a belief can only be justified by appealing to the rules of the society in which the agent exists. The rules of Pit should be defined to ensure that the submission of an agree to trade (*att*) statement is only done with the intent of subsequently changing state if the other player also agrees to trade. The act of submitting an *att* statement should therefore be treated as having the secondary effect (within the institution of Pit) of making a conditional commitment to change to the *TradingWith*(B) state in the appropriate circumstances. In the next section we show how this can be formalised, and demonstrate that the use of the publication mechanism allows the conditional part of this commitment to be defined in a way that guarantees that both agents have a common understanding of when each other becomes committed to the trade.

4. Commitment and institutional action in Pit

Following the path set by researchers of electronic institutions [5] we seek an understanding of Pit as a social interaction governed by norms, permissions, obligations, commitments, etc. In this section, a preliminary analysis of the "seek trade agreement" scenario in Figure 1 is presented, showing how the submission of an *att* statement to the channel can be defined to imply the performance of an institutional action: making a commitment to change to the *TradingWith*$(\ldots)$ state if both players' *att* messages are published with no intervening *cancelled* or group membership change. Based on the formalism of Verdicchio and Colombetti [20, 21, 4] for modelling commitments, agent communication and institutional action, we model the publication mechanism and the institutional meaning of an *att* submission.

4.1. Verdicchio and Colombetti's formal model

Verdicchio and Colombetti use a temporal logic CTL$^{\pm}$, based on CTL* [7]. The semantics of CTL$^{\pm}$ assumes that time is discrete, with potentially many possible next states for any given state, i.e. agents' choices and the nondeterminism of the environment are represented by a future-branching tree of states. Standard temporal primitive and derived modal operators are used, and these can be applied in the future direction (when adorned with a superscript '$+$') or the past (similarly indicated by a '$-$'). The operators include X (the next/previous state), F (eventually), G (always) and U (until). The until operator is a binary operator meaning that the formula on its left will remain true for a (possibly empty) sequence of states, followed by a state in which the formula on its right holds. In addition, the path quantifiers A (for all paths) and E (there exists a path) are used to constrain the set of possible paths through time that contain a given state.

The occurrence of an event e in a state is modelled by the truth of the proposition *Happ*(e) in that state. There is assumed to be a unique constant e to represent each distinct event, and this uniqueness constraint is represented by the following axiom:

$$\mathit{Happ}(e) \rightarrow \mathsf{X}^{-}\mathsf{G}^{-}\neg\mathit{Happ}(e) \wedge \mathsf{A}\mathsf{X}^{+}\mathsf{G}^{+}\neg\mathit{Happ}(e)$$

Other axioms are used to constrain the 'physics' of action, but are not presented here.

Events have types, represented by logical terms such as *rain* (it rains) or *inform*(a, b, ϕ) (a informs agent b of statement ϕ, where ϕ is a statement in an agent content language represented as a term). The type of an event is represented by a proposition of the form *EvType*(e, t). The following abbreviation is defined:

$$\mathit{Happ}(e, t) =_{\text{def}} \mathit{EvType}(e, t) \wedge \mathit{Happ}(e)$$

Some events represent actions that are brought about by an agent. The following predicates and abbreviations are used to represent actions:

Actor(a, e) : agent a is an actor of event e

$\mathit{Done}(e, a, t) =_{\text{def}} \mathit{EvType}(e, t) \wedge \mathit{Actor}(e, a) \wedge \mathit{Happ}(e)$

$\mathit{Done}(a, t) =_{\text{def}} \exists e\, \mathit{Done}(e, a, t)$

to which we add:

$\mathit{Happ}_t(t) =_{\text{def}} \exists e\, \mathit{EvType}(e, t) \wedge \mathit{Happ}(e)$

The formalism also includes a number of predicates used to represent the existence and state of commitments between agents. *Comm*(e, a, b, τ) means that event e has brought about a commitment between debtor a and creditor b to the truth of the CTL$^{\pm}$ formula τ (which is encoded here as a term). Action types *mc*(a, b, τ) and *cc*(e, a, b, τ) are defined to represent making and cancelling a commitment, and axioms are used to define the meaning of these. In particular, the performance by agent a of an action *mc*(a, b, τ) implies that in all possible futures *Comm*(e, a, b, τ) holds forever or until an act of cancelling the commitment is done

(in which case *Comm*(e, a, b, τ) ceases to be true)[5]. Temporal formulae are used to define the notion of fulfilment and violation of a commitment, represented by the predicates *Fulf*(e, a, b, τ) and *Viol*(e, a, b, τ) respectively. The definition for fulfilment is:

$$\textit{Fulf}(e, a, b, \tau) =_{\text{def}} Comm(e, a, b, \tau) \wedge \mathsf{AF}^{-}(\textit{Happ}(e) \wedge \lfloor \tau \rfloor)$$

where '$\lfloor . \rfloor$' represents the mapping from a term encoding a temporal formula to the formula itself. This definition says that to determine the satisfaction of τ as a commitment, τ should be evaluated in the (possibly prior) state in which the event e—the making of the commitment—was performed.

Within an institution, acts of one type (e.g. raising one's hand during an auction) can be deemed to "count as" an action of another type (e.g. offering to buy the item being auctioned at the current price). This institutional fact is represented by a proposition of the form *CountsAs*(t, t') and the following axiom defines its meaning:

$$\textit{Done}(e, a, t) \wedge \textit{CountsAs}(t, t') \rightarrow \textit{Done}(e, a, t')$$

The left hand side can include additional conjuncts testing for logical possibility and authorisation, but for the sake of brevity we do not discuss that here.

4.2. Modelling the publication mechanism

In our Pit protocol we consider the submission of an *att* statement to the channel as making a commitment to trade under certain conditions. To define this formally we must first model the publication mechanism using CTL$^{\pm}$. We use the following event types to represent agents leaving or joining a group (we do not model these as actions because leaving a group may occur due to a failure that is not brought about by an agent itself):

join_group(a, g)

leave_group(a, g)

The following predicates are used to model group membership and agents' views of group membership:

member(a, g) : agent a is a member of group g

member_in_view(a, g, b) : a is a member of g in b's current local view

These action types represent agents submitting and observing publications, and observing group membership changes:

$submit\_pub(\phi, g)$

$observe\_pub(e, a, \phi, g)$

$observe\_join\_group(e, a, g)$

$observe\_leave\_group(e, a, g)$

where ϕ is a statement for publication (represented as a term), g is a group name and e is the event that is being observed—either the submission of a publication or an agent joining or leaving the group. If more than one agent joins or leaves a group in a single state, this is modelled as separate but concurrent events.

CTL$^{\pm}$ axioms are used to define the meanings of the event and action types and the constraints on the publication mechanism; here we only present a sample:

$$\neg \textit{member}(a, g) \rightarrow \neg \textit{Happ}_t(\textit{leave\_group}(a, g))$$

$$\neg \textit{member}(a, g) \wedge \textit{Happ}_t(\textit{join\_group}(a, g)) \rightarrow \mathsf{AX}^{+}\textit{member}(a, g)$$

$$\neg \textit{member}(a, g) \wedge \neg \textit{Happ}_t(\textit{join\_group}(a, g)) \rightarrow \mathsf{AX}^{+} \neg \textit{member}(a, g)$$

$$\textit{Done}(a, \textit{observe\_join\_group}(-, b, g)) \rightarrow \textit{member\_in\_view}(b, g, a)$$

$$\textit{Done}(a, \textit{observe\_pub}(e, -, -, -)) \rightarrow \mathsf{G}^{-}\mathsf{X}^{-} \neg \textit{Done}(a, \textit{observe\_pub}(e, -, -, -)) \wedge \mathsf{AG}^{+}\mathsf{X}^{+} \neg \textit{Done}(a, \textit{observe\_pub}(e, -, -, -))$$

where, following Verdicchio and Colombetti, '$-$' is an abbreviation for an existentially quantified variable different from any other.

Three other important axioms (not shown here) state that a group member will observe its own publication unless a group membership change is observed first (the atomic delivery property), all group members observe a publication if any of them does, but the global order of these observations is not constrained, and any two group members observe any pairs of observations in the same relative order.

4.3. Modelling the trading commitment in Pit

In the Pit protocol in Figure 1 the agents communicate by submitting *att* and *cancelled* statements to the channel to be published. In this section we define the act of submitting an *att* statement for publication as making a commitment to change state if the appropriate conditions apply. To do this, we must first define another action type: *change_state*$(s, s', \textit{inst})$. This represents the act of changing from state s to state s' in institution *inst*. The institutional meaning of submitting an *att* message to the channel can now be defined as shown in Figure 2. The binary operator S^{+}, meaning "as soon as $\langle$*LHS*$\rangle$ then $\langle$*RHS*$\rangle$", is defined by Verdicchio and Colombetti as follows:

$$\phi \, \mathsf{S}^{+} \psi =_{\text{def}} (\phi \rightarrow \psi) \wedge (\mathsf{X}^{+}(\phi \rightarrow \psi)) \, \mathsf{W}^{+} \phi$$

where W^{+} is the future version of the usual "weak until" operator:

$$\phi \, \mathsf{W}^{+} \psi =_{\text{def}} \mathsf{G}^{+}\phi \vee \phi \, \mathsf{U}^{+} \psi$$

5 A counter-intuitive feature of this formalism is that a commitment is modelled as continuing to exist even after it has been fulfilled

$$
\begin{array}{l}
\mathit{CountsAs}(\mathit{submit\_pub}(att(a,n,b),g),\\
\quad \mathit{mc}(a,b,(\ (\mathit{Done}(a,\mathit{observe\_pub}(-,b,\mathit{att}(b,n,a),g))\\
\qquad \wedge (\neg\mathit{Done}(a,\mathit{observe\_pub}(-,a,\mathit{cancelled}(a,n,b),g))\\
\qquad\quad \mathsf{U}^{-}\\
\qquad\quad \mathit{Done}(a,\mathit{observe\_pub}(-,a,\mathit{att}(a,n,b),g))))\\
\qquad \vee\\
\qquad (\mathit{Done}(a,\mathit{observe\_pub}(-,a,\mathit{att}(a,n,b),g))\\
\qquad \wedge (\neg\mathit{Done}(a,\mathit{observe\_pub}(-,b,\mathit{cancelled}(b,n,a),g))\\
\qquad\quad \mathsf{U}^{-}\\
\qquad\quad \mathit{Done}(a,\mathit{observe\_pub}(-,b,\mathit{att}(b,n,a),g)))))\\
\qquad \mathsf{S}^{+}\\
\qquad \mathit{Done}(a,\mathit{change\_state}(\mathit{SeekingPartner}(b),\mathit{TradingWith}(b),\mathit{pit}))))
\end{array}
$$

Figure 2: The institutional meaning of submitting an agreement to trade

The declaration in Figure 2 states that the act of submitting an *att* statement for publication counts as making a commitment that whenever a publication from one of the two agents is observed, if when looking backwards to find the matching publication from the other agent no cancellation of that publication has occurred more recently, then a change of state from *SeekingPartner*(b) to *TradingWith*(b) will be performed. Analysing the axioms defining the properties of publication, together with the definition of fulfilment in Section 4.1, it is possible to infer[6] that if one of the two negotiating agents is committed to changing state then (eventually) they both are.

5. Related Work

Section 2 discussed some prior work on group communication in multi-agent systems. In particular, this paper extends the work of Busetta et al. [2] by considering the use of group communication channels that implement reliable multicasting.

Cheriton and Skeen [3] have presented some limitations of the use of causally and totally ordered (agreed order) multicasting as a generic mechanism for solving a variety of distributed computing problems. These limitations are: (i) messages can be delivered out of order from the application viewpoint if there are causal relationships unknown to the multicasting infrastructure, e.g. due to out-of-channel communications or interactions with the environment; (ii) particular sequences of messages cannot be considered as atomic operations so that no other messages are delivered between them; (iii) there can be causal relationships between events at the application level that are not captured by the semantics of causally and totally ordered multicasting, e.g. a particular message should only be sent if another message has not been received yet; and (iv) they lack efficiency and scalability compared to protocols directly based on application state.

The last limitation above is an important one, but we believe there is still a significant role for reliable multicasting to play within particular phases of an interaction protocol, where only a small group of agents (possibly a subset of all the participants) are involved. Furthermore, in some applications, such as Pit, interaction protocols are best described in terms of agent's individual states rather than a global state of the interaction. Our answer to the other three limitations is that designers of protocols using reliable multicasting do need to be aware of these issues, and they should not expect this technique to be a panacea. However, we believe that the problems identified can all be prevented by careful protocol design.

Paurobally et al. [17] proposed the use of *synchronisation protocols* that run in a layer between the network protocols and the interaction protocols that an agent follows. These synchronisation protocols ensure that the participants in an interation have a consistent belief in the state of the protocol. This is similar to the work discussed in this paper in that it relies on an underlying protocol layer to enable synchronisation without complicating the higher level interaction protocol. However, the approach of Paurobally et al. requires an interaction protocol defined in terms of global states—a protocol state is sent as part of every message. As shown in this paper, our approach can be used with interaction protocols in which each role is defined in terms of its own local state, but the state changes need to be coordinated with those of other agents.

The formal model of action and commitment used in this paper is that of Verdicchio and Colombetti [20, 21, 4], which uses the full power (and incurs the full complexity) of a CTL*-style logic. Mallya et al. [13] have investigated a simpler temporal language, with semantics based on CTL*, in which temporal interval expressions can be used as existential and universal quantifiers of propositional formulae. This language allows the satisfaction of commitments to be resolved automatically.

6 How agents, specialised auditor agents, or their designers could reason automatically using such axioms remains a subject for future research.

6. Conclusion

This paper has proposed the use of reliable group communication mechanisms in multi-agent systems and demonstrated its utility in a peer-to-peer electronic trading scenario where agents may require some guarantees about the state of other agents. It has also demonstrated how a publication on a virtually synchronous group communication channel can be formally defined to count as the establishment of a commitment that is dependent on the commonly understood order of future publications—thus allowing distributed agents to have a shared understanding of each other's commitments.

As in human society, software agents should have a range of communication mechanisms with varying properties available to them. Providing agent messaging infrastructure supporting various modes of group communication will allow the declarative definition of simpler interaction protocols where agents act in a peer-to-peer manner and, for particular phases, need some guarantees about the institutional state of their peers. However, reliable group communication comes at a computational cost—for example, with JGroups, the first node to connect to a channel for a group becomes the coordinator for that group. The channel for that node is responsible (via appropriate underlying protocols) for ensuring atomic delivery and agreed order semantics for multicasts. Reliable multicasting is also unlikely to be practically scalable to larger groups of agents. Therefore, the appropriate role of this technique is for particular phases of interaction protocols where synchronised agreements are needed amongst small groups of agents, with standard asynchronous messaging used elsewhere.

Acknowledgements

The author would like to thank Prof. Marco Colombetti for sharing his ideas during a visit to the University of Otago in 2003, and Francesco Fumarola for his work evaluating the use of the JGroups toolkit in a multi-agent implementation of Pit.

References

[1] K. P. Birman and T. A. Joseph. Reliable communication in the presence of failures. *ACM Transactions on Computer Systems*, 5(1):47–76, 1987.

[2] P. Busetta, A. Donà, and M. Nori. Channeled multicast for group communications. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, pages 1280–1287. ACM Press, 2002.

[3] D. R. Cheriton and D. Skeen. Understanding the limitations of causally and totally ordered communication. *Operating Systems Review*, 27(5):44–57, 1993. (Proceedings of the Fourteenth ACM Symposium on Operating System Principles).

[4] M. Colombetti. Agent communication. Unpublished seminar notes, University of Otago, 2003.

[5] U. Cortés. Electronic institutions and agents. *AgentLink News*, 15:14–15, September 2004. http://www.agentlink.org/newsletter/15/AL-15.pdf.

[6] D. Dolev and D. Malki. The Transis approach to high availability cluster computing. *Communications of the ACM*, 39(4):64–70, 1996.

[7] E. Emerson and J. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[8] R. Fagin, J.Y.Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, MA, 1995.

[9] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. M. Bradshaw, editor, *Software Agents*. MIT Press, 1997.

[10] FIPA. FIPA ACL message representation in string specification, Foundation for Intelligent Physical Agents. http://www.fipa.org/specs/fipa00070/, 2002.

[11] JGroups project home page. http://www.jgroups.org, 2004.

[12] 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 Seventeenth National Conference on Artificial Intelligence (AAAI 2000)*, pages 42–47. AAAI Press / MIT Press, 2000.

[13] A. U. Mallya, P. Yolum, and M. P. Singh. Resolving commitments among autonomous agents. In F. Dignum, editor, *Advances in Agent Communication, International Workshop on Agent Communication Languages, ACL 2003*, volume 2922 of *Lecture Notes in Computer Science*, pages 166–182. Springer, 2004.

[14] G. Neiger and S. Toueg. Simulating synchronized clocks and common knowledge in distributed systems. *Journal of the ACM*, 40(2):334–367, 1993.

[15] Object Management Group. UML 2.0 superstructure final adopted specification. Document ptc/03-08-02, http://www.omg.org/cgi-bin/doc?ptc/2003-08-02, 2003.

[16] Parker Brothers. Pit rules. http://www.hasbro.com/common/instruct/pit.pdf, 1904.

[17] S. Paurobally, J. Cunningham, and N. R. Jennings. Ensuring consistency in the joint beliefs of interacting agents. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*, pages 662–669. ACM Press, 2003.

[18] M. Purvis, M. Nowostawski, S. Cranefield, and M. Oliveira. Multi-agent interaction technology for peer-to-peer computing in electronic trading environments. In C. Zhang, H. W. Guesgen, and W. Yeap, editors, *Proceedings of the 8th Pacific Rim International Conference on Artificial Intelligence*, volume 3157 of *Lecture Notes In Artificial Intelligence*, pages 625–634. Springer, 2004.

[19] H. van Ditmarsch. Some game theory of Pit. In C. Zhang, H. W. Guesgen, and W. Yeap, editors, *Proceedings of the 8th Pacific Rim International Conference on Artificial Intelligence*, volume 3157 of *Lecture Notes in Artificial Intelligence*, pages 946–947. Springer, 2004.

[20] M. Verdicchio and M. Colombetti. A logical model of social commitment for agent communication. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*, pages 528–535. ACM Press, 2003.

[21] M. Verdicchio and M. Colombetti. A logical model of social commitment for agent communication. In F. Dignum, editor, *Advances in Agent Communication, International Workshop on Agent Communication Languages, ACL 2003*, volume 2922 of *Lecture Notes in Computer Science*, pages 128–145. Springer, 2004.

[22] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.