Knowledge-based Algorithm for Multi-Agent Communication {extended abstract}

E. van Baars & R. Verbrugge

Department of Artificial Intelligence, University of Groningen, Grote Kruisstraat 2/1, 9712 TS Groningen, The Netherlands, egon@vanbaars.com,rineke@ai.rug.nl

Abstract. Using a knowledge-based approach, we derive a protocol for the sequence transmission problem from one agent to a group of agents. The knowledge-based protocol is correct for communication media where deletion and reordering errors may occur. Furthermore, it is shown that after k rounds the agents in the group attain depth k general knowledge about the members of the group and the values of the messages. Thus, the knowledge-based protocol may be used in cooperative problem solving in order to attain an approximation of common belief among agents in a team.

1 Introduction

For cooperative problem solving (CPS) within multi-agent systems, Wooldridge and Jennings give a model of four consecutive stages [16], covering the entire process that starts with one initiating agent that sees a goal and ends with a group of agents cooperating in achieving this goal. Dignum, Dunin-Kęplicz and Verbrugge give a more in-depth analysis of the communication and dialogues that play a role during these four stages [1, 3]. Most writings about communication for CPS make a strong assumption: whatever is communicated between agents will arrive, and will arrive correctly at the desired agents; in particular, it is often silently assumed that public announcements lead to common knowledge or common belief [15]. This is based on idealization: in uncertain communication media, messages may for example be lost or they may arrive in the wrong order.

One of the great advantages of epistemic logic is that it can be used to model communication in distributed systems [6, 11]. For example, in their classical paper [9], Halpern and Zuck showed that epistemic logic enables perspicuous specification and verification for a number of protocols (like the alternating-bit protocol) that had been introduced for error-free transmission of sequences of messages over a distributed network. Let two processors be given, called the sender S and the receiver R. The sender has an input tape with an infinite sequence X of data elements. S reads these elements and tries to send them to R, which writes the elements on an output tape. The protocols are required to guarantee that (a) at any moment the sequence of data elements received by R is a prefix of X (safety) and (b) if the communication medium satisfies certain so-called fairness conditions, every data element of X will eventually be written by R (liveness). Fairness here means that infinitely many message instantiations from S to R and from R to S are delivered, guaranteeing that every message arrives eventually.

In [14] Stulp and Verbrugge extend the knowledge-based approach to the Transmission Control Protocol (TCP), which is indispensable for the Internet today. The main contribution there is the modelling of a sliding window, allowing agents to make

use of available bandwidth, and an epistemic analysis in which exact lower and upper bounds on the attained knowledge of the participants at every moment in the communication process are proved.

All above-mentioned algorithms are meant for one-on-one communication. For announcements from one agent to a (finite) group, however, using TCP or a similar protocol with all agents separately often does not create sufficient knowledge. For example, during the stage of team formation in CPS, the goal is to create a collective intention among a team. For this to happen communication between the initiating agent and a group of potential agents is needed. The algorithms for one-on-one communication from [14, 9] are not sufficient for this communication. What is needed is a knowledge-based algorithm that guarantees a reliable communication for one-on-group communication. In this article a knowledge-based protocol that can handle one-on-group communication will be presented, in which at every round the agents in the group attain a higher level of general knowledge of the make-up of the group as well as the contents of the announced message. The reader can run through a simulation of the protocol at `http://www.ai.rug.nl/alice/mas/macom`.

The analysis of this algorithm yields some interesting results. We will show that the depth of knowledge the sender and receiver can accumulate about messages sent is dependent upon the length of the tape and the position of information on the tape. If an infinite tape models the transmitted data, the following can be shown. For any k and any piece of data on the tape, at some point k-fold depth of general knowledge ('everyone knows') arises about the message, although common knowledge can never be achieved.

The rest of the paper is structured as follows. In section 2 we give an overview of knowledge and knowledge creation within a group as can be found in cooperative problem solving in multi-agent systems. In section 3, our knowledge-based algorithm for multi-agent communication is presented. An epistemic analysis of the algorithm follows in section 4. The paper ends with a discussion of related research.

2 Communication and knowledge

Communication between two or more agents consists of transmitting messages between these agents. This communication requires a *communication system* which consists of a connection in a communication medium between agents, together with a protocol by which the agents send and receive data over this connection. For a communication system to be reliable it has to satisfy the three properties mentioned in section 1: *fairness*, *liveness*, and *safety* [9]. During the transmission of packages over a connection, errors can occur which jeopardize these properties. To guarantee reliability in a communication system, the protocol has to handle the errors that can occur at a connection. Regardless which errors occur in a system, the connection has to satisfy the fairness condition for reliable communication to be possible. If this is not the case it could happen that none of the messages sent by S arrive at R which means that there is no communication at all. This leaves the protocol responsible for assuring the liveness and safety properties where they are jeopardized by the errors that occur at a connection. The goal of this article is to prove the gaining of group knowledge that emerges during the use of our protocol by a group of agents. For this reason we are not going to discuss all the possible errors that can occur and how our algorithm handles them. We just present a summarized protocol that deals with the knowledge-based part of group communication and only handles reordering and deletion errors.

In the communication system our protocol works in, every agent is connected to every other agent through one or more connections. These connections are faultless

except that messages can get lost, causing deletion errors. Also, the agents and the communication system are asynchronous, which means that they don't have access to a shared clock. Thus the sending and receiving part of the algorithm work independently. The transmission speed of the connections can differ so that a message that was sent after another message can arrive before this other message, causing reordering errors.

2.1 Logical background: knowledge and time

When proving properties of knowledge-based protocols, it is usual to use semantics of interpreted systems $\mathcal{I}$ representing the behaviour of processors over time (see [6, 11]). We give a short review. At each point in time, each of the processors is in some *local state*. All of these local states, together with the environment's state, form the system's *global state* at that point in time. These global states form the possible worlds in a Kripke model. The accessibility relations are defined according to the following informal description. The processor R "knows" φ if in every other global state which has the same local state as processor R, the formula φ holds. In particular each processor knows its own local state; for the environment, there is no accessibility relation. The knowledge relations are equivalence relations, obeying the well-known epistemic logic $S5_n^C$ (see [6]), including e.g. the knowledge axiom $K_i\varphi \Rightarrow \varphi, i = 1, ..., n$, as well as axioms governing general and common knowledge such as $E_G\varphi \Leftrightarrow \bigwedge_{i\in G} K_i\varphi$ and $C_G\varphi \Rightarrow E_G(\varphi \wedge C_G\varphi)$. We also use abbreviations for referring to general knowledge at any finite depth. Inductively, $E_G^1\varphi$ stands for $E_G\varphi$ and $E_G^{k+1}\varphi$ is $E_G\varphi\left(E_G^k\varphi\right)$.

A *run* is a (finite or infinite) sequence of global states, which may be viewed as running through time. Time here is taken as isomorphic to the natural numbers. There need not be any accessibility relation between two global states for them to appear in succession in a run. Time clearly obeys the axioms of the basic temporal logic K_t (see [7]), in which the following principle (A) is derivable:
(A) $P(\Box\varphi) \rightarrow \Box\varphi$
To further model time, we extend $S5_n^C$ with the following axiom:
KT1. $K_i\Box\varphi \rightarrow \Box K_i\varphi, i = 1, ..., n$
This axiom holds for systems with perfect recall [8]. Halpern et al. [8] present a complete axiomatization for knowledge and time, however in this article we only need the axiom KT1.
As for notation, global states are represented as (r, m) (m-th time-point in run r) in the interpreted system $\mathcal{I}$. In particular for the temporal operators, we have the following truth definitions:
$(\mathcal{I}, r, m) \models \Box\varphi$ *iff* $(\mathcal{I}, r, m') \models \varphi$ *for all* $m' \geq m$
$(\mathcal{I}, r, m) \models P\varphi$ *iff* $(\mathcal{I}, r, m') \models \varphi$ *for some* $m' < m$
In the next table some formulas are given, together with their informal meanings that will be used in the rest of the article.

Formulas	Descriptions
$K_S\varphi$	Sender S knows φ
$K_{R_i}\varphi$	Receiver R_i knows φ
$E_G\varphi$	Every agent in group G knows φ (general knowledge)
$E_G^k\varphi$	Group G has depth k general knowledge of φ
R_G	G is the current group of receivers
$P\varphi$	At some moment in the past on this run, φ was true
$\Box\varphi$	φ is now and will always be true on this run

2.2 Knowledge creation within a group

The goal of one-on-group communication is that all the members of the group gain a certain level of knowledge about a fact φ sent by the sender, and that all the members gain a certain level of knowledge about the knowledge of the group of this fact φ. This implies that the members have to gain a certain level of knowledge about which members the group consists of. When the group consists of only a sender and one receiver we speak of one-on-one communication and the gaining of knowledge is quite straightforward as described in [14, 9]. When the group consists of a sender and two or more receivers, it becomes a bit more complicated. The receivers of a certain fact now also have to know to whom the sender is sending this fact for gaining the above mentioned knowledge. The solution for this is to send the information about the extension of the group together with the fact φ. Considering the general form of a message this can be achieved in two ways. Analogously to the TCP [13], we will refer to the general form of a message as a *package*. A package consists of a data part which contains the fact to be sent and of a *header* which contains meta-information about the data part. Thus, the sender can put the group information in the data part of the message or in the header. The group to whom the sender is sending a certain fact φ is meta-information about this fact φ, so it is preferred to store this group information in the header of a package instead of in the data part.

When the group of receiving agents is stored in the header, then as soon as any of the receiving agents R_i receives this package it knows the j-th fact φ_j stored in this package, $K_{R_i}\varphi_j$, and it knows to which other receivers this message is sent, $K_{R_i}R_G$. How does R_i know whether the other receivers received this package? The sender has to wait with sending a package with the next fact φ_{j+1} until it has received acknowledgements about the package with the previous fact φ_j from *all* the receivers. The sender then knows that all the receivers know the fact φ_j, and thus $K_S E_G \varphi_j$. Every receiver knows that the sender works this way, so when a receiver R_i receives a package with the next fact φ_{j+1}, it knows that the sender knows that all other receivers did receive the previous package and thus know the previous fact φ_j, so $K_{R_i} K_S E_G \varphi_j$. With every repeating step of this cycle the knowledge of the sender and receivers of each others' knowledge of the facts grows for previously sent facts and the knowledge about each others' knowledge of the group they are in grows as well, $K_S E_G^k \varphi_j$ respectively $K_{R_i} K_S E_G^k \varphi_j$. The depth k of knowledge gained by the members of a certain fact is equal to the amount of consecutive facts sent successfully after this fact. The depth of knowledge within the group about the members of the group is equal to the depth of knowledge within the group about the first fact sent by the sender.

If one of the one-on-one algorithms from [14, 9] had been used, the receiving agents would not have known that the facts φ_j they received were sent to other receivers as well. Each of the receivers would have known not more than that the group consist of just the sender and itself $G = \{S, R_i\}$ instead of $G = \{S, R_1, ..., R_n\}$. The gaining of knowledge works in this case the same as mentioned above. However, the knowledge that is gained differs. The knowledge a receiver R_i now has gained after having received two packages with the consecutive facts φ_j and φ_{j+i}, is $K_{R_i} K_S E_{\{S,R_i\}} \varphi_j$, and not the much stronger $K_{R_i} K_S E_G \varphi_j$. So when the goal is to attain a certain depth of group knowledge, the algorithms from [14, 9] are not sufficient.

3 The Algorithm

The packages from our algorithm have the following form:

$K_{source}(destination, -, group, position, -, data)$

source = source port where this package is sent from $[S, R_i]$;
Ksource = the source who sends this package knows this package;
destination = destination port of package $[S, R_i]$;
group = group receivers to which the message is sent $[R_G, -]$ ("$-$" means that the sender communicates only to the **destination** (one-on-one communication));
position = position of data from the input tape;
data = data that has to be transmitted.

The fields filled with "$-$" are the checksum and window_size fields which deal with package mutation errors and congestion control [2]. Because they are not of interest for the the knowledge-based part of our algorithm they are left out in this summarized protocol version. The algorithm for the sender as well as for the receiver consists of two parts, because the algorithm works in an asynchronous system as mentioned in section 2. One part handles the sending of the packages and the other part handles the received packages. The reception of messages is independent and works asynchronously to the sending process. These two processes affect the same local knowledge state of an agent. Though being independent the sending and receiving algorithm influence each other's behaviour through the local knowledge state of an agent.

Sender (incoming packages)

```
1  for (i = 1 to n)
   {For all agents who sender is sending to, ... }
2     ack_Ri = 0
      {... initialize the acknowledgement number.}
3  end
   {ack_Ri's initialized}
4  while true do
   {Get ready for receiving acknowledgements from the receivers, ... [12]}
5     when received K_Ri(S, -, -, seq, -, -) do
      {You have received a package. Prepare for processing, ... [11]}
6        if (seq = ack_Ri+1) do
         {If this acknowledgement from Ri is equal to the next ack_Ri, ... [10]}
7           ack_Ri = seq
            {... this is the new current acknowledgement from Ri, ...}
8           store K_S K_Ri(-, -, -, seq, -, -)
            {... store the fact that you know that Ri knows it.}
9           ack_RG = min(ack_Ri (for i = 1 to n)
            {The highest acknowledgement by the group is equal to the lowest ack_Ri.}
10       end
         {[6] ... acknowledgement from Ri, and highest group acknowledgement updated.}
11    end
      {[5] ... finished processing of incoming package.}
12 end
   {... [4].}
```

Sender (outgoing packages)

1 **seq = 0**
{*Reading of a tape starts at position 0.*}
2 **while true do**
{*Start reading and sending an infinite tape, ... [13]*}
3 **read(seq,alpha)**
{*... read the value from the tape, ...*}
4 **store** $K_S(-,-,-,seq,-,alpha)$
{*... and store this information in your knowledge base.*}
5 **while (ack_RG $\neq$ seq) do**
{*While not al receivers have acknowledged the package with sequence seq ...[11]*}
6 **for (i = 1 to n) do**
{*For all receiving agents, ...*}
7 **if not** $K_S K_{R_i}(-,-,-,seq,-,-)$ **do**
{*... check if package 'seq' has been acknowledged yet by Ri, ...*}
8 **send** $K_S(R_i,-,R_G,seq,-,alpha)$
{*... (re)send the package to Ri.*}
9 **end**
{*A package that was unacknowledged by Ri, has been (re)sent.*}
10 **end**
{*A package has been (re)sent to all agents that didn't acknowledge it.*}
11 **end**
{*[5] ... all agents Ri have acknowledged the package with sequence number seq.*}
12 **seq = seq +1**
{*Move the sequence number to the next position.*}
13 **end**
{*... [2].*}

Receiver (incoming packages)

1 **while true do**
{*Get ready for receiving an infinite tape, ... [5]*}
2 **when received** $K_S(R_i,-,R_G,seq,-,alpha)$ **do**
{*You have received a package (from S). Prepare for processing, ... [4]*}
3 **store** $K_{R_i} K_S(-,-,R_G,seq,-,alpha)$
{*Store the received package.*}
4 **end**
{*[2] ... finished processing incoming package.*}
5 **end**
{*... [1].*}

Receiver (outgoing packages)

1 **when** $K_{R_i} K_S(-,-,-,0,-,-)$
{*Wait until the first message is received.*}
2 **seq = 0**
{*Initiate the sequence at 0.*}
3 **while true do**
{*Get ready to acknowledge incoming packages, ... [8]*}
4 **while not** $K_{R_i} K_S(-,-,-,seq+1,-,-)$ **do**
{*Still not received package with sequence number 'seq+1', ...*}
5 **send** $K_{R_i}(S,-,-,seq,-,-)$
{*... (re)send acknowledgement.*}
6 **end**
{*You've received message seq+1.*}
7 **seq = seq +1**
{*You now the sequence number of the next message. Increment seq.*}
8 **end**
{*... [3].*}

4 Epistemic analysis and proof

In this section a proof is given for the gaining of knowledge as described in section 2.2. For the readability of the proof the form of the package is shortened to $Ksource(position, data)$. We assume that the group stays unchanged and for the destination we assume that the sender S sends to a receiver R_i and vice versa, so the *destination* field and the *group* field are left out.

Definition 1. *The following abbreviations are used in the proof:*

$K_{R_i}(p, \alpha)$*: "Receiver i knows that the p-th data segment is* α*"; similar for* $K_S(p, \alpha)$*;*
$K_{R_i}(p, -)$*: "Receiver i knows the value of the p-th data segment"; similar for* $K_S(p, -)$*;*
$E_G(p, \alpha)$*: "Every receiver of group G knows that the p-th data segment is* α*";*
$E_G(p, -)$*: "Every receiver of group G knows the value of the p-th data segment".*

Theorem 1. *Let* $\mathcal{R}$ *be any set of runs consistent with the knowledge-based algorithm from section 3 where:*

- *the environment allows for deletion and reordering errors, but no other kinds;*
- *The safety property holds (so that at any moment the sequence* Y *of data elements received by each* R_i *is a prefix of the infinite sequence* X *of data elements on* S*'s input tape).*

Then for all runs in $\mathcal{R}$ *and all* $k \geq 0, j \geq 0$ *the following hold:*

[Forth]: R_i *stores* $K_{R_i}K_S(j+k, \alpha) \rightarrow \Box K_{R_i}K_S(E_GK_S)^k(j, \alpha)$.
[Back_i]: S *stores* $K_SK_{R_i}(j+k, -) \rightarrow \Box K_SK_{R_i}K_S(E_GK_S)^k(j, -)$.
[Back_G]: S *stores* $K_SE_G(j+k, -) \rightarrow \Box K_S(E_GK_S)^{k+1}(j, -)$.

In the proof below we use the following general principle from temporal logic, see subsection 2.1:

A $P(\Box\varphi) \rightarrow \Box\varphi$

From the assumptions of the theorem, we can derive some consequences that we will regularly use in the proof:

B Because $\mathcal{R}$ is consistent with the knowledge-based algorithm, S and R_i store all relevant information from the packages that they receive. Moreover, packages that are sent have the following form: $K_{R_i}\varphi$ or $K_S\varphi$, from which the following can be concluded. If R_i receives $K_S\varphi$, then R_i stores $K_{R_i}K_S\varphi$, thus al.1so $\Box K_{R_i}K_S\varphi$. Similarly for S.

C Under the same assumption of $\mathcal{R}$ being consistent with the knowledge-based algorithm, system $\mathcal{R}$ can be viewed as a system of perfect recall. Now we have in general that $K_S\Box\varphi \rightarrow \Box K_S\varphi$, see axiom KT1 from subsection 2.1.

Proof
We prove *theorem 1* by induction on k.

First we look at the situation for $\mathbf{k = 0}$.
From B follows the **Forth**-part for $(k = 0)$ namely

$$R_i \text{ stores } K_{R_i} K_S (j, \alpha) \rightarrow \Box K_{R_i} K_S (j, \alpha) . \tag{1}$$

R_i sends an acknowledgement only if it received a package. Together with A and B we have:

$$\text{if } R_i \text{ sends } K_{R_i} (j, -) \text{ then } P (R_i \text{ stores } K_S (j, \alpha)) , \tag{2}$$

$$\text{so } P \Box K_{R_i} K_S (j, \alpha) , \text{ and } \Box K_{R_i} K_S (j, \alpha) .$$

S only stores an acknowledgements if it also received it from R_i, thus it knows that R_i has sent it in the past.

$$\text{If } S \text{ stores } K_S K_{R_i} (j, -) \text{ then } K_S P (R_i \text{ sends } K_{R_i} (j, -)) ... \tag{3}$$

With A, C and the fact proven at (2) it can now be derived that:

$$K_S P (\Box K_{R_i} K_S (j, -)) , \text{ and } K_S \Box K_{R_i} K_S (j, -) , \text{ so } \Box K_S K_{R_i} K_S (j, -) . \tag{4}$$

If (3) and (4) are put together, then we have the **Back_i**-part of the theorem for the j-th data segment $(k = 0)$.
S receives acknowledgements from all the receivers and is able to retrieve information out of this. We go back two steps and look at another knowledge level of S instead of the knowledge level between S and just one receiver.
S only stores acknowledgements if it did receive those. If S has received acknowledgements of a certain package from R_G where $G = \{1, ..., n\}$ then S knows that $R_{i<i=1..n>}$ have sent these acknowledgements in the past.

$$\text{If } S \text{ stores } K_S E_G (j, -) \text{ then } K_S P (R_{i<i=1..n>} \text{ sends } K_{R_i} (j, -)) ... \tag{5}$$

With A, C and the fact proven at (2) it can now be deduced that:

$$K_S P (\Box E_G K_S (j, -)) , \text{ and } K_S \Box E_G K_S (j, -) , \text{ so } \Box K_S E_G K_S (j, -) . \tag{6}$$

If (5) and (6) are put together, then we have the **Back_G**-part of the theorem for the j-th data segment $(k = 0)$.

What knowledge about the j-th data segment will emerge for $k \neq 0$? This will be shown in the induction step.

induction step Suppose as induction hypothesis that **Back_i**, **Back_G** and **Forth** are valid for $k - 1$, with $k \geq 1$. Now a proof follows that **Back_i**, **Back_G** and **Forth** are also valid for k.
S only starts sending packages with position mark $(j + k)$ if it has received from all the receivers R_i an acknowledgement for package with position mark $(j + (k - 1))$.

$$S \text{ sends } K_S (j + k, \alpha) \rightarrow P (S \text{ stores } K_S E_G (j + (k - 1) , -)) . \tag{7}$$

With the **Back_G**-part of the theorem for $k - 1$ and A, the following can be deduced:

$$S \text{ sends } K_S (j + k, \alpha) \rightarrow \Box K_S (E_G K_S)^k (j, -) . \tag{8}$$

R_i knows this fact. So if R_i receives a package from S with position mark $j+k$, then R_i knows that S has sent this package somewhere in the past. From the fact given at (8) together with A and B, the following can be derived:

$$R_i \text{ stores } K_{R_i}K_S\left(j+k,\alpha\right) \rightarrow \Box K_{R_i}K_S\left(E_GK_S\right)^k\left(j,-\right). \tag{9}$$

This is exactly what the **Forth**-part of the theorem says.
R_i only sends an acknowledgement for the $(j+k)$-th data element if he did store $K_{R_i}K_S\left(j+k,-\right)$ in the past. With A, now the following can be derived:

$$R_i \text{ sends } K_{R_i}\left(j+k,-\right) \rightarrow \Box K_{R_i}K_S\left(E_GK_S\right)^k\left(j,-\right). \tag{10}$$

S knows this fact. So if S receives an acknowledgement from R_i for the $(j+k)$-th data segment, then S knows that R_i has sent this acknowledgement in the past. Using A and B it can now be concluded that:

$$S \text{ stores } K_SK_{R_i}\left(j+k,-\right) \rightarrow \Box K_SK_{R_i}K_S\left(E_GK_S\right)^k\left(j,-\right). \tag{11}$$

and this is exactly the **Back_i**-part of the theorem.
S receives acknowledgements from all R_i. At a certain time S will have received an acknowledgement for the $(j+k)$-th data segment from all R_i. Thus,

$$S \text{ stores } K_SE_G\left(j+k,-\right).$$

With A and B it can now be concluded that:

$$S \text{ stores } K_SE_G\left(j+k,-\right) \rightarrow \Box K_S\left(E_GK_S\right)^{k+1}\left(j,-\right). \tag{12}$$

and this is exactly the **back_G**-part of the theorem.

5 Discussion and conclusion

This paper extends the knowledge-based protocol TCP presented in [14] in order to allow true one-on-group communication, creating general knowledge up to any desired level about the identity of the group and the announcement sent to the group by an initiating sender. This is very useful for creating collective motivational attitudes such as collective intentions in CPS [1, 3].

In [12], a procedure is given for establishing shared beliefs (of the form E-BEL$_G$E-BEL$_G(\varphi)$) between two or more agents, and it is argued that, by reasoning, these shared beliefs lead to a common belief between the agents. We believe that their protocol is correct, but there is unfortunately a gap in their proof that E-BEL$_G$E-BEL$_G(\varphi)$ implies C-BEL$_G(\varphi)$, which they use to prove correctness. Our protocol, instead, establishes very precise levels of knowledge within a group of agents using a knowledge-based one-on-group protocol.

Another related paper is [5] where a procedure is presented that, under some strong assumptions about the communication channels, trust among group members and temporary persistence of some relevant beliefs (e.g. the group should be aware of the procedure), establishes a common belief C-BEL$_G(\varphi)$. The idea is essentially that one initiator first broadcasts the message φ to all agents in the group, based on a standard low-level communication protocol such as TCP, ensuring that it knows at a certain point that E-BEL$_G(\varphi)$; then the initiator broadcasts the message that C-BEL$_G(\varphi)$ to all of them. Typically, such strong assumptions are only true in very fixed multi-agent systems, such as participants in a rescue operation who work in a fixed team according

to a commonly known fixed procedure [5]. The one-on-group procedure presented in the current paper does not establish common beliefs but only fixed levels of group knowledge, but can work in open environments because the prerequisites are much weaker than those in [5]. This is in line with the argument in [4] that developers of multi-agent systems can decide beforehand, according to organization structure, goal and environment, which level of team-awareness of relevant propositions is appropriate for a given application.

As future work, it would be interesting to design a logic exactly suited to communication protocols such as TCP and the one-to-many protocol given here, in a similar fashion as the sound and complete system TDL developed by Lomuscio and Woźna in [10] for authentication protocols. For such a system with a computationally grounded semantics of interpreted systems, it may even be possible to develop model checking techniques in order to check relevant properties automatically.

References

1. F. Dignum, B. Dunin-Kęplicz, and R. Verbrugge. Creating collective intention through dialogue. *Logic Journal of the IGPL*, 9(2):289–303, 2003.
2. D. E. Douglas. *Internetworking with TCP/IP, Volume 1: Principles, Protocols and Architectures*. Prentice Hall, Upper Saddle River, NJ, USA, 2000.
3. B. Dunin-Kęplicz and R. Verbrugge. Dialogue in teamwork. In J. M. Fonseca et al., editor, *Proceedings of The 10th ISPE International Conference on Concurrent Engineering: Research and Applications*, pages 121–128, Rotterdam, 2003. A.A. Balkema Publishers.
4. B. Dunin-Kęplicz and R. Verbrugge. A tuning machine for cooperative problem solving. *Fundamenta Informaticae*, 63:283–307, 2004.
5. B. Dunin-Kęplicz and R. Verbrugge. Creating common beliefs in rescue situations. In B. Dunin-Keplicz, A. Jankowski, A. Skowron, and M. Szczuka, editors, *Proceedings of Monitoring, Security and Rescue Techniques in Multiagent Systems (MSRAS)*, Advances in Soft Computing, pages 69–84, Berlin, 2005. Springer.
6. R. Fagin, J. Y. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge*. MIT Press, Cambridge (MA), 1995.
7. R. Goldblatt. *Logics of Time and Computation*. Number 7 in CSLI Lecture Notes. Center for Studies in Language and Information, Palo Alto (CA), 1992.
8. J. Halpern, R. van der Meyden, and M. Vardi. Complete axiomatizations for reasoning about knowledge and time. *SIAM Journal on Computing*, 33(4):591–612, 2000.
9. J. Y. Halpern and L. D. Zuck. A little knowledge goes a long way: Simple knowledge-based derivations and correctness proofs for a family of protocols. *Journal of the ACM*, 39(3):449–478, 1992.
10. A. Lomuscio and B. Woźna. A complete and decidable security-specialised logic and its application to the TESLA protocol. In P. Stone and G. Weiss, editors, *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 145–152. ACM Press, 2006.
11. J.-J. Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Schience*. Cambridge University Press, 1995.
12. S. Paurobally, J. Cunningham, and N. R. Jennings. Ensuring consistency in the joint beliefs of interacting agents. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 662–669. ACM Press, 2003.
13. J. Postel. Transmission control protocol (TCP). Technical Report RFC 793, Internet Society, September 1981. ftp://ftp.rfc-editor.org/in-notes/rfc793.txt.
14. F. Stulp and R. Verbrugge. A knowledge-based algorithm for the internet protocol TCP. *Bulletin of Economic Research*, 54(1):69–94, 2002.
15. H.P. van Ditmarsch and B.P. Kooi. Unsuccessful updates. In E. Álvarez, R. Bosch, and L. Villamil, editors, *Proceedings of the 12th International Congress of Logic, Methodology, and Philosophy of Science (LMPS)*, pages 139–140. Oviedo University Press, 2003.
16. M. Wooldridge and N. R. Jennings. The cooperative problem-solving process. *Journal of Logic and Computation*, 9(4):563–592, 1999.