Distributed Coding in Multiagent Systems*

Filip Miletić
Faculty of Electrical Engineering,
Mathematics and Informatics
Delft University of Technology
Delft, The Netherlands
Email: f.miletic@ewi.tudelft.nl

Patrick Dewilde
Faculty of Electrical Engineering,
Mathematics and Informatics
Delft University of Technology
Delft, The Netherlands
Email: p.m.dewilde@ewi.tudelft.nl

Abstract – *This article describes using distributed coding of logical information in multiagent systems. First we identify failures of the agent platform in case of an unreliable underlying network. Next a simple, but inefficient way of distributing the agent state is given. Finally, we expose the way of solving failure problems by use of state snapshots and distributed coding. We develop a model of the network, based on the notion of* erasure graph channel. *We treat the properties of the erasure code encoding data efficiently with respect to a prescribed level of* distortion. *This work introduces information-theoretic ideas to multiagent middleware. It should enable applications of multiagent systems in environments where it is hard to establish a reliable communication infrastructure. The exposed work is a part of project Combined, funded by DECIS Lab, Delft.*

Keywords: Distributed coding, multiagent systems, erasure graph channel

1 Introduction

Multiagent systems consist of multiple processing entities (agents) connected together by the means of a middleware-type *multiagent platform*. Agents communicate by means of *communication acts* [12]. The communication acts are messages in a high-level language (KQML, SL), transported by a reliable network protocol, such as TCP/IP. The QoS guarantees made by TCP/IP ensure that messages are delivered in the order they were sent. But TCP/IP alone is not suitable for dynamic environments, e.g. the wireless network environment (WLAN). In WLAN, links between nodes change as environment and antenna positions change. These occurrences can cause communication acts to fail. Figure 1 gives an UML diagram of a failed communication act. After the Receiver agent is destroyed, the transaction cannot continue. A simple resolution to the problem of failed transaction is in replicating the agent services across multiple, functionally identical agents. For the sake of illustration, the number of replicating agents is two. The distribution requires that both agents offering a replicated service synchronize their internal

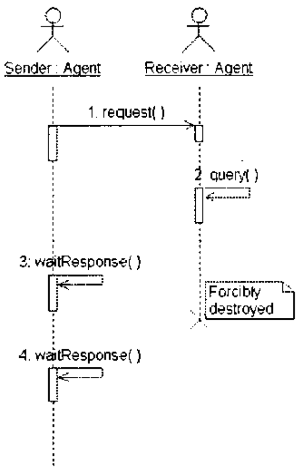

Figure 1: Failed communication act. The figure shows an UML *sequence diagram*. The title boxes represent UML classes. The vertical *lifelines* represent flow of time. Boxes on the lifelines indicate threads of control, and arrows connecting the threads of control stand for the messages exchanged between the threads.

states so that repeated queries can resume at the place where they were interrupted. Each agent holds a private copy of the transaction state, and total storage used is twice as large than in the original case, but now the probability of failed transaction is less: if in the case of the transaction in Figure 1 the probability of failure is ϵ, the probability of error in the case of replicated transaction is $\epsilon^2 < \epsilon$. With n replicated agents, it is possible to have the probability of error of ϵ^n. By using replication, the probability of error can be lowered with more replicated agents, at the expense of storage. With n replication agents, only a $1/n$ fraction of total storage contains useful information and bandwidth required to ensure synchronism grows at least linearly with n. When storage and bandwidth resources are scarce or unreliable, replication schemes are not feasible. Scarcity of resources and unreliability is illustrated by a crisis-management prototype application [3].

In an emergency service, every worker is equipped with a digital device that helps gathering information about a crisis to the command center; it also helps propagate instructions back to the workers. The data transported in such a way is precious, but cannot be buffered locally as the existence of a single device is uncertain. To resolve problems of data storage and communication, we search for an architecture that would help alleviate the adverse environment conditions.

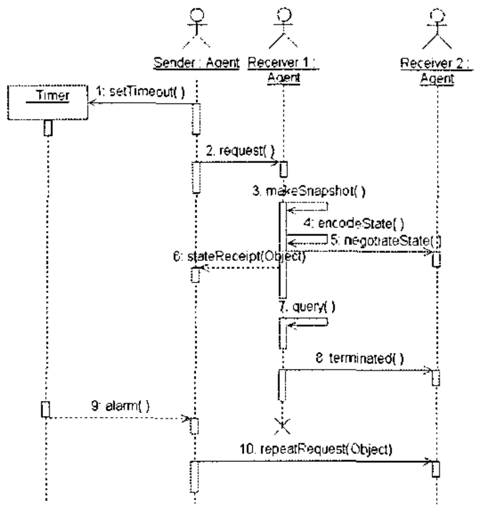

Figure 2: Coded communication.

2 Architectural Constraints

As the computational environment we assume a networked multi-agent system. In this setting there exists a finite number of separate threads of control, each executing on a single host. An agent is comprised of a number of such threads, which are able to communicate with each other through an uniform network abstraction layer. The underlying network may be physically realized in any way permitted by technology. The model presented herein is built on top of a wireless network model due to specific demands of the application [2]. However, the discussion is independent of the type of networking. One platform instance providing such interconnect is Jade [4]. It implements agency services in accordance with recommendations of FIPA [1]. The intended deployment diagram builds on top of this infrastructure and is given in figure 3 where the agent *virtualization* is proposed. The new platform must enable load balancing between several physical hosts. On the user side, the platform should keep a functionally identical interface towards the agent proper and should appear as a normal FIPA-compliant agent platform. The agent running on top of the distribution layer is called a *virtual agent*. The purpose of agent virtualization is increased resistance to processing errors. In order to be able to analyze the properties of virtualization layer, it is required that *failure model* be analyzed. The *Erasure* failure occurs when data are stored in a subset of

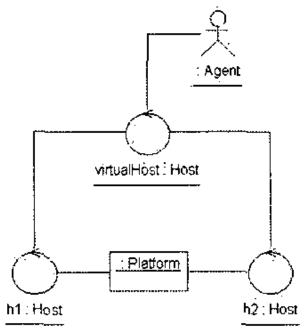

Figure 3: Agent virtualization. Intended deployment diagram of the new agency framework.

the available nodes. Due to the activity of the environment, any node of the subset can be lost, along with all the data it stores. When later retrieving the data, the segment which was located at lost nodes cannot be retrieved. The failures are handled in terms of the erasure-graph described in the following section. Following the exposition given in [11], we define software code abstraction, to be able to represent internal data managed by the software code underlying an agent.

Definition 1 (Software code $S \equiv \langle T_S, F_S, C_S \rangle$) We may characterize the code on top of which an agent is built as a triple $S \equiv \langle T_S, F_S, C_S \rangle$ where: T_S is the set of all data types managed by S; F_S is a set of predefined functions which makes access to the data objects managed by the agent available to external processes, and C_S is a set of type composition operations.

Notions of definition 1 are used to represent the programs that the agents execute, and here we use them to explain what portions of agent state are subject to encoding. The reader interested in formalization of these notions is referred to [11].

Definition 2 Let there be given a program code S that is associated with an agent, and let there be given a function SF*, mapping the set of all software codes into a set of binary strings of variable length. Then* SF(S) *is called a* serialization *of S.*

Definition 3 (Code inclusion) Let $S_1 = \langle T_S^1, F_S^1, C_S^1 \rangle$ and $S_2 = \langle T_S^2, F_S^2, C_S^2 \rangle$ be two pieces of software code. We will say that S_2 is included by S_1 ($S_2 \preceq S_1$) if the following holds: $(T_S^2 \subseteq T_S^1) \wedge (F_S^2 \subseteq F_S^1) \wedge (C_S^2 \subseteq C_S^1)$.

Definition 4 (Functional subset, equivalence) Let there be two agents A_1 and A_2, which possess collections of software code, labeled S_1 and S_2 respectively. Then: (a) Agent A_2 is a functional subset of A_1 ($A_2 \preceq A_1$) if $S_2 \preceq S_1$; (b) Agents A_1 and A_2 are functionally equivalent ($A_1 \equiv A_2$) if it holds $A_1 \preceq A_2 \wedge A_2 \preceq A_1$.

Definition 5 (Functional identity) An agent is a functional identity if it can accept a serialization SF(S) *of program code* S, *and is able to repeat it upon request at a later time.*

In particular, communication can be looked at as a functional identity agent (which in itself may be a functional subset of a different agent) storing a serialization of program code S, SF(S).

Definition 6 (Agent virtualization) Given an agent A that runs on a conventional multiagent platform, virtualization is a function $A_v = \mathcal{A}_v(A)$, *yielding an agent running on a distributed platform, where* $A_v \equiv A$.

Here we focus on distribution of functional identity agents. In this setting the virtual agent does not perform any additional processing of the software code. Rather, the code (or its serialized image) is intended for transmission. Since that is not always possible, according to the failure model, the identity agent acts as a message buffer storing the message until new transmission window opens. In light of the exposition so far, the research goal of this effort is finding suitable virtualization function $\mathcal{A}_v$ for various instances of agents. The need to make agent state explicitly reachable by serialization primitives influences the internal architecture of the agent. The state is made explicit by introducing the *blackboard* (BB), a container object in which all stateful agent information is to be stored [12]. The blackboard is accessed by stateless operators, pieces of software code that are allowed to manipulate the stored objects. The operators can add, remove or modify blackboard objects. There also exist other benefits of employing a blackboard based internal agent architecture, some of which are found in [5]. The software code attributed to the BB is denoted as: S_{BB}. The serialization of BB software code, SF(S_{BB}) is then used as the input to the encoder. The agents are deployed on nodes connected by a short range radio-based network (e.g. WiFi). We assume that nodes in the network cooperate in relaying each others' data packets. We let $G_c = \langle V, E[r_-, r_+] \rangle$ be the network (graph) formed when $|V|$ nodes are placed uniformly and independently in $\mathcal{D}$, and two nodes $v_i, v_j \in V; v_i \neq v_j$ can communicate if $r_- < \|x_i - x_j\| < r_+$, where the norm used is Euclidean (L^2) norm, and x_k is the position vector of v_k. The radius r_+ is usually referred to as the range of a node in G_c [8], and $r_- = (1-\Delta)r_+$ is the minimum internode distance needed for interference-free transmission [8]. All messages that a node sends are broadcast.

3 The Erasure-Graph

This section addresses specifically multi-hop ad-hoc networks, consisting of symmetrical nodes. We assume that the environment can switch off any network node with a given probability. We consider the case in which a partition of a string of binary data, of a predetermined length is made, and each component of the partition is remembered at a particular node, and determine the average number of bits that can be retrieved. To pursue this goal we define the erasure-graph channel and give a way to encode data over its nodes.

Let V be a set of vertices, and $E(V)$ be a set of pairs $\langle v_1, v_2 \rangle$, where $v_1, v_2 \in V$. Let the graph G be defined as $G = \langle V, E \rangle$. We interpret this in the usual sense: V is a set of vertices, and $E(V)$ is a set of edges, constructed by connecting pairs of vertices from V. Erasure is the construction of a new graph G' from G, by picking a (possibly improper) subset V' of V, with respect to some probability distribution.

Definition 7 (Erasure pattern) Let there be given a graph $G = \langle E, V \rangle$. *An erasure pattern is a mapping* $E_p : V \to \{0, 1\}$, *which associates each vertex of G with* 1 *if this node is to be erased from the graph, and* 0 *if this node is not to be erased. The set of all possible erasure patterns is denoted as* $\underline{E}$.

Definition 8 (Degradation model) Let there be given $E_p \in \underline{E}$ *and an induced probability* $\Pr(E_p)$ *that a given pattern* E_p *occurs. A degradation model is given by* $\underline{D} = \langle \underline{E}, \Pr(E_p) \rangle$.

Definition 9 (Erasure graph) Let there be given a graph $G = \langle V, E \rangle$, *and the degradation model* $\underline{D}$. *The tuple* $\underline{G} = \langle G, \underline{D} \rangle$ *is called the erasure-graph.*

As a special case, we consider the degradation model where all single vertex erasures are i.i.d. (independent identically distributed) with probability ε that each component of V disappears in V'. $G' = \langle V', E' \rangle$ is now a partial subgraph of G induced by a subset of vertices $V' \subseteq V$. We assume that G can be transformed into any of the subgraphs G' that can result from removing a number of vertices from V. The transition probability depends upon ε, the outage probability. It defines the probability that a node will go offline. Since outages are i.i.d., the probability of $k \leq |V|$ nodes going offline is:

$$\Pr(k \text{ nodes fail}) = \binom{|V|}{k} \varepsilon^k (1-\varepsilon)^{|V|-k}. \qquad (1)$$

Definition 10 (I.i.d. degradation model) Let there be given an erasure graph $\underline{G}$ *and an outage probability of each its node,* ε. *Let* $E_p \in \underline{E}$. *Define* $k(E_p)$, *the number of nodes that are online in* E_p *as:* $k(E_p) = |\{x | x \in \operatorname{dom} E_p, E_p(x) = 0\}|$. *An i.i.d. degradation model is a degradation model* $\mathcal{D}_i = \langle \varepsilon, \Pr(k(E_p)) \rangle$, *where the probability of any erasure pattern* E_p *is given by equation (1).*

Definition 11 (Erasure-graph degradation) Let there be given an erasure graph $\underline{G}$. *A degraded graph, with respect to a degradation model* $\underline{D}$ *is an erasure graph* $\underline{G}' = \langle G', \underline{D}' \rangle$, *where* $G'(E', V')$ *is a partial subgraph of* G *induced by a set of vertices* $V' = \{v : E_p(v) = 0\}$, *and* $\underline{D}' = \langle E_p', \Pr(E_p)' \rangle$. *A state transition of the graph* $\underline{G}$ *to* $\underline{G}'$ *by an erasure pattern* E_p *is denoted as:* $\underline{G} \xrightarrow{E_p} \underline{G}'$. *The* *erasure pattern that corresponds to this transition is denoted as $E_p(\underline{G}, \underline{G}')$.*

A communication channel is defined by specifying a tuple $\langle A_x, P, A_y \rangle$ [6], where A_x is a source alphabet used for signaling, A_y is the alphabet of the received symbols, and $P = [p_{ij}]$ is the channel transition matrix, where for $i \in A_x, j \in A_y$, $p_{ij} = \Pr(Y = j \in | X = i)$. This section will give a mapping from the parameters of newly introduced erasure graph channel to this tuple. Let there be given an alphabet Ξ. Let there be given an erasure graph $\underline{G}$, and let each vertex V be attributed by a symbol from alphabet $\Upsilon = \Xi \cup \{\epsilon\}$. Let $\underline{\Upsilon} = \Upsilon^{|V|}$. To define the source and received alphabets, we set $A_x \equiv \Xi, A_y \equiv \underline{\Upsilon}$. *Next we construct P from the parameters of an erasure graph $\underline{G}$.* Consider a single erasure pattern $E_p \in \underline{E}$. For a given $x \in \Xi$ set $(y_i) = y \in \underline{\Upsilon}$ in such a way that $y_i = \epsilon$ if $E_p(V_i) = 1$, or $y_i = x$ otherwise. Define each element of P as: $p_{xy} = \Pr(E_p)$.

Definition 12 (Erasure-graph channel) Let there be given alphabets $\Xi, \underline{\Upsilon}$ and P a channel transition matrix, as described above. The erasure graph channel is given by: $C = \langle \Xi, P, \underline{\Upsilon} \rangle$.

An (M, n) code for the channel $\langle \Xi, P, \underline{\Upsilon} \rangle$ consists of the following [6]: (a) an index set $\{1, \ldots, M\}$; (b) an encoding function $X : \{1, \ldots, M\} \to \Xi$, yielding codewords $X(1)$, $X(2), \ldots, X(M)$. The set of codewords is called the codebook; and (c) a decoding function: $g : \underline{\Upsilon} \to \{1, \ldots, M\}$ which is a deterministic rule assigning a guess to each possible received vector. Let that Ξ be a set of binary strings of length n, and $X \in \Xi$ has to be transmitted through this channel. Let X_k be the k-th bit of the binary string, and let there be given an erasure graph $\underline{G}$ that is used for encoding. Transmission has two distinct operations: encoding and decoding. Encoding consists of determining a partition $\Pi = \langle \pi_1, \ldots \pi_{|V|} \rangle$. In it, each π_i is a set of bits X_k from X that are assigned to a vertex $v_i \in V$. For all pairs of i, j such that $i \neq j$, it holds true that $\pi_i \cap \pi_j = \emptyset$, and $\bigcup_{k: v_k \in V} \pi_k = S$. Decoding starts with a particular vertex $v \in V$. A set R is constructed of all vertices $v_m \in V$ such that a path exists between v and v_m in G (v is included). Then all bits from π_m such that $v_m \in R$ are gathered. We assume that it is easy to determine index for a bit K_j into the binary string K when decoding. This is a reasonable assumption if we consider Π fixed, as it is always possible to search all π_i to determine which one contains X_j, but here we do not go into efficient ways of doing it. The decoded string will be denoted as Y. Only the message bits that were not erased by transition *and* are accessible from the current reader position are considered readable. These bits are arranged in a received vector Y, with all the erased bits set to ϵ, so each bit position of Y can take on a value from the set $\{0, 1, \epsilon\}$. We denote as p_{ij} the probability that due to erasure, a certain X would be received as Y. By considering in turn all of the 2^n distinct messages X, and all the possible received messages Y, we can in principle determine p_{ij} for all $i, j \in \overline{1, n}$.

4 Message Encoding

Under failure models given here it is not the best decision to keep a serialized software code on a single host only, as it could be lost as a consequence of failure. In this section we discuss in more detail how to formalize this requirement, and define optimality criteria for finding the best distribution. The distribution optimality is governed by several design rules which put constraints on the type of data distribution, and the properties of the encoder/decoder pair: (i) The decoder should be such that, given a threshold ε_T, decoding can succeed if number of erased components is less than $\varepsilon_T n$; (ii) Given two equivalent data distributions, the encoder should decide in favor of the "more distributed" one; (iii) The encoder should be able to generate a code for arbitrary number of hosts ($|V|$), and arbitrary message block length (n). The rule (i) can be satisfied up front. It is known that for iterative decoders such a threshold exists [10], and thus iterative decoder can be used. The remaining requirements are more involved to satisfy. To treat requirement (ii), a notion of distortion needs to be introduced and an equivalence of codes needs to be established. To satisfy requirement (iii), the size of the code must be runtime-variable and should not be sensitive to changes in partitioning. We remark that a rateless, linear parity generator code such as Luby code [9] or a variety can be used as a starting point in the analysis. A compact representation of the configurations and their characteristics is captured by the configuration generating function.

Definition 13 (Coefficient (following [7])) Let there be given a polynomial $P(x) = \sum_i a_i x^i$ over an arbitrary field F, where $i \in Z$ and $a_i \in F$. Let the coefficient function be defined as: $[x^i]P(x) \equiv a_i$.

Definition 14 (CGF) Let there be given an erasure graph $\underline{G}$ and a corresponding partition $\Pi = \langle \pi_i \rangle$, for $i \in \overline{1, |V|}$. Let each node in C be described by the corresponding erasure probability ε_i. The configuration generating function (CGF) $G(y)$ is given by:

$$G(y) = \prod_{i=1}^{|V|} \left[\varepsilon_i + (1 - \varepsilon_i) y^{|\pi_i|} \right]. \qquad (2)$$

If the number of bits retrievable from $\underline{G}'$, for which $\underline{G} \xrightarrow{E_p} \underline{G}'$ holds, is q, $\Pr(E_p)$ is given as $\Pr(E_p) = [y^q]G(y)$.

Proposition 1 Let there be given a CGF $G(y) = \sum_i p_i y^i$, as in definition 14. Then the probability of occurrence of a configuration with b bits not erased is given by $[y^b]G(y)$.

Proof: Consider the case in which there exists only one node. The generating function, $G_1(y)$ in this case is: $G_1(y) = \varepsilon_1 + (1 - \varepsilon_1) y^{|\pi_1|}$. In the case of a single node, $n = |\pi_1|$. The probability of the configuration of 0 bits is then: $[y^0]G_1(y) = \varepsilon_1$, and the only other possible configuration of $|\pi_1|$ bits is $[y^{|\pi_1|}]G_1(y) = 1 - \varepsilon_1$, so for $|V| = 1$, the claim holds. Assume now that the claim holds for $G_i(y)$, $i < n$, and add a new node, n. The new generating function is: $G_n(y) = \varepsilon_n G_{n-1}(y) + (1 - \varepsilon_n)y^{|\pi_n|}G_{n-1}(y)$. The first term, $\varepsilon_n G_{n-1}(y)$ generates all configurations in which the node n is not present. If we denote $G_k(y) = \sum_i p_{ki}y^i$, where $p_{ki} = \Pr(k \text{ nodes}, i \text{ bits})$, the term becomes: $\sum_i \varepsilon_n p_{n-1,i}y^i$. As $p_{ni} = \Pr(n \text{ nodes}, i \text{ bits}) = \varepsilon_n p_{n-1,i}$, the claim holds for the first term. The second term equals to: $\sum_i (1 - \varepsilon_n)p_{n-1,i}y^{i+|\pi_n|}$. As $p_{n,i+|\pi_n|} = \Pr(n \text{ nodes}, i + |\pi_n| \text{ bits}) = (1-\varepsilon_n)p_{n-1,i}$, the claim holds for the second term too. Noting that the two terms partition the set of possible configurations yields the claim. $\square$

The binary string to be distributed is pre-coded by a linear parity generator code. Each bit of the string is obtained by computing a modulo-2 sum of a certain number of bits of the original message [9]. The set of modulo-2 sums for all the bits of the binary string induce a bipartite graph [10], where the bits of the original (input) message correspond to *data nodes*, and the bits of the binary (output) string correspond to *check nodes*. We look at the node degree distribution of the check nodes in the induced graph. The distribution is given by the following degree generator function.

Definition 15 (DGF) Let d_i be the number of nodes of degree i in a partition Π of an erasure graph $\underline{G}$. The degree generator function (DGF) is a polynomial: $d(x) = \sum_i d_i x^i$.

From definition 15 we can see that for an erasure graph with π retrievable bits, $\pi = d(1)$ holds. We extend the configuration generator function to contain the probabilities for node distributions.

Definition 16 (CDGF) Let there be given an erasure graph $\underline{G}$ and a corresponding partition $\Pi = \langle\pi_i\rangle$, for $i \in \overline{1,|V|}$. Let each node in C be described by the corresponding erasure probability ε_i, and let each node i have its degree distribution $d_i(x) = \sum_k d_{ik}x^k$. The configuration-distribution generating function (CDGF) $G(x,y)$ is given by:

$$G(x,y) = \prod_{i=1}^{|V|} \left[\varepsilon_i + (1 - \varepsilon_i)y^{d_i(x)}\right]. \tag{3}$$

Let $G(x,y)$ be a CDGF. Then, $G(1,y)$ is the corresponding CGF, as by definition $d_i(1) = |\pi_i|$ must hold. The existence of CDGF allows us to talk in compact terms about the combinatorial properties of the degree distributions.

Proposition 2 (CDGF properties) Let $G(x,y)$ be a CDGF of a given graph $\underline{G}$, and $i \in \overline{1,|V|}$ $d_i(x) = \sum_j d_{ij}x^j$. Let $G^*(x,y)$ be a CDGF generated by $d_i^*(x) = \sum d_{ij}^2 x^j$. The expected degree distribution is given by:

$$\mathrm{E}\{d(x)\} = \left(y\frac{\partial G}{\partial y}\right)_{y=1}. \tag{4}$$

The variance of the degree distribution is given by:

$$\mathrm{Var}\{d(x)\} = \left[y\frac{\partial G^*}{\partial y} - \left(y\frac{\partial G}{\partial y}\right)^2\right]_{y=1}. \tag{5}$$

Distortion corresponds to the distance between the complete and the received message: any decodable message has zero distance from the complete message. Any undecodable message has distance proportional to the information lost in the channel. It is reasonable to assume that the distortion of any configuration is therefore the distance from the decoder threshold, $(1 - \varepsilon_T)n$ [10]. The distortion D is the expected distortion of the configurations reachable from a given one.

Definition 17 (Distortion) Let there be given a message of length n, a partition Π and let C be a configuration and $D_c = \{C' : C \xrightarrow{E_p} C'\}$. Let $R = (1 - \varepsilon_T)n - \sum_{i\in C'}|\pi_i|$. The distortion *is given by:*

$$d(C) = \sum_{C'\in D_c} \Pr(E_p(C,C')) \cdot R \cdot I(R > 0), \tag{6}$$

where the sequence $\{n_i\}$ is the number of code elements per agent i, and $I(x)$ is the indicator function: equal to 0 when x is false, or 1 otherwise.

Given a maximum distortion $d_{\max}$ and the area $v(d_{\max}) = \{C'|d(C') \le D_{\max}\}$, the optimal configuration C^* is given by

$$C^* = \arg\min_{C\in v(D_{\max})} f(C), \tag{7}$$

where $f(C)$ is a disambiguation function that helps in the choice of unique C if multiple solutions are available. We suggest $f(x) = (\sum_i |\pi_i|^2)^{1/2}$. Other disambiguation functions, as well as distortion definitions could also be possible. This choice ensures that there exists only one solution to equation (7). This solution is found by a gradient descent on the surface defined by equation (7). An example contour plot of $d(C)$ for a configuration with two nodes, 1 and 2, is given in figure 4. The contours connect points with equal distortion. For a given maximum distortion e.g. $d_{\max} = 20$ the solution is the lattice point (n_1^*, n_2^*) closest to $(0,0)$, and within the area bounded by the contour $d_{\max} = 20$.

Proposition 3 (Properties of Distortion) Let the distortion $d(C)$ be defined as in definition 17. $d(C)$ is a positive piecewise linear function of $\langle|\pi_i|\rangle$, and is bounded above by $(1-\varepsilon_T)n$. Relative distortion is then $\delta = d(C)/((1-\varepsilon_T)n)$.

Proof: Every term of equation (6) is positive, and defined only for nonnegative values of $|\pi_i|$. By setting all π_i to be empty sets (i.e. maximum distortion), the maximum of $d(C)$ is given by:

$$\begin{aligned} d(C) =& (1 - \varepsilon_T)n \sum_{C'\in D_c} \Pr(E_p(C,C')) \cdot \\ & I[(1 - \varepsilon_T)n > 0] = (1 - \varepsilon_T)n \end{aligned} \tag{8}$$

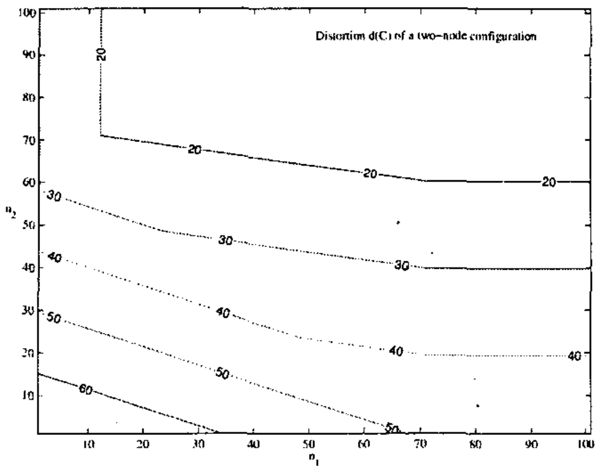

Figure 4: The level lines of a two-node configuration distortion $d(C)$. Number of bits stored in each node are denoted as n_1, and n_2. The erasure probabilities are $\varepsilon_1 = 0.7, \varepsilon_2 = 0.3$, and message length is 100.

By re-expressing $d(C)$ in terms of $|\pi_i|$, we get: $d(C) = a_{0,C} - \sum_i a_{i,C}|\pi_i|$, where $a_{0,C} = (1 - \varepsilon_T)n$, and $a_{i,C} \neq 0$ in general. □

To illustrate the evolution of the erasure graph, we set up a *coding game*. The number of hosts is determined and a list of reliability figures is given. Let the number of hosts be $|V|$, with host $i \in \overline{1, |V|}$ having erasure probability ε_i. In this experiment, $|V| = 3$. The code distribution is computed so that the distortion is minimal, for relative distortions ranging from $\delta_{\min} = 0$ to $\delta_{\max} = 1$.The environment then removes a random subset of hosts, with given probabilities ε_i and empty replacement hosts are introduced. The number of hosts removed in the example is 1, and its associated erasure probability is $\varepsilon_1 = 0.8$. The replacement host has $\varepsilon_1' = 0.3$. Given the new host connectivity, the new equilibrium point is computed. The difference in number of coding elements n_{11} in the first and in the second case (n_{12}) is computed and given in figure 5.

5 Conclusion

This article gives the reader an overview of the architectural properties of a novel multiagent platform at different detail levels. We motivated the architectural choices that make decodable the explicit agent state. The network connectivity model was introduced in form of the erasure-graph channel and its relevant notions were presented in order to build a vocabulary for talking about distributed coding. The notion of distortion is introduced, along with an explanation of the design requirements and choices that follow. Once the acceptable distortion is given, and optimality criterion defined, it is possible to determine the partition of the code and an example thereof was given.

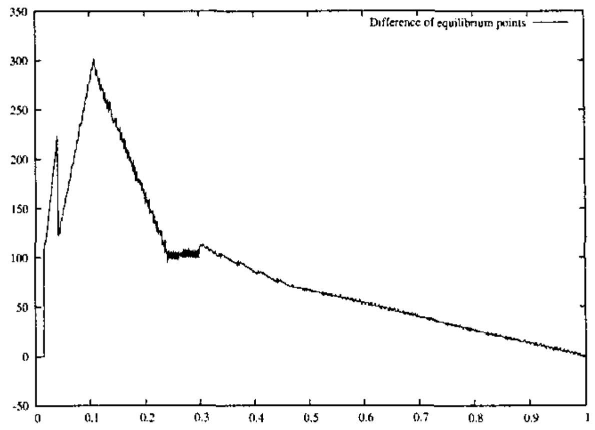

Figure 5: Equilibrium points difference for n_{11} (x axis) and n_{12} (y axis), as the relative distortion ranges from $\delta_{\min} = 0$ to $\delta_{\max} = 1$. For a given δ, the difference of the number of code elements in the cases of $\varepsilon_1 = 0.3$ and $\varepsilon_2 = 0.8$ are shown.

References

[1] Foundation for intelligent physical agents. http://www.fipa.org.

[2] Combined architecture technical concepts, 2003.

[3] Combined system: Case architecture, 2003.

[4] Elisabetta Cortese, Filippo Quarta, and Giosue Vitaglione. Scalability and performance of jade message transport system.

[5] The cougaar architecture guide. http://cougaar.org/docman/view.php/17/56/CAD_11_0.pdf.

[6] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, New York, NY, USA, 1991.

[7] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics; Second Edition*. Addison-Wesley, 1997. GRA r 94:1 1.Ex.

[8] P. Gupta and P. Kumar. Capacity of wireless networks, 1999.

[9] Michael Luby. Lt codes. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, page 271. IEEE Computer Society, 2002.

[10] T. Richardson and R. Urbanke. Modern coding theory, June 2003.

[11] V. S. Subrahmanian, Piero Bonatti, Jürgen Dix, Thomas Eiter, Sarit Kraus, Fatma Ozcan, and Robert Ross. *Heterogeneous Agent Systems*. MIT Press/AAAI Press, Cambridge, MA, USA, 2000.

[12] G. Weiss, editor. *Multiagent Systems: A Modern Approach*. MIT Press, 2002.