An Adaptive Choice of Messaging Protocol in Multi-Agent Systems

Chris Preist, Siani Pearson
Living the Vision Department
HP Laboratories Bristol
HPL-98-08
January, 1998

E-mail: [cwp,siani]@hplb.hpl.hp.com

decentralised systems, communication protocols

There are a variety of choices which need to be made when setting up a multi-agent community. In particular, which agents communicate with which, what protocols they use, and what information flows from one to another. Such design choices will affect the efficiency of the community with respect to several parameters – accuracy, speed of solution, and message load.

In this paper, we consider one class of problem in which multi-agent systems engage – service provision. Using a simple, abstract, form of this problem, we use a mathematical analysis to show that three different messaging protocols result in varying message loads, depending on certain parameters such as number of agents and frequency of request.

If the parameters are fixed, we can conclude that one of these three protocols is better than the others. However, these parameters will usually vary over time, and hence the best of the three protocols will vary. We show that the community can adopt the best protocol if each individual agent makes a local decision based on which protocol will minimise its own message load. Hence, local decisions lead to globally good behaviour. We demonstrate this both mathematically and experimentally.

1. INTRODUCTION

The designer of a multi-agent system needs to make various choices as to how the agent community is organised: in particular, how the agents communicate and co-ordinate with each other. A variety of different approaches have been proposed in the literature, such as the contract net [12], the facilitator approach [5], distributed blackboard architectures [7,9] and market-based control [2].

Increasingly, flexible multi-agent toolkits and languages are being developed which do not constrain the developers to any one of these approaches [1,3,8,11]. They must make a choice between the alternatives proposed. Such a choice involves making decisions about:

- How to partition the tasks performed by the system between different agents and what reasoning each agent can perform.
- How to partition the information used by the system between different agents, and whether to allow information to be duplicated.
- What messages one agent can send to another, and in what circumstances.

When comparing alternative approaches, it is unlikely that any one will be the 'best'. Instead, the system designer needs to make trade-offs between various criteria for assessing the performance of the system, depending on the characteristics of their application.

These criteria include, amongst others;

- Quality of solution - how well the system task is done, according to some measure.
- Communications efficiency - how much communication takes place to produce a solution.
- Time - how long it takes to produce a solution.

A more extensive list of such criteria is given in [10].

If a designer knows that, for problems similar to their application, one set of design choices leads to a fast system, while another set leads to a system which is communications efficient, they can choose which to adopt, based on what is important in their circumstances. In a time-critical application, the system designer would be prepared to use more communication if it produces a solution more quickly. They may even be prepared to accept a poorer quality solution in less time. In other circumstances, time may not be a major issue.

For this reason, we believe that systematic comparative analysis of how different design choices affect different performance criteria is necessary. In this paper, we present work of this nature.

We present a simple form of the service provision problem in section 2, and three different messaging protocols in section 3. In section 4, we assess the communications efficiency of these protocols with respect to this problem. As we argue above, other factors must be taken into account when deciding which protocol to use. However, for the purpose of this paper, we will assume that, (providing the protocol produces a quality solution), communications efficiency is the priority. For brevity, we will refer to the most communications efficient protocol in a given circumstance as "optimal".

Given certain parameters of the problem, such as rate of service requests and number of service providers, we determine the number of messages agents would send under each protocol. Hence, for given parameter values we can determine which protocol is optimal.

However, the parameters of the problem are rarely static. New agents may join, increasing the amount of activity in the system. Agents may become increasingly loaded, and so less available to perform tasks. For this reason, the optimal protocol at one time may not be optimal at another. It will change as the environment the system is in changes. It may therefore be better to allow the system to make such choices dynamically, rather than fixing them at design time. As circumstances change, the system can adapt, choosing a new protocol as its current one ceases to be optimal.

In section 5, we demonstrate mathematically that a decentralised adaptive approach can be used by a system carrying out our simple form of service provision. Local decisions, taken by each individual agent, can lead to the agent community adopting a protocol that is optimal. Section 6 presents an overview of experimental results to support this mathematical analysis.

2. THE SERVICE PROVISION PROBLEM

We have chosen to use the service provision problem as the initial focus of our work. In the abstract, service provision consists of matching client agents with certain needs with service provider agents able to meet those needs. The service provider agents may themselves act as clients, and subcontract parts of the service they provide to other agents.

Examples of service provision problems could be:

1. Connecting an agent requiring the fax number of a customer with any database able to provide that information.
2. Connecting an agent wishing to purchase a CD with an Internet supplier able to provide it at the best price.
3. A nurse's patient has a cardiac arrest. The nurse's agent contacts the first cardiac specialist available in the area, asking for their assistance.

The different characteristics of these examples will have an affect on the choice of protocol (and other design decisions). An efficient protocol for example 3 is not likely to be the most efficient for example 1; it will be too complex for such a simple problem. Hence, we cannot analyse the effect of choosing different protocols with respect to service provision in general. Instead, we must analyse it with respect to various service provision problems with different characteristics.

In this paper, we focus on a relatively simple service provision problem. We use this to demonstrate our approach, as even this case is quite rich. There is no single 'best' protocol choice in it, so we can consider how the agent community can swap protocols in response to changes in the environment.

The characteristics of the problem are;

- All service providers give the same service, with the same quality.
- Service providers get no benefit from providing the service.
- Service providers can be unavailable; they all have a probability p of being unavailable at any given time.

We consider an abstract service provision problem with these characteristics, and analyse the communications efficiency of three different messaging protocols. By performing the analysis in the abstract, it can be applied to any service provision problem with these characteristics. One example would be the routing of a client request to one of a team of people on an information helpline.

3. THE PROTOCOLS TO BE COMPARED

For the purposes of this analysis, we assume that certain design decisions have been made about the system. In particular, we assume that there is a single facilitator with which all service providers must register. It operates in recommend mode [4], giving clients lists of service providers when they request it to.

We consider three alternative protocols, to explore the effect of two design decisions. Firstly, whether to broadcast requests, or to send requests to individual agents one at a time. Secondly, whether to provide the facilitator with updated information on the availability of service providers. We call the three protocols embodying these design decisions naïve broadcast, naïve one-to-one, and informed one-to-one.

(a) Naïve broadcast

The recommender contains a list of all service providers offering this service, and no other information. It provides a list of providers to a requesting client. The client contacts all providers on the list by broadcasting a message which does not require a response. Providers reply if they are currently available, and the client selects one. This can be viewed as a simple variant of the contract net operating in general broadcast mode [12].

(b) Naïve one-to-one

Again, the recommender simply provides a list of service providers to a client on request. This time, the client contacts one of these[1] with a request for service, using a message which requires a reply. The service provider either replies that it is available, or that it is busy. In the latter case, the client contacts another, repeating the process. This is a variant of the contract net operating in point-to-point mode [12].

[1] Here, and in the informed one-to-one subsequently, we assume that if a client makes a selection of one of a set of alternative providers which appear equivalent, then it makes this choice randomly.

(c) Informed one-to-one

This time, the recommender contains information about whether each service provider is currently available or busy. The service providers must keep this up to date by sending a message to it whenever their state changes. On request from a client, the recommender gives a list of all service providers currently available. The client contacts one of them.

4. ANALYSIS OF THE PROTOCOLS

Each of these protocols is designed in such a way that, if there is a service provider available, it will be found. However, the number of messages required will vary. To consider in what circumstances each is optimal, we must see what the average message load in each system is, and how it varies as the parameters of the problem vary.

The parameters we consider are:

N	- The number of providers of this service.
p	- The probability of a service provider being available.
t_u	- The average time a service provider is unavailable.
M	- The average number of service requests made by clients per second.

We now derive formulae that give the average number of messages used to satisfy one client request, for each of the three protocols**.**

(a) Naïve broadcast

The client sends a message to the recommender and receives a reply. It then broadcasts a message to all N service providers. Each service provider has a probability p of being available and replies only if it is. Hence, the average number of replies is Np.

Therefore, the average number of messages generated by the naive broadcast protocol for the client to get an offer of service, $c(nb)$, is given by;

$$c(nb) = 2 + N + Np$$

Note that, as *p* can range between 0 and 1, *c*(*nb*) can range between *N*+2 and 2*N*+2.

(b) Naïve one-to-one

Again, the client sends a message to the recommender and receives a reply. It then contacts any one of the providers which the recommender proposed. The provider replies that it is able to perform the service, with probability *p*, or that it is not, with probability (1 - *p*). In the latter case the client contacts another provider, and so on.

Hence the average number of messages generated by the naive one-to-one protocol, *c*(*no*), is a probablistic summation;

$$c(no) = 2 + \sum_{i=1}^{N} 2ip(1-p)^{i-1} + 2N(1-p)^{N}$$

The last term represents the messages generated in the case that no provider is available.

For simplicity of notation, we let

$$E(N,p) = \sum_{i=1}^{N} 2ip(1-p)^{i-1} + 2N(1-p)^{N}$$

Hence the equation becomes:

$$c(no) = 2 + E(N,p)$$

Solving the summation, we can show that, for $p \neq 0$,

$$E(N,p) = \frac{2(1-(1-p)^{N})}{p}$$

As *p* ranges from 0 to 1, *E*(*N*,*p*) is a monotonically decreasing function which ranges between 2*N* and 2. Hence, *c*(*no*) can range between 4 and 2*N*+2.

(c) Informed one-to-one

In the case of informed one-to-one, there are two kinds of message exchange to be considered; messages sent to connect a client with a service provider, and messages sent by the service provider to keep the status information in the recommender up-to-date.

When a client wishes to connect with a service provider, it firstly contacts the recommender and receives a reply listing the service providers currently available. It then contacts one of these and receives a reply. Therefore, 4 messages are generated.

We now consider the number of status updates a provider sends. A provider has a probability 1-*p* of being unavailable at any given time, and is unavailable for t_u seconds on average. Hence it will become available once every $t_u/(1-p)$ seconds, and similarly, will become unavailable once every $t_u/(1-p)$ seconds. Therefore, each service provider sends, on average, $2(1-p)/t_u$ update messages per second to the recommender.

To compare this protocol with the others, we need to calculate the number of update messages sent per client request. There are *N* service providers sending update messages, and there are *M* client requests per second. Hence, there are $2N(1-p)/Mt_u$ update messages per client request.

Hence, the average number of messages generated by the informed one-to-one protocol per client request, *c*(*io*), is given by;

$$c(io) = 4 + \frac{2N(1-p)}{Mt_u}$$

We now have equations which give the average number of messages to connect a client with a service provider for each protocol. Hence, given specific parameter values, we can determine which protocol is most communications efficient.

Based on these equations, we can make some general observations;

As *p* tends towards 1, *c*(*nb*) will increase, tending towards $2N + 2$. However, *c*(*no*) will decrease, tending towards 4. As $4 < 2N+2$ for all $N>1$, we can conclude that, for high values of *p*, the naïve one-to-one protocol will be more communications efficient than the naïve broadcast strategy in any system with more than one service provider.

As *p* decreases towards 0, *c*(*nb*) will decrease tending towards $N + 2$, *c*(*no*) will increase tending towards $2N + 2$. Hence, for low values of *p*, the naïve broadcast strategy will be more communications efficient than the naïve one-to-one strategy.

$c(io)$ increases as N increases and p decreases, but the most significant factor in $c(io)$ is the size of Mt_u. As Mt_u tends to zero, then $2N(1-p)/ Mt_u$ tends to infinity (provided $p \neq 1$), and therefore $c(io)$ does too. Hence, one of the other strategies will be more communications efficient.

However, if $Mt_u > N$ then $c(io) \leq 4 + 2(1-p)$. Furthermore, as Mt_u increases, $c(io)$ rapidly decreases towards 4. Hence, in almost all circumstances where $Mt_u > N$, informed one-to-one will be the most communications efficient strategy.

Mt_u can be viewed as a measure of the busyness of the system; Busyness increases as the number of requests per second increases, and also as the downtime of service providers increases. Hence, informed one-to-one is most communications efficient when a system is reasonably busy. If a system is not very busy, then another protocol is better.

Hence, even with the simplifying assumptions we have chosen, no one protocol is the most efficient in all circumstances. Any one of the three may be most efficient, depending on how many providers there are, how many requests are made, and how often and for how long providers can be unavailable.

For given parameter values, the equations derived above can be used to determine which of the three protocols would be most efficient. However, the parameters that determine this decision may vary with time. New service providers may arrive, or existing ones may leave, resulting in a change of N. The probability, p, of a provider being unavailable is likely to fluctuate dramatically, as client demand varies. Hence, it is not possible to decide which protocol is most efficient at design time. Rather, it is necessary to allow the decision to be made dynamically by the agent community, in response to changing circumstances. We will now consider a possible mechanism for doing this.

5. DYNAMIC CHOICE OF MESSAGING PROTOCOL

5.1 Local choices lead to an optimal global choice

The decision to change from one protocol to another could be either centralised or decentralised. In a centralised approach, a monitoring agent would make the decision to change protocol on behalf of the entire community, and then inform all the agents. This has the advantage that the monitoring agent could make decisions based on what is best for the community as a whole, but has the disadvantage that it would need to gather vast amounts of data, and would need to handle agents joining and leaving the system. In a decentralised approach, agents alter their protocol in response to what is happening locally to them, and to what task they are performing. This has the advantage that no single agent needs to gather vast amounts of data, and hence the decision process should be simpler. However, it has the disadvantage that no agent takes a global view, and hence their decisions may not be best for the community as a whole. We will consider the latter approach, and show that, in this case, local decisions do lead to an optimal choice globally.

Firstly, we consider the choices available to each agent. A client agent has a choice between adopting a naïve broadcast protocol, or a naïve one-to-one protocol. Once it has sent messages of the given type, the service providers are constrained in how they react. However, the service providers have the choice of whether to provide availability information or not; they control the decision of when to move to an informed one-to-one protocol. (The client, of course, still has the option of using one of the other protocols).

Now we will look at what information is available locally to each agent. We focus on the number of messages an agent sends and receives, and compare this with the total number of messages sent and received in the community. We will show that if each agent chooses the protocol which will result in it minimising the number of messages it sends and receives, then the community overall will adopt the protocol which is most efficient. In section 5.2, we will look at how an agent can determine this.

For the naive broadcast and naive one-to-one protocols, a client agent either sends or receives every message involved in the provision of its service. Hence, if the client chooses the protocol which minimises the number of messages it sends and receives, this choice will be the better for the community as a whole.

The situation is more complex when the third protocol, informed one-to-one, is also considered. We would like the service providers to offer availability information only if the informed one-to-one protocol would be the most efficient for the agent community. Furthermore, we would like the client agents to adopt the informed one-to-one protocol as soon as the providers offer availability information. If the providers are offering the information and it is not being used, then the messages which update this information are wasted.

Assume that service providers offer availability information only if the informed one-to-one protocol would result in them sending and receiving less messages, on average, than either of the other two protocols. We now show that, at such a time, the informed one-to-one protocol would be the most efficient for the community to adopt.

We do this by considering the average number of messages a provider sends and receives per client request. We consider this for each protocol;

- Under the naïve broadcast protocol, each service provider agent sends and receives, on average,$1 + p$ messages per client request.
- Under the naïve one-to-one protocol, each service provider agent sends and receives, on average, $E(N,p)/N$ messages per client request. As $2 \le E \le 2N$, this varies from between $2/N$ to a maximum of 2.
- Under the informed one-to-one protocol, each service provider agent sends and receives, on average, $2/N + 2(1-p)/Mt_u$ messages. The first term represents the average number of messages it sends and receives to/from the client, while the second represents the number of update messages it sends to the facilitator.

Firstly, we consider the choice between the informed one-to-one protocol, and the naive one-to-one protocol.

A single service provider agent will choose the informed one-to-one protocol if;

$$\frac{E(N,p)}{N} > \frac{2}{N} + \frac{2(1-p)}{Mt_u}$$

As $E(N,p) \ge 2$, and $N \ge 1$, this is equivalent to;

$$E(N,p) - 2 > \frac{2N(1-p)}{Mt_u} \qquad \text{(Equation 5.1)}$$

We now consider the total number of messages. The informed one-to-one protocol is more communications efficient if $c(no) > c(io)$.

Hence, this gives the inequality;

$$2 + E(N, p) > 4 + \frac{2N(1-p)}{Mt_u}$$

Subtracting 4 from each side gives:

$$E(N, p) - 2 > \frac{2N(1-p)}{Mt_u}$$

This is identical to equation 5.1. Therefore, a service provider will select the informed one-to-one protocol in preference to the naive one-to-one protocol only if it is more efficient for the community as a whole.

We now consider the naïve broadcast, and compare it to the informed one-to-one protocol. A service provider will select the informed one-to-one protocol in preference to the naive broadcast protocol if;

$$1 + p > \frac{2}{N} + \frac{2(1-p)}{Mt_u}$$

Multiplying this by *N*, $(N \geq 1)$, and adding 2 to each side gives:

$$2 + N + Np > 4 + \frac{2N(1-p)}{Mt_u}$$

Recalling the equations from section 4, this is equivalent to *c*(*nb*) > *c*(*io*).

Hence, a service provider will select the informed one-to-one protocol in preference to the naive broadcast protocol only if it is more efficient for the community as a whole.

Combining this with the previous result, we have shown that a service provider will provide the information necessary for the informed one-to-one protocol if and only if the informed one-to-one protocol is more communications efficient than the other two protocols.

We must now show that if the service provider agents make the effort to provide the information necessary for the informed one-to-one protocol, then the client agents will use it. We assume that they will do so if and only if it will result in them sending and receiving less messages than either of the two other protocols they could adopt. Recall that, in the other two protocols, all messages are either sent or received by the client.

Hence, the client sends and receives $c(no)$ messages in the naïve one-to-one protocol, and $c(nb)$ messages in the naïve broadcast protocol. In the informed one-to-one protocol, the client sends and receives 4 messages. So we must show that, if the service providers offer the information for informed one-to-one, then $4 < c(no)$ and $4 < c(nb)$.

We have already shown that the service provider agents will offer the information for informed one-to-one if and only if $c(io) < c(no)$ and $c(io) < c(nb)$.

As $c(io) = 4 + 2N(1-p)/Mt_u$, where N, M and t_u are all greater than zero, and $1 \geq p \geq 0$, it follows that $4 \leq c(io)$. Hence $4 < c(no)$ and $4 < c(nb)$, as required.

So we have shown that, if service providers offer availability information when the informed one-to-one protocol is most efficient from their local perspective, then the protocol will also be most efficient from the perspective of the client.

To summarise, we have shown that;

- If a client agent makes a choice between the naïve broadcast and naïve one-to-one strategy based on which minimises the number of messages the agent sends and receives, this choice will also globally be the more communications efficient of the two alternatives.
- If a service provider agent offers the information needed for the informed one-to-one protocol only if this protocol would result in it sending and receiving, on average, fewer messages than either of the other two protocols, then it will do so only when the adoption of this protocol would globally be the most communication efficient.
- If a service provider agent offers the information needed for the informed one-to-one protocol in the circumstances described above, then a client agent will always make use of it.

From these results, we can draw an important conclusion. If each agent decides which protocol to adopt based simply on which will minimise the average number of messages it sends and receives, then the agent community as a whole will adopt the protocol which is most communications efficient.

5.2 How agents choose the protocol

We have shown that in a simple form of the service provision problem, a local choice of protocol by each agent can lead to a choice which is the best for the community as a whole. Each agent simply needs to choose the protocol that minimises the total number of messages it sends and receives. We now consider how an agent is able to make this decision.

Client agents need to make two decisions; whether to use the informed one-to-one protocol if some service providers are offering the necessary information, and whether to use naïve broadcast or naïve one-to-one otherwise. The first decision is trivial; we showed in section 4.1 that it is always better for a client to use informed one-to-one if it is offered.

Secondly, the client must decide what to do if informed one-to-one cannot be used. This can occur either because no service provider is offering availability information, or because all service providers offering availability information are currently busy.

In this case, the client can use the two equations which give the number of messages it will send and receive under the naïve broadcast and naïve one-to-one protocols;

$$c(nb) = 2 + N + Np$$

$$c(no) = 2 + E(N, p)$$

Given the estimate of *N* and *p*, it can use these equations to give an informed guess as to which protocol should be adopted.

N can be known for certain - it is the number of service provider agents on the list given by the recommender agent. If a hybrid protocol is being used, and all service providers providing information are busy, then we remove these agents from the list. (This can be done either by the client, or by the recommender sending the list).

The probability *p* can be estimated by the agent keeping track of how often a service provider agent is free when it tries to make contact with it. If, as is likely, *p* is expected to vary with time, more recent experience could be weighted more strongly when calculating the estimate.

Alternatively, a client can adopt a more empirical approach. It can try using the protocol it isn't currently using every now and then, and swap when it finds that the other protocol usually results in less message traffic.

A service provider needs to decide whether to offer availability information or not. It can do this by comparing the number of update messages it would send with the number of messages it would receive and send when busy if it didn't keep this information updated.

For an agent which is not offering availability information, it simply counts how many times it switches from available to unavailable, or vice-versa, in a given time period, and counts how many messages it receives and sends while unavailable. When the number of messages it receives and sends while unavailable is usually above the number of switches it makes, it is worthwhile providing availability information.[2] Hence, the agent will switch protocol at this point.

If an agent is offering availability information, then it needs to determine when it is no longer worth it doing so, i.e. at what time the number of messages it sends to keep the availability information updated is greater than the number of messages it avoids having to send and receive while unavailable.

From the agent's perspective, assuming all service provider agents are also using the informed one-to-one protocol, it receives M/N service requests, on average, per second. It can get a value for N from the recommender agent. (Possibly, it would keep a note of N, and whenever N changes, the recommender would inform it). Hence, it can calculate M. It can estimate p by observing what proportion of the time it is unavailable. It can then compare the actual number of messages it currently receives per second, with an estimate of what it would expect to receive under the other protocols.

It would expect to send and receive $M(1+p)$ messages per second under the naïve broadcast. Under the naïve one-to-one, it would expect to send and receive $ME(N,p)/N$. If either of these is consistently lower than the actual number of messages sent and received, then the agent should stop giving availability information.

[2] Exactly how to define 'usually above' in this context will require experimentation.

Hence, in this way, a service provider agent can monitor its behaviour, and decide when it wishes to swap from providing availability information to not providing it. In an idealised homogenous agent community, all providers will make this decision at the same time, as they will all be receiving and handling the same number of service requests. However, in practice, this will not occur. Some providers may wish to change, while others do not. For the purposes of this paper, we assume that we wish to ensure that the community as a whole adopts the same protocol.[3]

To ensure this, we add a co-ordination agent, which acts as a vote collector from the service providers. If a provider wants to change strategy, it registers a vote. When enough votes are registered, the co-ordination agent sends a message to all service providers, and they swap protocol. Hence, we can maintain a homogenous protocol among service providers, using a loose form of centralised control.

6. Experimental Comparison of Protocol Efficiency

We have carried out experiments to provide empirical support for the mathematical analysis provided above. Our methodology is similar to that used in [6]. We have developed a system that can generate a community of agents for given values of the parameters N, p, t_u and M, with a given protocol. The community consists of a single client, a single facilitator, and N service providers, each which are available with a probability of p and have a down time of t_u time units. The client issues 100 task requests, at a rate of M requests per time unit[4]. The system then counts the total number of messages that are generated in the community by these requests.

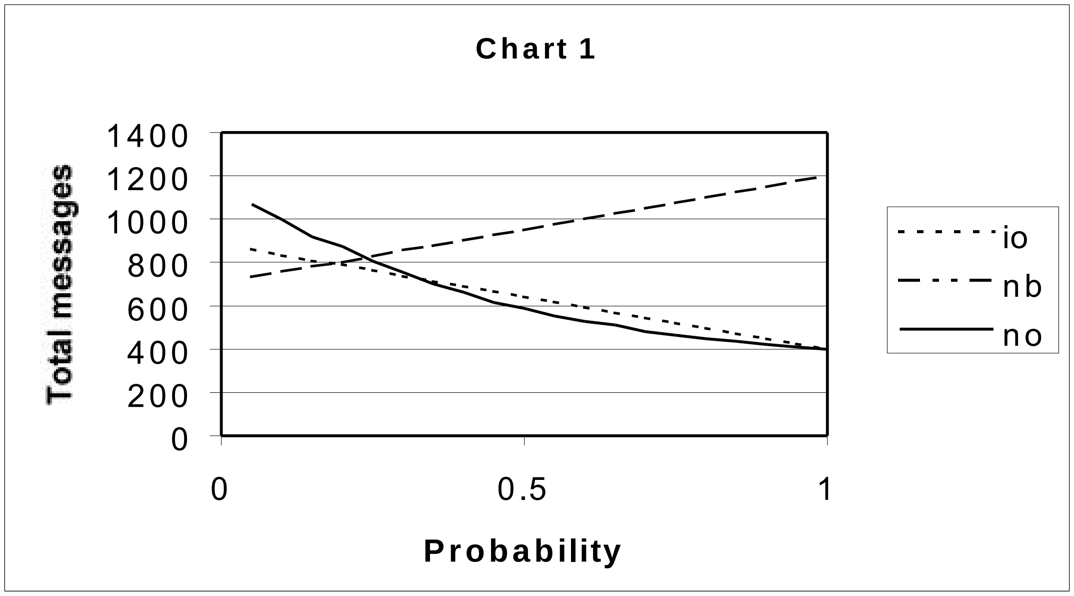
We present here the results of three series of experiments. In each case, we fix three of the four parameters, and consider a sequence of values of the fourth. For each value, we run the experimental system 100 times, and plot the mean number

[3] If we do not place this restriction, a hybrid protocol can develop, with some agents providing availability information and others not. This can be more communications efficient than any of the three protocols discussed, but needs careful management to prevent oscillation of behaviour.

[4] As the rate of requests by all clients is the factor which determines how many messages are generated, rather than the number of clients, we can safely use only one client agent without loss of generality.

of messages as a point on a graph. We repeat this for the different protocols, to get graphs of how the efficiency of the different protocols varies as the fourth parameter changes.

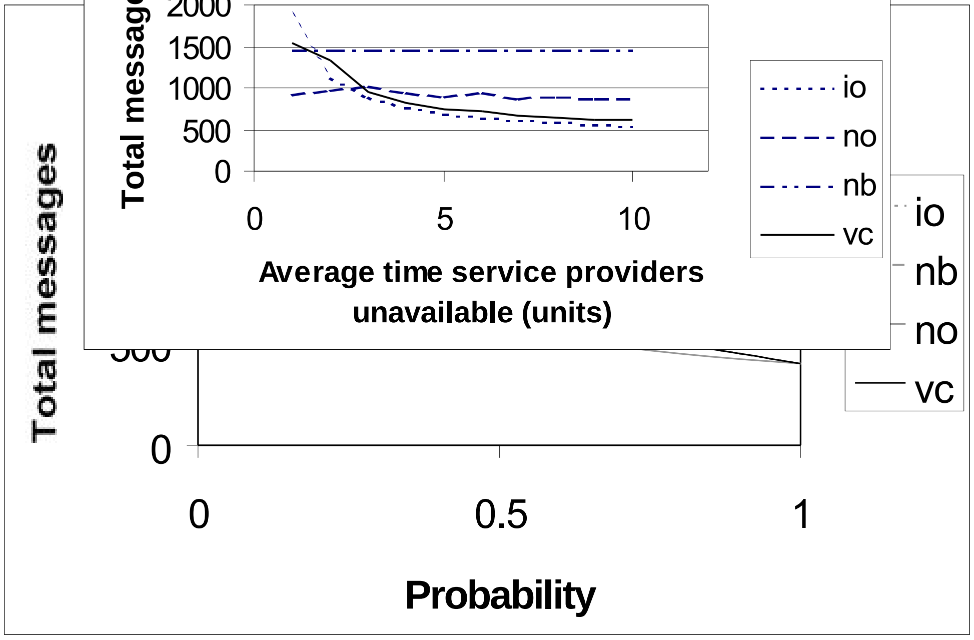
In experiment 1, we use the parameter values $N = 5$, $t_u = 2$ and $M = 1$. We allow p to range from 0.05 to 1 in increments of 0.05. Chart 1 gives the resulting graphs for each of the three protocols. These graphs corroborate the mathematical equations of section 4 The average difference between the theoretical predictions and the mean values found by experimentation is 0.22% for naïve broadcast, 1.22% for naïve one-to-one and 1.11% for informed one-to-one.

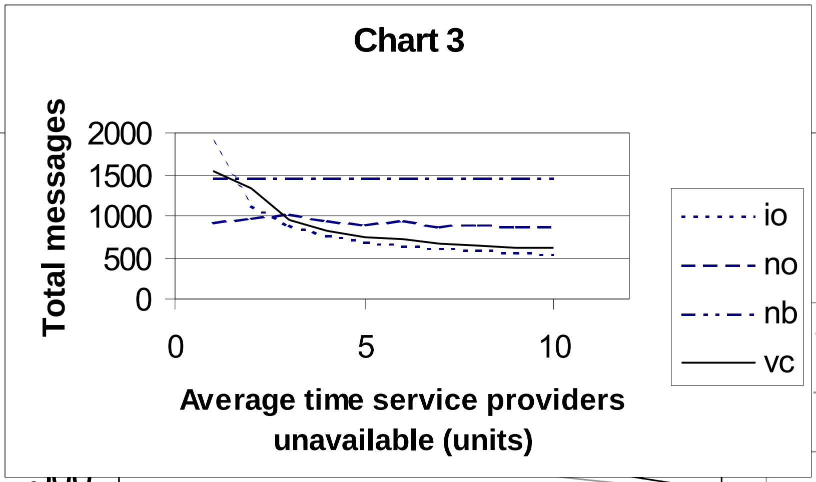


In experiment 2, we use the same parameter values, but introduce a fourth protocol: the vote collector. Service providers vote if they wish the community to swap from informed to naïve or vice-versa, using the techniques described in section 5.2 to make their decision. This vote takes place every 10 time units. The facilitator acts as the vote collector. The initial protocol is informed one-to-one. If a majority of service providers vote for a protocol swap, it sends a message out to indicate that the swap should take place. Chart 2 plots the resulting graphs. Both vote and swap messages are included in the message count.

We can see from chart 2 that the vote collector protocol tends to choose more efficient protocols at different probability values. The community chooses to use the naïve broadcast for values of *p* between 0.05 and 0.2. It wavers between all three protocols in the range 0.2 to 0.4, though tends to choose informed one-to-one in general. Finally, it choose informed one-to-one consistently for the range 0.4 to 1. In this last range, the most efficient protocol is naïve one-to-one; however, because the difference between this and informed one-to-one is small, the system remains on informed one-to-one.

In experiment 3, we plot all four protocols for parameter values *N* = 10, *p* = 0.25





and M = 1. We allow t_u to vary from 1 to 10 time units. We see in chart 3 that, as the equations predict, the number of messages used by naïve one-to-one and naïve broadcast remains constant, while the message count for informed one-to-one is very high for low values of t_u, but reduces rapidly as t_u increases. The vote collector successfully chooses naive one-to-one if $t_u \leq 2$, and informed one-to-one otherwise. On the graph, the vote collector message count is higher for $t_u \leq 2$ than the naïve one-to-one. This is because of the large number of messages sent in the first 20 time periods before it gets a chance to swap away from informed one-to-one, and the messages used in the vote collection process. If the experiment were run for more tasks (say 1000), we would expect this discrepancy to become less significant.

Conclusions and Future Work

In this paper, we have demonstrated that mathematical and experimental analysis of multi-agent system protocols can provide useful information to guide design choices when developing such systems. We have shown that some of these choices may best be made at runtime by the system in response to changes in the demands placed on it. Furthermore, we have shown that this decision can be made in a decentralised way, by allowing client agents to swap their protocol freely, and service provider agents to vote on the protocol to be used by their community.

We hope to be able to allow such decision making to be taken in an even more decentralised way, without the use of a vote collector. Such a protocol, the *hybrid protocol*, allows some service providers to provide availability information, while others do not. We believe that such a system will be more efficient than any of the other protocols, though care needs to be taken to ensure it settles in a stable state.

We believe that the approach of allowing an agent community to make architecture decisions dynamically, in a decentralised fashion, can lead to systems which are robust and efficient in the face of change. We hope to extend this approach to other architecture decisions, such as the number of facilitators and the mode in which they operate, and to more complex forms of the service provision problem.

Acknowledgements

Thanks to Martin Merry, Janet Bruten, Miranda Mowbray and Lin Jones for their assistance.

References

1. J. L. Alty, D. Griffiths, N.R. Jennings, E. H. Mamdani, A. Struthers, and M. E. Wiegand. ADEPT - Advanced Decision Environment for Process Tasks: Overview & Architecture. In *Proc. BCS Expert Systems 94 Conference* (Applications Track), Cambridge, UK, 359-371, 1994.

2. *Market-Based Control: A Paradigm for distributed resource allocation*. ed S.H.Clearwater. World Scientific, 1996.

3. dMARS product brief. http://www.aaii.oz.au/proj/dMARS-prod-brief.html

4. T. Finin and R. Fritzson. KQML as an Agent Communication Language. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94),* ACM Press, November 1994.

5. M.R. Genesereth and S.P. Ketchpel. Software Agents. *Communications of the ACM*, 37:7, 48-53, 1994.

6. C.Gu and T.Ishida. Analyzing the social behavior of the contract net protocol. *Agents Breaking Away*, *Proc. MAAMAW 96*. pp 116-127, 1996.

7. B.Hayes-Roth. A Blackboard Architecture for Control. *Artificial Intelligence Journal 26,* pp 21-321. 1985

8. H. Jean. JATlite overview. http://java.stanford.edu/java_agent/html/

9. L.V.Leao and S.N.Talukdar. An Environment for rule-based blackboards and distributed problem solving. *International Journal for Artificial Intelligence in Engineering*, 1(2): 70-79, 1986.

10. T.Sandholm. Agents in Electronic Markets. *Tutorial notes, Autonomous Agents 97 conference.*

11. A.Sloman. The SIM_AGENT toolkit. http://www.cs.bham.ac.uk/~axs/cog_affect/sim_agent.html

12. R.G. Smith. The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Trans. Comput*., 29, 1104-1113, 1980.