

Communication Efficiency in Multi-Agent Systems

Mary Berna-Koes
Robotics Institute
Carnegie Mellon University
mberna@cs.cmu.edu

Illah Nourbakhsh
Robotics Institute
Carnegie Mellon University
illah@cs.cmu.edu

Katia Sycara
Robotics Institute
Carnegie Mellon University
katia@cs.cmu.edu

Abstract—Despite the growing number of multi-agent software systems, relatively few physical systems have adopted multi-agent systems technology. Agents that interact with a dynamic physical environment have requirements not shared by virtual agents, including the need to transfer information about the world and their interaction with it. The agent communication languages proven successful in software based multi-agent systems incur overheads that make them impractical or infeasible for the transfer of low level data. Instead, real world systems typically employ application specific protocols to transfer video, audio, sensory, or telemetry data. These protocols lack the transparency and portability of formal agent communication languages and consequently are limited in their scalability. We propose augmenting the capabilities of current multi-agent systems to provide for the efficient transfer of low level information, by allowing backchannels of communication between agents with flexible protocols in a carefully principled way. We show that this extension can yield significant performance increases in communication efficiency and discuss the benefits of incorporating backchannels into a search a rescue robot system.

I. INTRODUCTION

Agents in a multi-agent system (MAS) must be able to interact and communicate with each other. This usually requires a common language, an Agent Communication Language, or ACL. Much work has been done in developing ACLs that are declarative, syntactically simple, and readable by people. KQML [1] and FIPA-ACL [2] are two of the most widely used ACLs in multi-agent systems. These languages have been very successful in facilitating the communication and coordination of software agents in a variety of domains including organizational decision making [3], [4]; financial management [5]; and even aircraft maintenance [6]. This approach to interagent communication, while well suited to communication related to negotiation or the transfer of high level information, has significant overhead which frequently proves a drawback in systems that require the transfer of low level data or systems with stringent time and bandwidth limits. Real robot systems typically fall into both of these categories. They may require the transfer of telemetry, video, audio, and sensory data at high frequencies in real time over relatively slow wireless networks or RF modems.

The data transferred by real robot systems tends either to be relatively small messages, such as telemetry commands, or large multimedia files, such as streaming video. The overhead due to the ACL is the most significant for small messages. Multimedia files are also ill-suited for transfer via ACLs as the intermediary representation of the files is not compatible

with a textual representation. Furthermore, most ACLs require ASCII text messages which results in an inflation of the message size plus additional processing. Consequently, most physical robot teams do not use the agent communication languages developed by the multi-agent systems community. Instead, they typically use an ad hoc solution, defining their own protocols specific to their system. This approach lacks the semantic power and transparency afforded by a language like KQML and does not allow robots from various teams to communicate.

The robot community would benefit from the adoption of a formal ACL inside the framework of a MAS, provided that the efficiency of the current approach of hard coding various protocols were not significantly decreased. This adoption would enable different robot systems to communicate and cooperate, allow robots to negotiate the flow of information, and provide increased transparency. Likewise, the MAS community could benefit from a more efficient method of transferring low level data, such as media files, particularly as web cameras become more ubiquitous. Thus we have the apparently opposing goals of improving efficiency for the transfer of media files and small messages at high frequencies and preserving the portability, readability, and declarative nature of an agent communication language like KQML.

We propose a two tiered communication strategy, augmenting the current multi-agent system architecture. This solution realizes both these goals by uniting the strengths of ACLs and the methods used by the robot community. This extension, which we call *backchannels*, has been implemented on the RETSINA MAS [7], which uses KQML. We present the backchannel extension, detail the necessary supporting network drivers, show significant analytical and experimental performance improvements achieved with backchannels, and relate the successes of this approach in a search and rescue robot system.

II. TWO TIERED COMMUNICATION

The current approach to multi-agent communication is to allow one channel of communication between agents and to constrain communication to one language. This works well for software agents communicating high level information (such as commitments or negotiations) when efficiency is not of paramount importance. Systems operating in a dynamic physical environment need to send low level information (e.g. telemetry or video data). These low level communications

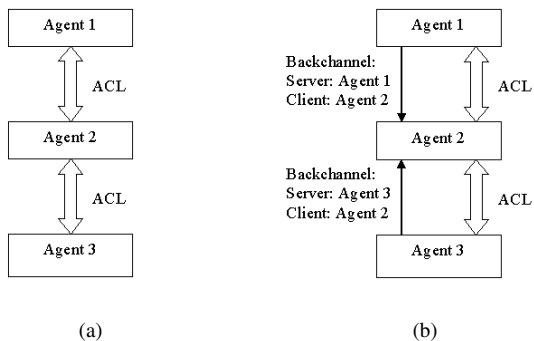


Fig. 1. Figure (a) shows the current MAS architecture. Figure (b) illustrates how backchannels augment the current system. The backchannels do not replace current lines of communication using the ACL, nor can they exist between two agents not communicating at the ACL level.

often occur at high frequencies, which can clog the main line of communication. We extend the current architecture to allow multiple lines of communication between agents, as shown in figure 1. The additional lines, or backchannels, are for the transfer of low level information.

In order to preserve the functionality and elegance of the ACL, all meta-information pertaining to the backchannels is related at the ACL level. Accordingly, as shown in figure 1, the backchannels are simplex, so that the sender, denoted as the server, and receiver, or client, are always evident from ACL level communication. The content of the messages sent over a backchannel is likewise fixed and agreed upon at the ACL level. As backchannels are a resource facilitating the transfer of information, negotiations on the regulation of this resource, namely the establishment of backchannels and the frequency of communication, take place at the ACL level.

As explained above, the use of backchannels is not to replace the current agent communication languages, which are necessary for communication between heterogeneous agents, but for the transfer of low level messages. The content and meaning of the messages to be exchanged is specified by referencing a user defined format description library. Each entry in the library consists of details of how to parse one message and a semantically meaningful description of the type of format, for example, “video” or “teleoperation-imperatives.” Some applications may use a translator agent to convert messages into the agent communication language or another human readable form for transparency. Since the efficiency of the message format can have a large impact on performance, as explained in the Analysis section, protocols should be carefully designed.

The purpose of the agent communication language is to facilitate communication between agents in a multi-agent system. Extending the system architecture therefore necessitates the augmentation of the language with communication acts relating to this extension. Accordingly, the ACL must support the establishment, flow control, and termination of backchannels.

Communication Act	Description
backchannel request	:request <line-type> :protocol <message-type> :reference-number <N> :server-name <name> <implementation specifics>
accept request	:accept <line-type> :reference-number <N> :server-name <name> <implementation specifics>
decline request	:decline <line-type> :reference-number <N> :server-name <name> :reason <details>
connection status	:connection <status> :reference-number <N> :server-name <name>

TABLE I

COMMUNICATION ACTS TO SUPPORT ESTABLISHMENT OF BACKCHANNELS

A. Establishing a backchannel

The formation of a backchannel begins with one agent desiring to send or receive low level information to or from another agent. If the initiator wishes to be the sender, the server, she requests a “client line” of the other agent. If the initiator wishes to receive information, acting as client, she requests a “server line” of the other agent. In either case, the request should indicate the type of data to be transferred by referencing the format description library. Additional technical information relating to the establishment of the backchannel, discussed in the Network Driver Details section and illustrated in figure 2, accompanies this request. The communication acts necessary for the formation of a backchannel are described in table I.

These communication acts were designed for KQML but can easily be adapted to other ACLs. Backchannels are distinguished by a reference number, unique to the server. All communication regarding a particular backchannel must specify the server and reference number. The <line-type> may be either server-line or client-line. The connection status, <status>, of the backchannel connection may be requested, accepted, connected, failed, declined, or terminated. Implementation specifics for TCP are described in the Network Drivers section of this paper.

B. Flow control of backchannels

Backchannels are simply a means of facilitating the transfer of information between agents. As information transfer is a commodity that consumes system resources, negotiations regarding the flow control of backchannels are necessary. The subject of negotiation is well explained in many papers on multi-agent systems [8], [9], and the details are not presented here. We examine the communication acts specific to backchannels which must be supported by the ACL. It is necessary to start and stem the flow of messages, regulate the frequency of transmissions, repeat messages, confirm the connection status of the backchannel, and inquire and respond to the total number of transmissions sent. The communication

Communication Act	Description
begin transfer/ halt transfer	:transmission <start/stop> :reference-number <N> :server-name <name>
repeat messages	:repeat <number of messages> :reference-number <N> :server-name <name>
request frequency	:request <frequency> :reference-number <N> :server-name <name>
accept frequency	:accept <frequency> :reference-number <N> :server-name <name>
deny frequency	:deny <frequency> :reference-number <N> :server-name <name>
request total number of transmissions	:request num-transmissions :reference-number <N> :server-name <name>
report total number of transmissions	:tell num-transmissions <n> :reference-number <N> :server-name <name>

TABLE II

COMMUNICATION ACTS TO SUPPORT CONTROL OF BACKCHANNELS

Communication Act	Description
request termination (<i>Handshake method</i>)	:request termination :reference-number <N> :server-name <name>
termination warning (<i>Warning method</i>)	:termination-warning <Time> :reference-number <N> :server-name <name>
report termination (<i>All methods</i>)	:connection terminated :reference-number <N> :server-name <name>

TABLE III

COMMUNICATION ACTS TO SUPPORT TERMINATION OF BACKCHANNELS

acts necessary to the flow control of backchannels is described in table II.

C. Termination of backchannels

As either the server or the client may be the initiator of the backchannel, so must the ability to end the connection over a backchannel fall equally to both parties. We provide three protocols for termination at various levels of social etiquette. The support required of the ACL for each method is described in table III. Consider again two agents, Agent 1 and Agent 2, where Agent 1 wishes to terminate the connection.

In the most polite way to sever the line of communication, the *handshake method*, Agent 1 sends a request to Agent 2 through the ACL to end the connection. Agent 2 closes the TCP connection when ready and then sends a response in ACL alerting Agent 1 that the connection has been closed.

The second protocol for ending communication across a backchannel is more similar to the agent shaking her head than shaking hands. In the *warning method*, Agent 1 sends warnings in ACL that the line will be shut down. Warnings are sent at 30 seconds and 5 seconds. During this time, Agent 2 has the opportunity to close the TCP connection on her side and then notify Agent 1 that the connection has been closed.

At the end of the time, if Agent 2 has not terminated the connection, Agent 1 terminates it and then sends a message of notification to Agent 2.

The final protocol, the *cold shoulder method*, is far less friendly than the other methods as data may be lost. Here, either agent simply closes the TCP connection without notice. A message in the ACL is then sent notifying the other agent that the connection has been closed.

Independent of which protocol is chosen, we insist that a message is always sent through the ACL when a connection is closed. With this restriction, termination protocols are consistent with the rule that all metacommunication be handled at the ACL level so that the state of a connection be apparent from the messages. An exception to this rule is the case when neither agent intentionally terminates the backchannel but the connection is severed due to a physical loss of communication between the agents. If this occurs, the agents must recognize that the connection was terminated from the TCP level, which sends a warning when a connection is terminated. Our protocol does not allow agents to reopen a closed backchannel for any reason, so a new backchannel would need to be established if the agents wish to reestablish backchannel communication.

III. NETWORK DRIVER DETAILS

Although backchannels, like agent communication languages, are independent of the transport protocol, their implementation naturally depends heavily on the transport protocol used. This section provides sample implementation details in order to demonstrate the interleaving of low level and ACL level communication. The specific implementation presented here is for the TCP low level transport protocol, the most widely used protocol in real robot systems. The analysis and experimental results also assume TCP, though for systems where efficiency is of paramount importance, UDP may be a better alternative.

The protocol for establishing a connection between a pair of agents varies slightly depending on whether the initiator wants to send or receive the information. To illustrate this distinction, consider Agents 1 (the initiator) and 2 as shown in figure 2. In the first case, Agent 1 wishes to send information to Agent 2. For example, Agent 1 may wish to control Agent 2 through teleoperation and needs to send many commands. In this case, Agent 1 acts as the server while Agent 2 is the client. In the second case, Agent 1 wishes to receive information from Agent 2. As an example of this relationship, Agent 1 may want streaming video at a certain resolution from Agent 2. Now Agent 1 is the client and Agent 2 is the server.

Since the first step in establishing a TCP connection is for the server to open a passive line on a port, the server must acquiesce to the agreement before any progress can be made. The case in which the initiator is the server is consequently the simplest (figure 2(a)). Agent 1, the server, opens a port for connections and then sends a message to Agent 2 through the ACL requesting permission to set up a line to send information (from Agent 2's perspective a client line). Agent 1 also provides the necessary information for Agent 2

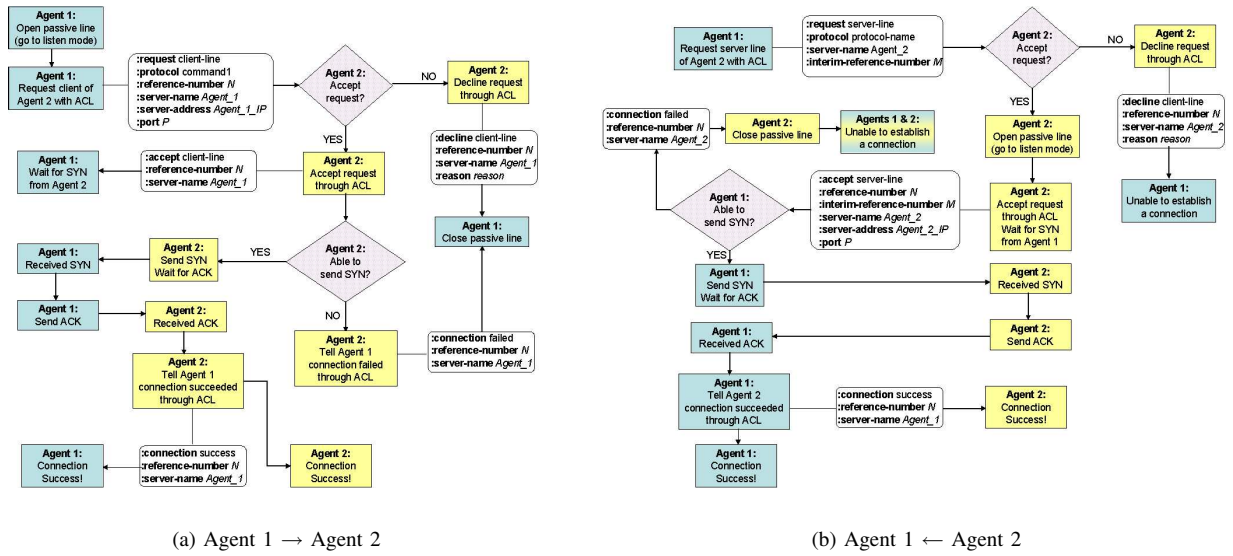


Fig. 2. Flowchart of interleaved high and low level communication for establishment of a backchannel in the case where Agent 1 wants Agent 2 to be the client (a) and where Agent 1 wants Agent 2 to be the server (b). Messages sent through the ACL are in boxes with rounded edges.

to connect to the available port as well as a reference number. The reference number is a unique number on Agent 1's side referring to this particular channel for communication. Finally, Agent 1 specifies what information will be sent over the line with the format descriptor, which is a reference into a library of format descriptions. Agent 2 either accepts the request and opens a client connection over TCP, or declines, optionally providing a reason. Once the connection has been established, Agent 2 notifies Agent 1. The backchannel is now ready to be used, controlled through communication between the agents in the ACL.

The protocol for the case in which the initiator wishes to receive information and act as the client differs only in the logistics. For a complete description of the interactions of the agents when the initiator is the client, refer to figure 2(b).

IV. ANALYSIS

There are several reasons to believe that backchannels will improve system performance, but it is first necessary to define our metrics. Two of the most important elements of network performance are bandwidth and latency. Following the lead of the networking community [11], this analysis uses frequency, directly related to throughput, and latency as metrics for evaluating the performance of a system. This paper equates the bandwidth and throughput of the system with the frequency of message exchanges though there are some differences. Bandwidth is the transmission capacity of the network, usually measured in bits per second. Throughput is the measurement of real world data across the network and can never be more than bandwidth, and is frequently less due to network traffic. We use frequency to describe the number of messages successfully transmitted in a given time, rather than the number of bytes transmitted as throughput. Frequency is important from the multi-agent system perspective because it is

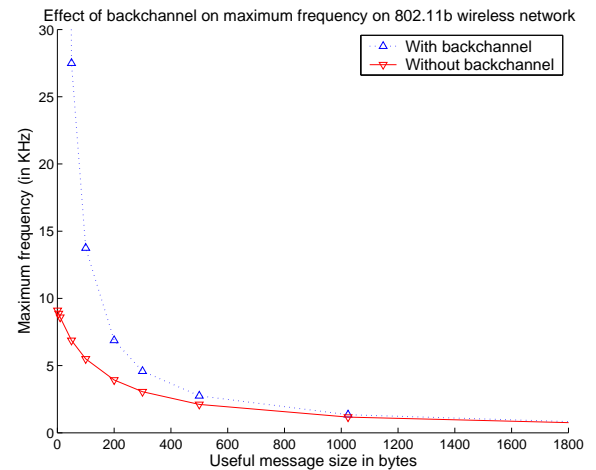


Fig. 3. A theoretical analysis of the maximum frequency at which messages can be sent over an 11 Mbps network assuming the same message content with or without backchannels and ignoring TCP effects.

the number of messages sent between agents which determines the maximum number of agents and the maximum rate at which information can be exchanged in a network.

Network latency is the amount of time it takes for a packet to travel from the source to the destination. We use latency to describe the amount of time it takes for a message to be sent from the source and be processed at the destination. Latency is of interest in systems with real-time constraints. Although bandwidth and latency are partially properties of a given network (i.e. a network that uses phone lines has lower bandwidth and higher latency than one that uses Ethernet lines), they are also dependent on the size of the messages transmitted over the network. A theoretical analysis the effect of message size on both throughput and latency illuminates

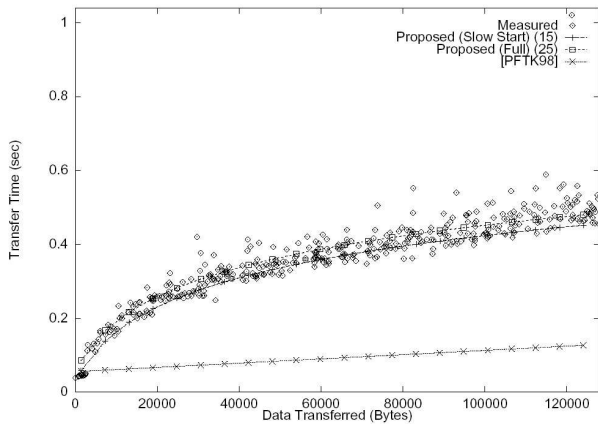


Fig. 4. Modeled and measured TCP latencies for 403 transfers from the University of Washington to the UC-Davis, used with permission [10].

the benefits of the more efficient system we propose.

For this analysis, consider a multi-agent system composed of identical agents, each sending the same messages. The bandwidth required for such a system would depend on the number of agents, the frequency at which each agent is sending its messages, and the size of each message.

$$\text{Bandwidth consumed} = \text{Number of Agents} \times \text{Frequency} \times \text{Message Size}$$

When using the backchannel, the overhead of sending messages can be significantly reduced. Consider the case of teleoperating a robot with a joystick. An interface agent wishes to send the left and right wheel velocities to the robot agent. Using the system implemented for a urban search and rescue multi-agent system [12], which uses KQML, the message has the following format:

```
(tell
  :sender Agent2
  :receiver Agent1
  :language default-language
  :ontology default-ontology
  :reply-with nob
  :in-reply-to nob
  :forward-to nob
  :content
    (:leftWheelVelocity 145
     :rightWheelVelocity 152))
```

Alternatively, if the message were passed over the backchannel with a format description stating that each message was two characters long and the first and second characters were to be interpreted as the left and right wheel velocities respectively, each message would only be two bytes long, as opposed to the 200 bytes required to send the same information over KQML. On the other hand, setting up the backchannel and controlling the backchannel through the ACL incurs some overhead.

The efficiency of the format description is very important.

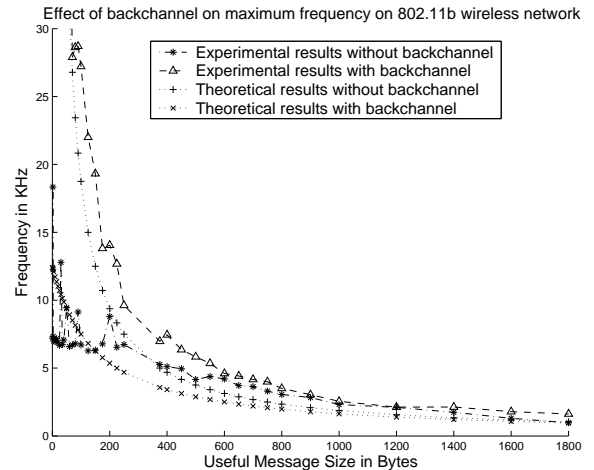


Fig. 5. Measured and theoretical effect of backchannels on maximum frequency

To remove this dependency, we assume that the same message :content is sent over the backchannel as through the ACL line and all overhead is due purely to non-content overhead. We also ignore the fact that large messages are most likely either media files or compressed, which means that they would need to be encoded and thus enlarged in order to be transmitted as ASCII text, incurring additional overhead (e.g. 20% using UU-encoding). Finally, we ignore TCP level overhead in this analysis.

Latency is studied by networks researchers because it affects the quality of service of a network [10]. In a multi-agent system, it is particularly important in real-time domains where small changes in latency may mean the difference between success and failure. The work of Cardwell *et al.*, reproduced with permission in figure 4, shows analytically and empirically that TCP latency increases with message size. For example, doubling the message size from 20000 to 40000 bytes results in a nearly 75% increase in latency. Furthermore, the results indicate an increased sensitivity of latency to message size for smaller messages. This data suggests that more efficient protocols may significantly decrease latency in a system. As a caveat, however, the latency of a system depends on the protocols and algorithms used. The Nagle algorithm [13] or kernel buffers can cause small messages to be accumulated and packaged together, increasing the latency.

V. EXPERIMENTAL RESULTS

We augmented the RETSINA multi-agent system, [7], to support backchannels as discussed in Section 2. We then conducted tests to determine the maximum frequency at which messages could be interchanged between a pair of agents on two different networks. The results (shown in figure 5) confirm our claim that the use of backchannels can lead to significant savings for small messages. The maximum frequencies observed on a 11 Mbps network match the predicted results of figure 3 very well.

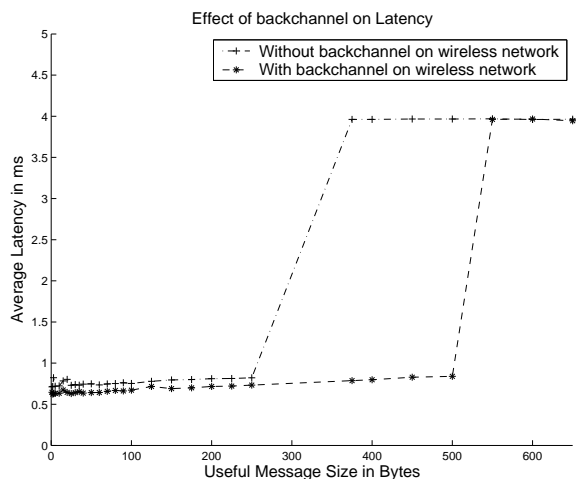


Fig. 6. Measured effect of backchannels on latency

Tests to determine how latency is affected by the addition of a backchannel also match theoretical results. The tests described do not measure only TCP latency because we timed the round trip which includes time to parse the message.

The total savings in latency is an average of 20% on the wireless network with our implementation, but can be as much as 80% savings for messages in the 375 to 500 byte range, a significant benefit for real-time systems.

VI. APPLICATION TO SEARCH AND RESCUE ROBOTICS

Urban Search and Rescue (USAR) involves coordination between people, agents, and robots to explore a space with many environmental challenges, including limited and sporadic communication. We have built a heterogeneous set of wheeled differential drive robots, and several user interfaces to allow robots to explore a mock up disaster site. In order to enable robots, agents, and people to work together, we are using the RETSINA multi-agent system architecture. Based on empirical testing of our system, we found that the overhead of KQML messages was very significant. Additionally, sending real-time video to the human operators could only be done at enormous cost through RETSINA and so was controlled outside the architecture. After integrating backchannels into the RETSINA architecture, we found that the stability and reliability of the robot system improved significantly as latency was decreased and throughput was increased.

VII. RELATED WORK

Discussion of the balance between efficiency and readability is common in the networking community where HTML is mixed with other formats using MIME headers to allow media types other than simple ASCII text to be encoded into a message. Much work has also been done to improve the speed of transferring images and multimedia. This work [10], [11], [14], [13] has contributed to the analysis on throughput and is an excellent resource for more information on the effect of message size on latency. To our knowledge, this is the first attempt to integrate backchannels into a MAS in a formal way.

The MAS community has largely ignored the importance of throughput efficiency. Work has been done to develop real-time multi-agent systems, [15], [16], but this work has focused on higher level algorithms, ignoring the low level protocol and the potential benefits of improving throughput.

VIII. CONCLUSION

We have presented arguments that a single line and language for agent communication is inadequate for systems that require the transfer of multimedia files and low level data at high frequencies. We describe a two-tiered communication architecture using backchannels and the steps necessary to integrate backchannels into existing MAS architectures in a principled way. We hope this flexibility will open up multi-agent systems for use with physical agent systems, helping to bridge the multi-agent multi-robot system research gap. Theoretical and experimental tests show improved communication efficiency and the addition of backchannels to the MAS RETSINA enabled the use of RETSINA for an urban search and rescue multi-robot system.

REFERENCES

- [1] T. Finin, R. Fritzon, and R. McEntire, "KQML as an agent communication language," in *Proceedings of the 3rd International Conference on Information and Knowledge Management*, November 1994.
- [2] (1997) Foundation for intelligent physical agents. [Online]. Available: <http://www.fipa.org>
- [3] K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng, "Distributed intelligent agents," *IEEE Expert*, December 1996.
- [4] H. Chalupsky *et al.*, "Electric elves: Agent technology for supporting human organizations," *AI Magazine*, Summer 2002.
- [5] K. Decker, K. Sycara, and D. Zeng, "Designing a multi-agent portfolio management system," in *Proceedings of the AAI Workshop on Internet Information Systems*, 1996.
- [6] O. Shehory, G. Sukthankar, and K. Sycara, "Agent aided aircraft maintenance," in *Proceedings of Autonomous Agents '99*, May 1999, pp. 306–312.
- [7] K. Sycara, M. Paolucci, M. van Velsen, and J. Giampapa, "The RETSINA MAS infrastructure," *special joint issue of Autonomous Agents and MAS*, vol. 7, no. 1 and 2, July 2003.
- [8] C. Li, J. Giampapa, and K. Sycara, "A review of research literature on bilateral negotiations," Carnegie Mellon University, Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-03-41, Nov. 2003.
- [9] G. Zlotkin and J. Rosenschein, "Negotiation and task sharing among autonomous agents in cooperative domains," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989, pp. 912–917.
- [10] N. Cardwell, S. Savage, and T. Anderson, "Modeling tcp latency," in *Proceedings of IEEE INFOCOM*, March 2000.
- [11] M. Harchol-Balter. (2002, November) Quality of service lectures. [Online]. Available: <http://www-2.cs.cmu.edu/~srini/15-441/F02/>
- [12] J. Wang, M. Lewis, and J. Gennari, "USAR: A game based simulation for teleoperation," in *Proceedings of the 47th Annual Meeting of the Human Factors and Ergonomics Society*, Denver, CO, Oct. 13–17, 2003.
- [13] J. Nagle, "Congestion control in IP/TCP internetworks," Network Information Center, SRI International, Menlo Park, CA, RFC 896, January 1984.
- [14] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley, July 2000, ch. 3.7 TCP Congestion Control.
- [15] M. Allouche, O. Boisser, and C. Sayetta, "Temporal social reasoning in dynamic multi-agent systems," in *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*. IEEE Computer Society, 2000, pp. 23–28.
- [16] V. Julian and V. Botti, "Developing real-time multi-agent systems," in *Fourth Iberoamerican Workshop on Multi-Agent Systems, Iberagents 2002*. IBERAMIA 2002, the VIII Iberoamerican Conference on Artificial Intelligence, November 2002.