Using IP multicast to improve communication in large scale mobile agent systems

Jörn Hartroth and Markus Hofmann
Institute for Telematics, University of Karlsruhe
[hartroth/hofmann]@telematik.informatik.uni-karlsruhe.de

Abstract

Two major challenges for the operation of mobile agent systems in a global networking environment are the tasks of providing an efficient infrastructure for remote inter-agent communication and of locating agents during their remote execution. The potential partners of a mobile agent form an overwhelmingly large population that is constantly fluctuating as individual agents move between different locations in pursuit of their tasks. Communication patterns range from simple two-party exchanges to long lasting coordination cycles among sizable agent groups. While adequate for the simpler cases, conventional messaging and directory mechanisms fail to scale up to the requirements of ensuring consistency and mutual location among many highly agile parties as demanded by more complex agent configurations. In this paper we present an approach to the remote coordination of dynamic agent groups based on the IP multicast model. By dynamically associating a multicast address with an agent group we gain a cost-efficient means of communication that can be easily adapted to dynamic changes in location and membership.

The mobile agents paradigm of distributed processing relies on small software units, named mobile agents, to traverse global data networks and meet with information providers or other agents at certain locations, often called agent meeting points (AMPs). A mobile agent encapsulates a set of application specific code and state variables that enable it to carry out an assigned task (e.g. preselection of data from a database) autonomously through local communication with the other entities present at a remote agent meeting point. Communication over data networks is thus reduced mainly to the travels of mobile agents between different agent meeting points. Mobile agents are attractive whenever it is desirable to minimize transfer volume and message exchanges for a data-intensive distributed application [2].

While the essence of the mobile agents paradigm consists in avoiding costly network transfers it is often desirable to be able to exchange small control messages with an agent to direct its operation, and agents cooperating remotely need to exchange status reports with their peers. One of the major challenges for mobile agent computing is the task of locating and communicating with agents in the field. The conditions found in popular visions of mobile agent systems (e.g. [19]) assume a numerous population of highly agile entities roaming all over the world and are thus especially ill suited for location strategies employed in present day distributed middleware (see [15]). The problem deteriorates further when communication is not restricted to two partners but includes a large community of agents that wish to exchange data with several partners at the same time. Typical applications of this scenario are found in the field of agent coordination scenarios in DAI and parallel approaches to highly data intensive tasks such as agent based electronic library access, data mining and the electronic marketplace.

In this paper we propose a solution to the location and communication problem in mobile agent systems through a very direct use of the IP multicast technology. We capture the inherent agility of mobile agents in a scheme of dynamically adapted multicast groups and thereby facilitate the design of an efficient location service under a restricted visibility assumption at the level of agent meeting points.

IP multicasting technology has demonstrated its efficiency for wide area multiparty communication and is thus a suitable basis for a communications infrastructure. However, the characteristics of mobile agents require some features, such as reliable delivery and high dynamics in topology changes, that are not met by IP multicast in its simple form. This added functionality has to be provided by extensions which must be carefully chosen to yield an efficient overall solution.

The argumentation of this paper is arranged as follows: In chapters 1 and 2 we give a short introduction into the principles of mobile agent computing and the IP multicasting technology, respectively. Chapter 3 details the practical application of IP multicast as a communications infrastructure for mobile agents. In chapter 4 we provide a close analysis of the special demands placed on

the transport system by mobile agents and show how the simple multicast model must be extended to reach the desired efficiency. We illustrate our approach with a sample application from the electronic marketplace in chapter 5. We conclude in chapter 6 with a discussion of the effects of our approach on other components of mobile agent computing and give an outlook to further research directions.

1. Mobile agents

The mobile agents paradigm is a recent approach to the development of distributed applications. It has its origins in the concept of remote program evaluation [18] and builds on the areas of distributed object oriented systems (e.g. [10]), user interface design [4] and models of the artificial intelligence community [20]. From the first it inherits the attribute of mobility, from the second the attribute of asynchronicity, and from the last the attribute of autonomy. In the technical view a mobile agent comprises the encapsulation of an application's code, data and control flow into a single compound. With the assistance of a suitable runtime environment, an agent running at one computing node can suspend its execution, be transferred across a network link to a different computing node, and seamlessly continue its task in the new location at the point where it left off [2]. Containing its own flow of control means for the agent that it can perform internal computations autonomously and explicitly interface with external data sources. It is thereby possible for the agent to keep a tight control over its internal data structures. Since a mobile agent can carry a computation across several locations, it is capable of solving its task asynchronously of the entity that started the agent. These three attributes distinguish the mobile agent from the several other agent concepts used in the artificial intelligence, user interface, and systems management contexts (see [14], [16]).

The growing interest for mobile agents in the distributed systems community arises from the fact that agents appear well suited as a cost efficient means for distributed information gathering and processing tasks in environments with low available bandwidth or unreliable network connections such as mobile computing and wide area networking. Moving an application across a network to the location of relevant data – often referred to as code shipping – can result in considerable savings of transfer volume compared to the data shipping approach predominantly employed in today's network-related applications from networked database systems to the world wide web. A mobile agent can also be used to asynchronously monitor remote events and data without the need for repeated network accesses [2]. In order to achieve their primary objective, the reduction of network traffic for distributed computations, mobile agents are designed to perform the majority of their interactions with service providers and other agents inside their local AMP environment. There is often an additional message-based mechanism for low bandwidth communication with remote peers, whenever an agent cannot solve its task entirely autonomously, to avoid the overhead of transferring the agent for simple status reports.

While there are many other questions still to be answered for a comprehensive cost model of mobile agent computing, we shall focus on *data volume transferred over network links*, which is related to both transfer delay and consumed bandwidth, as the basis for our further argumentation and show how it can be improved by the use of a multicasting communication infrastructure.

2. Fundamentals of IP multicast

Existing mobile agent systems generally like [12] or [5] forward messages to multiple receivers by repeatedly transmitting them using successive point-to-point transfers. The data source transmits various copies of one and the same data unit, each of them addressed to a different receiver. This approach has two major drawbacks. Firstly, the data source needs to know the identity and location of each receiver. However, mobile agents may change their location at any time, thus generating the need for an explicit location mechanism. Secondly, the transmission of one copy per receiver does not scale well with the overall number of recipients. Network load is increased with the group size. Broadcasting, on the other hand, might be a good solution in local environments. However, it is obviously not practicable in wide-area networks.

Rather than broadcasting information or using multiple point-to-point transfers, a much better approach is to make use of the forthcoming multicast technology. *Multicast* refers to a communication pattern whereby a single sender transmits data to multiple receivers. The receivers form a multicast group, which is assigned a certain multicast address. Forwarding messages to a multicast group is optimized by delaying the replication of a data packet until it has to traverse different links. Therefore, routers and switches have to incorporate group management facilities as well as mechanisms to establish and maintain multicast routing trees. Fortunately, the current Internet already provides support for multicast transfer. The *Multicast Backbone (MBone)* [11] forms an overlay network on top of today's Internet providing a multicast facility to the Internet community. It makes use of Steve Deering's IP multicast extensions [3]. The MBone comprises only a small fraction of currently installed Internet routers and uses so called *tunnels* to link multicast-capable islands together. These tunnels

presently need to be manually configured by system administrators. They are used to forward multicast packets through non-multicast routers by encapsulating them inside regular IP packets. The MBone has been established to gain practical experience with the new multicast technology. In the near future, it is expected that most of the routers will be multicast-capable and that the MBone will encompass the whole Internet.

According to the group delivery model of IP multicast, data sources simply send a single copy of multicast messages to the group's multicast address. Senders do not need to have any advance knowledge of the group membership. It is the task of IP multicast to forward messages to all the current group members. To receive any multicast data, receivers simply announce their interest in messages destined to a certain multicast group. They do not need to have any knowledge of group membership or active senders. In addition, hosts may join and leave multicast communication at any time without affecting the data transmission to any other member.

The *Internet Group Management Protocol (IGMP)* provides mechanisms for hosts to announce and to reannounce their membership in certain multicast groups. Local multicast routers listen to these membership reports to track the group membership in their subnets. Based on the obtained information, they decide whether to forward multicast messages addressed to specific groups. However, these mechanisms are transparent to the service user. Application programmers simply use an extended socket interface to send or receive multicast data. Group membership is announced by setting the appropriate socket options. Therefore, writing IP multicast applications is no more complicated than using traditional point-to-point communication via a socket interface.

3. Applying IP multicast to mobile agent systems

Communication in mobile agent computing occurs in two shapes: Firstly, mobile agents themselves are transported between different hosting computers in a process commonly known as agent migration, and secondly agents may enter into communication with remote entities in addition to their interactions with resources local to their present place. Mechanisms to support both types of communication have been incorporated in all present agent architectures (e.g. [1], [12], [9], [19]). So far, emphasis has been put on the technical realization of code shipping and the establishment of security mechanisms for agent transfer and communication. The transfer of the involved raw data is handled by conventional transport connections between the different agent runtime environments. While this approach is sufficient for prototype scenarios dealing with only a small number of agents it quickly becomes a problem when faced with real world conditions.

Application of mobile agents in the electronic marketplace (an example is discussed in detail in chapter 5) or distributed simulations [13] can easily amount to a community of thousands of agents distributed over continental distances. New agents may be introduced into the system at any time through the process of spawning or cloning subagents which allows the creation of distributed agent teams. In this kind of environment both modes of communication, the migration of agents themselves through multiple-degree cloning and the sending of messages between teams of agents, involves several parties at once instead of just two partners. Studies in the coordination of multiple agents carried out in distributed artificial intelligence research like [17] assume broadcast-type communication facilities in the shape of a shared blackboard. At the same time the large-scale employment of mobile agents in practically relevant applications will bring about a closer control of the resources consumed by mobile agents in their remote host environments. Here, an agent creator will want to keep track of the costs accumulated by his wayward agents and occasionally stop an agent whose task has become obsolete. One of the keystones for the practical suitability of mobile agents for real-world applications is therefore an efficient service for locating and communicating with mobile agents in a huge and dynamically varying population.

In present approaches different agent runtime environments are typically connected by unicasting links and provide a service component at the application level to keep track of locations of the active mobile agents. Inherently parallel communication patterns like cloning have to be serialized and multiplexed onto the links. The preservation of consistent location information after an agent migration or cloning requires an application-level protocol between AMPs and results in several message exchanges per movement or location request. By analogy to the multiparty communication problem on the network layer it is obvious that this approach does not scale well with the large of the envisioned scenarios of mobile agent applications. Nor are the well known application-level location mechanisms for mobile objects such as forward pointers, home nodes or broadcast particularly suited to the dynamism of mobile agents environments.

3.1. Technical realization

We observe that the principle of autonomous task solution mentioned as a key classifying attribute of mobile agents has the effect that a potential for remote communication and an interest in the present location of any particular agent does not extend to the entire cosmos of a worldwide mobile agent community. Contrarily,

knowledge of the existence and location of a specific mobile agent and the need to exchange coordination messages remains in the scope of a group of mutually acquainted entities, e.g. all agents working on the same top-level task or all agents belonging to the same user. A surprising solution to both the problem of multiparty communication and of locating dynamically moving entities under this *restricted visibility* assumption is found in the application of the very same mechanism chosen for multiparty streaming applications below the application protocol layers: the IP multicasting technology.

For our further elaboration we define the concept of a mobile agent *communication context*. This term is to stand for a group of agents and the messages exchanged between them in pursuit of a common well defined task as in the anonymous group communication variant in [1]. A communication context may involve a number of agents varying over time, extend in duration from a single message exchange to a multi-message and long-lasting conversation, and the type of messages may range over several magnitudes from small status reports of only a few bytes to encapsulated migrating agents and large amounts of accumulated result data. For each communication context contained in an agent-based computation we assign an IP multicast group if the number of agents that are to receive messages under this context is greater than a certain threshold. All network transfers in a communication context are sent to the assigned multicast group and equally delivered to all context members.

We will denote the underlying multicast group of a communication context as the *context carrier* because it closely reflects the topological state of a communication context and its dynamic evolution. For any communication subcontext, an additional multicast group will be created either through initialization by the original context creator or dynamically at runtime by one of the group members, whereafter it can be propagated over the global context multicast group.

A context carrier can be used to improve the communication for arbitrary groups of agents. When used for the temporary collaboration of otherwise unrelated agents the group needs a previous agreement on the ID of the underlying multicast group. An especially efficient case arises for *teams* of multiply cloned agents that are simultaneously created by a single parent.

The setup of a context carrier requires that an initiator agent acquires an unused address for an IP multicast group. Currently, no mechanism has been defined for IP multicast to dynamically assign group addresses. Instead, today's MBone Tools use a program called *Session Directory (SDR)* [11] to ensure uniqueness of randomly chosen multicast address at an application level. We have implemented a client/server based variant of this tool, which allows programs to get an unused group address via

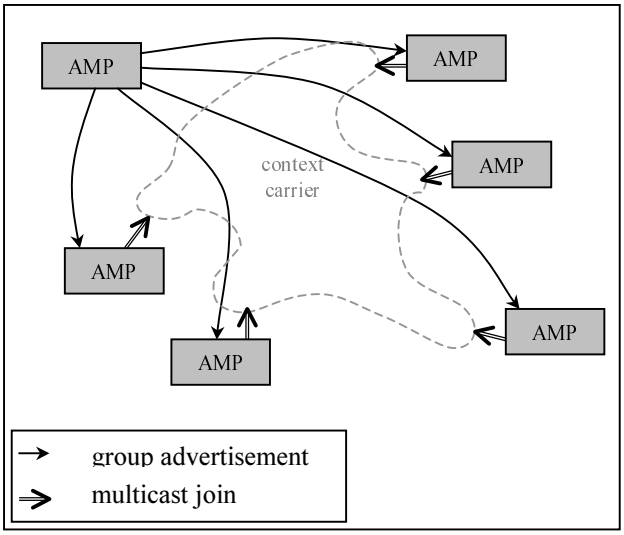

Figure 1: Establishment of the context carrier

Remote Procedure Call (RPC) and ensures uniqueness of the used address. After getting an group address, the initiator agent sends a (very small) setup packet by conventional unicast to each AMP that carries an agent group member (or is to be the target of a clone agent) to spread knowledge of the chosen multicast address. For the clone team case this message also contains information specific to the individual clones such as agent ID and initialization state. The target sites then join the advertised multicast group and wait for incoming messages (see figure 1) thus completing the context carrier setup. At this point, a clone parent, which itself has not joined the multicast group, may send a single copy of packaged agent code as the first multicast message, and the transport system efficiently handles the delivery to all of the registered target sites (see figure 2 below).

The established multicast group can henceforth be used to relay any messages of global scope among the communication context. For subsequent migration of

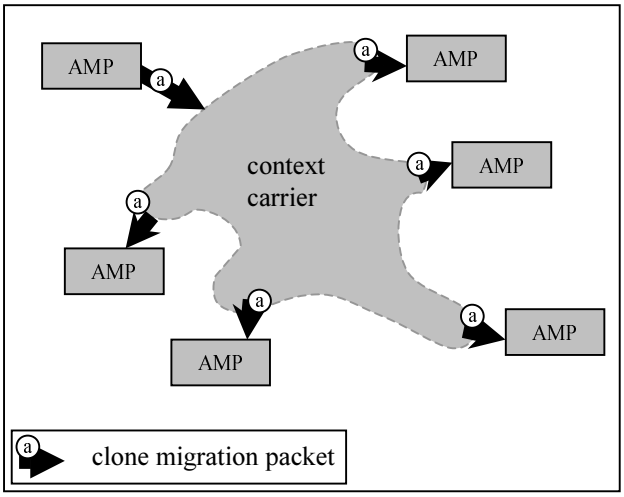

Figure 2: Clone team setup via context carrier

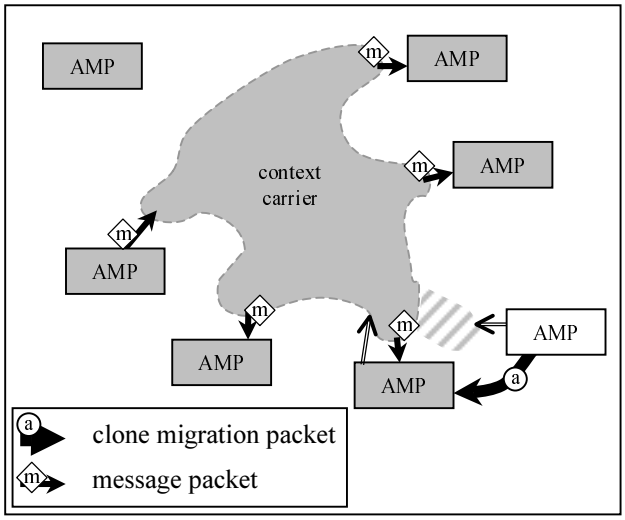

Figure 3: Transparent location of moved agents

context members the multicast setup is adapted by deregistering the old site and adding the new site to the multicast group (shown below in figure 3). New agents may be dynamically spawned and added to the group and group members can resign at any time, causing the multicast group to be updated with join and leave events.

A special case arises whenever two members of a communication context reside on the same AMP. While a doubled multicast *join* is simply ignored by the networking infrastructure, a premature *leave* operation issued by the AMP on departure of the first agent leaves the remaining agent disconnected from the group. The AMP is thus required to keep a counter per context and will only issue a multicast leave operation when all agents of a context have departed.

A reliable delivery mechanism is required to ensure that the migrating agents reach any target sites that received the setup notification but were late to join the multicast group, entering only after the migration message had been delivered to all receivers.

It is important to note that at the application level changes in the setup of the context carrier through migration of participating agents involves only local interactions with the transport infrastructure. Specifically, only the origin and destination AMPs of a migration leave and join, respectively, the context carrier. Full locatability of the moved agent is provided in an entirely transparent way by the multicast routing infrastructure. This is unlike conventional locating schemes where remote home pointers or reference chains have to be updated to reflect the occurred migration.

3.2. Changes in the AMP interface

The implementation of all communication functionality in our architectural framework is located in an AMP service component named the *communication service*. Agents access this service through an interface that offers methods for migrating or cloning an agent and sending arbitrary messages to other agents among a set of acquaintances of the sending agent. The acquaintance set in our approach contains individual agents, such as the creator of an agent group, as well as agent groups, represented by their communication context. The mapping between agent or group IDs and their corresponding unicast or multicast addresses and the actual transfer of a message is encapsulated in the communication service as are join- and leave-operations for the AMPs at both ends of an agent migration. Thus, the individual agent does not need to know any details about the shape of an acquainted agent group and stays entirely unconcerned with the joining or leaving of members. Allowing set-valued acquaintances and managing the necessary bookkeeping for dynamic groups in the AMP infrastructure extends the (single-valued) options offered in existing systems and relieves the agent developer of the burden of organizing multi-party communication by himself. The clone operation used to spawn one or several copies of an agent comes in two flavors. By default, new agents are added to the enclosing communication context. Alternatively, it is possible to create a new communication context for the clone team enabling the user to form a hierarchy of subcontexts that limit multicast exchanges to precisely the group communication pattern desired for the application.

3.3. Advantages over conventional approaches

The advantage attainable over the conventional approach of application-level controlled communication via unicast transport links is manifested in two areas.

Firstly, the accumulated data volume transferred in a single multicast is almost always significantly lower than that of a matching number of unicast transfers. Teams of clone agents are to the largest degree made up of identical code and initialization data (with differences e.g. in the individual IDs and itineraries) and are therefore suitable for our multicast transport scheme. The volume generated in agent migrations, one of the major remaining cost factors for mobile agents on the communication side, can thus be reduced at least for the initial cloning phase.

Secondly, the task of locating the addressees of a communication event is transparently handled by the multicast protocol on the networking layer. We argue that the network layer is the appropriate choice to carry out routing and locating functionality because the required protocol mechanisms are already there and have undergone considerable development. The functions for agent location performed at the application level inside the AMPs are greatly reduced in our approach and lead to a simplification of the overall AMP architecture.

3.4. Required extensions to the simple IP multicast

While the advantages to be gained by the direct use of multicast technology are promising, the discussion so far has not considered all aspects of the technical realization. First and foremost, the simple IP multicast service does not provide guarantees for the reliable delivery of a packet to any of the receivers which is an absolute requirement for agent communication.

The literature mentions several extensions that add reliability to the multicast service. The communication patterns generated by a context of agents are, however, fundamentally different from those generally assumed as the basis for reliable multicast protocols in a number of ways: traffic is highly bursty instead of streaming, receivers may move during the existence of the context, data volumes are comparatively small. The ratio of the topology change rate over the communication event rate may be high, which means that an agent may choose to migrate between two packets of a conversation and expect to be reached by all packets at its new location. With respect to organization, the member AMPs of a context carrier do not join the group out of their own choice but react to the request of a peer AMP (the origin of an immigrating agent).

This set of constraints necessitates a careful analysis of the employed extensions for reliable delivery to ensure that the profit gained in the use of multicast is not lost through carelessness at the other end.

4. Analysis of multicast-based communication

IP multicast provides efficient and scaleable routing of data packets to multiple receivers. However, the unreliable bearer service provided by IP does not fit the requirements of mobile agent systems. Remote inter-agent communication relies on reliable and error-free data delivery. Measurements have shown that packet losses in the current MBone are significant. The data sets collected in [21] state that in several scenarios almost 70% of transmitted packets were not correctly received by at least one receiver. This illustrates the need for efficient and powerful error correction schemes.

When designing reliable multicast protocols for large scale mobile agent systems, scalability becomes a key issue. Several hundred agents may be involved in a single multicast communication. In addition, mobile agents may be spread all over the world. As the size and the geographic span of agent groups increase, efficient connection management schemes become ever more essential. Some kind of interaction between sender and receivers is required to ensure correct data delivery. Neither system has enough information to control data transfer on its own. The provision of reliable data transfer is based on a comparison between sent and received data. The transmitter has knowledge about which data units have been sent and the receivers about which data units have been received correctly. Therefore, the provision of a reliable communication service requires the transmission of receiver status back to the sender or vice versa.

Another issue of high importance is to identify error correction schemes suitable to provide reliable multicast delivery. Common protocols use Go-Back-N or selective repeat to retransmit lost and corrupted data. Receivers request missed data units directly from the transmitter without any consideration of network topology and current network load. In the case of group communication, it is also possible to exchange data with neighboring receivers. It is preferable to request lost and corrupted data from a group member placed next to the host which is missing some information. An optimal error correction scheme would stimulate retransmissions of missed data units by the receiver located closest to the failing host. This would minimize transfer delay and network load. Studies of packet loss correlation in the current MBone [21] show that packet loss is more likely to occur on the path between the multicast backbone and the local host rather than on the backbone links of the multicast tree. The measurements also show that, on average, there is little pair-wise spatially associated loss in the MBone. Therefore, the probability that a receiver is able to obtain a missed data unit from a nearby group member is quite high.

4.1. LGMP – A scalable reliable multicast protocol

The described problems and the characteristics of the MBone have strongly influenced the design and the development of a novel multicast protocol for mobile agent systems, which is called *Local Group based Multicast Protocol (LGMP)*. It is based on the Local Group Concept presented in [7]. The mechanisms of LGMP are designed to support reliable data transfer on top of unreliable IP multicast. The communication service provided by LGMP is perfectly in conformity with the requirements of mobile agent systems. A comparison with related work can be found in [6].

The basic aim of LGMP is to distribute the burden of acknowledgment handling and error recovery among all the members of a global multicast group. LGMP divides global communication groups into so-called *Local Groups* to improve scalability of point-to-multipoint services. The receivers which belong to the same Local Group should be located in close vicinity. Different application-specific metrics are supported to establish a hierarchy of Local

Groups. In each subgroup a *Group Controller (GC)* is responsible for processing status information from the assigned receivers. The integration of acknowledgment processing capabilities into Group Controllers reduces the implosion problem. A GC evaluates control messages from all the members of its subgroup and forwards them to the multicast sender or a higher-level Group Controller in a single composite control unit. This concerns error reports as well as messages to control data flow. Parallel processing of status reports and their combination into a single message per Local Group relieves the multicast sender by reducing the number of control units to be processed at the transmitter.

Group Controllers also support the coordination of local retransmissions. In principle, any group member that has correctly received a certain data unit is able to perform retransmissions. It is not just the original sender which may retransmit lost data units. LGC recovers packet errors inside Local Groups first. A Group Controller requests missing data units from the sender or a higher-level GC only if not a single member of its subgroup has received the missing data unit correctly. However, retransmissions performed by regular receivers have to be coordinated in some way. Redundant retransmissions should be avoided and an appropriate group member should perform the retransmission. The receivers have to come to an agreement on which of them is performing a requested retransmission. They coordinate local recovery from data loss to avoid expensive retransmissions from the multicast sender or a higher level Group Controller. This reduces delay and decreases the load for sender and network.

In principle, any regular receiver can become a Group Controller. However, each communication participant decides on its own whether it is willing to be a potential Group Controller or not. A receiver may refuse to perform local acknowledgment processing and local retransmissions due to its restricted processing power or memory space. Other reasons for the refusal could be the cost structure of the Internet service or security considerations.

The definition of Local Groups and their formation into a hierarchy is supported by the Dynamic Configuration Protocol (DCP) [7]. It provides mechanisms for an automated establishment of Local Groups and for dynamic reconfiguration in accordance with the current network load and group membership. No manual administration is necessary. DCP is self-healing, meaning that it is tolerant of the failure of certain receivers or a Group Controller.

An important feature of LGMP is implicit connection establishment. Short term, transaction oriented inter-agent communication are not delayed by long-lasting information exchange during connection establishment. Instead, connections are established implicitly by the first incoming data packet.

4.2. Performance issues

Appropriate use of IP multicast is a novel approach to solve the location problem in mobile agent systems. However, it needs to be clarified how fast IP multicast will reflect changes of agent location and whether the observed delay is acceptable for the envisaged application environment. In addition, the question arises whether the approach generates more multicast routing load than it prevents in data traffic load. In order to answer these questions, simulations as well as measurements in the MBone have been applied.

A mobile agent travelling from one agent meeting point (AMP_1) to another one (AMP_2) has to perform several tasks. It causes AMP_1 to leave the corresponding multicast group and stimulates AMP_2 to join it. Before being able to receive multicast packets at AMP_2, the updated group membership information needs to be distributed by the underlying multicast routing protocol and the routing tree must be changed to reflect the new location of the mobile agent. The delay ΔT incurred by these updates is defined as the interval between time T_1 and time T_2, whereby T_1 refers to the moment of issuing a join request and T_2 defines the earliest point of time at which multicast packets could be received. To determine realistic values for ΔT, several experiments in the MBone have been performed. Multicast capable skeleton AMPs have been placed on workstations at the University of Karlsruhe, at the University of Braunschweig and at the University of Hannover. One of them has been set up to home a sending agent, which transmitted test messages at the maximum possible rate to a certain multicast group *m*. In order to determine the delay ΔT, an agent located at another AMP stimulated a join request a time T_1. After reception of the first multicast packet at time T_2, the delay

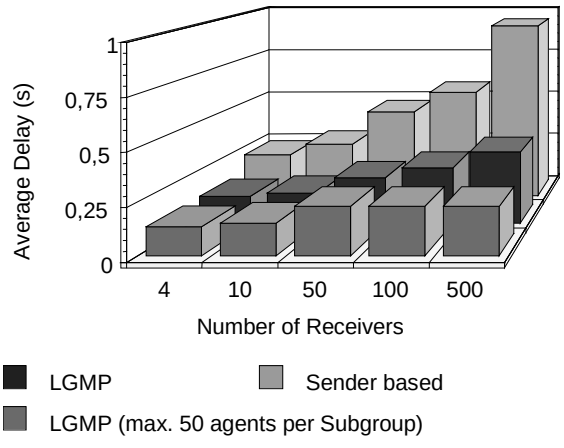

Figure 5: Delay of multicast messages

is calculated to $\Delta T = T_2 - T_1$. Various experiments have been performed to determine the join delay for different scenarios. The results show highly volatile values, which do not allow any definitive statement about the join delay to be expected for agent migration. However, values for ΔT ranged between 1 ms and 140 ms. This timescale seems to be quite acceptable, because a newly migrated agent is able to participate in inter-agent communication after this relative short time bound. No further location mechanism is necessary any more.

Besides the benefits in agent location, the use of scalable multicast techniques also results in reduced network load and decreased average transfer delay. Because benefits of multicast communication compared to successive unicast transfer are obvious, our performance evaluation focused on a comparison of LGMP and classical, sender-based multicast schemes.

All simulations were based on BONeS/Designer, an event-driven network simulation tool by the Alta Group of Cadence. The simulation scenario consists of a sending agent and various receiving agents which are connected to a certain subnet. The sending agent is linked to the subnet across a wide area network. The evaluation examined the impact of group size on average transfer delay and network load. The values obtained for LGMP were compared to corresponding results for two common sender-based techniques using multicast and unicast retransmission. Transfer delay was assumed to 20 ms for the wide area link, which is approximately the delay for transferring data between the East and West coast of the USA, and was fixed to 2 ms within the subnet. Error probability was assumed to be 10^{-1} for message loss through buffer overflow or bit errors, which is not uncommon for highly loaded internetworks. This value was derived from many measuring cycles to randomly selected internet hosts using the ping command. Other simulation parameters included data rate, processing delay within communication systems, status request rate, and burst length.

The impact of group size on average transfer delay is given in the above diagram. The graph shows for all techniques an increase in the average transfer delay with increasing numbers of receiving agents. The sharp increase for the common, sender-based approach can be explained with the large number of acknowledgments that have to be processed solely by the sending agent. The relative high values for average transfer delay are mainly caused by retransmissions across the wide area link. Local retransmissions, as performed by LGMP, effect a less strong increase in transfer delay. The benefit for LGMP is extremely high for large agent groups, but transfer delay still increases with the number of receivers. The establishment of several subgroups with a restricted number of agents leads to a distribution of acknowledgment processing to different local Group Controllers. In the third simulation, the maximum size of Local Groups was defined to 50 members. Therefore, every Group Controller just has to process a maximum of 50 acknowledgments. This results in an soft upper bound for transfer delay, because every Local Group with more than 50 receivers is separated into diverse subgroups with fewer members. This division distributes the burden of acknowledgment processing to different controllers and avoids further increase of transfer delay due to acknowledgment processing overhead.

To clarify the difference between unicast and multicast retransmission for both of the common techniques, 50 receivers were added at the sending side of the wide area network for evaluation of network load. The result is illustrated in the diagram shown below. For both common techniques, the ratio of retransmissions and total data traffic over the wide area link depends directly on the number of recipients. LGMP, in contrast, is not influenced by the number of receiving agents and shows very low values compared with the results of the other techniques.

5. An example application

We illustrate the application of our approach to mobile agent communication in an example from the electronic marketplace. The task which we would like to solve with the help of a group of mobile agents consists in finding the cheapest arrangement for a new personal computer we intend to buy. Being eager to keep abreast of the rapidly evolving technological advances we scoff at pre-installed turn-key solutions and wish to configure the system ourselves from individual components such as a mainboard, a CPU, a harddisk, video components, memory, a metal case, input devices, and a monitor. Our budget for the new machine is, unfortunately, strictly limited, therefore we are willing to expend some effort at finding the best deal. Prospective part dealers are situated

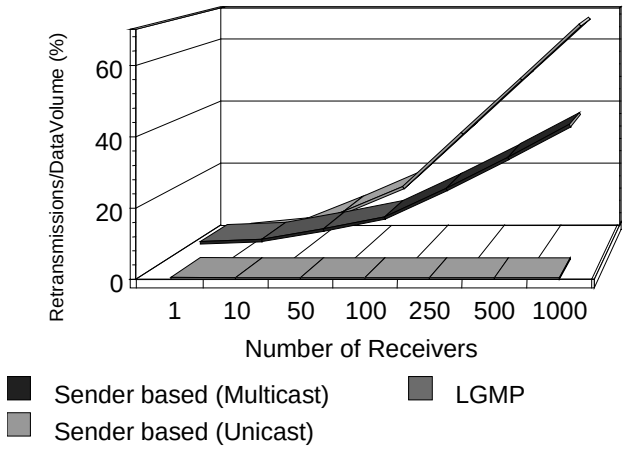

Figure 6: Generated network load

all over the country and since we are still reasonably patient there is enough time for shipment even from the farther locations. Comparing prices is, of course, a tedious job so we instantiate a team of mobile agents to carry out the search for components on the global network. Since we are rather sure of our preferences we can provide a set of simple rules for acceptable component types (board manufacturer, HD brand, etc.), prices we expect to pay for the respective variants, and the date we need them delivered to us. With these, we initialize our mobile agents and set them to their tasks.

While the setting unrolled so far is nothing untypical, it goes beyond the simpler examples given e.g. in [19] in that the overall task to be solved encompasses a number of closely related subtasks which makes the linear approach of sending a single mobile agent to all sites in sequence inappropriate. Pricing structures and differences in components in stock at different merchants turn the problem into a complex distributed coordination task that would require the single agent to accumulate all the available information in a first step and subsequently analyze the data pile to come up with the solution. A team of clones, however, can be sent to all known component dealers simultaneously, inquire about the present offers locally and exchange their findings with other team members. Use of several agents in parallel allows us to exploit one the second strength of mobile agents: remote monitoring of data. If the search for components is initiated some time before the delivery deadline, the team of agents can wait for price drops – at times a safe bet in some market segments such as harddisks.

The communication pattern for this application is dominated by multiparty exchanges. In the first step, a team of virtually identical agents migrates from the home site of the prospective PC owner to agent meeting points run by or located in the vicinity of component dealers. For the location of suitable component dealers we rely on a trading service [8] either in the shape of an external service provided by the networking infrastructure or as an integral component of the AMP, as has been suggested in several architectural concepts for mobile agent systems (e.g. [2], [19]). The initial migration of the agent group can be very effectively supported by setting up a multicast group for the transfer of the encapsulated agents. The very same multicast group will be used during the coordination phase between the agent team. An agent which finds an acceptable component in the offer of its assigned dealer spreads this information via the multicast group to all its peers. Depending on the market model underlying the agent system, the peer agents can then try to bargain with their dealers or simply stop looking for a specific component class when it is clear that they cannot surpass the advertised offer.

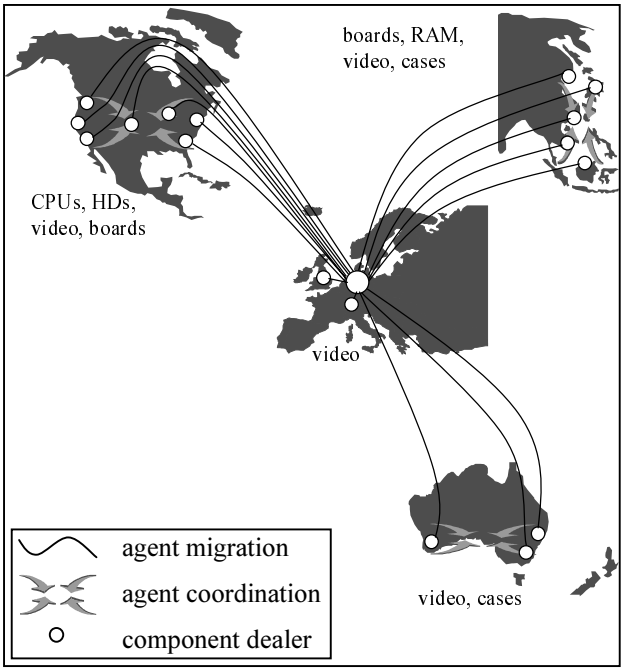

Figure 4: Using mobile clones to find PC components

To further limit the network traffic it is possible to separate the agent team into subteams concerned with specific components. It may be, for instance, advisable to coordinate the selection of the harddisk, the CD-ROM, and the disk controller because they are technologically dependent. The choice of monitor and graphics board, however, is largely independent of the mass storage aspect. After finding a solution for these subtasks, an additional negotiation may be necessary to meet the global pricing constraints or converge on a consistent delivery date. This hierarchical pattern of communication is equally well suited to our approach. In addition to the application-global multicast group a set of component-specific multicast groups is formed which carry the messages related to a specific component type.

The scenario presented in this chapter is easily extended to the even more practically relevant setting of a computer dealer who configures systems according to his customer's specifications and entertains business relations to several component wholesalers. In this case there is a fundamental business interest connected with finding cheap and timely deliverable components, and the repetitive character of the task makes it worthwhile to choose an efficient communication pattern for the agents.

6. Conclusion and outlook

We have presented a new approach to the problem of realizing an efficient and scalable mechanism for the location of and communication between large-scale

mobile agent populations. We have based our architecture on multicasting technology on the network protocol layer and have discussed necessary extensions to provide the quality-of-service required by mobile agents, namely reliable delivery and suitability for highly dynamic reconfiguration among the receiving agents. Initial measurements of migration delays and error-induced retransmissions for a protocol extension based on local groups appear promising for a practical application to mobile agent computing.

We intend to widen our focus to integrate other as yet unsolved aspects of mobile agent systems in our framework. In terms of agent security, for example, our approach to the underlying transport mechanism can be treated largely like any other transport infrastructure. Authentication and encryption schemes between agent meeting points can just as readily be implemented on top of a multicasting transport as on unicast systems. Preference should be given to mechanisms with a minimal number of exchanged messages (ideally one-way schemes) to avoid effects similar to the sender implosion mentioned in the fundamentals section. One additional concern is the possible monitoring of multicast group development by a hostile agency which might serve to gain knowledge on agent distribution patterns. Since the IP multicast does not provide any explicit membership information an attacker would have to break into the communication service of a group member AMP to find the association between an agent group and the underlying multicast group and would still learn no more about group membership than in the case of multiplexed unicast connections. We therefore conclude that the vulnerability of our scheme to security attacks is on par with conventional systems in all practically relevant aspects.

In the mid term we will incorporate our findings into a more complete mobile agent runtime architecture and carry out detailed studies on the performance of our multicasting scheme under a number of different communication patterns. Eventually we plan to design an integrated communication infrastructure that gives assistance in choosing an efficient group hierarchy suited to the specific communication demands of an application.

7. References

[1] Baumann J., Hohl F., Radouniklis N., Straßer M., Rothermel K.: Communication Concepts for Mobile Agent Systems, In: Proceedings of the First International Workshop on Mobile Agents, Berlin, Germany, April 7-8, 1997

[2] Chess D., Grosof B., Harrison C., Levine D., Parris C.: Itinerant Agents for Mobile Computing, IEEE Personal Communications, October 1995

[3] Deering S.: Host Extensions for IP Multicasting. RFC 1112, August 1989.

[4] Etzioni O., Levy H., Segal R., Thekkath C.: The Softbot Approach to OS Interfaces, IEEE Software, July 1995, pp. 42-51

[5] Gray R.: Agent Tcl: A transportable agent system, In Proc. of the CIKM Workshop on Intelligent Information Agents, 4th Int. Conf. on Information and Knowledge Management (CIKM 95), Baltimore, MD, December 1995

[6] Hofmann M., Scalable Multicast Communication in the Internet. ConneXions, Vol. 10, No. 10, October 1996

[7] Hofmann M.: Enabling Group Communication in Global Networks. Proceedings of Global Networking'97, Calgary, Alberta, June 1997

[8] ISO: ODP Trading Function, ISO/IEC JTC/SC 21 Draft Rec. X.950 / ISO/IEC DIS 13235; March, 1996

[9] Johansen D., van Renesse R., Schneider F.: An Introduction to the TACOMA Distributed System Version 1.0, Technical Report 95-23, Department of Computer Science, University of Tromsø, Norway, June 1995

[10] Jul E., Levy H., Hutchinson N, Black A.: Fine-Grained Mobility in the Emerald System, ACM Transactions on Computer Systems Vol. 6 No. 1, 1988, pp. 109-133

[11] Kumar V.: MBone: Interactive Multimedia on the Internet. New Riders Publishing, Indianapolis, Indiana, USA, 1995

[12] Lange D., Chang D.: IBM Aglets Workbench – Programming Mobile Agents in Java, IBM White Paper, IBM Co., http://www.trl.ibm.co.jp/aglets/ whitepaper.htm

[13] Langton C., Minar N., Burkhart R.: The Swarm Simulation System: A tool for studying complex systems, (draft available at http://www.santafe.edu/projects/swarm/ swarmdoc/swarmdoc.html), Santa Fe Institute, 1995

[14] Mendes M., Loyolla W., Magedanz T., Assis Silva, F.M., Krause S.: Agents skills and their roles in mobile computing and personal communications, IFIP Workshop on Mobile Communications, Canberra, Australia, September 1996

[15] Mullender S.: Distributed Systems, chap. 5 & 12, Addison-Wesley, 1993

[16] Nwana H.: Software Agents: An Overview, Knowledge Engineering Review, Vol 11, No. 3, Cambridge University Press, September 1996

[17] Rosenschein J., Zlotkin G.: Rules of Encounter, MIT Press, Cambridge, MA, 1994

[18] Stamos J., Gifford D.: Remote evaluation, ACM Transactions on Programming Languages and Systems 12(4), 1996

[19] White J.E.: Mobile Agents, In: Bradshaw (Ed.): Software Agents, AAAI Press / MIT Press, Menlo Park, CA, 1996

[20] Wooldridge M., Jennings N.: Agent Theories, Architectures, and Languages, In: Wooldridge M., Jennings N. (eds.): Intelligent Agents, Springer-Verlag, January 1995

[21] Yajnik M., Kurose J., Towsley D.: Packet Loss Correlation in the MBone Multicast Network. UMASS CMPSCI Technical Report # 96-32, University of Massachusetts at Amherst, 1995