Towards a minimal hosting specification for open agent systems : the lessons of IP

Cefn Hoile
Intelligent Systems Lab
BTexact Technologies
Ipswich, UK
cefn.hoile@bt.com

Erwin Bonsma
Intelligent Systems Lab
BTexact Technologies
Ipswich, UK
erwin.bonsma@bt.com

ABSTRACT

In this paper we observe that the strategies adopted by many standards projects may unnecessarily restrict the range of agent control, addressing and communication strategies which can be hosted, and hence limit the adoption of conformant hosting platforms. We argue that a minimal hosting platform is required to achieve convergence to common standards. Using the example of Internet Protocol - a minimal *communication* standard addressing issues common to all supervening protocols - we hope to stimulate discussion of a minimal *hosting* specification, which could address issues common to the hosting of all supervening agent strategies. We draw the reader's attention to issues of resource contention and other conflicts of interest which are expected to arise in open, heterogeneous, agent systems. An analogy is drawn between the resource contention strategies adopted to manage IP route conflicts and the those required to manage hosting resource conflicts.

We have attempted to address some of these issues in the core design of a lightweight platform for mobile process control and process intercommunication known as the DIET platform [6][3][9]. Some of the strategies adopted within the DIET core are presented for discussion.

1. INTRODUCTION

Networks of users and devices are an archetypal case of Brian Arthur's "increasing returns" [7]. In the last few years, many examples of networks have arisen in which the value of the system to each participant is a function of the degree of participation by others: fax machines, videotape standards, and the internet. More recently, examples have emerged in the peer to peer application space [11]. Such cases have led to conjectures detailing the relationship between adoption and network value, such as Metcalfe's law, "the total value of a network is equal to the square of the number of subscribers, while the value to a subscriber is equal to the number of subscribers". As Bell [1] points out "The law describes why it is essential that everyone have access to a single network instead of being subscribers on isolated networks."

Agent systems are no different. The value of an agent system to an developer or user consists in the resources and services available in the system as a whole. A significant proportion of these resources are contributions from other users and developers: content and data, channels for real-time or asynchronous interaction with the user themselves, and channels for interaction with other agents.

In the first section of this paper we argue that the drive for standardisation of agent toolkits within the research community may not achieve the appropriate kind of openness; By overspecifying for selected agent problem domains and building in expectations regarding the scale of agent systems, communication structures and protocols, alternative agent approaches could become excluded. The wide applicability of Internet Protocol as a scalable underlying networking standard for a huge variety of application-specific protocols is discussed. In particular, the minimal specification of IP, and the role of IP as a means of resolving contention over shared network resources is discussed.

In the second section, the architectural implications of this change in paradigm are noted; Adopting an open and heterogeneous approach - in which entirely different types of agent share common resources - raises issues of resource contention and other conflicts of interest which cannot be resolved through explicit negotiation. Similar strategies to those adopted in IP networking are considered for the management of resources in a shared agent hosting platform.

In the third section we briefly describe the specific countermeasures which we have adopted in the development of the DIET core - BTexact's minimal agent hosting platform - as a basis for further discussion.

2. A COMMON HOSTING SPECIFICATION

2.1 Agents as the coordinators of networked resources

There is a very good fit between the complex management tasks involved in exploiting networked resources and the strengths of agent systems. Computational resources and services can be shared within user groups for mutual benefit. Where agents take responsibility for coordinating resource access, the human user can be isolated from the complexity of negotiation. It is possible to envisage more and more complex interchanges taking place between many different resources and services in pursuit of users' objec-

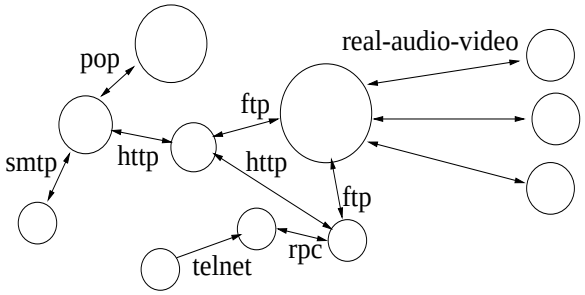

Figure 1: Proliferation of IP-based protocols through pairwise interactions

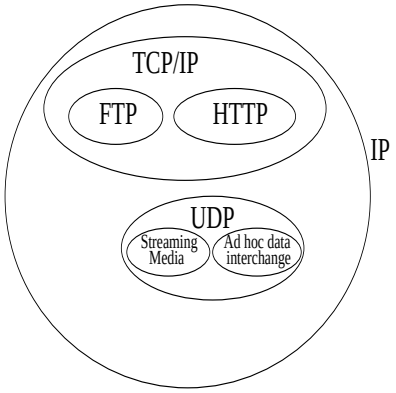

Figure 2: IP as minimal common denominator for supervening protocols

tives. Agent systems could be a key technology for the exploitation of the network. Agent interoperability, through the determination of shared standards is vital in this context, and represents a core objective for many high profile projects in agent research [4][12].

In the ideal scenario all resources are shared within a single system. However, where multiple incompatible standards exist, the global set of resources becomes subdivided into isolated subsets, and each competing system has fewer resources on offer than are available in the network at large. This has certainly taken place with the explosion in the variety of Peer to Peer file sharing systems [11], and could be said to have also taken place in the domain of agent systems, with a survey of the web in January 2001 revealing 71 distinct agent toolkits [8].

2.2 Ubiquitous standards

The success of the internet is built upon the huge range of interactions which can take place when *arbitrary* machines in the network can pass *arbitrary* data between each other. Application developers only need to determine the way in which machines *participating* in the application interact. Often, complex services are delivered through multiple machines which do not all share a common interaction protocol, such as the provision of web based e-mail, in which SMTP and POP interactions are coordinated through HTTP.

To compose such services, the basic requirement is that pairs of machines can exchange data with each other. Standards of interaction are important. However, supervening protocols can be designed on an ad hoc basis as they are needed by individual developers.

Where these supervening protocols are published, ratified, or simply dominate through wide adoption, other developers' programs are able to benefit from the new interactions available, simply by writing code conforming to the protocol. Over time, *de facto* standards may emerge, but this does not have to be achieved before interoperation can commence. It is through interoperation that these standards arise. Crucially, the developers of networked software can assume the existence of underlying layers which manage network traffic.

If a single agent platform is to achieve dominance as a *de facto* standard, it must be flexible enough to support a wide variety of agent implementation strategies, allowing various competing developers, companies, agent based toolkits, and standards organisations to converge on a single common hosting platform. In addition to this flexibility, it needs to be scalable in order to permit very large numbers of agents to interact across indefinitely many hosts.

This common hosting platform, offering minimal services, could be extended as necessary, for example to support specific agent control and coordination strategies, communication languages, naming and directory schemes. Convergence by all developers to a specific set of agent control strategies, agent communication conventions, interaction protocols and host services may follow. However, as in the case of internetworking, this is not a necessary precursor for large scale interoperability to arise. Perhaps no single set of standards can be asserted in these areas for all possible agent-based strategies. However, where a common hosting platform is available, supporting the minimal subset of all agents hosting requirements, pairwise interactions may be formed on an ad hoc basis between agents.

Since, in this scenario, the host's computational resources may be the only thing which all different forms of agents share, managing contention for these resources is a key aspect for a minimal hosting specification to address.

3. THE LESSONS OF IP

Internet Protocol, or IP, offers a good example of the the way in which interoperability can be catalysed by a minimal set of standards. Application-specific protocols for the internet are predicated upon the ability for machines to address each other and pass data. Many interaction protocols can share this underlying transport system, yet have message based interactions adapted for a specific problem domain.

Importantly, IP makes no commitments to the specific interactions which take place in supervening protocols. It is rather focused on solving the problem of coordinating shared resources - the common network infrastructure - in a scalable way. As such it remains the common denominator for the vast majority of machine interactions.

3.1 Lesson 1 : Less is more

The ubiquity of IP, despite its minimal specification, suggests that detailed examination should be made of the minimal common foundation required for the hosting of interacting agents of arbitrary kinds.

Although all hosted agents may not share a common set of interaction protocols, sharing an execution platform and addressing layer enables pairwise interactions between agents on an ad hoc basis, and the formation of clusters of cooperative units, as found in the interactions between machines on

the internet. In the case of mobile agents, co-locating interacting agents can also lead to reductions in communication latency and increases in efficiency.

In contrast to many existing models of standardisation [4][12][2] which focus to a great extent on common semantics, we propose a minimal common base set of *hosting* functions allowing agents of multiple different kinds, including mobile agents, to occupy and interact in shared environments. By contrast with Wooldridge and Jennings [13] we believe it is possible to construct a generic agent platform. They are correct to observe that initial platform development efforts are often focused on problem-specific design and advise "Only attempt to apply the architecture to problems with similar characteristics". However we hope to establish minimal agent hosting requirements, avoiding commitment to problem-specific architectural features or semantic schemes, and thus address problems which are characteristic of all agent systems.

3.2 Lesson 2 : Address resource contention without explicit coordination

One common issue which must be addressed in a scalable, open agent system is similar to the issue addressed by IP - the problem of resource contention.

In a IP packet routed network, it is hard to imagine how all the individuals using a specific route could coordinate their usage by explicit negotiation. A single route could be used to connect a very large number of possible pairs of machines, making the communication overhead of explicit coordination very large. Perversely, this is a 'Catch 22' [5] situation : for coordination, stable communication routes are required. The solution adopted in IP networks is a fail fast regime, in which finite buffers are associated with each contended resource. Demands made beyond this point are simply rejected. Feedback notifying failure may be returned to each node in an IP route. Compliant nodes respond to this by reducing their traffic, and potentially selecting a different route to deliver packets to their destination.

In agent systems, similar issues must be faced. Contended resources in agent host environments include: memory, network bandwidth, threads, read/write locks on shared data structures, and the attention of other hosted agents. The problem is that the initiator of a request cannot negotiate to minimise contention on all of the resources which are dedicated to the completion of that request. This too is a 'Catch 22' : explicit negotiation requires host resources.

The problems of the management of host resources described above shows strong similarities to those solved by IP. However, despite the analogy with IP and network communication it should be stressed that resource contention is not restricted to mobile agent systems or agent systems incorporating networked communication.

As the agent pattern is used more widely, we may expect more aspects of of our computing environment to be delivered in this way. A large number of agents could be hosted at one time, with the majority in an inactive state. In this situation, the allocation of threads to all hosted agents simultaneously may lead to host overload. A shared service provider targeted by more messages than it can handle may lead to large delays in the completion of its clients' tasks. With demand originating from a large number of hosted agents, (or potentially from anywhere in a very large network), peaks of activity must be handled gracefully at all levels. The dynamics of an agent system may even encourage peaks, as the demands made by one agent may trigger the activation of several others. Without appropriate constraints, free access to host resources could lead to unintended crashes or denial of service through unexpected interactions between co-hosted agent communities.

Examples of this form of resource overload in agent systems are rare at present, owing to the small populations of agents which share resources in experimental scenarios, to the scaling of available resources to the demands of the experiment which is being executed, or to the close coupling of all individuals in a single community, allowing implicit coordination. However, in open agent systems without these constraints, the lessons of IP suggest that finite limits and fail fast behaviour with feedback may be needed to control the behaviour of a large, diverse population of agents.

3.3 Lesson 3 : The ends are the means

Internet Protocol is often encountered as a component of TCP/IP. TCP or Transmission Control Protocol enables applications to use the IP network without being aware of the fact that IP packets arrive at variable rates, through various routes, and can arrive out of order or fail to arrive at all. TCP/IP hides the complexity of managing asynchronous packet deliveries and packet loss from the software which uses it. Conformant implementations adopt backoff strategies, responding to packet loss by reducing the number of packets sent. Consequently, the reliability of connections is guaranteed not by the behaviour of the shared resource - the network - but through a higher level behaviour which adapts to the strict limitations on that resource.

TCP/IP is not the only strategy to manage failure in the network. The appropriate strategy depends upon the application. In some cases, the resending of lost packets is unnecessary, for example where delays arising from network latency, or redundancy in the encoding of informations into packets, make packet redelivery inappropriate. This suggests that it is more appropriate for demand on host resources to be managed *end to end* , by the participants in transactions, rather than the host platform itself.

It is part of the agent paradigm that subtasks are delegated to autonomously executing individuals. Commonly, the initiator only receives feedback from the agent handling the request. The initiator of a request may not even share common communication protocols with the agents which deliver its intermediate objectives. The existence of resource bottlenecks must responded to by the initiator at each point in the chain of agent interactions which composes a service. Responses to failure could include; resending messages, rescheduling routine tasks to a time when activity is lower, relocation to less loaded environments or the identification of more responsive service providers.

4. DIET CONTENTION STRATEGIES

Example scenarios illustrating the features of an indefinitely scalable peer to peer system for process intercommunication - the core of the DIET, (Decentralised Information Ecosystem Technologies[6][3][9]), platform - incorporating resource contention strategies analogous to those in IP, are presented in Table 2 as a starting point for discussion of a minimal hosting specification. More implementation details are available in [6].

The strategy adopted in DIET could be compared in some

Table 1: Internet Protocol and Hosting Platforms compared

	IP (Internet Protocol)	*HP (Hosting Platform)*)
Contended	route bandwidth	hosting resources
Participants	machines	processes
Application delivery	software behaviours	agent behaviours
Supervening protocols	TCP/IP HTTP FTP UDP	FIPA-ACL KQML KIF
Result	software control	agent coordination

Table 2: DIET Scenarios: Contended resources and strategies adopted

Contended Resource(s)	Scenario	Threat to Host	Contention Measure
Processor capacity, Stack memory	new agents are spawned or idle agents reactivated	environment failure	finite population and limited thread pool - feedback for event failure
Processor capacity, Stack memory	many agents are idle waiting on a thread	memory overload and environment failure	permit thread surrender and reallocation for event handling
Processor capacity, Stack memory, Sockets and Ports	agents overload environment through migration	loss of responsiveness, environment failure	finite population, socket pool and thread pool - no feedback for migration failure
Agent Service Providers, Heap Memory	too many transactions or transaction states are maintained in parallel	client delay, environment failure	number of open communication channels per agent finite - feedback for channel failure
Agent lookup tables	query operations require list synchronisation	delays of lookup and registration	restrict queries to lookup by direct match (clock cycles for hash lookup independent of population size). Do not support wildcard lookups or copy exhaustive list of hosted agents

respects to that of the OPAL project [10] which describes a micro-agent kernel from which more complex services can be composed. However, in addition, the DIET core is specifically designed to address key technical obstacles in the deployment of very large populations of agents across an internet-scale host network, focusing on scalability, efficiency, decentralisation and robustness.

The DIET core is highly scalable. The CPU load to deliver its core services is not systematically dependent on the number of processes hosted, or the number of hosts in the network. It is highly efficient, the memory footprint of a JVM running the DIET core occupies significantly less than 1Megabyte. DIET applications have scaled to more than 500,000 addressable processes on a modest desktop PC. It is fully decentralised, maintaining the separation of each host's addressing tables whilst connecting hosts via peer to peer links.

DIET has deliberately omitted several agent services or features in the core, owing to problem-specific semantics, or to scalability or resource contention concerns. These include; predetermined ACLs or parsers; yellow pages services; the forwarding of yellow and white pages requests to remote hosts; the forwarding or storing of messages for agents not currently hosted; the guaranteed startup of agents; the guaranteed availability of hosted agents; the guaranteed delivery of messages within or between hosts.

In many cases, we have implemented add-on services on top of the minimal DIET core which *do* commit to problem specific semantics or particular failure handling strategies but deliver these services in turn through the execution of resource-constrained agents. These 'parasitic' hosting services may be considered the equivalents of the higher level protocols which supervene on IP, each satisfying different functional criteria. In this way, a variety of services with different behaviours, performance and load characteristics can be made available through a common platform, leaving agent developers free to choose the most appropriate implementation, or to develop their own. However, all levels of service provision can benefit from the resource management and scalability inherent in the DIET core.

5. CONCLUSION

We have presented analogies between the functional requirements of Internet Protocol, and the functional requirements of a minimal agent hosting platform. We claim that the benefits to developers of a shared hosting platform are also clear from this analogy, permitting interactions between heterogeneous agents on an ad hoc basis.

A short list of the features found in DIET has been put forward for discussion. The DIET core has been designed from the outset not only to offer a scalable and lightweight platform for process migration, management and intercommunication, but also to address issues of resource contention inherent in heterogeneous multi-agent systems.

In common with IP, its implementation is characterised by finite limits, fail-fast behaviour and feedback. This feedback provides a basis for the initiator of a transaction to moderate the demand on host resources according to a diverse range of failure handling strategies, in a similar way to the diversity of IP based flow control approaches.

Building upon minimal foundations as a common standard could permit a broader variety of agent strategies to coexist in a common hosting environment, allowing both the functions of the hosting environment and the degree of interoperation between different agents to increase gracefully according to the needs of participating developers.

A common hosting environment could offer a basis for the emergence of *de facto* standards in inter-agent messaging, communication languages, directory listing, and other hosting services for specific problem domains. With minimal hosting, these standards can diversify yet interoperate within a common, scalable infrastructure.

6. REFERENCES

[1] G. Bell and J. N. Gray. *The revolution yet to happen*, volume Beyond Calculation. Springer Verlag, 1997.

[2] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Jade a fipa-compliant agent framework. In *Proc. of the 4th International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents, London*, 1999.

[3] Erwin Bonsma and Cefn Hoile. A distributed implementation of the swan peer to peer lookup system using mobile agents. In *AAMAS Proceedings of Agents and Peer to Peer workshop*, 2002.

[4] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, Maryland, 1994. ACM Press.

[5] Joseph Heller. *Catch 22*. Simon and Schuster, 1999.

[6] Cefn Hoile, Fang Wang, Erwin Bonsma, and Paul Marrow. Core specification and experiments in diet: A decentralised ecosystem-inspired mobile agent system. In *AAMAS Conference Proceedings*, 2002.

[7] Kevin Kelly. *New Rules for the Network Economy : 10 Ways the Network Economy is Changing Everything*. Barnes and Noble, 1999.

[8] Paul Marrow. Survey of agent toolkits, 2001. Available to ECOMAS forum members after free registration.

[9] Paul Marrow, Cefn Hoile, Fang Wang, and Erwin Bonsma. Evolving preferences among emergent groups of agents. In *AISB Proceedings of Adaptive Agents and Multi-Agent Systems Workshop*, 2002.

[10] Mariusz Nowostawski, Geoff Bush, Martin Purvis, and Stephen Cranefield. A multi-level approach and infrastructure for agent-oriented software development. In *Proceedings of the 2nd Workshop on Infrastructure for Agents, MAS and Scalable MAS at the 5th International Conference on Autonomous Agents, Montreal*, 2001.

[11] A. Oram, editor. *Peer-to-peer : Harnessing the Power of Disruptive Technologies*. Sebastopol, CA, 2001.

[12] S. Poslad, P. Buckle, and R.G. Hadingham. The fipa-os agent platform: Open source for open standards. In *Proceedings of PAAM , Manchester, UK, 355-368*, 2000.

[13] Michael Wooldridge and Nicholas R. Jennings. Pitfalls of agent-oriented development. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 385–391, New York, 9–13, 1998. ACM Press.