53

An Agent-based Approach to Service Management - Towards Service Independent Network Architecture

Gísli Hjálmtýsson
AT&T Research
600 Mountain Avenue, Murray Hill, NJ 07974, (908)582-5495, gisli@research.att.com

A. Jain
AT&T Laboratories
101 Crawfords Corner Rd, Holmdel, NJ 07733-3030, (908)949-5856, akj@hostare.att.com

Abstract

With deregulation of the telecom industry the intense competition for customers is driving service providers to offer new and sophisticated services at an increasing rate. Simultaneously, the Internet is attracting vendor creativity and putting a fatal pressure on the traditional telecom pricing structure. Whereas the telecom industry still leads in service quality, there is growing need for a cost effective architecture for network and service management comparable in responsiveness and flexibility to the Internet, yet capable of maintaining high service quality for increasingly complex services.

Although autonomous agents have been proposed for networks and distributed systems, they have largely been considered as reasoning entities exhibiting some form of intelligent behavior. Viewed more as autonomous objects, however, agents provide a powerful abstractions even when the agents task is more mundane in nature. In particular, the ability to move from one location to another goes beyond the strong level of modularity provided by object orientation, by disassociating each autonomous agent from a particular location or environment

In this paper we propose a new agent-based architecture for service management and provisioning. We describe an agent-based service environment and argue how such an environment supports rapid service creation and enables transparent services across authority domains.

Keywords

Service management, agents, active networking, network architecture, signaling

A. A. Lazar et al. (eds.), *Integrated Network Management V*

1 MOTIVATION

As more players rush to become providers of communication services, the intense competition for customers is driving service providers to offer increasingly sophisticated services at accelerating rate. Although partially caused by deregulation of the telecom industry, the explosive growth of the Internet is fundamentally changing the landscape of communication, creating a new class of service providers that are putting fatal pressure on the traditional telecom pricing structure. The uniform service model of the Internet and localized control provides flexibility and responsiveness for service creation that is attracting vendor creativity. A key factor in this architecture is separation of responsibility and limited integration of service semantics into the underlying network. The network is responsible for delivering packets, the end systems are responsible for providing semantics to the packets delivered.

In contrast, the telecom infrastructure is a tightly knit web of hardware and software, where service logic is interwoven with more primitive capabilities at all levels of the network. For example, the most successful enhanced service in telephony, the 1-800 service, has connotation at all levels of abstraction. At the lowest level it represents indirect addressing, whereas at service level it means name resolution, load balancing and time-of-day sensitive routing. At the highest level it implies reverse charging. In addition to the blurring of concepts, the implementation is even more tightly coupled, with network element recording sensitive to reverse charging, and service level performing the indirect address resolutions. Whereas integration promotes performance, introducing new services becomes complex and costly, particularly since introducing a new service incurs cost for all existing services. Moreover, this rigid architecture causes long delays when introducing new services. Although some of this can be attributed to legacy, without decisive departure from current service architectures, newer transport networks will inherit this legacy (ATM Forum, 1995).

However, when compared to the Internet, the telecom industry still holds a significant lead in service quality. For example, whereas Internet telephony has become available, its quality is comparable to what the phone network provided for international calls a decade or more ago. A similar difference holds for other multimedia applications. The availability of the phone network is unsurpassed (99.99999%). In essence, whereas the Internet excels in recovering from failures, the phone network hardly ever has them*. Moreover, although the Internet never completely blocks a connection, service is frequently too poor to be of value. In contrast the predictable high quality service of the phone network is offered with practically no call blocking. Whereas demands for high standards of quality are in part responsible for the tight integration, there is a growing need for cost effective architecture for service and network management comparable in responsiveness and flexibility to the Internet, yet capable of maintaining the high service quality of the telecom industry for increasingly complex services.

* Although network elements may fail, there is a tremendous amount of hot sparing in the telecom world, thus hiding element failures from customers. Even at micro level every quantum of a conversation is treated reliably.

We therefore seek an architecture that uniformly separates network management from service management, while offering customized support to dynamically created services. Uniformity and separation of responsibility provides flexibility and responsiveness. Since all services are treated the same, no new facilities are needed to add or enhance services. Separation of responsibility localizes the scope of modifications, physically, at abstract level, and in terms of component modification. Whereas a uniform service model simplifies service management, removing service semantics from network elements simplifies network management. Maintaining high service quality of increasingly complex services requires customized support, including the ability to intervene or control an ongoing conversation. Agents provide a powerful abstraction that provides customized support, while operating in a uniform environment. Separation of responsibility is achieved by the agent environment providing an opaque interface - the service semantics are encapsulated in the respective service agents; the network facilities support primitive and generic abstractions but are oblivious to services and their semantics.

This paper is a paradigm paper centered on two themes. Uniformity and separation of responsibility bring flexibility and responsiveness. Customization and control provide service quality. Current network architectures trade one for the other but do not support both. We propose a new agent-based architecture for service management and provisioning, which is uniform and strongly separates service management from the low level network, yet provides sufficient customized control to maintain quality service. We show how this architecture, supports rapid service creation and real time provisioning while hiding the underlying network specifics. We furthermore argue how the agent-based architecture simplifies service management and enables transparent services across domains of authority.

After discussing related work, we analyze in Section 3 how these themes translate into architectural requirements. In Section 4 we describe the agent-based architecture. A key component of this architecture is a uniform agent environment supported by a limited set of basic network primitives, that together act as an opaque interface separating the network proper from the service level. Services are implemented by autonomous service agents that encapsulate service semantics. We show how this architecture and the agent abstraction simplifies service management. Several examples of service agents are given:

- elementary examples in Section 4.1,
- an example of mobile services in Section 4.2.2,
- an example of a call screening agent providing transparent service across domains in Section 4.2.3, and
- a customer agent in Section 4.3 illustrating rapid service creation and its enhancement in Section 4.4.

The paper concludes with a summary and discussion in Section 5.

2 RELATED WORK

Although autonomous agents have been proposed for networks and distributed systems, they have largely been considered as reasoning entities exhibiting some form

of intelligent behavior (Etzioni and Weld, 1994), (Genesereth and Ketchpel, 1994), (Lashkari, Metral, and Maes, 1994), (Magendanz, Rothermel, and Krause, 1996). However, viewed more as autonomous objects, agents provide a powerful abstraction for distributed computing and networking, even when the agents task is more mundane in nature. In particular, the ability to move from one location to another (Gray, 1996), (Gray, Kotz, Nog, Rus, and Cybenko, 1996), goes beyond the strong level of modularity provided by object orientation, by disassociating each autonomous agent from a particular location or domain.

The work herein is related to (Tennenhouse and Wetherall, 1996), proposing the use of mobile agents to dynamically change network functionality. In comparison, our work is more aimed at complex services and service management and emphasizes separation of services from the underlying network. (Magendanz, Rothermel, and Krause, 1996), provides a taxonomy of intelligent agents for network management and discusses the potential of agent-based solutions for network and service management. Providing some of the functionality of agents, several projects are experimenting with Java (Arnold and Gosling, 1996) for service creation and management. In a related project we are working on mechanisms, implemented in a C++ library, to implement and support agents (Hjálmtýsson and Gray, 1996).

Other management frameworks that share some of the objectives of our work, include CORBA (Spec., 1995), (Vinoski, 1993) and the TINA-C (Chapman, 1995), (de la Fuente, 1994), (Berndt and Minerva, 1995) effort. Whereas the CORBA effort focuses on enabling heterogeneous systems to inter-operate, by defining interfaces, services, and information exchange mechanisms, our proposal is an environment and a methodology to create and manage the services themselves. While the TINA consortium is addressing most of the issues we cover our approach differs substantially. In particular, this work draws on some of the work on open signaling (Lazar, 1994), (Lazar and Stadler, 1993), (Wu, 1996), (Hjálmtýsson, 1997), on streamlining of signaling and management.

3 MOVING SERVICE SEMANTICS OUT OF THE NETWORK

In addition to the above arguments for removing service semantics from the network, the benefits of integration and service specific network optimizations are diminishing. The current phone network is engineered to support telephone conversations. Accordingly, engineering and controls make assumptions about call duration, bandwidth, and quality requirements to optimize interactive voice conversations. However, large proportion of the network capacity is currently used for other services, including data modems, facsimile and very short request-reply exchanges (e.g., credit card authorizations). Moreover, the telephone infrastructure transports most data traffic, including Internet traffic, on leased virtual private networks. Currently growing at much faster rate than telephony, the volume of these traffic classes already invalidates many of the phone-call-engineering assumptions.

Furthermore, as resource capacity increases, radically changing assumptions about signaling bandwidth, database performance and processing capabilities, the need for tight integration is further reduced. Current signaling in the telecom world, conducted

on a physically separate network, is static in topology and capacity is limiting both in terms of performance and flexibility. Furthermore large proportion of signaling is service related, thus representing service semantics within the network. Carrying service level signaling as an ordinary (guaranteed service) connection, strengthens the separation between the network and the service level, and enables service level signaling to scale up with transport capacity. Exploiting modern processing power and high speed networking, combined with algorithms to hide latency, out-of-network service management can perform competitively with existing signaling systems.

A streamlined transport infrastructure, free of service semantics, with network signaling and control limited to element communication and primitive information exchange, amounts to a RISC like networking architecture. Each network component, and the network as a whole performs very basic and streamlined operations, accessed by higher levels through a very limited set of basic primitives (reduced instruction set). The service agents, playing the role of a smart compiler, translate service logic into series of primitive instructions. Such a network architecture is inherently flexible and promotes the use of commodity parts for processing and transmission, both of which steadily provide increased performance per unit cost. In contrast, customized facilities are becoming increasingly costly.

3.1 Network infrastructure that contains no service semantics

The challenge in building a network supportive of the service quality of the telecom world, still having a service model as flexible and responsive as the Internet's, requires a uniform service environment maintaining strong separation of network and service level concerns, while enabling customized service control of network resources. Figure 1 depicts the two layers of such an architecture, the **network level** separated from the **service level** by a thin interface. This interface consists primarily of basic primitives to network capabilities. The service environment, in which services are created, controlled and performed, consists of an execution environment and virtual network defined by the network interface. Uniformity is achieved by making this environment uniform throughout the network infrastructure.

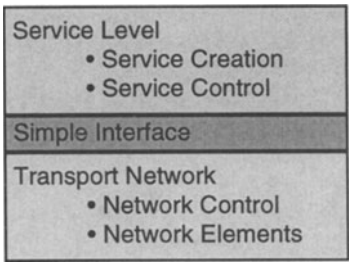

Figure 1 Moving Service Semantics out of the Network

Separation of network concerns from service concerns is achieved by limiting the interaction between the layers to the primitives defined by the interface. As a result, since the service level accesses network resources only through the interface primitives, the physical resources, the hardware and the software implementation, is hidden and of limited consequence to the service level. Conversely, since all services are treated the same by the network new services can be introduced or existing services enhanced without any new facilities or support from the network. This way, service and network management becomes independent, significantly reducing their respective complexities.

Whereas service logic is moved **up** from the transport network infrastructure, the service level is supported throughout the network. Therefore, service logic exists throughout the network, customizing network control (still through the interface's primitives) for each service, to ensure highest service quality. In contrast, the traditional Internet service model relegates service semantics to the edges of the network, disabling service dependent inner network behavior.

4 AGENTS PROVIDE NETWORK INDEPENDENCE

To construct a uniform service environment enabling customized service control of network resources while maintaining strong separation of network and service level concerns, we propose an agent-based service level, with network resources hidden beneath an agent environment. Services are implemented in **service agents**, which encapsulate service semantics, including service specific data and the logic to interpret the data. The service agents operate in an agent environment which provides generic agent support - e.g., creation, destruction and execution of agents - plus access to the network primitives. Together, the network primitives and the agent environment provide an abstract network interface, separating services from the underlying network. Assuming that the generic agent support, and the limited set of network primitives is the same across the network, service agents see a uniform service environment.

The agents themselves represent autonomous objects that are mobile within the environment. Beside the ability to move, the agents are not required or assumed to exhibit intelligent behavior. The service agents themselves can, however, be of arbitrary complexity, notwithstanding being composed of other components, or cooperating with objects or agents that exist outside the agent environment. Access to the network is limited to the primitive interface provided through the agent environment. In many ways therefore these agents can be viewed as a generalized processes, and the agent environment as an operating system, controlling access to the underlying network hardware by limiting it to a small set of network primitives.

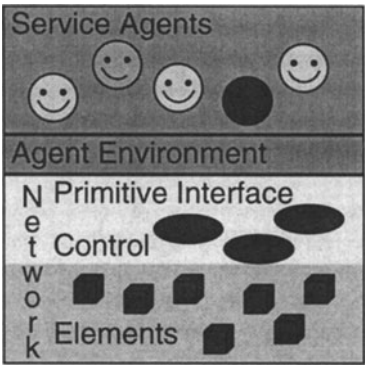

Figure 2 Separating the Network and the Service Infrastructure.

In this framework, all services are implemented by a service agent. Moreover, all service semantics is contained and encapsulated within the agent. We distinguish a subset of service agents as **customer agents**, whose responsibility is primarily towards the service user. Whereas the goal of all service agents is to make a service available, the goal of a customer agent is to customize it for individual users. In particular, all user specific information - data, rules, or user supplied logic - is encapsulated in a corresponding customer agent. From the networks point of view all agents look the same. Conversely, seen through a service agent all networks look the same. In general, **network interface agents**, a generic low capability service agents, encapsulate

network specifics; more elaborate service agents define a service, with customer agents providing customized user support. A service agent can execute at personal devices, customer premise or within the network, in any combination, provided that they provide an agent execution environment. This way the agent abstraction helps separating the service function from concerns about service location.

4.1 Some Elementary Agents

To make the discussion more concrete consider some examples of service agents, the most basic being a regular phone agent, a POTS[†] agent. Given a phone number (or in general a name) it establishes a phone connection, performing an indirect address resolution if needed. In particular, the agent will translate the name into an Internet address, or resolving a 1-800 name to a network address. Upon connection the agent identifies the type of receiving device (fax, voice mail, busy, no answer), thereby enabling enhanced service agents to react accordingly. Furthermore, the agent issues an indication upon call completion. While simple, this agent is already an enhancement of current POTS service. More importantly, the agent unifies telephony service, making it independent on transport technology (e.g., current phone network, Internet phone, etc.), while inviting other agents to build on it to provide an enhanced service.

For a more elaborate example, consider a call forwarding service where incoming calls are forwarded to a set of destination depending on "caller" identification. Assume the service is implemented by a Forward-Agent, which uses a network primitive to forward a connection. When subscribing the customer supplies a list of names (e.g., phone numbers) and rules to determine whereto each name is to be forwarded. The customer specific information and logic is encapsulated in a "customer" agent, and stored in the network. Upon an incoming connection request, the customer agent is invoked, who then screens the incoming name, applies the customer rules and forwards the call. Assuming an agent environment and that a network primitive to forward a connection exist, this service can be provided without any service specific modification to the network or the agent environment. The service semantics is fully encapsulated within the Forward-Agent and the customer specific "helper" agent.

In contrast, whereas the Internet service model would put such a service completely out of the network, the traditional telecom approach would involve modifications at all levels of the network. Unlike the Internet however, for performance, cost and customer service reasons it may be advantageous to offer the service from within the domain of the service provider. However, introducing new database(s) to store the list of numbers and their associated forward number, plus modifying the infrastructure to implement the service logic, affects existing services and results in growing complexity for network and service management. Instead, an agent approach encapsulates the service semantics - the data and the logic to interpret the data - in an agent that operates in a general purpose agent infrastructure accessing network facilities only through an interface to very basic primitives.

[†] POTS - Plain Old Telephony Service

4.2 Mobile Agents Enhance Service Management

In addition service management benefits from enhanced modularity and separation from network management, the mobility of service agents support service management, by conceptually separating the service abstraction from location of execution, by simplifying resource allocation, and by enabling transparent services across service domains. Various "interface" approaches achieve the separation of service definition from service implementation, resulting in a growing set of interface primitives (e.g., an API). However, a major conceptual problem arises when a part of this interface is to be implemented in one location, e.g., inside the network, and the rest in another, say customer premise. For example, consider a service API provided at the network interface. Suppose a subset of services is moved locally to the customer premise (e.g., become available at a new local PBX). The conceptual problem is: what does this mean for the interface? One option is to say that the full interface is available locally, partially implemented locally and partially relayed to the network interface. An alternative is to say that there is a locally available interface for some services, but not for others, the locally available interface overriding a portion of the full interface residing at the network. Neither one is very elegant. Conceptually either the local interface is relaying requests to the network interface, or the two implementations are conceptually different and thus so are the interfaces.

On more practical terms, after decades of debate about whether the network intelligence belongs inside, outside, or at the boundary of the network, the reality is that it appears at all of these locations. Whereas a home-office corporation may employ an answering machine and its only workstation, a large corporation may implement a rich set of services in their local PBX. Yet some other services may only be available from a service provider. Of course the users are indifferent to such details, assuming that the service is consistently delivered. Service agents provide a metaphor that captures this rather simply: offering a service at a different location conceptually corresponds to moving the agent. More generally service agents allow definition, and implementation of services independent of their domain of execution. Indeed, there is an increasing market for various services offered only within a local domain.

4.2.1 Agents Enhance Resource Management

Whereas transport capacity is largely managed below the agent environment, the agent abstraction enhances resource management through mobility, uniformity and encapsulation. In particular mobile agents facilitate load balancing by migrating agents. While objectives in a communication system may not require strict balancing of load, overloaded processing nodes recover by migrating agents to lightly loaded nodes in the network. In fact, to economize further on network resources an agent might be ejected to customer premise for processing. Service encapsulation within the service agent gives a conceptually convenient unit of relocation. Uniformity of the agent environment, reduces migration complexity, as simple cost measures are sufficient to select a destination of migration. In fact, a new service management dimension opens

up with agent mobility as the service environment and service domains can be reconfigured rapidly.

Agent mobility furthermore enhances service quality, as services are moved to where needed on demand. This improves service availability, and response seen by users as most of the time services are provided locally, still available globally. Moreover, migrating service to a local domain improves resource utilization without sacrificing service sophistication. In contrast, consider for example mobile communication. If a user from New Jersey, currently visiting California receives a call from across the street (in California of course), it would be desirable to most of the call processing locally. In particular minimizing transport resources implies routing the conversation instate. However, except for the most elementary services, where migrating the dumb state and leaving it to the Californian network provider to interpret the state is insufficient. Mobile service agents solve this problem as the agent encapsulates the state and the logic to interpret the state.

The uniformity of the agent environment simplifies resource allocation and provisioning, as resources become interchangeable. Since databases become agent-bases, storing encapsulated agents, they are used as generic data stores, and thus equally suited for any agent. As mentioned above, same applies to processing units, all of which implement the generic agent environment. Apart from the current residency of agents at a particular node, every node appears equal to a service agent. Therefore, per service resource forecasting is not needed (or can be estimated more crudely) significantly reducing resource provisioning, and enhancing the scalability of the infrastructure.

The service agent abstraction is valuable in maintaining the service itself, as service support updating service logic and the data that defines the service without interruption of service. Service agents are transportable autonomous program entities, inherently encompassing the concept of dynamically introducing and removing a code fragment into/from an executing program. Exploiting the encapsulation properties of agents, this is achieved while preserving program level type safety and maintaining service level abstractions. Honoring the agents conceptual integrity thus helps in resolving version problems resulting from and update in the code implementing a service. As a consequence, the agent abstraction supports service evolution on reasonably small component granularity, and helps solve the difficult problem of service maintenance.

4.2.2 Transparent Service Across Domains

Consider a new service that is offered only in a limited geographical area. Such limitations might be transient for new services, or could arise from market segmentation or operational reasons. For example, while the infrastructure for wireless services is being built, the intra-domain service support is physically limited to the region covered by the already deployed equipment. To provide maximum coverage, the service provider may therefore sub-contract (and resell) services from other network providers. Similarly, marketing strategies in foreign markets (or otherwise geographically segregated markets) may differ significantly, thus resulting in different services being supported in different parts of the infrastructure. Although owned by the

same provider, each service is therefore supported only in some sub-domain. In either case, however, customers do not want to be inconvenienced, and expect the service they subscribe to be available even when they cross the domain boundaries.

In spite of this desire, with current network architectures transparent service across segmented domains is not really an issue - it simply cannot be done. One reason for this is that current network architectures are not designed for heterogeneous support for sophisticated services. Currently, either relatively simple services are negotiated and standardized, and then offered throughout the network (and across multiple carriers); otherwise services are only available within a particular domain, and simply not available elsewhere. This remains true even when older switches/routers - supporting only a fraction of the latest standard - are inter-operable with newer switches, since typically the newer services are unavailable on paths using an old switch.

Instead, consider the uniform agent environment. Rather than encompassing service semantics for all supported services, the agents ability to move and perform throughout the environment, provides for transparent services across domains. A service normally offered in one domain, is in fact universally available, by having the corresponding service agent migrate on demand. Heterogeneity arises as service agents reside primarily where their services are offered (or deemed likely to be needed by service management) but are not universally distributed throughout the network.

4.2.3 An Example - Advanced Call Screening

Consider an advanced call screening service, which beside offering call blocking allows the customer to have incoming calls be processed based on caller identification. Capabilities could include, redirecting to other numbers (e.g. secretary), redirecting to voice mail, or pass the call through all based on customer preference. While not inconceivable in the current network two difficult problems arise. The more general is the problem of added network complexity as databases and code must be changed within the network and potentially some additional (special) equipment added. Still, the more problematic is how to this type of service on more global scale. Provided that a service provider operates globally, it is only natural for a customer subscribing to a service in one region, say the US, to demand the same abroad, e.g., Europe while within the service region of the provider. However, for various reasons a service offered in on region may not be offered in others.

Implementing this service in an agent environment, a call screening agent supports transparent service by migrating to where the service is requested. Unlike services like mobile, where state information (data) is migrated to where the mobile is registering, the agent brings not only data but also the knowledge of how to interpret the data. This removes the need for advance service provisioning, as the migration of the agent constitutes service provisioning on demand. Furthermore, since the agent only utilizes general purpose facilities, the marginal impact is negligible, and thus service specific resource provisioning is not required.

4.3 Agents Support Rapid Service Creation

The agent environment supports rapid service creation, as a new service does not require any new network support, nor coordination with other service or network providers. In particular a new service does not imply provisioning of service specific facilities - databases, or code - other than encapsulated within the respective service agent. Furthermore, a service provider could unilaterally introduce a service without any negotiations or announcements to other service providers. Still the new service would receive support comparable to any existing service. Indeed, a new player, potentially without any network infrastructure, could become a service provider simply by creating a new service agent and introducing it into the environment (of course financial settlements would need to be negotiated).

To illustrate rapid service creation consider the following example service, which we will call "*the panic button.*" Subscribers identify a list of phone numbers, in some order of preference, and can specify what constitutes a successful *panic*-resolution. As an example parents could leave a *panic-button* capable device (could be a cellular phone, smart card etc.) with their children. The ordered phone number list would contain, each parents work phone number, home number, number of friends and relatives. Upon a *panic-call*, the list is processed in order, either until successful or the list is exhausted.

A traditional telecom solution would be to introduce a database to store the list of phone-numbers, and the processing logic for a panic-call. Panic-call processing then would involve identifying the customer (which could for example be the device identifier), fetching from the database the panic-list of phone numbers, and finally processing each of the numbers. Since, this new service may interfere with other services already in the network, careful system integration and service provisioning (and perhaps re-provisioning of some of the existing services) would have to be performed.

An alternative would be to push all of the intelligence out of the network, and have a smart terminal, for example a PC, implement the service completely out of the network. Call processing in this framework is simply seen from the network as a series of call requests coming from the same source, the smart terminal providing all the call processing logic. While possible, it requires substantial processing capabilities of the terminal device and requires control information (e.g. about call completion status) to propagate out of the network. Whereas the latter requires standardized information exchange, the former translates into more costly equipment.

Using a customer agent - a *panic-agent* on the panic-button device is used to implement this service. We assume that an POTS service agent exists, capable of processing regular phone calls, returning either a successful connection, or a completion indication (busy/no answer/voice mail). Upon a panic-call the panic-agent is invoked from the panic-button agent pool with the list of numbers. The agent then cooperates with the POTS service agent, processing each number on the list by issuing requests to the POTS-agent, and receiving back call completion information.

The key advantages of the agent approach are:

1. No service specific database or any other support needs to be provisioned before the panic-button service can be offered. Still, the service is able to take advantage of being executed within the network. In particular, internal network information is available to the customer agent.
2. Given that a network supports the POTS service agent, the panic-button service can be offered by a third party network services reseller who has no control of the physical network or network resources.

4.4 Out of Network Service Enhancement

Although the above panic-button example could potentially be offered out-of-network, while non-traditional, the above example is still as an in-network service offering. This means that although storage and other resources are not customized for the service, they are committed and owned by the service provider and reside within the network. In this context, out-of-network service offering, therefore means that the service is offered without any network resources commitment or control except during execution. In particular the user data and the code to interpret it - i.e., the customer agent - is not retained within the network or the service providers domain, but is instead injected into the network from customer premises upon service request. Out-of-network services for example facilitate third party service provider owning no network infrastructure.

To illustrate, out-of-network service enhancement, consider the following enhancement of the above panic-button service. Suppose now, that beside a POTS-service agent, an email service agent - an agent who accepts messages and delivers them to an Internet email address - becomes available in networks. Suppose an independent third party service provider decides to offer an enhanced "panic-button" service, by allowing the list of addresses to contain email addresses as well as phone numbers. Furthermore, since the service provider is not a network provider, the service will be offered as a small panic-device similar to a beeper. The service is implemented by an enhanced customer agent, which in addition to recognizing phone numbers recognizes email addresses.

Upon a panic-call, the device establishes a connection to the network, and injects the customer agent, carrying the panic-list, into the network. Once within the network, the agent processing the panic list as, issuing requests to the POTS agent, or issuing a panic message to the designated email recipient with the help of the email service agent. After the agent completes the list, it is destroyed by the agent environment. Observe that as before, no service specific support or provisioning is needed. Moreover, the network maintains no persistent knowledge of this enhanced service, no data nor logic. Indeed, after the panic-agent is destroyed the network acts as if nothing had happened.

In contrast a traditional network centric solution would require redesign of the supporting databases, and other provisioning, in addition to code modification. As for any new service, introducing the enhancement into the network could also cause potential interference with other existing services.

5 SUMMARY AND DISCUSSION

We have described an agent-based paradigm for service management, that exhibits the flexibility of the Internet while supporting customized high quality of the telecom world. We have proposed an architecture for this paradigm and shown how agents provide simple solutions to many complex service management tasks. The architecture factors services out of the network into a separate service layer, separated from the network by an opaque interface. Services are implemented by service agents, completely encapsulating service semantics, the data and the logic to interpret the data. In particular, the underlying network is free of service specific support. Instead, the service agents operate in a agent environment, accessing the network only through a limited set of basic primitives. The environment and the set of network primitives is the same across the network, resulting in a uniform view from the service agents.

We have shown how the agent abstraction, the uniformity of the agent environment and the agents ability to move helps in reducing service management complexity. Specifically, the agent-based service environment supports rapid service creation and real time provisioning. Moreover, agent-based service management enables transparent services across domains of authority. The agent abstraction conceptually clarifies some common problem in service management, such as service migration to a local domain, and service maintenance. More generally, the ability of agents to move from one location to another goes beyond the strong level of modularity provided by object orientation, by disassociating each autonomous agent from a particular location or environment. The uniformity of the agent environment simplifies resource management, thereby reducing provisioning cost and signaling requirements, as network resources become interchangeable.

This work is part of a larger project on lightweight networking aiming to streamline signaling, simplify network and service management, and exploit high level modeling abstractions at low levels of networking. In (Hjálmtýsson, 1997) we report on a lightweight call setup primitive and protocol, supporting both connection and connectionless service, while encompassing enough generality to facilitate arbitrary parameters on call setup. In particular, the light weight call setup supports the agent ideas contained herein. Although agents have been proposed before for networks and distributed systems, the low performance of agent-based systems have relegated them to tasks like user interfaces or web-searching, where performance is not a primary issue. We have developed a C++ library implementing agents, the agents themselves written in C++, compiled and run natively. This paper is a strand of a general theme investigating flexibility not only as a design goal but as a performance metric, with the aim to develop methods and metrics to quantify some of the guidelines of software engineering that postulate loose coupling and encapsulation. With the deregulation of the telecom industry and the omnipresence of the Internet, we believe that the ultimate performance of any communication architecture will hinge on its ability to handle service diversity and volatility.

5.1 Acknowledgements

Special thanks to Albert Greenberg for his help on the presentation of this work.

6 REFERENCES

K. Arnold and J. Gosling (1996) *The Java Programming Language*. Addison-Wesley, Reading, MA.

ATM Forum 95-0221R2, (1995) *Draft PNNI Signaling*.

H. Berndt and R. Minerva editors (1995) Definition of Service Architecture. *TINA-C Baseline Document*, Version 2.0.

M. Chapman editor (1995) Overall Concepts and Principles of TINA. *TINA-C Baseline Document*, Version 1.0.

The Common Object Request Broker : Architecture and Specification, Rev. 2.0 (1995)

O. Etzioni and D. Weld (1994) A Softbot-Based Interface to the Internet. *Communications of the ACM*, **37-7**, 72-6.

L. A. de la Fuente editor (1994) Management Architecture. *TINA-C Baseline Document*, Version 2.0.

M. R. Genesereth and S. P. Ketchpel (1994) Software Agents. *Communications of the ACM*, **37-7**, 49-53

R. S. Gray (1996) Agent Tcl: A flexible and secure mobile-agent system. *Proceedings of the Fourth Annual Tcl/Tk Workshop (TCL 96)*, Monterey, California

R. Gray, D. Kotz, S. Nog, D. Rus and G. Cybenko (1996) Mobile agents for mobile computing. *Department of Computer Science, Dartmouth College.*

G. Hjálmtýsson and R. Gray (1996) Dynamic Classes Enhance Maintainability of Critical Applications. *AT&T Research Manuscript.*

G. Hjálmtýsson (1997) Lightweight Call setup - Supporting connection and connectionless services. To appear at *the International Teletraffic Congress ITC'97.*

Y. Lashkari and M. Metral and P. Maes (1994) Collaborative Interface Agents. *Proceedings of AAAI'94*

A.A. Lazar and R. Stadler, (1993) On Reducing the Complexity of Management and Control of Future Broadband Networks. *Proceedings of the Workshop on Distributed Systems: Operations and Management*, Long Branch, NJ.

A.A. Lazar (1994) A Research Agenda for Multimedia Networking. *The Workshop on Fundamentals and Perspectives on Multimedia Systems, International Conference Center for Computer Science*, Dagstuhl Castle, Germany.

T. Magedanz, K. Rothermel, and S. Krause (1996) Intelligent Agents: An Emerging Technology for Next Generation Telecommunications? (1996) *in Proceedings of INFOCOM'96*, San Fransico, California, 464-472

D. L. Tennenhouse and D. J. Wetherall (1996) Towards an Active Network Architecture. *Computer Communication Review.*

S. Vinoski (1993) Distributed Object Computing with CORBA. *C++ Report.*

D. Wu (1996) An Efficient Signaling Structure for ATM Networks. *Proceeding of INFOCOM'96*, San Francisco, 844-854

7 BIOGRAPHY

Gísli Hjálmtýsson received a B.S degree in Applied Mathematics and Computer Science (1987) from University of Rochester, and M.S. (1992) and Ph.D. (1995) in Computer Science from University of California, Santa Barbara.

Dr. Hjálmtýsson joined AT&T Bell Laboratories, Murray Hill, in 1995, and is currently at AT&T Labs - Research. In 1993, he was a visiting scientist at Telecom Australia's, Telecom Research Laboratories in Melbourne.

His current research is in performance evaluation of networks and information systems, lightweight signaling, active networking, and new opportunities resulting from the convergence of computer networking and traditional telecommunication.