OPTIPRISM: A Hierarchical Distributed Network Management System for All-Optical Networks

Bilal Khan* Dardo D. Kleiner† David Talmage*
Center for Computational Science, Naval Research Laboratory, Washington D.C.
http://www.nrl.navy.mil/ccs/project/public/DC/web/
{bilal,dkleiner,talmage}@cmf.nrl.navy.mil

***Index Terms*— Network management, optical network, multi-agent system.**

***Abstract*— This paper describes the design and implementation of Optiprism, an agent-based network management system (NMS) providing configuration and fault management services for all-optical networks. Optiprism is designed to support (1) a scalable architecture consisting of a distributed hierarchy of software agents, or managers (2) the ability to alter the hierarchy as the network evolves by adding, removing or upgrading managers (3) reorganization of physical deployment for better responsiveness (4) an innovative browser agent providing scalable end-user interaction with the distributed NMS.**

I. Introduction

Traditional network management software implementations have used centralized paradigms based on SNMPv1 or SNMPv2c, or weakly distributed hierarchical paradigms based on SNMPv2, RMON, CMIP, or CMIP derivatives such as TMN [15, p. 5]. While these approaches are feasible in small networks, their communication costs grow linearly with the number of devices [19, p. 4]. Wavelength division multiplexing (WDM) networks present additional difficulties since the central problem of routing and wavelength assignment (RWA) [18] is NP-complete [20] and even heuristic approaches to it are computationally expensive [4, p. 2]).

An effective optical NMS must thus address the core problem of scalability. We contend that a *strongly distributed* deployment of a hierarchy of *cooperating* intelligent agents [15, p. 9] or "managers" would yield significantly reduced processing requirements at the client-side. These managers would maintain aggregated information such as route availability and fault reports about recursively smaller sections of the network. Moreover, if the network's state was hierarchically distributed, then management applications would not need to establish direct connections to every network element. Instead, the administrator would interact with high-level supervisory managers. Control operations (e.g. lightpath provisioning and teardown) would be issued to these high-level managers, which would compute routes and delegate partitioned connection requests to their subordinate managers. Monitoring of alarms and alerts would operate in the reverse direction: subordinate managers would report fault conditions to their supervisor. A management application would only need to communicate with high-level supervisors in order to manipulate and monitor the optical network. The next sections describe the design and implementation of such a network management system.

∗ Advanced Engineering & Sciences, ITT Industries
† Computer Integration & Programming Solutions, Corp.

II. Design

In designing Optiprism, we adopted a distributed architecture because it enabled us to meet four important objectives. The most critical is *scalability*. In large networks, the processing of management requests (e.g. route selection) presents computational burdens that would ultimately choke a centralized NMS. In contrast, a distributed architecture can amortize this computational overhead against a set of processes distributed throughout the computational environment [10, p. 1]. Second, a distributed architecture is *maintainable* because it is easier to augment as the network grows. Third, a distributed architecture permits computations to be closer to information sources, reducing latency and total control traffic [7] [12], thereby yielding better *responsiveness*. This benefit is amplified if the architecture supports dynamic re-distribution of managers, since then the NMS can adapt to circumvent computation and communication hot-spots in its environment [11, p. 32]. Finally, adopting a distributed architecture makes it possible to develop end-user management applications which exhibit *scalable interaction*, i.e. applications that interact with only a scalable subset of the NMS at a given time. We now describe how the design of Optiprism strives to meet these objectives.

A. Scalability

An effective optical NMS must able to coordinate the control planes of hundreds of optical switches. This objective led to the choice of a hierarchical architecture. In Optiprism, each manager can be a *supervisor*, composed of several *subordinate* managers. Conversely, each manager—with the exception of a unique "root"—is subordinate to some supervisor. In a supervisory role, each manager provides an interface to the services it can implement using the functionality of its subordinates. Two managers are called *peers* if they have the same supervisor.

Complications arising from this design choice include: (i) higher level managers may experience greater load and (ii) failures at higher levels may have non-local negative side-effects on the NMS. Presently these concerns are addressed by assigning high-level managers to more reliable machines that have larger memory and processing power. We are investigating the possibility of addressing both issues through replication and clustering of managers.

B. Maintainability

The NMS architecture should be easy to alter as the network evolves. In a hierarchical NMS, this would be achieved by addition and removal of managers, and by restructuring of the hierarchy. Adding new hardware to the NMS domain should require little more than inserting a new specialized subordinate into the hierarchy. Let us see how Optiprism achieves this.

In Optiprism, there are three types of managers:

1. *Element managers* exist at the lowest level of the manager hierarchy. Each manager controls and monitors a physical device via specialized communication protocols.
2. *Subnet managers* delegate to and aggregate from lower level managers.

These two types of managers expose a command interface to the next higher level and a notification interface to the next lower level. All command and notification interfaces are functionally identical, regardless of the manager's level.

Making all subnet and element managers indistinguishable makes it possible to add, remove and splice managers into an existing Optiprism hierarchy at run-time. It has also yielded benefits of simplicity in their implementation and interactions, while providing encapsulation at the manager level. Subnet managers are truly "virtual optical switches".

One issue with this approach is that element managers for new devices must adhere to a specification representing the least common denominator of the functionality of all devices. As vendors adopt standards for optical network provisioning and management, this penalty will be alleviated. An agent-based attempt at such standardization is [8] by FIPA.

Physical network topology is reflected by deployment of:

3. *Link managers*, each of which represent a physical connection between two elements/subnets.

Subnet managers determine their internal topology (i.e. the connectivity among their subordinate subnets/elements) by consulting subordinate link managers. In addition, they discover connectivity with peer subnets/elements by consulting their peer link managers. In the terminology of [5], all Optiprism managers can be considered *netlets* because they have a persistent process-based life-cycle model [11].

C. Responsiveness

The performance of an agent-based NMS is influenced by the characteristics of both the hardware on which the agents reside and the network over which they communicate. Cost factors make it impractical to dedicate entire machines and separate networks solely for the NMS. On the other hand, permitting managers to mingle with external processes on multi-purpose machines means that the system needs to sense fluctuations in performance characteristics and act to minimize impact on the NMS. This requirement underscores the need to support process mobility [11, pp. 26-33] as a core feature of the NMS. One drawback of allowing managers to be mobile is the added complexity of inter-manager communication: managers need to communicate with each other reliably despite their ability to move. Another complexity introduced is that the NMS must collect and provide sufficient information, from which decisions about manager migration can be made. Section III-E describes how Optiprism addresses some of these concerns.

D. Scalable Interaction

An NMS must provide an application for network administrators to access network management services. This application needs to communicate with the NMS's managers so as to obtain information about the state of the network and the range of commands that may be initiated. This information would then be used to populate the application's user-interface. Scalability dictates that the application cannot expect to communicate simultaneously with *all* running managers at any time.

Optiprism provides a *browser agent* as a scalable solution to user interaction with a large hierarchical NMS. This agent is a leaf in the hierarchy of managers and may only communicate with managers that are *visible* from it. This set is defined to be the browser's peers, its supervisor's peers, its supervisor's supervisor's peers, and so on up to a configurable number of levels that we call its *horizon*. Visibility ensures a "graceful degradation of resolution" which provides the administrator with full access to parts of the network "near" the task at hand, while still maintaining a perspective on the "bigger picture".

An administrator can change the browser agent's location within the hierarchy in one of two ways: (i) *promotion* causes it to become a peer of its supervisor; (ii) *demotion* causes it to become the subordinate of one of its peers. This *logical navigation* of the browser agent causes its set of visible managers to change in a manner that corresponds to (i) zooming out and (ii) zooming in on particular regions of the network. Many browser agents can be instantiated simultaneously, to provide management from various vantage points in the hierarchy.

III. Implementation

Optiprism is implemented using a Java-based multi-agent framework called CHIME (Cellular Hierarchical Information Modeling Environment [14]), developed at the Naval Research Laboratory. Like other agent frameworks [6], [16], [21], it provides an execution environment for mobile agent code. This execution environment is called a *depot*. Every machine that is part of CHIME runs a depot. CHIME also provides a component API for agent development similar to the Java Agent Specification [1]. Notable differences between CHIME and prior frameworks include (i) intrinsic support for agent hierarchy, (ii) support for logical navigation, and (iii) enforcement of the visibility constraints (as presented in section II-D). A CHIME agent may interact with the depot in which it resides and request (i) *migration* to a different depot, (ii) *logical navigation* via promotion or demotion, or (iii) a structured directory of *visible* agents. Optiprism managers and browsers are derived from CHIME's agent classes and thus inherit the same capabilities.

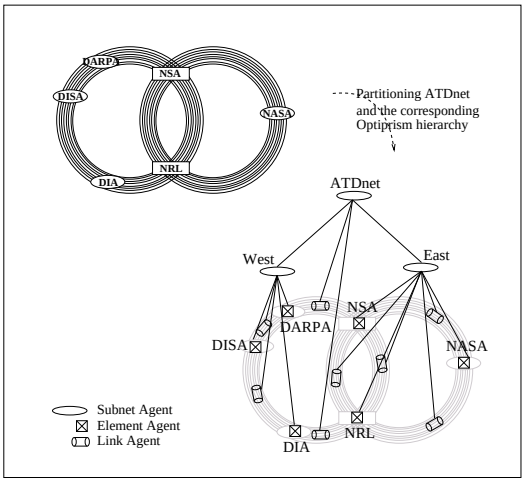

Fig. 1. Device-based network partitioning.

A. Installing Optiprism

Optiprism has been deployed and tested on the Multi-wavelength Optical Network* (MONET) switches [2] of the Advanced Technology Demonstration Network (ATDnet). ATDnet presently consists of six sites connected in the dual-homed multi-ring topology [17] (see top left of figure 1). Two of the sites (NRL and NSA) have Wavelength Selective Cross-Connect (WSXC) switches while the remaining four have Wavelength Add/Drop Multiplexer (WADM) units. Each WSXC supports four transport interfaces (TI). Each TI carries eight wavelengths using wavelength division multiplexing (WDM). The WADM units support two similar TIs. Each network element has several single-wavelength client interfaces (CCI) where the optical signal enters and exits the WDM layer.

In general, to install an Optiprism system, the network topology is determined by a network administrator, who partitions it hierarchically by assigning an Optiprism address to each network element and indicating link endpoints. Each address is a dotted sequence of unique names. An installer utility takes this description and instantiates a corresponding hierarchy of element, link, and subnet managers, distributing these in available depots. Each element manager immediately initiates a session with its corresponding physical device. The manager then uses this session for transmitting commands and receiving notifications from the device. Figure 1 shows the hierarchy for ATDnet.

B. Management Subsystems

The OSI management model categorizes network management into several functional areas. Optiprism presently addresses two areas needed in the ATDnet research environment: (i) Configuration management (CM), which addresses the problem of lightpath provisioning, and (ii) Fault management (FM), which enables monitoring of hardware alarms and alerts. Each functional area is embodied in a *management subsystem*, and a manager is then composed of a set of subsystems. Presently Optiprism subnet and element managers contain CM and FM subsystems. In the future, performance and security management subsystems will be supported.

Communication between managers takes place via delegation agents, or *deglets* (see [5]). A deglet is a lightweight agent with a transient task-based life-cycle model [11]. Optiprism defines two classes of deglets: downward flowing control deglets and upward flowing monitoring deglets.

When a subnet manager receives a request, it formulates a set of subtasks for its subordinates. Each subtask is transported to a subordinate by a *control deglet*. Upon reaching its target manager, each control deglet attempts to perform the intended subtask. The deglet then encapsulates a report of the side effects and carries this back to the initiating manager. When all the deglets have returned, the manager aggregates the reports from below into a report for the original request. Collectively, control deglets are referred to as *control flow*.

A manager may send asynchronous notifications to its supervisor by using *monitoring deglets*. Monitoring deglets encapsulate information about changes in the *beliefs* [9] of their sender. Upon reaching its target supervisor, each monitoring deglet attempts to notify the supervisor of the change in the subordinate's beliefs. The deglet then carries an acknowledgment of this notification back to the originating manager. Collectively, monitoring deglets are referred to as *monitoring flow*.

C. Configuration Management

To illustrate the operation of control and monitoring deglets, we describe how the connection management subsystem (CM) provides support for lightpath provisioning. The procedure for handling teardown requests is similar but simpler.

1) CM Monitoring Flow : CM monitoring flow takes the form of *CAT-Status* deglets. These contain a Connection Availability Table (CAT) which describes the availability of routes across a subnet/element. At the element level, the CAT is the complement of the fabric table modulo the wavelength conversion capabilities of the device. At higher levels, each subnet manager generates its own CAT by aggregating the information from the CATs of its subordinates as follows.

Each CM periodically obtains a CAT from each of its subordinates. The CM maintains two graphs: (i) a *compressed graph* that has one vertex for each of its subnet/element subordinates and one edge for each of its link subordinates, and (ii) an *exploded graph* derived from the compressed graph by replacing each link with a set of parallel edges (one per wavelength) and replacing each vertex with the CAT obtained from the corresponding subordinate. Figure 2 depicts the relationship between the compressed and exploded graphs. A vertex in the exploded graph corresponds to a *particular wavelength* on an interface advertised by some subordinate. The CM considers each pair of wavelengths λ_1, λ_2 where either (i) λ_1 is a wavelength on a border input transport interface (TI) and λ_2 is a

*MONET is sponsored by the Defense Advanced Research Project Agency (DARPA)

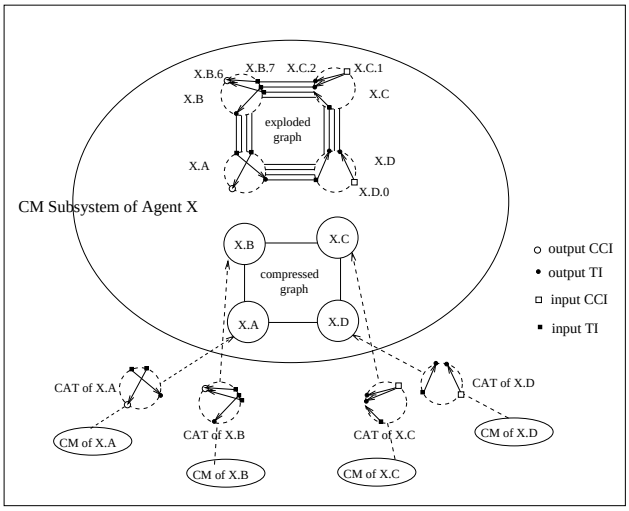

Fig. 2. CM graphs.

wavelength on a border output TI, or (ii) λ_1 is a wavelength on an input compliant client interface (CCI) and λ_2 is a wavelength on a border output TI, or (iii) λ_1 is a wavelength on a border input TI and λ_2 is a wavelength on an output CCI. For each such pair, the CM uses its exploded graph to compute a route between the corresponding vertices. If a route is found, the CM makes an entry in its *own* CAT. Once the CM has considered all such pairs λ_1, λ_2, it sends the constructed CAT upwards to its supervisor. This procedure recurses upwards.

Several schemes are used to speed up CAT aggregation. To reduce the number of computations required in CAT aggregation, access points (CCIs) are grouped based on their connectivity within that subnet. The precise criteria for determining "similar connectivity" is tunable, in order to obtain an acceptable trade-off between accuracy and computational cost. To reduce the frequency of CAT computation, a random sampling of CAT entries is recomputed periodically and used to estimate the likelihood that a new CAT would be "significantly different" from the one previously advertised. Whenever this likelihood exceeds a threshold, the entire CAT is recomputed. We also use techniques similar to those proposed for reducing routing traffic in optical OSPF [3].

2) CM Control Flow : Lightpath provisioning is achieved by CM control flow. Requests are delivered via deglets to the highest subnet manager containing both endpoints of the desired trail. From there, requests proceed recursively in parallel down the tree until they reach element managers, which create individual fabric connections in hardware. The trail partitioning process follows the guidelines of ITU-T G.805 [13]. To perform routing, each manager uses its exploded graph to determine a suitable path across the subnet. The path determines a set of lightpath provisioning subtasks that are then sent to appropriate subordinates via control deglets. Returning deglets indicate the success or failure of each subtask. A failure can result in a *fail-fast* response (i.e. rollback of any completed subtasks, and immediately report failure to the supervisor) or a *reroute* response (i.e. attempt to route around uncooperative subordinates).

D. Fault Management

The purpose of the Fault Management subsystem (FM) is to detect and diagnose network faults. We describe the roles of control and monitoring deglets in the FM.

1) FM Monitoring Flow : The monitoring flow for the FM consists of fault notifications. These are encoded in *Fault-Indication* (FI) and *Fault-Clear* (FC) deglets which convey severity, location, and type of network failure. FI/FC messages propagate upwards in the tree. Intelligent filtering is performed at each level, customized to the particular monitoring characteristics desired (e.g. severity, location, type, etc). Each FM filters and aggregates fault information received from its subordinates and passes this upward to the next higher level.

2) FM Control Flow : The control flow of the FM enables run-time configuration of the corresponding monitoring flow for an FM-enabled subnet or element manager. For example, the parameters determining the fault aggregation policy of each FM are configurable via control deglets. Similarly, control deglets are used to register Fault-Handlers inside an FM. Whenever an FM receives an FI/FC message from a subordinate, it reports this message to each registered Fault-Handler, which can then determine how to respond to the error condition.

E. Manager Communication & Mobility

Allowing managers to be mobile introduces complications to inter-manager communication. Optiprism addresses these issues by using CHIME's two-layer inter-agent communication protocol stack. The Inter-Cell Transport Layer (ICTL) provides FIFO delivery between pairs of agents, and below it, the Inter-Depot Transport Layer (IDTL) provides FIFO delivery between pairs of depots. Managers communicate via ICTL messages which are encapsulated into IDTL messages during inter-depot transit. The address of the target depot is obtained by resolving the name of the destination agent using a distributed agent look-up service. Inbound messages are unpacked and delivered to their target only if the target's name is found in the directory of local agents. Otherwise, the sending agent is blocked from further communication with the target, until its local look-up service has obtained a new binding.

Optiprism uses CHIME's Traffic Analyzer Module (TAM) and Microbenchmark Facility (MBF) to give managers information needed to make decisions about migration. The TAM maintains statistics on round-trip latency and cumulative volume of traffic from each locally resident manager to the depots with which it communicates. The MBF takes local measurements of average CPU and memory usage. A manager may use this information to determine when to request migration, and to where. CHIME follows the paradigm of "Agent proposes, Depot disposes". Either the source or the destination depot can

reject an agent's request to migrate. We are further investigating optimal criteria for (i) when managers should request to migrate and (ii) when depots should allow managers to migrate into or out of them.

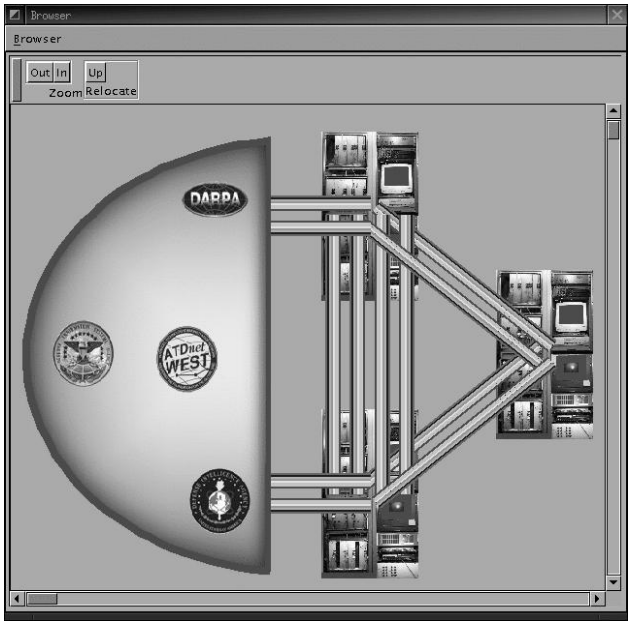

Fig. 3. The browser agent's window when it is a subordinate of the East subnet manager.

F. *The Management Browser*

The browser communicates each visible manager A by collecting a *model* of A. This model $\mathcal{M}(A)$ is an active object created dynamically by A, with functionality specialized to the capabilities of the browser agent. $\mathcal{M}(A)$ maintains a bidirectional channel to its backing manager A; this channel operates transparently to physical mobility of the manager.

The browser displays a window to the user and asks each collected model to render itself as a user-interface component within this window. The visual representation of each model depicts the state of its backing manager (e.g. network elements are rendered as images reflecting their operational characteristics). Figure 3 shows the window when the browser agent is a subordinate of the East subnet manager (see figure 1) and has obtained models of the West, NRL, NSA, and NASA subnet managers plus six link managers.

The browser agent is "featureless" except for its ability to navigate within the hierarchy. As the browser is made to navigate, it updates the set of models it owns based on visibility, and refreshes the window by requesting the models to render themselves. All other functionality comes directly from the models. This design makes it possible to perform live upgrades of manager software without altering the browser.

The user can interact with the visual representations of models to get more information or issue requests. The browser dispatches mouse clicks and key presses to the model over which they occur. The model can perform immediate action or present additional dialogs for extended input. For example, each subnet manager's model provides a dialog to select pairs of input/output connection points for trail provisioning (see figure 4). These models also offer extended FM information in a dialog that lists the outstanding fault conditions (see figure 5).

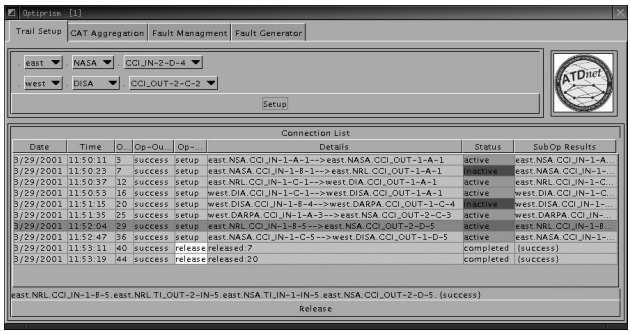

Fig. 4. Configuration management dialog.

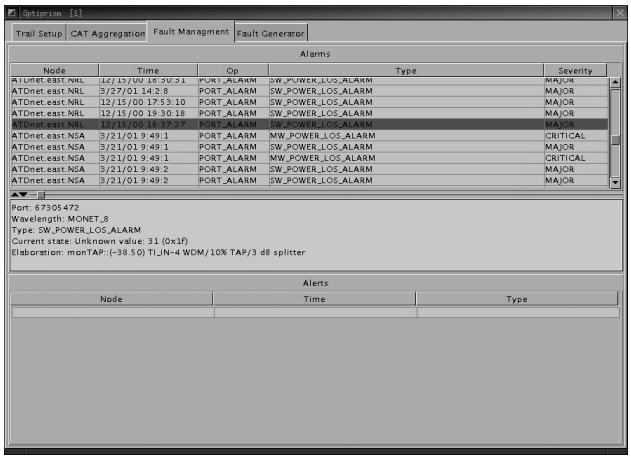

Fig. 5. Fault management dialog.

IV. **Conclusion**

Optiprism's scalable and maintainable architecture relies on the distributed deployment of a hierarchy of cooperating intelligent manager agents. By using CHIME services, managers and browsers have access to physical mobility and logical navigation. The Optiprism browser provides a management application which supports scalable interaction with NMS services. Optiprism has been successfully deployed within the ATDnet optical network.

Enhancements to Optiprism will include (i) design and implementation of the performance and security management subsystems, (ii) devising algorithms for fast CAT aggregation within the CM subsystem, and (iii) determining effective policies for manager migration, to enable the NMS to circumvent computation and communication hot-spots in its environment.

REFERENCES

[1] http://java.sun.com/aboutJava/communityprocess/jsr/jsr_087_jas.html.

[2] W. T. Anderson, J. Jackel, G.-K. Chang, H. Dai, W. Xin, M. Goodman, C. Allyn, M. Alvarez, O. Clarke, A. Gottlieb, F. Kleytman, J. Morreale, V. Nichols, A. Tzathas, R. Vora, L. Mercer, H. Dardy, E. Renaud, L. Williard, J. Perreault, R. McFarland, and T. Gibbons. The monet project–a final report. *JOURNAL OF LIGHTWAVE TECHNOLOGY*, 18(12):1988–, 2000.

[3] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi. Quality of service based routing: A performance perspective. In *Proceedings of SIGCOMM*, 1998.

[4] D. Banerjee and B. Mukherjee. A practical approach for routing and wavelength assignment in large wavelength-routed optical networks. *IEEE Journal of Selected Areas in Communications*, 14(5):903–908, 1996.

[5] A. Bieszczad, T. White, and B. Pagurek. Mobile agents for network management. *IEEE Communications Surveys*, 1998.

[6] M. Breugst, I. Busse, S. Covaci, and T. Magedanz. Grasshopper – A Mobile Agent Platform for IN Based Service Environments. In *Proceedings of IEEE IN Workshop 1998*, pages 279–290, Bordeaux, France, 1998.

[7] D. Chess, B. Grosof, C. Harrison, D. Levine, C. Parris, and G. Tsudik. Itinerant Agents for Mobile Computing. *IEEE Personal Communications*, 2(5):34–49, 1995.

[8] FIPA. Fipa network management and provisioning specification. 2000.

[9] M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge. The belief-desire-intention model of agency. In J. Müller, M. P. Singh, and A. S. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555, pages 1–10. Springer-Verlag: Heidelberg, Germany, 1999.

[10] C. Ghezzi and G. Vigna. Mobile code paradigms and technologies: A case study. In *Proceedings of the First International Workshop on Mobile Agents*, Berlin, Germany, 1997.

[11] S. Green, L. Hurst, B. Nangle, P. Cunningham, F. Somers, and R. Evans. Software agents: A review. Technical Report TCS-CS-1997-06, Dublin, 1997.

[12] L. Hurst, P. Cunningham, and F. Sommers. Mobile agents — smart messages. In *Proceedings of the 1st International Workshop on Mobile Agents*, Berlin, Germany, 1997.

[13] ITU-T. G.805 - generic functional architecture of transport networks. 2000.

[14] B. Khan, D. D. Kleiner, and D. Talmage. CHIME: The Cellular Hierarchy Information Modeling Environment. In *Proceedings of International Conference on Parallel and Distributed Computing and Systems 2000*, Las Vegas, Nevada, 2000.

[15] J.-P. Martin-Flatin and S. Znaty. A simple typology of distributed network management paradigms. In *8th IFIP/IEEE Int. Workshop on Distributed Systems: Operations & Management (DSOM'97)*, 1997.

[16] N. Minar, M. Gray, O. Roup, R. Krikorian, and P. Maes. Hive: Distributed agents for networking things. In *Proceedings of ASA/MA'99, the First International Symposium on Agent Systems and Applications and Third International Symposium on Mobile Agents*, 1999.

[17] A. Proestaki and M. Sinclair. Wavelength routing in all-optical dual-homing hierarchical multi-ring networks. In *European Conference on Networks and Optical Communications - Core Networks and Network Management (NOC'99)*, pages 52–59, Delft, The Netherlands, 1999.

[18] R. Ramaswami and K. Sivarajan. Optimal routing and wavelength assignment in all-optical networks. In *IEEE INFOCOM'94*, pages 970–979, 1994.

[19] M. G. Rubinstein and O. C. M. B. Duarte. Evaluating tradeoffs of mobile agents in network management. *Networking and Information Systems*, 2(2):237–252, 1999.

[20] Z. Zhang and A. S. Acampora. A heuristic wavelength assignment algorithm for multihop WDM networks with wavelength routing and wavelength re-use. *IEEE/ACM Transactions on Networking*, 3(3):281–288, 1995.

[21] A. Zunino and A. Amandi. Brainstorm/J: a Java framework for intelligent agents. In *Proc. of the* 2^{nd} *Argentinian Symposium on Artificial Intelligence (ASAI'2000 -* 29^{th} *JAIIO)*, Tandil, Buenos Aires, Argentina, 2000.