Towards an Agent-based Distributed Hierarchical 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 system, optical networks, multi-agent systems, distributed hierarchy, network management agents.**

Abstract— **We describe the design and implementation of Optiprism, an agent-based network management system (NMS) providing configuration and fault management services for all-optical networks. Optiprism provides support for**

1) **a scalable architecture consisting of a distributed hierarchy of intelligent mobile software agents, or managers,**
2) **the ability to adapt as the network evolves by adding, removing or upgrading managers and restructuring the hierarchy,**
3) **manager mobility services permitting reorganization of physical deployment for better responsiveness,**
4) **an innovative browser agent which provides a scalable solution to the problem of end-user interaction with a large distributed NMS.**

Optiprism has been developed and tested on the Multi-wavelength Optical Network (MONET) switches of the Advanced Technology Demonstration Network.

I. Introduction

Traditionally, network management software has been based on centralized paradigms using on SNMPv1 or SNMPv2c, or weakly distributed hierarchical paradigms using SNMPv2, RMON, CMIP, or CMIP derivatives such as TMN [24, p. 5]. While these approaches are feasible in small networks, their communication costs grow linearly with the number of devices [31, p. 4]. In many such architectures, as network size and complexity grows, the manager's workstation becomes burdened with resource intensive *micromanagement* operations [5, p. 2]. Wavelength division multiplexing (WDM) networks present additional difficulties since the central problem of routing and wavelength assignment (RWA) [30] is NP-complete [35] and even heuristic approaches to it are computationally expensive [6, 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 mobile agents [24, p. 9] or *managers* would yield significantly reduced processing requirements at the client-side.

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

The architecture of our NMS is inspired by theories of organizational hierarchies, as developed in the works of J. R. Galbraith [16], Mount and Reiter [26], Radner [29], Patrick and Dewatripont [27] and others. We draw upon the compelling analogies comparing distributed computer systems with distributed human organizations, as presented by Fox [15], et al. In particular, our NMS employs the idea of vertical integration within a hierarchy of managers: state information is condensed and flows recursively upwards at each level of the heirarchy to facilitate the analysis of state and the making of decisions. This process can result in actions which are executed by a recursive downward flow of subtasks to subordinates. Within the NMS, managers maintain aggregated information such as route availability and fault reports about recursively smaller sections of the network. The two flows facilitate both decentralized decision-making and decentralized information processing, respectively, in a manner described by Van Zandt [34, pp. 1]. Because the network's state is hierarchically distributed, management applications do not need to establish direct connections to every network element. Instead, the administrator interacts with high-level supervisory managers. We draw on Galbraith's mechanistic model of organizational design theory, incorporating the strategy of permitting *lateral relationships* between managers across groups. This model of planning achieves integrated action and reduces the need of continuous communication between interdependent sub-units. Within the NMS, control operations, such as lightpath provisioning, are issued to the high-level managers, who then compute routes and delegate partitioned connection requests to their subordinate managers. Monitoring of alarms and alerts operates in the reverse direction: subordinate managers report fault conditions to their supervisor. Depending on the task, the management application communicates with some subset of the managers to monitor and manipulate the network. The next sections describe the design and implementation of the Optiprism network management system.

II. Design

In designing Optiprism, we adopted a distributed architecture because it enabled us to meet four important objectives. The most critical of these is *scalability*. In large networks, the processing of management requests (e.g. route selection, alarm filtering) 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 [19, p. 1]. Second, a distributed architecture is *maintainable* because it is easier to extend and adapt when the network evolves, without requiring downtime for NMS reconfiguration. Third, a distributed architecture permits computations to be closer to information sources, reducing latency and total control traffic [12] [21], thereby yielding better *responsiveness*. This benefit is amplified if the architecture supports dynamic re-distribution of managers (e.g. using agent mobility) since then the NMS could be adapted to circumvent computation and communication hot-spots in its environment [20, 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-sized subset of the NMS at any given time. We now describe how the design of Optiprism strives to meet these objectives.

A. Scalability

An effective optical NMS must be 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 may 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 a high-level interface to the services it can implement using the functionality of its subordinates. Two managers are called *peers* if they have the same supervisor.

Optiprism thus consists of a logical hierarchy of managers whose members are physically distributed across the agent environments (see figure 2). Optiprism takes a *device-oriented* [4] approach to hierarchical network partitioning (as opposed to *traffic-oriented* approaches such as those explored in [32] and [33]). Each manager encapsulates the capabilities of its subordinates managers by (i) *aggregating* their notifications, and

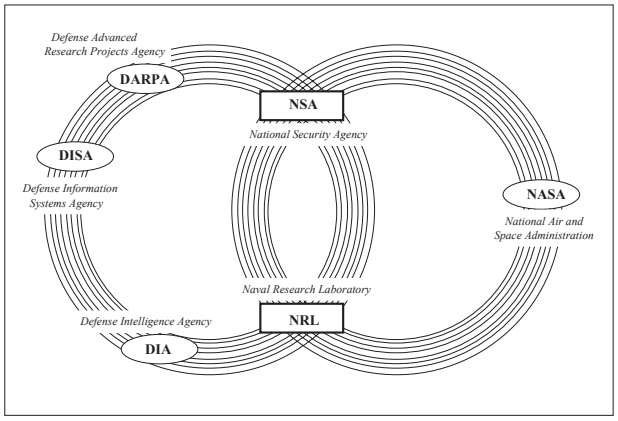

Fig. 1. The ATDnet dual-homed multi-ring topology.

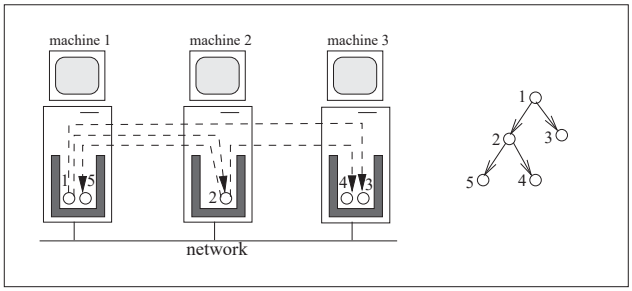

Fig. 2. Logical agent hierarchy deployed in physical execution environments.

(in the spirit of management by delegation [11, pp. 12]) by (ii) *delegating* control requests to them. As an example of (i), a manager aggregates reports of faults from subordinates; as an example of (ii), a manager partitions a given lightpath provisioning task into a sequence of requests for smaller constituent connections.

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 having greater memory and processing power. We are investigating the possibility of addressing both these issues through automatic replication and clustering of high-level managers.

B. Maintainability

The NMS architecture should be easy to alter when the network evolves. In a hierarchical NMS, this would be achieved by the addition and removal of managers, and by dynamic 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 element manager controls and monitors a physical device using 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 required to be functionally identical, regardless of the manager's level. Without this uniformity, the NMS would have to address difficult problems involving dynamic discovery and mapping of manager interfaces.

Requiring all subnet and element managers to implement the same interface makes it easy to add and remove managers in an existing Optiprism hierarchy at run-time. Supporting a new type of hardware device involves developing a specialized element manager that implements the command and notification

interfaces in terms of the device's proprietary protocols. A new element/subnet manager may be inserted into the hierarchy at run-time, making it possible for Optiprism to evolve with the network, without implying downtime for the NMS. Adopting symmetric interfaces has also yielded benefits of simplicity in the implementation and interactions of managers, and provided encapsulation at the manager level. Optiprism's subnet managers are truly "virtual optical switches".

One issue with this approach is that element managers for new (more feature-rich) 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. One such standardization attempt in the agent research community is the FIPA Network Management and Provisioning Specification [13].

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 [8], all Optiprism managers can be thought of as *netlets*, because they have a persistent process-based life-cycle model [20].

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 their impact on the NMS. This requirement underscores the need to support agent mobility [20, pp. 26-33] as a core feature of the NMS.

Allowing managers to be mobile does introduce some complications to the NMS. First, it adds to the complexity of inter-manager communication: managers need to communicate with each other reliably despite their ability to move. In addition, the NMS must collect and provide sufficient information on the basis of which decisions regarding migration may be made. Sections III-E and III-F describe how Optiprism begins to address some of these concerns.

D. Scalable Interaction

A distributed NMS must provide an application for network administrators to access network management services. This application needs to communicate with the NMS's managers in order to obtain information about the state of the network and the range of commands that the administrator may initiate. This information could then be used to populate the application's user-interface. Scalability arguments dictate that the application cannot expect to communicate simultaneously with *all* running managers at any time.

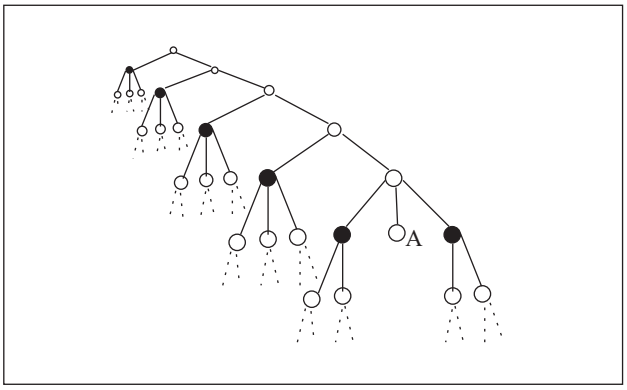

Fig. 3. Visibility for graceful degradation of control resolution.

Optiprism provides a *browser agent* as a scalable solution to user interaction with a large hierarchical distributed NMS. The browser agent is a leaf in the hierarchy of managers and only communicates 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" that 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". Figure 3 depicts a hypothetical hierarchy and distinguishes the agents that are visible to browser agent *A* by shading them black.

In the language of organizational design theory, the browser agent is considered to be a "consultant" whose position within the management hierarchy can be changed at will. This browser can establish lateral relationships with a limited set of managers relevant to its current position in the tree, and can accomplish tasks by issuing requests to them. For example, since network state is aggregated as information flows upwards (see II-A), to provision a lightpath the browser only needs to contact the highest level manager of the logical subnet containing both endpoints of the desired trail. This manager will have suitable information in order to partition the trail request into lower-level subnet cross connections. Despite having no *a priori* knowledge of the actual network topology within a logical subnet, a network administrator can use the browser to effect provisioning across that subnet.

An administrator can change his/her browser agent's location within the manager 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), developed by the authors [23] at the Naval Research Laboratory. Like other agent frameworks [10], [25], [36], 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. Like the Java Agent Specification [1], CHIME provides a component API for agent development. Notable differences between CHIME and prior agent 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 abstract agent classes, and thus inherit these same capabilities.

A. Installing Optiprism

Optiprism has been developed 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 [28] shown in 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. In addition, each network element has several single-wavelength client interfaces (CCI) which are access points where the optical signal enters and exits the WDM layer.

In general, to install an Optiprism system, the network topology is determined and partitioned hierarchically by assigning an Optiprism address to each network element and indicating link endpoints†. Each Optiprism 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 across the set of available depots. Each element manager immediately initiates a session with its corresponding physical device (e.g. a WSXC switch).

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

†Currently on ATDnet, this process is carried out manually by a network administrator, but we expect to support dynamic topology discovery and assisted partitioning in the future, when we integrate Optiprism with the Toolkit for Routing in Optical Networks (TRON) [9].

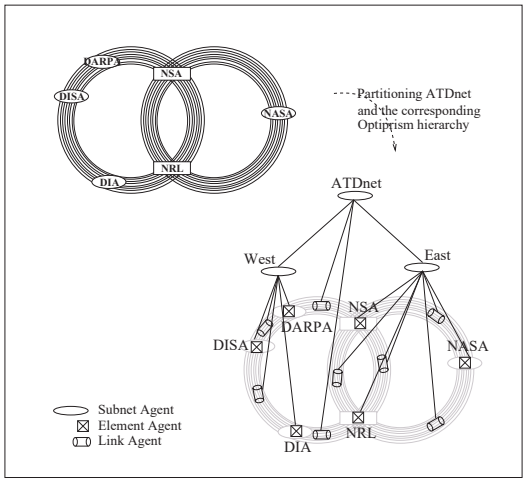

Fig. 4. Device-based network partitioning.

The element manager subsequently uses this session for transmitting commands and receiving notifications from the device. Figure 4 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*; a manager is constructed by composing a wrapper object with a set of subsystems. Presently Optiprism subnet and element managers contain CM and FM subsystems. In the future, performance and security management subsystems will also be supported.

Communication between managers takes place via delegation agents, or *deglets* (see [8]). A deglet is a lightweight agent with a transient task-based life-cycle model [20]. 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 *plan* consisting of subtasks for its subordinates. Each subtask is transported to a suitable subordinate as 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, the downward-flowing control deglets faciltate decentralized decision-making and 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* [18] of their

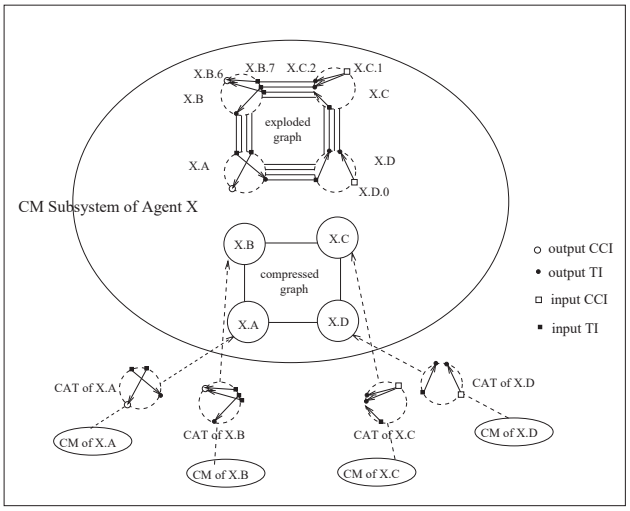

Fig. 5. CM graphs.

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, the upward-flowing monitoring deglets faciltate decentralized information processing and 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 manager level, the CAT is the complement of the physical device's fabric table, computed 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 according to the procedure that follows.

Each CM periodically obtains a CAT from each of its subordinates. The CM maintains two graphs:

- A *compressed graph* that has one vertex for each of its subnet/element subordinates and one edge for each of its link subordinates, and
- 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 5 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 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 a corresponding 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. CAT aggregation recurses upwards, operating in a manner similar to topology aggregation [7] in PNNI [14].

Optimizations: Several schemes are used to minimize the total computational cost of CAT aggregation.

- To reduce *the cost of each CAT aggregation*, access points (CCIs) are grouped together, based on their connectivity within the subnet. A tunable criterion is used to determine when two access points have connectivity that is similar enough to warrant grouping them together. Adjusting this policy permits selection of the trade-off point between accuracy and computational cost.
- To reduce *the frequency of CAT aggregation*, 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 apply timer and threshold based techniques similar to those proposed to reduce flooding of routing information 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, control deglets 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 [22], as depicted in figure 6. To perform routing, a manager uses its exploded graph to determine a suitable path across the subnet. The path then determines a set of subtasks that are sent to appropriate subordinates via control deglets. Returning deglets report the success or failure of each of the subtasks.

A failure report from a subtask is indicative of a significant discrepancy between the beliefs of the higher-level manager and the underlying reality of one of its subordinates. Stated from the subordinate's perspective, if its supervisor has requested a connection for which no route was found, then this is evidence that the supervisor is using an outdated CAT which no longer accurately reflects wavelength availabilities within the subordinate. The frequency of such failures is recorded by the subordinate, and used to trigger on-demand CAT re-aggregation. This process operates independently of the CAT

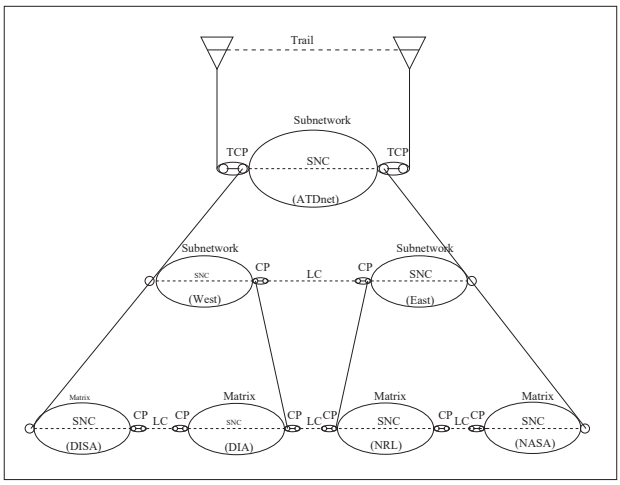

Fig. 6. Trail partitioning of a connection from NASA to DISA in ATDnet.

aggregation optimizations described in the previous section.

If a subnet manager receives a failure report from one its subordinates during routing, the subnet manager may take either

- a *fail-fast* response, where any completed subtasks are rolled back using teardown requests, and a failure report is sent upward to the supervisor, or
- a *reroute* response, where some number of attempts are made to re-route around uncooperative subordinates. If these all fail, a report is sent to the supervisor indicating failure.

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 deglets propagate upwards in the tree. Intelligent filtering is performed at each level, customized to the particular monitoring characteristics of interest (e.g. severity, location, type, etc). Each FM filters and aggregates fault information received from its subordinates and passes it upward to the next higher level in new monitoring deglets.

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 deglet from a subordinate, it permits each registered Fault-Handler to interact with the deglet, to determine if and how to respond to the error condition.

E. Manager Communication

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 for 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 agent only if the target's name is found in the depot's directory of local agents. Otherwise, the sending agent is blocked from further communication with the target, until the look-up service at the sender has obtained a new binding for the target.

F. Manager Mobility

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 housing the managers with which it communicates. On the other hand, the MBF takes local measurements of average CPU and memory usage. A manager may use information from the TAM and MBF in order 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 may reject an agent's request to migrate. In one of our testing scenarios, the receiving depot refuses if, upon consultation with its local MBF, it determines local CPU/memory usage exceeds a configurable threshold. In contrast, the sending depot refuses to transfer the manager if it determines that the average number of manager departures per unit time is in excess of a configurable threshold (this dampening policy was used to prevent ping-ponging of managers between depots in high-load settings). We are at present investigating good criteria for (i) when managers should request to migrate and (ii) when depots should allow managers to migrate into or out of them. These investigations will be the subject of a later report.

G. The Management Browser

The browser communicates with a visible manager A by collecting a *model* of A. This model $\mathcal{M}(A)$ is an active object created dynamically by A, with interfaces and functionality specialized to the capabilities of the browser agent. $\mathcal{M}(A)$ maintains a bidirectional channel with its backing manager A. The operation of this channel is transparent to physical mobility of A. Injection of model $M(A)$ into the browser represents the beginning of an active connection between the browser and A. As a consequence, the browser gains access to aspects of A, with

the model $\mathcal{M}(A)$ acting as mediator [17] between the two (see figure 7).

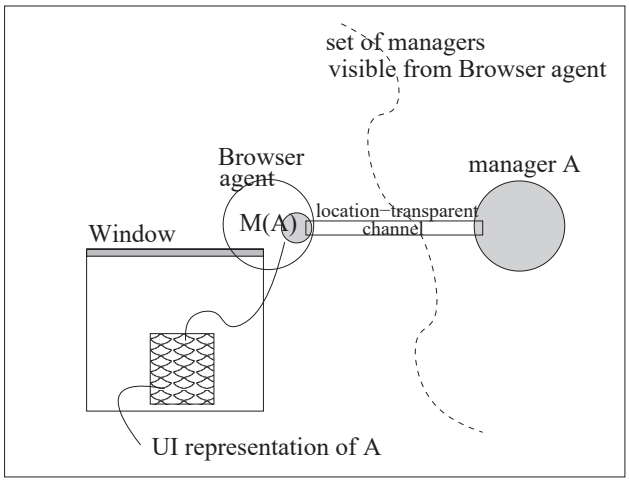

Fig. 7. How the browser permits models of visible managers to render themselves.

The browser displays a window to the user, and requests each collected model to render itself as a user-interface component within this panel. For example, a subnet manager's model might render itself as a simple icon, while an element manager's model might render itself as a more complex image reflecting current operational characteristics like status of ports. The visual representation of each model embodies the state of its backing manager (e.g. color-coding for the presence of faults, etc). Figure 8 shows what the browser window looks like when the browser agent is a subordinate of the East subnet (see figure 4), and has obtained models of the managers that are visible from there (i.e. West, NRL, NSA, and NASA subnet managers, and 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 collects models from visible managers, and refreshes the window by requesting these 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 each model in order to get more detailed information (e.g. examining faults) or to affect control flow based requests (e.g. provisioning connections). The browser dispatches mouse clicks and key presses to models, based on where in the window the events occur. The model can then 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 9), and offers extended FM information in a dialog that lists the outstanding fault conditions (see figure 10).

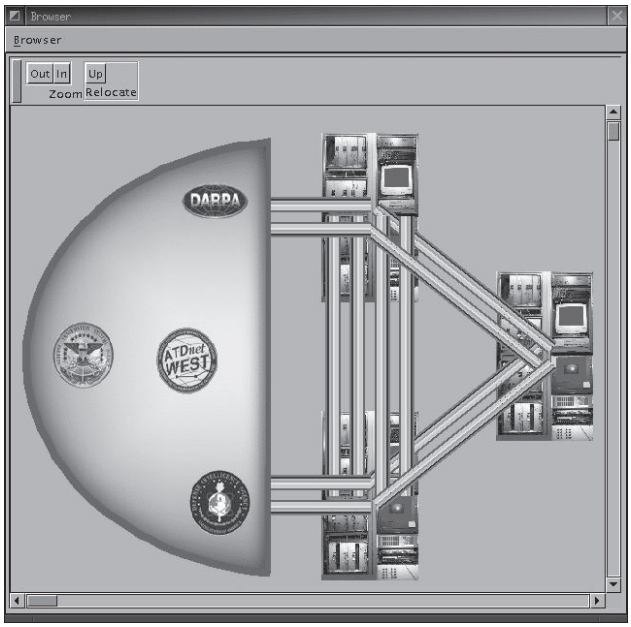

Fig. 8. The browser window when the browser agent is a subordinate of the East subnet manager.

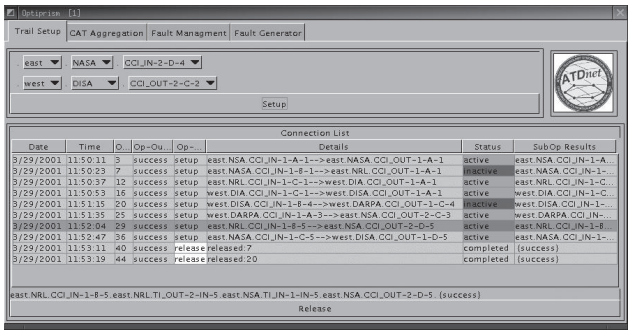

Fig. 9. Configuration management dialog.

IV. **Conclusion**

Optiprism's scalable and maintainable architecture is built using a distributed deployment of a hierarchy of cooperating intelligent mobile 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 developed and successfully deployed within the ATDnet optical network.

Future enhancements and research using 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, (iii) determining effective policies for manager migration, to enable the NMS to circumvent computation and communication hot-spots in its environment, and (iv) evaluating fault-tolerance schemes based on replicating high-level managers into clusters.

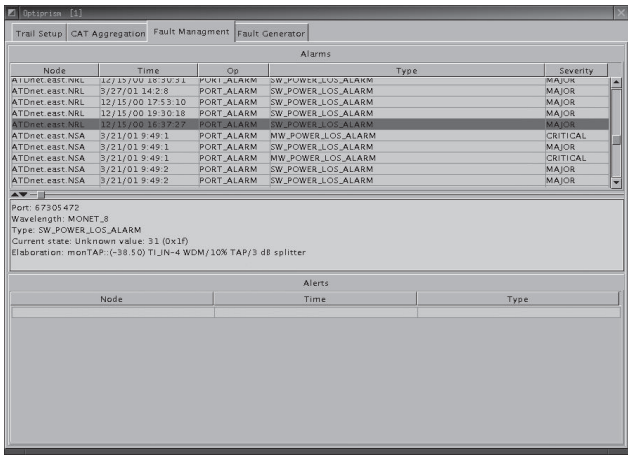

Fig. 10. Fault management dialog.

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] M. Baldi, S. Gai, and G. P. Picco. Exploiting code mobility in decentralized and flexible network management. In *Proceedings of the First International Workshop on Mobile Agents*, pages 13–26, Berlin, Germany, 1997.

[5] M. Baldi and G. P. Picco. Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications. In R. Kemmerer, editor, *Proceedings of the 20^{th} International Conference on Software Engineering*, pages 146–155. IEEE CS Press, 1998.

[6] 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.

[7] A. Battou, K. Bhutani, and B. Khan. Two approaches for aggregation of peer group topology in hierarchical pnni networks. *International Journal of Intelligent Automation and Soft Computing*, 2000.

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

[9] G. B. Brahim, B. Khan, A. Battou, M. Guizani, and G. Chaudhry. TRON:the Toolkit for Routing for Optical Networks. In *submitted to GLOBECOM 2001*. IEEE.

[10] 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.

[11] M. M. Cheikhrouhou, P. Conti, and J. Labetoulle. Intelligent agents in network management, a state-of-the-art. *Networking and Information Systems*, 1(1):9–38, 1998.

[12] 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.

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

[14] A. Forum. Private network-network interface specification. (Version 1.0), 1996.

[15] M. S. Fox. An organizational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):70–80, 1981.

[16] J. R. Galbraith. *Organization Design*. Addison-Wesley Publishing Company, 1977.

[17] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns — Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman, 1995.

[18] 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.

[19] 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.

[20] 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.

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

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

[23] 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.

[24] 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.

[25] 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.

[26] K. Mount and S. Reiter. A model of computing with human agents. Technical Report 890, Evanston, IL, 1990.

[27] B. Patrick and M. Dewatripont. The firm as a communication network, 1994.

[28] 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.

[29] R. Radner. The organization of decentralized information processing, 1993.

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

[31] 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.

[32] S. Waldbusser. Remote network monitoring management information base. *Request For Comment 1757*, 1995.

[33] S. Waldbusser. Remote network monitoring management information base version 2 using smiv2. *Request For Comment 2021*, 1997.

[34] T. V. Zandt. Real-time hierarchical resource allocation.

[35] 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.

[36] 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.