Routing Patterns for Self-Management

Constantin Adam and Rolf Stadler
Laboratory of Communication Networks
Department of Microelectronics &
Information Technology
KTH Royal Institute of Technology
Stockholm, Sweden
{ctin, stadler}@imit.kth.se

May 1, 2003

Abstract

This paper contributes towards engineering self-managing systems, starting from known routing schemes. Routing schemes are self-managing, because they create and maintain a set of distributed states in a dynamic environment, enabling service delivery under various constraints. Using the approach of separating the update propagation from the routing table computation in routing protocols, we construct a navigation pattern, which underlies DUAL and OSPF, two known Internet routing protocols. This pattern, called progressive wave, defines a generic information distribution mechanism. We instrument it and show how to collect global statistics about any mechanism using this pattern, and apply these statistics to performance monitoring or troubleshooting. In particular, we demonstrate how the propagation of waves carrying routing updates can be monitored and analyzed. We argue that the progressive wave pattern can be used in self-organizing systems that locate resources under constraints – on the network level, as well as on the service level.

1. Introduction

Network management technology is constantly evolving, adapting to the advances and diversity of networks. The first generation of management systems was based on a centralized architecture and built for traditional telecom networks, as well as IP-based computer networks. Over the last ten years, networking and end systems technology evolved dramatically. Network sizes increased by many orders of magnitude, multiple services were introduced on the same networking infrastructure, a large percentage of the nodes became mobile, and the processing capability of the nodes increased tremendously.

To cope with these changes, the need for decentralizing management tasks has been recognized [1, 2, 3], and several approaches have been developed [4, 5]. In addition to that, an important aim of current management research is to restrict the role of human

operators to defining management policies, while the management system carries out automatic configuration of services and resources, fault isolation, diagnosis and repair, as well as reactions to any other type of events, while conforming to the operators' policies.

Routing systems are well-understood examples of self-managing systems. They create and maintain the distributed network state in the form of routing tables, which allows service delivery under certain constraints, such as transporting packets along shortest paths and balancing the network load. Routing systems dynamically adapt to changes in the network configuration, link state (link cost), and various network faults.

Our current work aims at developing generic self-management and resource management functions, starting from known routing schemes. Following some of our earlier work in decentralized management [6, 7, 8], we use the approach of separating information propagation in the network from information processing in the nodes. In the context of routing, this means that a routing update consists of two parts: message processing and message propagation. The message processing on a node handles the data received from neighboring nodes and modifies the routing table entries if needed. The message propagation part decides where to forward the updated information. The propagation part is encapsulated in an object which we call a *navigation pattern*.

The study of the two main families of unicast Internet routing protocols, based on distance-vector and link-state algorithms, shows that, while most of these protocols rely on a flooding scheme to propagate information, there is no separation in their specification between the data processing and the data forwarding functions. In our work, we have examined and implemented a distance-vector (DUAL or EIGRP [9]) and a link-state (OSPF [10, 11]) routing protocol. We found that both protocols can be built using the same navigation pattern. In the implementation of DUAL or OSPF, different information is propagated in the network using the same pattern, which can be presented in a very compact form.

In this paper, we present this navigation pattern, which we call *progressive wave*. We instrument it and show how global statistics about any mechanism using this pattern can be gathered and used for performance monitoring or troubleshooting purposes. In particular, we demonstrate how the propagation of waves carrying routing updates can be monitored. We argue that the progressive wave pattern can be used in self-organizing subsystems that facilitate locating resources under constraints, on the network level, as well as on the service level.

The paper is organized as follows. Section 2 describes the concept of the navigation pattern and motivates its use for this research. Section 3 presents the progressive wave pattern for Internet routing protocols. Section 4 gives an example of how this pattern can be instrumented to make it manageable. Section 5 outlines further applications for this pattern. Section 6 discusses the results of the paper and outlines future work.

2. Pattern-Based Routing

We base our approach to implementing routing protocols on the concept of navigation patterns (which is different from that of software patterns). The realization of a decentralized network management framework using patterns has been extensively covered in [6, 7, 8]. The idea behind the pattern-based network management is to separate the distributed flow of control messages from the local operational semantics in a management task. Every management operation contains two independent algorithms: a generic graph traversal algorithm, called navigation *pattern*, and a local algorithm which implements the computations required by the task, called *aggregator*.

When a management operation is launched, it is injected into the network at a particular node. The navigation pattern propagates it through the network (generally in a parallel fashion), and the aggregators compute partial results at each node. After the navigation pattern has visited all the nodes involved in a particular management operation, it returns the final result to the management station and terminates.

For example, assume we want to find the node with the highest CPU utilization in a tree network. There are several ways to visit all the nodes in the tree. One way is to apply a sequential Breadth-First Search. Another way is to apply a sequential Depth-First Search. A parallel way of propagating the operation through the tree is to use an echo pattern [6], where each node in the tree sends a query to its children. When a query reaches a leaf node, it turns into an answer and it is returned to the parent. When a parent has received the replies from all its children, it sends a reply to its own parent. An aggregator instance runs on each node. The aggregator compares all the CPU utilizations received by that node, decides which one is the highest, and the pattern will forward the result. Note that the aggregator is unaware of the way in which the information travels in the network.

Routing is a good candidate for pattern-based design, since it is based on a distributed algorithm that runs on every network node and can be separated into two basic functions: (1) computing paths and maintaining the local routing information, and (2) propagating update information in the network. (A third function, often considered to be part of routing, which includes forwarding the packets from the input links to output links of a node according to its routing table, is not of interest for this paper.) During the path computation phase, each node calculates, using its local information, the best way to reach every other node in the network, taking into account the routing objective, *i.e.* minimum hop, lowest cost, or maximum throughput. During update propagation, the nodes exchange information about changes in network configuration and in link states. Developing a pattern-based design for a specific routing protocol thus means finding a pattern that captures the update propagation and defining an aggregator that encapsulates that path computation and maintenance of local information.

There are two main differences between the pattern-based routing and the pattern-based network management algorithms presented in [7, 8]. First, in the case of routing, the events that activate a computation are triggered from inside the system, rather than from a management station. Second, once a routing update has been triggered and propagated

within the network, the pattern will not terminate, but remain dormant on the network nodes, waiting for events that will re-activate it.

3. A Pattern for Internet Routing Protocols

The routing protocols currently used in the Internet are based on shortest path algorithms that can be classified as either distance-vector or link-state algorithms [9]. A distance-vector algorithm, also called distributed Bellman-Ford, works as follows: every node knows the length of the shortest path from each of its neighbors to every other node in the network and uses this information to compute the shortest path and the next hop to each destination. Examples of distance-vector routing protocols include BGP, RIP, DUAL or EIGRP, and Netchange [12, 13, 9, 14]. In a link-state algorithm, every node knows the entire network topology and computes the shortest path to each other node. Examples of link-state routing protocols are OSPF and OSI IS-IS [10, 15].

For this work, we studied in detail one protocol of each class. From the distance-vector class, we chose DUAL, because it addresses in the most systematic and rigorous way the problems inherent to the distance-vector approach, namely, the possible formation of transient routing table loops and counts to infinity. From the link-state class, we chose OSPF, as the most widely used link-state protocol in the Internet. (For the purpose of this paper, we restrict the discussion of OSPF to its function within a single Autonomous System and exclude so-called designated routers.)

In both of the above protocols, three classes of events trigger a routing table update and propagation of routing information: a link cost decrease, a link cost increase and an internal event, mostly a timer. In the context of shortest path routing, adding a link can be seen as decreasing the link cost from infinity to a finite value; while a link failure can be viewed as a link cost increase from a finite value to infinity. Moreover, adding a node is equivalent to adding a set of links, and a node failure is equivalent to a set of link failures.

If a link failure isolates one node from the rest of the network, the routing update will propagate through the network like a progressive wave (see Figure 1). If a link failure breaks the network into two disconnected parts, then an update will propagate inside each disjoint network part. In case a link failure does not partition the network, several updates will be propagating through the network at the same time. These updates will not interfere with each other, because each of them will affect different parts of the routing tables. A node will stop the further propagation of an update, if that update will not trigger a change of the local routing information. In the case of a link cost decrease, two separate updates will be initiated at both ends of the link. In the case of an internal event, the node on which the event occurred will start the update.

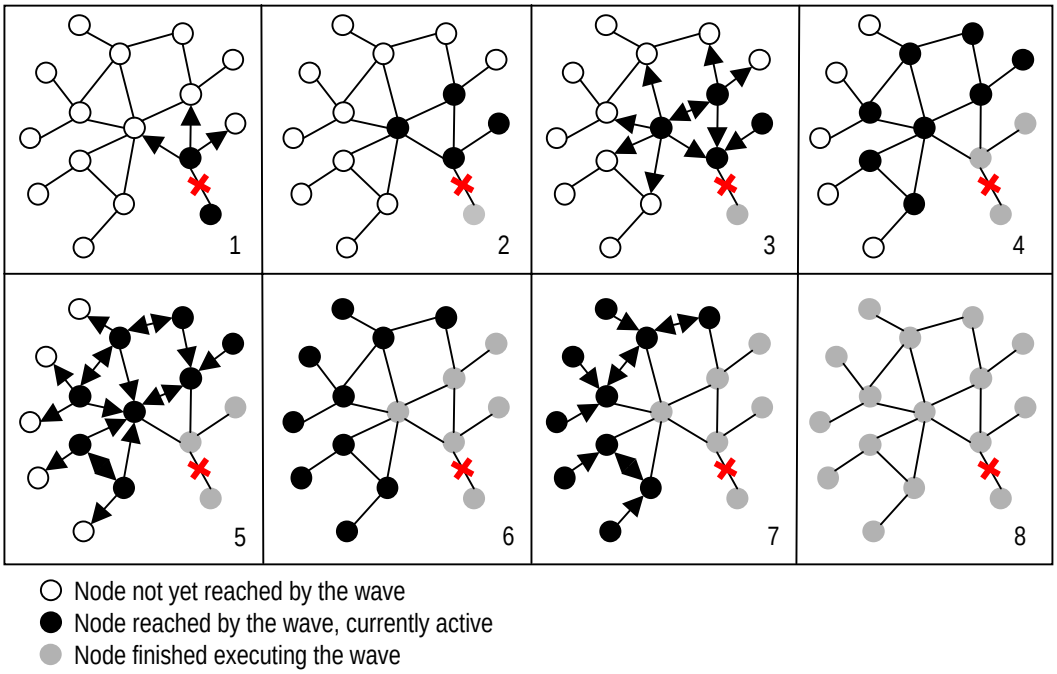

Figure 1: Progressive wave propagation.

The above discussion motivates us to interpret the propagation of routing information in the network as a set of *progressive waves*, originating from different nodes in the network, expanding for some time and then disappearing. Since patterns (see Section 2) describe the propagation of information in a distributed operation, we have attempted to identify patterns that describe these progressive waves. In the course of our investigation, we discovered that a single pattern defines the progressive waves for both DUAL and OSPF. For obvious reasons, we call this pattern *progressive wave*. Figure 2 gives the pseudo-code and Figure 3 the state machine of this pattern.

```
Progressive_Wave::run( in_msg: Message, from: Node ) {

my_address := get_local_address();
initialized := false;
G := get_neighbors();

If from = undefined
  Aggregator->Start( in_msg );

If initialized = false {
  Aggregator->Initiate( in_msg );
  Aggregator->Wait( in_msg, out_msg );
  Send_message( my_address, out_msg );
}

If from = my_address {
  Aggregator->Wait( in_msg, out_msg );
  Send_message( my_address, out_msg );
  Aggregator->Originate_Wave( in_msg, out_msg );
    If out_msg != NULL
```

```
      Send_message( G, out_msg );
}
else {
  Aggregator->Reply( in_msg, out_msg );
  If out_msg != NULL
    Send_message( from, out_msg );
  Aggregator->Propagate_Wave( in_msg, out_msg );
  If out_msg!= NULL {
    G_from := G – from;
    Send_message( G_from, out_msg);
  }
}
```

Figure 2: Pseudo-code of progressive wave.

The local state machine of a node running a routing algorithm is presented in Figure 3.

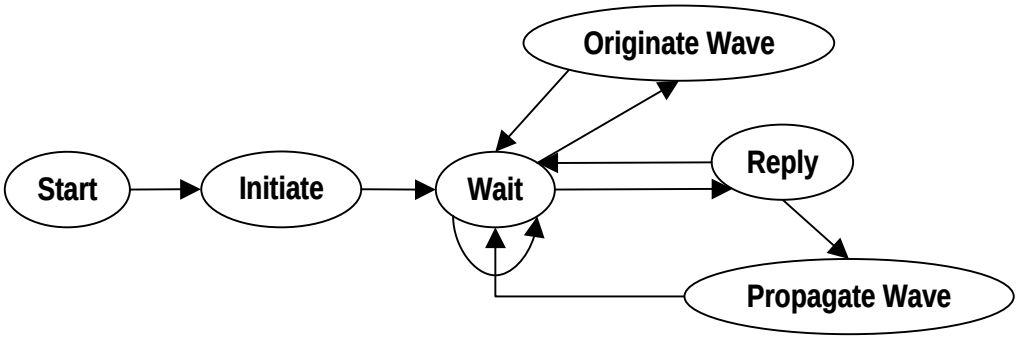

Figure 3: Local state machine of progressive wave.

Following the pattern approach, the operations shown in Figures 2 and 3 (`Start, Initiate, Wait, Originate Wave, Reply, Propagate Wave`) are defined and implemented as methods in the Routing Aggregator object. An aggregator interprets the updates received from its pattern, maintains the local routing information, and builds the new updates that the pattern will further propagate. As the data structures used in the above operations for DUAL and OSPF are different, their aggregators capture the identifying characteristics of these respective protocols. The same event can change the routing information of certain network nodes in one protocol, but not in the other. For example, adding a link that does not change any of the shortest paths in the network will not trigger any update in DUAL, but it will cause updates to the routing information of all the network nodes in OSPF.

We have implemented pattern-based versions of OSPF and DUAL on the SIMPSON simulator [16] and are currently porting them to Weaver [6], our experimental networking platform. While the pattern code is compact, the aggregator code is more complex: about 500 lines of C++ code for DUAL and about 400 C++ lines for OSPF. We have performed a functional validation of our protocol implementations on the simulator, by running a series of experiments on different topologies. We looked at the following scenarios: initialization, sequences of link and node failures and recoveries with constant

and exponential mean times to failure and mean times to recovery. We verified that for each scenario the routing tables, the estimation table for DUAL and the topology database for OSPF are computed correctly.

4. Managing Routing by Instrumenting the Pattern

The routing pattern described in Section 3 can be instrumented for management purposes. This opens up an array of possibilities for monitoring routing systems in various ways and troubleshooting them. Note that by instrumenting the pattern every routing system based on this pattern "inherits" these additional capabilities. This section is devoted to illustrating some of the capabilities that an instrumented progressive wave pattern can provide. In Section 5, we discuss another potential benefit of the progressive wave, namely to use it as a generic building block for various resource discovery and location schemes, on the network, as well as the service level.

In our work, we instrumented the pattern in a way that allowed us to generate data to monitor the dynamics of the waves triggered by routing updates. Since each progressive wave has a unique originating node, this node assigns a globally unique identifier to each wave launched from it. The wave propagation is tracked by having the pattern write into the local state of each node it passes a timestamp, the wave id, and the distance from the originating node, *i.e.*, the number of hops the wave has expanded before reaching the current node. Note that this instrumentation will not increase the total number of messages exchanged in the system, but only their size.

The information in the local node state can be used to compute a number of local and global statistics that reflect the routing traffic and its dynamics. The local statistics include the number of waves that reaches a node in a given time interval, and the number of times a specific wave passes a node. The global statistics are obtained by aggregating local information from each node. Examples of global statistics are the maximum number of hops traversed by a wave and the number of waves propagating in the network at a specific time.

These statistics can be used to trace a single routing update and see how far it propagates in the network. Alerts can be triggered when abnormal wave propagation is detected (*e.g.* the number of hops traversed by the wave exceeds a certain threshold). The records that log past activity at different nodes can be retrieved, providing the capability to playback routing behavior during a certain time interval. For example, one can see when a failure occurred, and how the routing dynamically changed as a consequence of this failure event. If the capabilities exist to dynamically collect and aggregate such statistics in the network (as can be done on our Weaver testbed [6]), it becomes possible to compute and visualize in near real-time the propagation of the waves and the changes of routing inside the network.

We have instrumented the pattern on the SIMPSON simulator [16], and are in the process of porting it to the Weaver platform, a testbed for pattern-based management [6]. On the

SIMPSON platform, we have observed DUAL and OSPF for different network topologies and have gathered statistics for a variety of failure scenarios. While it is obviously easier to implement such functionality and perform experiments on the simulator, we are confident that we can perform the same kind of investigation on the Weaver testbed. The two main problems encountered when moving from the simulator to the network testbed, namely gathering and aggregating the local statistics and synchronizing clocks, can be addressed. Synchronizing clocks in a network within millisecond resolution can be achieved by known techniques. The echo pattern implemented in the Weaver platform can collect and aggregate the local statistics [7, 8].

As an illustration of the type of statistics we can produce with the instrumented progressive wave pattern, we present graphs showing the number of waves over time and the distribution of waves according to their diameter for an 18 node grid network during the simulation of initialization and failure scenarios.

The initialization begins when the routing protocol is started on the first node. Each node initializes its internal routing structures when it receives the first routing message, and afterwards exchanges routing messages with its neighbors. The initialization phase stops when all the nodes reach a stable state, in which their routing tables are synchronized. We simulate then a series of link failures and recoveries. The mean time to failure and mean time to recovery for each link are modeled as exponential variables with different means.

Figure 4 shows the number of waves propagating in the network during the initialization phase for both OSPF and DUAL. We have added to Figure 4 the percentage of nodes that have been reached by the algorithm at a given time. We see that the maximum activity occurs shortly after the last node is reached by the routing pattern.

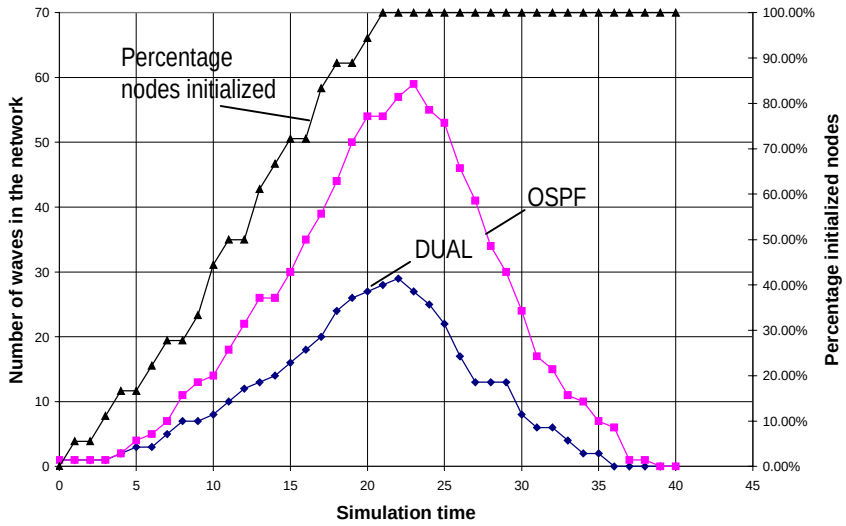

Figure 4: Number of waves in the network during initialization.

The distribution of waves according to their diameter is a global statistics that provides information about traffic locality. The diameter is the maximum number of hops a wave

propagates in the network from its origin before extinction. Figure 5 shows this distribution for the initialization phase.

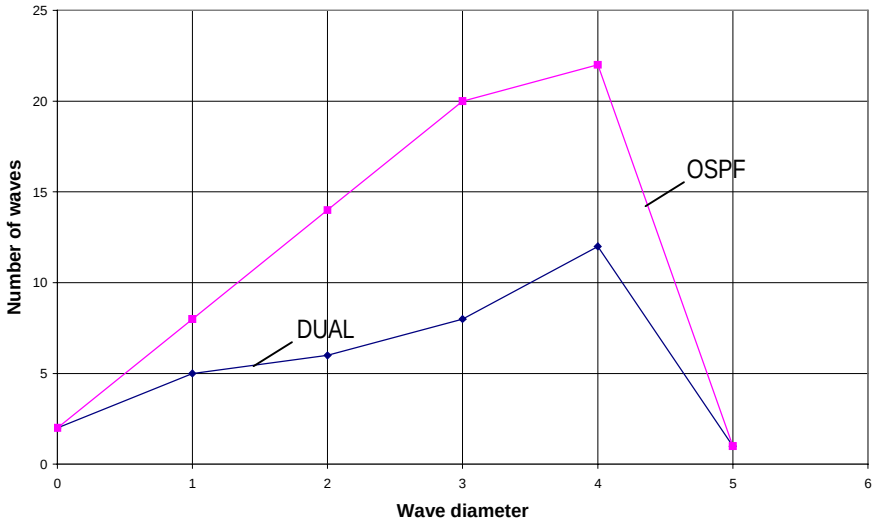

Figure 5: Distribution of waves according to their diameter during initialization.

Figures 6 and 7 represent the number of waves propagating in the network and the distribution of waves according to their diameter during a series of link failures and recoveries. Figure 6 also plots the number of link failure events in time, showing how link failures or recoveries affect the routing traffic.

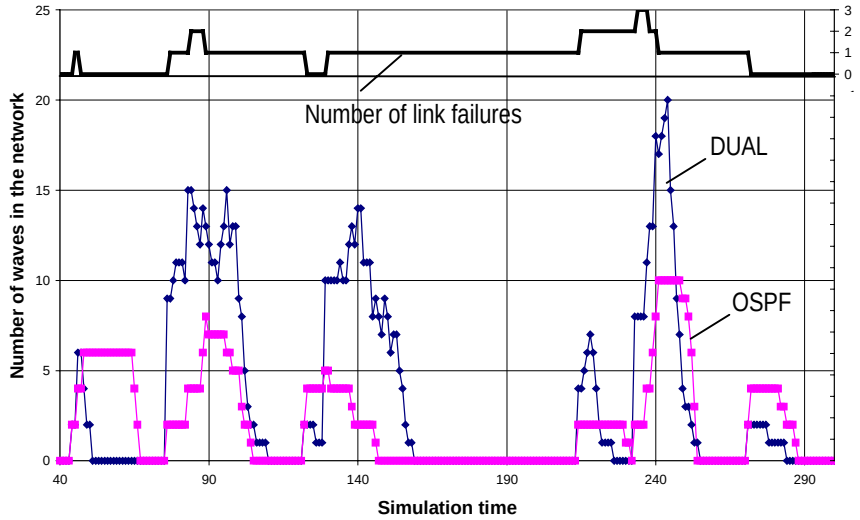

Figure 6: Number of waves after multiple link failures and recoveries.

From Figure 7, one can see that DUAL generates many local waves, while OSPF generates a smaller number of waves, each wave covering larger parts of the network. This shows that updates in DUAL are more localized, while updates in OSPF affect the whole network (the diameter of the network is 5).

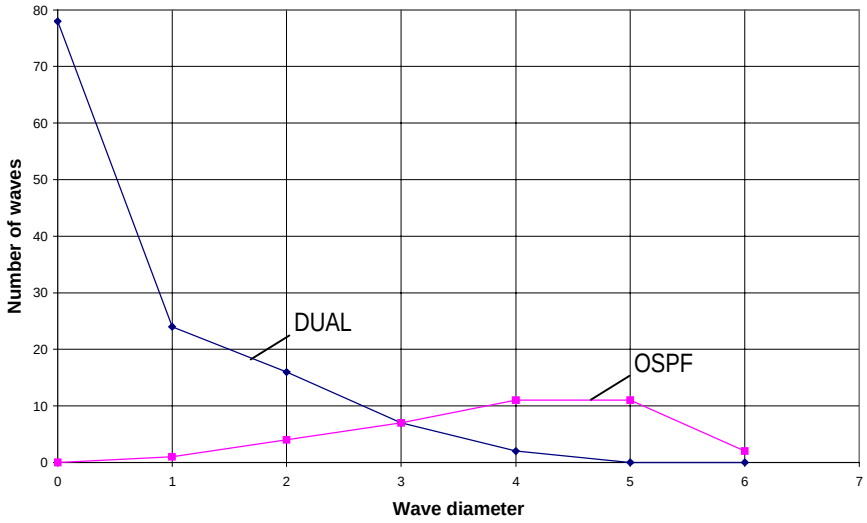

Figure 7: Distribution of waves according to their diameter after multiple link failures and recoveries.

5. Using Progressive Waves in Self-Managing Systems

Progressive waves can be used to build self-managing network services. Every request for a network service will result in the reservation, consumption and later release of a set of resources. Examples of resources are: processing power, capability to run a certain software application, storage capacity, allocating secure sockets, etc. The end user will measure the service performance according to high-level criteria such as availability, immediacy of the response, low price, or high security. The network service provider needs to meet these criteria, but at the same time needs to process the request in a way that will optimize its network utilization, efficiency and profitability. New service requests and completion of service executions will result in a constant redistribution of resources inside the network of the service provider. In order to offer an efficient network service, one needs to solve a problem of resource distribution under a set of constraints. The updated availability of these resources can be propagated inside the service provider network by a progressive wave pattern. When a new request arrives in the network, it can be automatically forwarded to a server that will process the request in such a way that both user and service provider constraints are met. The same navigation pattern can be used for managing different types of resources, but every type of resource needs a new type of aggregator.

To illustrate how a simple application could be optimized using pattern-based routing, consider the example of a distributed ftp service. A company has a new software demo version available online, and an end user is interested in downloading it. Usually, the user is presented with a list of ftp servers where the file can be downloaded. Information about the location and number of users already connected to each ftp server is also provided. The user then chooses, based on these criteria, an ftp server and starts the download. The

software provider could hide the internals of its ftp network from the user and always pick the best server by running a routing algorithm inside the ftp service overlay network. The routing algorithm propagates in the network updated utilization statistics of each server, and when a request is received it can be automatically directed to the server that will provide the fastest download time to the user and will maintain an appropriate load balancing in the ftp server network. The process of selecting an ftp server from which to download the file is automated.

6. Discussion

While studying OSPF and DUAL, we came to the conclusion that information exchange in these two Internet routing protocols can be modeled and implemented as a set of waves propagating in the network. We encapsulated the information propagated in these waves in a navigation pattern which we call progressive wave.

Our study of patterns for routing to date is limited to DUAL and OSPF. While we expect it, we cannot yet prove that all common Internet protocols can be based on this pattern. The same applies to other routing schemes, such as multicast Internet routing, peer-to-peer systems routing, and routing in mobile ad-hoc networks.

The routing and resource location systems described in Sections 3, 4, and 5 are distributed and adaptive. Since they maintain a distributed state they can be characterized as self-managing. By trying to identify common characteristics among them, we aim at contributing to a methodology for engineering self-managing systems.

We observe that the progressive wave pattern behaves similarly to propagation phenomena known from electromagnetism, acoustics, hydrodynamics, and anatomy. An interesting question to ask is whether other types of wave propagation patterns can be recognized in natural phenomena and what the properties of these patterns in the networking domain would be.

We are currently porting the progressive wave, as well as the DUAL and OSPF routing aggregators to the Weaver networking platform [6]. We want to experiment on a network testbed with scenarios similar to those that currently run on the simulator (Section 4). Further, we plan to realize a set of adaptive “e-sourcing” services on the Internet, similar to the ones we described in Section 5.

7. Acknowledgements

This work has been supported by the Swedish SSF under grant number Dnr SSF 2001/0243.

References

[1] M. Baldi, S. Gai and G. Picco, “Exploiting Code Mobility in Decentralized and Flexible Network Management,” *First International Workshop on Mobile Agents (MA’97)*, Berlin, Germany, April 1997, pp. 13-26.

[2] M. Baldi, G. Picco, “Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Appli-cations,” *First International Working Conference on Active Networks (IWAN’99)*, June/July 1999, Berlin, Germany.

[3] Y. Yemini, G. Goldszmidt and S. Yemini, “Network Management by Delegation,” *IM’91*, Washington, DC, April 1991, pp. 95-107.

[4] A. Liotta, G. Knight, G. Pavlou, “On the Performance and Scalability of Decentralized Monitoring Using Mobile Agents,” *DSOM ‘99*, Zurich, Switzerland, October 1999.

[5] M. Daniele and B. Wijnen, “Agent extensibility (AgentX) protocol, version 1,” *RFC-2741*, IETF, January 2000.

[6] K.S. Lim and R. Stadler: "Weaver: Realizing a Scalable Management Paradigm on Commodity Routers," *IFIP/IEEE International Symposium on Integrated Network Management (IM'03)*, Colorado Springs, CO, March 24-28, 2003

[7] K.S. Lim and R. Stadler: "Developing Pattern-Based Management Programs," *IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS 2001)*, Chicago, IL, October 29 - November 1, 2001.

[8] K.S. Lim and R. Stadler: "A Navigation Pattern for Scalable Internet Management," *IFIP/IEEE International Symposium on Integrated Network Management (IM'01)*, Seattle, Washington, 14-18 May, 2001.

[9] Garcia-Lunes-Aceves, J.J.: “Loop-free routing using diffusing computations**”** *IEEE/ACM Transactions on Networking*, Volume: 1 Issue: 1, Feb 1993 pp: 130 -141

[10] J. Moy, “OSPF Version 2”, *STD-0054*, IETF, April 1998

[11] R. Coltun, “OSPF: An intemet routing protocol,” *ConneXions*, vol. 3, no. 8, pp. 19-25, Aug. 1989.

[12] Y, Rekhter, T. Li, “A Border Gateway Protocol 4 (BGP-4)”, *RFC-1771*, IETF, March 1995

[13] G. Malkin, “RIP Version 2”, *STD-0056*, IETF, November 1998

[14] G. Tel, “Introduction to Distributed Algorithms”, *Cambridge University Press*, 2000

[15] D. Oran, “OSI IS-IS Intra-domain Routing Protocol”, *RFC-1142*, IETF, February 1990

[16] SIMPSON - A SIMple Pattern Simulator fOr Networks, http://www.comet.columbia.edu/adm/software.htm