A Biological Approach to Network Computing

James W. Fehlig ~
Information and Computer Science

Author

James' research led him to a career with Novell, Inc., an industrial network software company. He plans to attend graduate school to further pursue his research. His project gave him the chance to apply the concepts of large biological systems to his knowledge of network services and applications. In the development and execution of this research project, James learned that the key to success is to "define goals and milestones and steadily progress to accomplish [those] goals." James feels that "the essence of research is that you are investigating something new and unknown, which makes it a very challenging yet rewarding activity."

Key Terms:
- Bio-Networking Architecture
- Internet Topology
- Next Generation Internet (NGI)
- Object-Oriented Techniques
- Theoretical Graph

Abstract

Services and applications in the Next Generation Internet (NGI) have several key requirements. They must scale to billions of users and network nodes, adjust to heterogeneous and dynamic networking environments and conditions with minimal administration, resist large-scale attacks and failures, and support deployment and extensibility with minimal complexity. The proposed Bio-Networking Architecture is inspired by the observation that the biological world has already developed the mechanisms necessary to achieve these key requirements. This project examines the biological concepts of autonomous behavior and adaptation through the conduct of simulations. An autonomous, adaptable population of cyber-entities is compared to the static, non-adaptive approaches that are currently used to implement network services. An example of the latter technique is the use of statically placed servers. This research has shown that an adaptable population of cyber-entities can deliver performance comparable to six statically placed servers at a fraction of the cost.

Faculty Mentor

James Fehlig attacked a new network architecture called Bio-Networking. Bio-Networking Architecture is inspired by the observation that the biological world has already developed the mechanisms necessary to achieve the key requirements of future network services and applications, such as scalability, adaptability to heterogeneous and dynamic conditions, evolution, security, survivability, and simplicity. James developed a simulator to evaluate feasibility and efficiency of the Bio-Networking Architecture. The research James has carried out will deepen our knowledge on how biological concepts can be applied to network architecture designs. Working with James has been a very exciting and rewarding experience for me.

~ Tatsuya Suda
Department of Information and Computer Science

Introduction

Due to the exponential growth of the Internet, both in terms of hardware devices and the number of users, new approaches to designing and implementing network services and applications in the Next Generation Internet (NGI) must be discovered and analyzed. NGI is a U.S. government-sponsored initiative involving government, research institutions, and the business sector, with the goal of creating an infrastructure that is 100 to 1000 times faster than today's Internet.

Any plausible approach must satisfy several key requirements. First, it must possess scalability, supporting billions of users and network nodes (Network Wizards, 1998). Next, it must be capable of adaptation and evolution in order to adjust to heterogeneous and dynamic networking environments and conditions with minimal administrator involvement. Finally, it must feature security and survivability to resist large-scale attacks and failures (Davis, 1997) and support deployment and extensibility with minimal complexity (Ellison et al., 1997).

This project has been inspired by the observation that the biological world has already developed the mechanisms necessary to achieve the aforementioned key requirements of the NGI. Many biological systems such as the immune system (Tonegawa, 1983; Branden et al., 1991), bee colony (Michener, 1974; Seeley, 1995), and ant colony (Franks, 1989; Gotwald, 1995) can grow to a large population, adjust to heterogeneous and dynamic environments, detect and eliminate foreign entities, and survive attacks. For example, a bee colony scales to support a huge number of bees. Bee colonies adapt to a wide variety of weather and food conditions. They secure their hive from intruding bees and other animals and are able to survive even when a significant percentage of the bees are destroyed by predators, accidents, or disease.

An important characteristic of biological systems is their ability to adapt to the surrounding environment, adjusting to both short-term and long-term changes. Short-term environmental changes can be characterized by day-to-day fluctuations in weather conditions such as temperature. Returning to the bee colony example, increases in ambient temperature, which result in an increased hive temperature, cause the colony members to fan the hive in order to maintain a constant internal temperature. Long-term changes, such as depletion of a local food supply, may result in a relocation of the entire biological system to a new area. The adaptation of biological systems is accomplished through migration, reproduction, evolution, and death. Although this list is by no means an exhaustive enumeration of adaptation mechanisms, it includes important fundamental concepts necessary to define an adaptable system.

Based on the above observations of biological systems (which already satisfy the NGI requirements), this project attempts to investigate the biological concept of adaptation and to demonstrate that an adaptable population of entities, which encapsulate a network service, can also achieve the key requirements of the NGI. The Netgroup in the Information and Computer Science Department at the University of California at Irvine has proposed the Bio-Networking Architecture, a computer network architecture based on the key biological concepts observed in large-scale biological systems. In the Bio-Networking Architecture, a service is implemented by a distributed, collective entity called the super-entity. The super-entity is composed of multiple, autonomous entities, each of which is called a cyber-entity. This is analogous to the concept of a bee colony consisting of multiple bees. Each cyber-entity provides a basic service (e.g., a web page) and has a set of simple behavior rules. The desired behavior and functionality (e.g., an entire web site) emerge in the super-entity from the collective actions and interactions of its cyber-entities.

Another key concept in the Bio-Networking Architecture is the exchange of energy for services among the cyber-entities. For example, one type of cyber-entity is a service cyber-entity that provides information (e.g., a web page) to a user type cyber-entity. When the service cyber-entity provides information to a user, the user cyber-entity must give energy units to the service cyber-entity. The service cyber-entity also expends energy, in exchange for the network resources it consumes. A specific example of energy in the biological world is food. As such, energy strongly affects and limits cyber-entity behavior. Cyber-entities whose services are in high demand and have more suitable behavior algorithms will receive more energy and reproduce more. Inefficient cyber-entities will eventually die from lack of energy.

The Bio-Networking Simulator (hereafter referred to as the Simulator) provides a framework for investigating adaptation of the Bio-Networking Architecture. Biological concepts used in this project are energy exchange and storage, migration, replication and reproduction, and death. Through these biological concepts, this project attempts to demonstrate that an adaptable population of cyber-entities will provide an improved service over existing techniques, such as static servers. The Simulator is capable of emulating an arbitrary network topology and typical user load scenario. This provides a realistic network environment in which to evaluate the performance of an adaptable population of cyber-entities against statically placed servers.

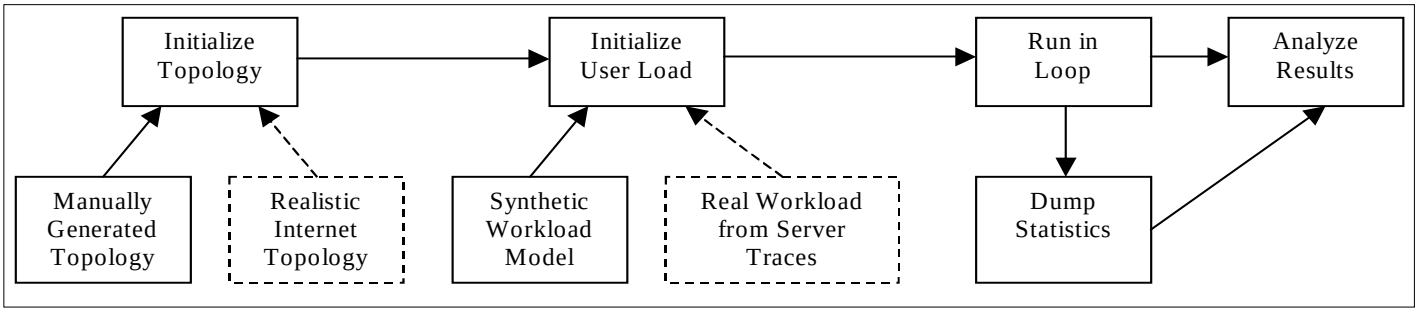

Figure 1

Bio-Networking Simulator Architecture

Bio-Networking Simulator Design/Architecture

The Bio-Networking Simulator is implemented in the Java™2 programming language using Object-Oriented Techniques, which involve the use of objects (i.e. components containing data and instructions for operations to be performed on the data) to create applications. The Simulator is capable of emulating arbitrary network topologies as defined by a topology description file. Currently, the topology description file is manually generated, however the design will accommodate topology files generated using automated techniques. Later in this section, there is a description of the format of the topology description file. User load (number of users on the network at any given time) is synthetically generated in the current Simulator version. A future enhancement includes modeling user load from server access logs, which would provide a more realistic load scenario. A description of the current user load modeling technique used by the Simulator is also given later in this section. After initializing the network from the topology description file, the Simulator iterates through a sequence of events, each sequence defining a 1 sec interval of the simulation. The end of this section offers a detailed description of the tasks performed during each simulation cycle. The simulation proceeds until 3 months (7,776,000 sec) have transpired. Numerous statistics are collected throughout the conduct of the simulation. After the simulation is complete, the statistic files can be imported to a spreadsheet, such as Microsoft Excel, for graphing and analysis.

The simulation can be run numerous times, using a different type of cyber-entity each time in order to analyze the performance of cyber-entities with distinct behaviors. Figure 1 illustrates the high-level architecture for the Bio-Networking Simulator.

Cyber-Entity Design

Both users and service providers are implemented as cyber-entities. Users refer to both the conventional concept of a user (i.e. actual person requiring network services) and a software entity requiring the services of another entity in order to perform its task. Using this definition, it is reasonable to consider a user as a type of cyber-entity.

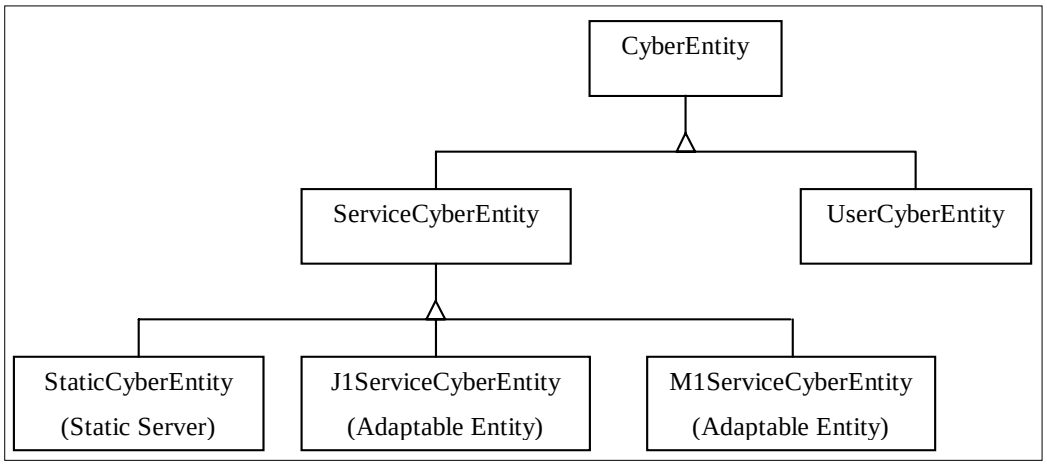

Figure 2

Cyber-entity Class Hierarchy

As shown in Figure 2, all entities are derived from the CyberEntity super class. The UserCyberEntity object represents a user (i.e., an entity that requires a network service). The ServiceCyberEntity branch represents entities that provide services to users. StaticCyberEntity is a specialized type of ServiceCyberEntity that models a traditional static server. J1ServiceCyberEntity (J1) and M1ServiceCyberEntity (M1) are service objects that implement the biological concepts of migration, replication (reproduction), and death. They each use different algorithms in order to evaluate diverse behaviors.

A CyberEntity object consists of three parts: 1) attributes, 2) behavior, and 3) body. The attributes portion contains information such as the cyber-entity's name and unique identification number. The behavior part contains logical units of executable code that allow the cyber-entity to perform useful services for a user (e.g., delivering a web page

or other multi-media content) and mimic simple biological behaviors such as migration, replication, and death. The body part contains data relevant to the cyber-entity, such as a web page or multi-media object. The specialized UserCyberEntity finds the closest ServiceCyberEntity and delivers a message requesting the service provided by it. The concept of being close is defined as the fewest number of hops between the two entities. The UserCyberEntity then waits for a response from the ServiceCyberEntity. A response contains the delay incurred in providing the requested service. In order to simplify the simulation, the actual service (a web page or multi-media object) is not provided in the response.

A ServiceCyberEntity can satisfy 50 user requests per second (20 msec per request). Excess requests are placed on the entity's work queue, which is assumed to be infinitely long. For each user request received during the current simulation cycle, a response is generated and sent to the user that generated the request. As previously noted, the response includes the delay incurred in servicing the user's request. Delay is calculated as a function of the number of hops between the user and service providing entities and the contents of the work queue. A ServiceCyberEntity with several hundred backlogged requests in the work queue will generate a longer delay than an entity with an empty work queue.

ServiceCyberEntity objects also receive energy units (which equate to food) for satisfying user requests, and expend energy units for the network resources used while providing their services. Migration, replication, and death behaviors of a ServiceCyberEntity object can be controlled as a function of their accumulated energy level.

The StaticCyberEntity object is used to emulate the behavior of a static server. The static entity inherits the functionality of the ServiceCyberEntity object defined above; however, there are no implementations for migration, replication, or death. Thus, the StaticCyberEntity represents a non-adaptive network service provider.

J1 and M1 represent two variants of adaptable cyber-entities studied in the simulation. Both entities inherit the ServiceCyberEntity functionality described above, but implement different migration, replication, and death behaviors.

J1 Migration Behavior

The J1 cyber-entity investigates the benefits of migrating every minute (60 simulator cycles) by calculating a score for each neighboring network node. The score is based on energy seeking and repulsion factors. The energy-seeking factor for a neighboring node is simply the amount of energy units that came from the direction of that neighboring node in the last 60 cycles. The repulsion factor increases or decreases a node's score based on the proximity of similar J1 entities, ensuring that similar cyber-entities do not congregate on the same node. The J1 entity then migrates to the neighboring node with the highest score provided that the score is above an "aggressiveness" threshold.

M1 Migration Behavior

The M1 cyber-entity considers migrating to another node each minute based on energy imbalance, migration cost trade-off, oscillation avoidance, migration rest, and hesitation factors. The energy imbalance factor favors migrating to a neighboring node if the energy coming from the direction of that node is higher than the average energy coming from all nodes by a configurable amount. The migration cost trade-off factor prevents migration if the amount of energy imbalance is less than the cost of migration. The oscillation avoidance factor prevents migration to a neighboring node if the M1 entity migrated from that node within the last hour. The rest factor prevents migration if the M1 entity migrated in the last 600 sec. The hesitation factor allows migration to a neighboring node only if the M1 entity wanted to migrate to that same neighboring node for a configurable number of simulator cycles. Newly created M1 entities do not consider the rest and hesitation factors which allows them to quickly migrate away from their parent.

J1 Replication Behavior

The J1 cyber-entity evaluates replication every minute based on workload and overcrowding factors. The J1 entity maintains a workload variable, which is a time decaying average of the number of requests it receives per second. If the workload is above a threshold and the neighboring nodes are not overpopulated with J1 entities, replication occurs.

M1 Replication Behavior

Every minute the M1 cyber-entity considers replication based on replication rest and workload factors. The replication rest factor prevents replication if the entity has replicated in the past 900 sec. The workload factor encourages replication using a probability distribution that increases exponentially with the workload. Thus, when the workload is high there is a high probability that the M1 entity will replicate.

J1 Death Behavior

Every second the J1 entity considers dying based on age and workload. The J1 entity will die if it is more than 1 hr old and the workload is below a threshold.

M1 Death Behavior

Every minute the M1 cyber-entity considers dying based on immortal and workload factors. The immortal factor

prevents certain M1 entities from dying. This behavior is needed to keep at least one M1 entity alive in the network even when there are no users. The workload factor causes the M1 entity to die based on a probability distribution that increases inversely with workload. Thus, when workload is low there is a high probability that the M1 entity will die.

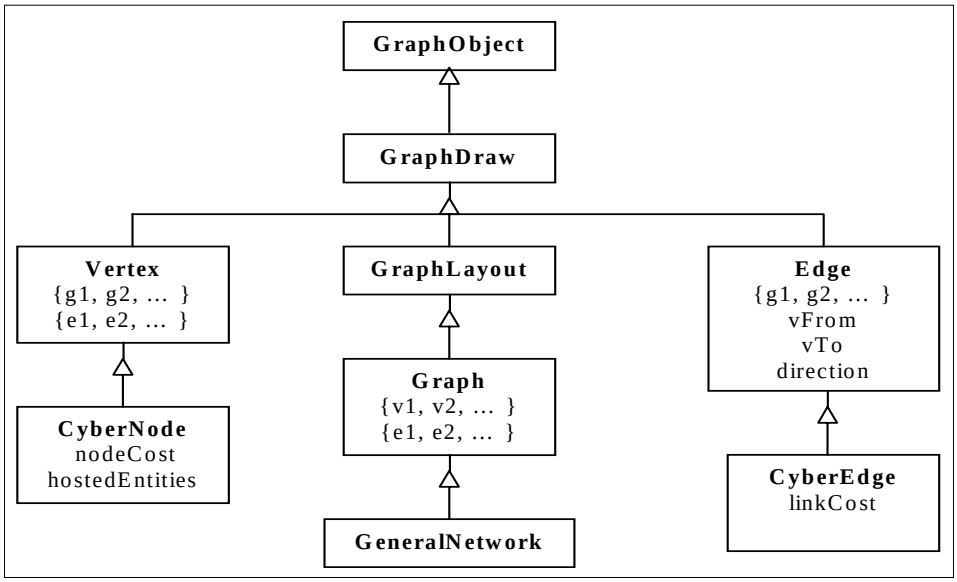

Figure 3
Network Class Hierarchy

Network Design and Cost Structure

The Simulator can model an arbitrary network topology defined by the topology description file. Internally, the network is maintained and manipulated by a data structure that supports theoretical graph representations and operations. Theoretical Graphs are mathematical analyses of graphs. The class hierarchy for the graph library is depicted in Figure 3. All network objects are derived from the GraphObject super class. The library is structured to support plug-ins for rendering the network in a Graphical User Interface (GUI). This design isolates theoretical graph operations from the mechanisms used to draw the graph, permitting the use of several different rendering techniques. A vertex in the graph is defined by a set of graphs (in theory, a vertex can belong to more than one graph) and a set of edges. An edge is defined by a set of graphs, a "from" vertex, a "to" vertex, and a direction. A graph is defined by a set of vertices and a set of edges. At the bottom of each branch are specialized graph objects that together constitute a network environment where cyber-entities can live, reproduce, migrate, and die. This framework also supports objects modeled after traditional, static servers such as the StaticCyberEntity described above.

A network of 50 nodes was used for the simulations conducted during this phase of the project. It approximates a global network that spans Asia, the Americas, Europe, and Australia. Each node has a resource cost of 3 units, which represents the cost to use the node's resources for 1 sec. Although unrealistic, a constant resource cost permits a simplified calculation of resources consumed over the conduct of the simulation.

Cyber-entities must also "pay" 180 units to migrate and 900 units to replicate. Migration and replication operations are not "free", otherwise an unfair resource consumption analysis would be made between static servers and adaptable cyber-entities. As an example of computing resource consumption over a one-month period, consider the case of a 1 static server. The server consumes 3 units of energy each second and there are 2,592,000 sec/month. Thus a static server will consume 3 x 2,592,000 = 7,776,000 energy units over a one-month period. In the simulations conducted for this project, 1,000,000 energy units are equivalent to 1 USD. Thus a content provider using 1 static server would spend approximately $7.26/month to host its service on the network.

Topology Description File

Internet Topology is the pattern in which computers are interconnected on the Internet. The topology description file provides a mechanism for describing an arbitrary network topology. The Simulator includes a parser capable of reading the description file and instantiating a graph data structure described by the file. This design permits independent creation of the topology file. Thus, the file can be created manually (using a text editor) or automatically (using an Internet Topology generation tool such as Transit-Stub or Tiers), both modes being developed by researchers at Georgia Tech (Calvert et al., 1997).

The format of the topology description file permits definition of network nodes and links that connect the nodes. Comments can also be inserted in the file. A comment begins with the asterisk character (*) and continues to the end of the line. An excerpt from a topology description file follows.

```
* These are the major nodes of the network.
*
* japan
BEGIN NODE NODEID=1
      NODECOST=3
      NODEX=100 NODEY=270
END

* hawaii
BEGIN NODE NODEID=2
      NODECOST=3
      NODEX=600 NODEY=600
END

* Here comes the links that connects all the nodes
*
BEGIN LINK LINKID=1
      LINKCOST=10
      LINKENDA=1 LINKENDB=14
END
```

The sample topology description file above would create a simple network of two nodes connected by a single bidirectional link. Each node has a numeric identification, a resource cost of 3 units, and a x, y coordinate that specifies its location for rendering within a GUI window. The link has its own numeric identification, the numeric identifications of the nodes to which it connects, and a cost currently not used in the Simulator, but which can represent capacity of the link.

Synthetic User Load Model

User load is synthetically generated, modeling typical weekly user activity. The weekends have moderate activity and the weekdays have high activity during the morning and late afternoon hours. The heaviest activity occurs on Monday, which simulates users accessing the network after a weekend away from work. Figure 4 depicts a one-week snapshot of the synthetically generated user load. Note that the maximum number of users at any given time is 250, which is reasonable considering the simulated network of 50 nodes used during this phase of the project.

Definition of a Simulator Cycle

Each cycle of the Bio-Networking Simulator corresponds to 1 sec of the simulated period. For this phase of the project, a three-month time period was simulated, requiring the execution of 7,257,600 cycles. Below is a description of the events that occur in a cycle.

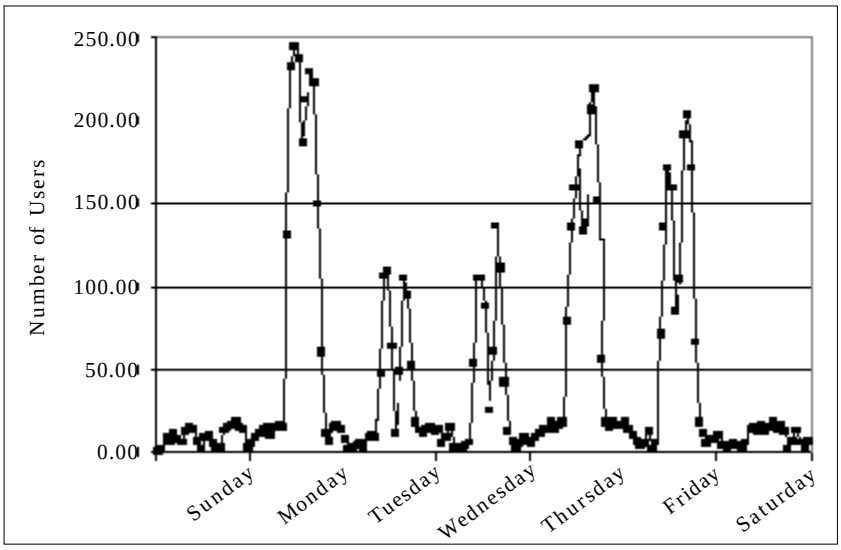

Figure 4

Symmetric User Load, One Week Excerpt

During each cycle, the network object is informed of the new cycle. This permits changing the cost structure of the network environment and simulating network failures such as breaking a link between two nodes or disabling an entire sector of the network. Next, the synthetic user load module is informed of the new cycle in order to add or remove users from the network. For each user in the network, the closest cyber-entity is found that can provide services to the user. A message containing the user's identification, an energy payment for satisfying the user's request, and the number of hops between the user and the service is delivered to the selected cyber-entity. The cyber-entity stores the message in its internal message queue. Then, for each service cyber-entity currently alive in the network, a routine is executed allowing the cyber-entity to respond to all messages in its message queue. It then determines whether or not to migrate, replicate, or die. Finally, the state of each user and service cyber-entity is saved to a disk for post-simulation analysis.

Results and Conclusions

Five different simulations were conducted in order to compare adaptive and non-adaptive techniques to providing network services. The five simulations are characterized as follows:

Non-Adaptive

- 1 static server placed in Chicago network node
- 4 static servers placed in Chicago, Japan, Los Angeles, and England network nodes
- 6 static servers placed in Chicago, Japan, Los Angeles, England, San Francisco, and Houston network nodes

Adaptive

- J1 variant of cyber-entity initially placed in Chicago network node
- M1 variant of cyber-entity initially placed in Chicago network node.

Figure 5 shows the resource costs for the five simulations over the three-month simulation period. For the static server cases, each server costs approximately $7.29/month. This value is consistent with the cost structure defined above. For the two adaptable cyber-entity simulations, the costs reflect a variable number of cyber-entities and the costs associated with their migrations and replications. Figure 5 clearly shows that both adaptable cyber-entity configurations are more expensive than the single server configuration, however they have considerable cost advantages over the 4 and 6 static server configurations.

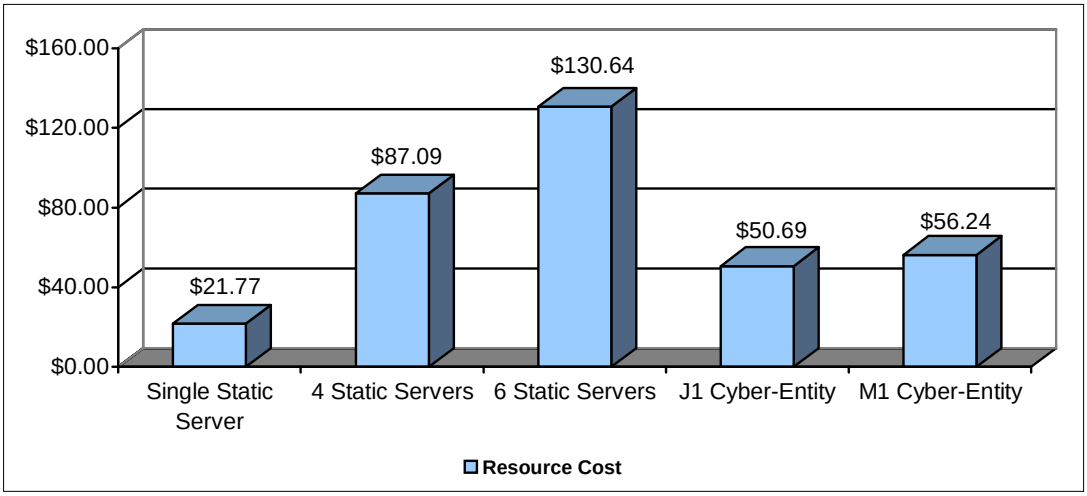

Figure 5

Resource Cost Over Three Months

Figure 5 shows the average service delay experienced by the users for each of the 3 static server simulation runs. Although each simulation ran for a three-month period, only the service delay results for the first week are shown in the graph. Succeeding weeks exhibit similar service delay patterns.

The key observation to note in Figure 5 is that the single server configuration is unable to adequately support the network users. Throughout a majority of the weekday work hours, service delay is well over 600 sec. With the exception of a small rise on Monday, service delay for the 4 and 6 static server configurations is not discernable over the excessive delays exhibited by the single server configuration. The single server will be removed from succeeding graphs due to its poor performance.

Figure 6 shows the service delay throughout week 1 for the 4 static server configuration and the J1 and M1 adaptable cyber-entities. During the Monday peak workload, average delay experienced by the user approaches 120 sec for the 4 static server configuration. Average delay is approximately 10 sec for the M1 cyber-entity and 80 sec for the J1 cyber-entity. The M1 cyber-entity exhibits aggressive replication and migration behaviors compared to the J1 cyber-entity, thus quickly adapting to the sharp increase in network activity on Monday. With the exception of Monday, all configurations are able to satisfy user demand with reasonable delay.

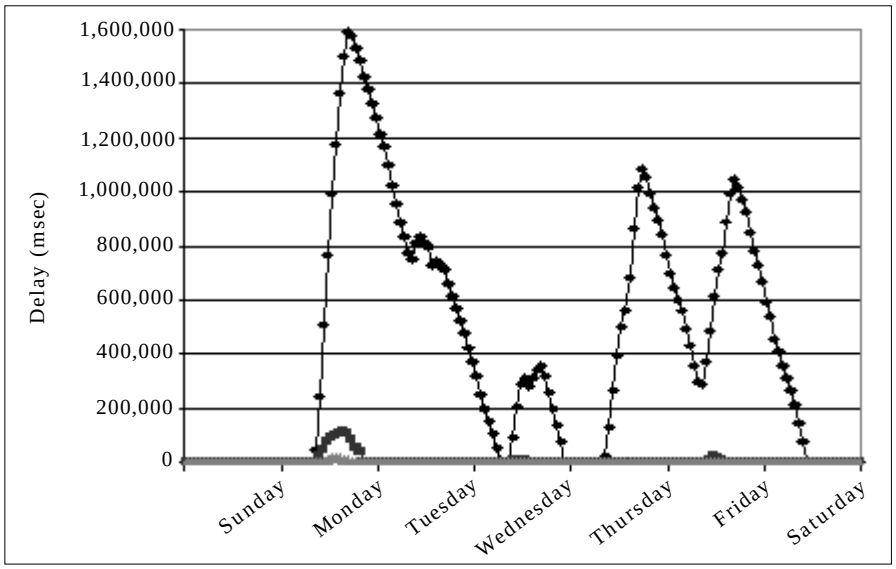

Figure 6

Average Service Delay For Static Server Configurations

Figure 7 illustrates the service delay throughout the entire three-month simulation period for the 6 static server configuration and the J1 and M1 adaptable cyber-entities. The 4 static server configuration has been excluded due to its inferior performance relative to the 6 static server and adaptable cyber-entity configurations. The y-axis has been expanded to examine service delay between 0 and 40 msec.

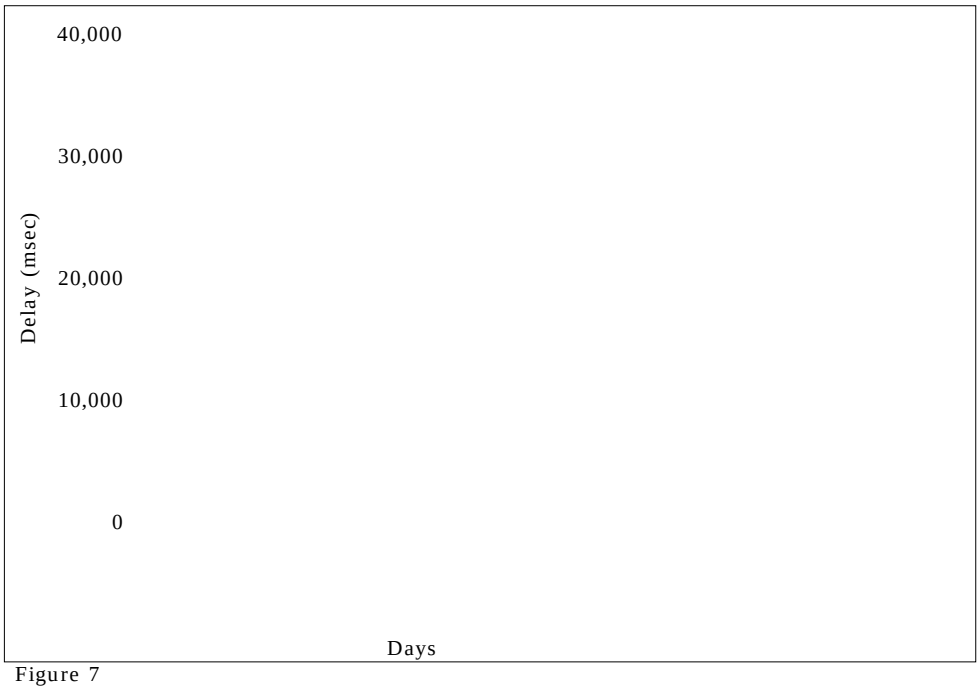

Figure 7

Average Service Delay Over 3 Months

The graph clearly shows that the M1 cyber-entity, which has more sophisticated migration and replication algorithms, outperforms the 6 static server configuration throughout most of the peak workload periods. Excluding peak workload periods, the J1 cyber-entity, with its simplistic migration and replication algorithms, rivals the performance of the 6 static server configuration.

Figure 7 reveals a far more interesting observation. The J1 and M1 cyber-entities adapt to the simulator environment, performing their programmed task more efficiently as the simulation progresses. Maximum peaks of the J1 and M1 cyber-entities decrease over time. Being unable to adapt to the environment, maximum peaks of the static servers gradually increase. Analogous to bees adjusting their behavior to maintain optimal conditions within the hive (the programmed task of the bee population), the cyber-entity population adjusts to the environment in order to efficiently perform its programmed task of providing minimal user delay at minimal cost.

This research has demonstrated that the Bio-Networking Architecture can provide a promising approach to creating adaptable network services. Although the J1 and M1 cyber-entity behaviors are far from optimal, the simulations show that adaptable cyber-entities can deliver performance comparable to 6 statically placed servers at a fraction of the cost. This project is only the first step in the research and development of the Bio-Networking Architecture; however, the results suggest that the biological world possesses features and characteristics that should not be overlooked when designing services and applications for the Next Generation Internet.

Works Cited

Branden, Carl, and John Tooze. Introduction to Protein Structure. Garland Publishing, 1991.

Calvert, Kenneth, Matthew Doar and Ellen Zegura. "Modeling Internet Topology." IEEE Communications Magazine June, 1997.

Davis, J. C., ed. President's Commission on Critical Infrastructure Protection. Online. Internet. June, 1997. Available <http://www.info-sec.com/pccip/web/>.

Ellison, R.J., Fisher, D.A., Linger R.C., Lipson H.F., Longstaff, T. and Mead N.R. "Survivable Network Systems: An Emerging Discipline." Carnegie Mellon University/Software Engineering Institute Technical Report (1997).

Franks, N. "Army Ants: A Collective Intelligence." American Scientist (1989).

Gotwald, William. Army Ants. Comstock Publishing, 1995.

Michener, Charles. The Social Behavior of the Bees. Belknap Press, 1974.

Network Wizards Internet Domain Survey. Online. Internet. July, 1998. Available <http://www.nw.com>.

Seeley, Thomas. The Wisdom of the Hive. Harvard UP, 1995.

Tonegawa, S. "Somatic generation of antibody diversity." Nature (1983).