PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

Hybrid evolutionary computing model for mobile agents of wireless Internet multimedia

William S. Hortos

SPIE.

Hybrid evolutionary computing model for mobile agents of wireless Internet multimedia

William S. Hortos

Florida Institute of Technology, Orlando Graduate Center, 3165 McCrory Place, Suite 161, Orlando, FL 32803

ABSTRACT

The ecosystem is used as an evolutionary paradigm of natural laws for the distributed information retrieval via mobile agents to allow the computational load to be added to server nodes of wireless networks, while reducing the traffic on communication links. Based on the Food Web model, a set of computational rules of natural balance form the outer stage to control the evolution of mobile agents providing multimedia services with a wireless Internet protocol (WIP). The evolutionary model shows how mobile agents should behave with the WIP, in particular, how mobile agents can cooperate, compete and learn from each other, based on an underlying competition for radio network resources to establish the wireless connections to support the quality of service (QoS) of user requests. Mobile agents are also allowed to clone themselves, propagate and communicate with other agents. A two-layer model is proposed for agent evolution: the outer layer is based on the law of natural balancing, the inner layer is based on a discrete version of a Kohonen self-organizing feature map (SOFM) to distribute network resources to meet QoS requirements. The former is embedded in the higher OSI layers of the WIP, while the latter is used in the resource management procedures of Layers 2 and 3 of the protocol. Algorithms for the distributed computation of mobile agent evolutionary behavior are developed by adding a "learning" state to the agent evolution state diagram. When an agent is in an indeterminate state, it can communicate to other agents. Computing methods can be replicated from other agents. Then the agent transitions to the "mutating" state to wait for a new information-retrieval goal. When a wireless terminal or station lacks a network resource, an agent in the "suspending" state can change its policy to submit to the environment before it transitions to the searching state. The agents "learn" the facts of agent state information entered into an external database. In the cloning process, two agents on a host station sharing a common goal can be merged or "married" to compose a new agent. Application of the two-layer set of algorithms for mobile agent evolution, performed in a distributed processing environment, is made to the QoS management functions of the IP multimedia (IM) subnetwork of the third-generation (3G) Wideband Code-division Multiple Access (W-CDMA) wireless network.

Keywords: Evolutionary computing, mobile agent, radio resource allocation, time-varying self-organizing feature map, genetic algorithms, wireless Internet protocol, multimedia services

1. INTRODUCTION

Mobile agents are software programs that can migrate from host to host in an entire network of computers, autonomously in time and place. Unlike applets, both the code and the execution state (heap and stack) move with the agent; unlike processes in process-migration systems, mobile agents move when and where they choose. They are typically written in a language that can be interpreted, such as Java, Tcl, or Scheme, to ensure interoperability and independence among operating systems and hardware architectures. Agent programmers typically structure their application so that the agents migrate to the host(s) where they can find the desired service, data, or resource, so that all interactions occur on the local host, rather than across the network. In some applications, a single mobile agent migrates sequentially from host to host; in others, an agent spawns one or more offspring or cloned agents to migrate independently.

A mobile-agent programmer thus has an option not available to the programmer of a traditional distributed application: to move the code to the data, rather than move the data to the code. In many scenarios, moving the code may be faster, if the agent's state is smaller than the data that would be moved. Or, moving the code may be more reliable, because the application is only vulnerable to network disconnection during the agent transfer, not during the interaction with the resource. For a survey of the potential applications and technology of mobile agents see [1].

Applications and Science of Computational Intelligence IV, Kevin L. Priddy, Paul E. Keller, Peter J. Angeline, Editors, Proceedings of SPIE Vol. 4390 (2001) © 2001 SPIE · 0277-786X/01/$15.00

These characteristics make mobile-agent technology especially appealing in wireless networks, which tend to have lower bandwidth and reliability when compared with wired networks. A user of a mobile computing device can launch a mobile agent, which jumps across the wireless connection into the wired Internet. Once there, the agent can safely roam among the sites that host mobile agents, interacting either with local resources or, when necessary, with resources at remote sites that are not willing or able to host mobile agents. Its task completed, it can return to (or send a message to) its user via the wireless network.

Clearly the agent case avoids the transmission of unnecessary data, but does require the transmission of agent code from client to server. The total bandwidth consumption from code transmission depends on the agent size and arrival rate. For most reasonable agent code sizes and arrival rates, the savings in data transmission may be much larger than the code transmissions. Of course, each client's code could be pre-installed on the server. This approach presupposes, however, that the clients are known in advance. In many of the application environments, new clients with new code can appear at any time, and possibly disappear only a short time later. In the wireless scenarios discussed in this paper, at least a dynamic-installation facility is required, and mobile agents supply the flexibility to move filtering code to any point in the network, and to move the code again as the network conditions change.

The technique can be used in distributed information retrieval which allows the computational load to be added to servers, but significantly reduces the traffic of network communications. Unfortunately, it is difficult to find a theoretical basis for mobile agent computing and interaction over the Internet, for both wired and wireless links. Due to the limitations on wireless network bandwidth, distributed Internet search engines promise to improve upon traditional multimedia information retrieval schemes, which perform the control of searching for and transmitting data on a single machine.

A mobile agent, in general, is more than a search program. For instance, a mobile agent can serve as an emergency message broadcaster, an advertising agent, or a survey questionnaire collector. A mobile agent should have the following capabilities: (1) achieve a goal automatically; (2) clone itself and propagate; (3) communicate with other agents; and (4) possess evolution states, including a termination state.

The environment where the mobile agents live is the Internet. Agents are distributed automatically or semi-automatically via some communication paths. Therefore, agents meet each other on the Internet. Agents with the same goal can share information and co-operate. However, if the system resources, e.g., network bandwidth, storage, link gain, channels available at a station or processing node, is insufficient, agents compete with each other. These phenomena are similar in type to those in a real-world ecosystem. A creature is born with a goal to live and reproduce. To defend against their enemies, creatures of the same species cooperate. However, a perturbation in an ecosystem can induce the creatures to compete with or even kill (terminate) each other. The natural world is built upon a law of balance. Food Web embeds the law of creature evolution.[2] With the growing popularity of Internet where mobile agents live and the emergence of multimedia services carried by the versions of the Internet Protocol (IP) in both wired and wireless domains, the objective of this work is to emulate the natural world with an agent evolution computing model over the IP subsystem in 3G wireless networks. The model, applied to mobile agent evolution here, can be generalized to solve other distributed/concurrent computation problems.

A logical network for agent connections/communications in a wireless multimedia network, called Wireless Multimedia Agent Communication Network (or WM-ACN). The WM-ACN is necessarily dynamic to evolve as agent communications proceed. It can be represented as a graph-theoretical model of agent evolution.[2] The objectives of this research are: (1) provision of a model for agent evolution and associated rules; (2) construction of routines to estimate agent evolution; (3) guidelines to write intelligent mobile agent programs; and (4) strategies to construct efficient WM-ACNs. Issues regarding network reliability and security mechanisms in the operating environment are deferred to a later work.

Given an WM-ACN, the model determines which agent evolution policy negotiates the required quality of service (QoS) associated with multimedia services activated by the user's request, or equivalently, which policy achieves the goal(s) of the agents. Given a change in the structure of the WM-ACN, the model is able to find out how to adapt the agent evolution policy in order to recover from the change or determine how multimedia QoS attributes are affected by the change.

A background in third-generation IP multimedia (IM) service networks and the application of mobile agents to their operation is presented in Section 2. Section 3 introduces terms and definitions to be used in the WM-ACN development. Agents evolve based on a state transition diagram, discussed in Section 4. A graph-theoretical model describes agent dependencies and competition in Section 5. Agent evolutionary computing algorithms, on which the simulations can be constructed, are addressed in Section 6. And, finally, conclusions based on the model development and future extensions in the context of wireless multimedia networks are discussed in Section 7.

2. BACKGROUND

The International Telecommunications Union (ITU) has developed requirements for third-generation (3G) mobile communication networks to provide anywhere, anytime, bandwidth-on-demand multimedia services to users. These services include toll-quality voice, variable-rate video, and high-speed data of 144 (384) kilobits per second (kbps) for high- (low-) mobility users with wide-area (microcell) coverage and up to 2 Megabits per second (Mbps) for stationary and indoor users with picocell coverage. The current leading proposal to meet the ITU 3G objectives is wideband direct-sequence (DS), code division multiple access (W-CDMA), based on a coded spread spectrum technique. The network concept is known alternately as W-CDMA in Japan and Korea and as the Universal Mobile Telecommunications System (UMTS) in Europe. A converged version of the concept has come to be called 3GPP, after the 3G Partnership Project. Details of the W-CDMA radio interface can be found numerous articles based on 3GPP specifications.[3] The 3G network architecture is comprised of circuit-switched, packet-switched and IP access networks to support emerging as well as legacy services and interfaces. The focus of the application of mobile agents is to the service management functions in the IM subnetwork.

2.1 Classes of wireless IP multimedia service

Several types of proposed 3G IM services can be grouped into two broad classes: real-time or delay-sensitive, such as voice and interactive video, and non-real-time or delay-insensitive, such as packet data, imaging, text file transfer, and Internet browsing. Each service type must be guaranteed a QoS level that can be related to received signal strength (RSS), signal-to-interference ratio (SIR), bit-error rate (BER), frame erasure rate (FER), system latency, and outage probability on the wireless connections between the distributed processing platforms at base stations (BSs) and mobile subscribers (MSs).

Real-time services can be variable-rate, such as, the 8-kbps and 13-kbps voice codecs or interactive video in which excessive delay or delay variation noticeably degrades service. In real-time modes, a large amount of digitized information is transmitted over a relatively long duration. Non-real-time services, such as, file transfers, Internet accesses, e-mail and other delay insensitive ones are transmitted by IP networks as high-rate bursts, characterized as on-off processes. For packet data services, transmission stops at the end of the data burst, since no information is generated during the unpredictable off intervals. Transmission of real-time services is continuously maintained during the call. Packet data services are provided to users with demand for high transmission rates, but short service times. Certain non-real-time packet data services differ in their tolerance of delay variation as opposed to fixed transfer delay. For example, many web pages already include real-time video and audio clips. Service providers are beginning to offer Internet access to full MPEG-coded films, documentaries and news programs. In these applications, overall delay is not as important as delay variation in order to avoid jerky pictures. Table 1 summarizes the delay requirements of various services.

Three classes of services, similar to those studied in [4], are accommodated in the network model. Both constant bit rate (CBR) and variable bit rate (VBR) services are assumed in each class.

- Class 1 services encompass highly delay-sensitive, real-time connections with very low delay-tolerance, such as voice and interactive video and video conferencing. Class 1 typically receives the highest service priority over other classes.

Service	Requirements	
	Delay	Delay Variation
File Transfer	Insensitive	Insensitive
Web Browsing	Insensitive	Insensitive
E-mail	Insensitive	Insensitive
Voice over IP	Very low	Very low
Video telecom over IP	Low	Low
Real-time Video	Insensitive	Low

Table 1. Delay-related requirements for typical packet data services

- Class 2 services include non-real-time, delay-sensitive, connection-oriented services with limited delay requirements such as remote login, file transfer protocol (FTP), and similar applications associated with the transport control protocol (TCP). This class typically receives lower priority than Class 1.

- Class 3 services are message-oriented and delay-tolerant. Typical services are paging, e-mail, voice mail, facsimile, and data file transfer. They can be packet- or circuit-switched. Class 3 services can be conveyed at the earliest possible time.

2.2. Quality of service and radio resources

Class-based QoS measures are incorporated in the model. BER, FER, delay limits, call dropping and blocking are among the QoS measures selected to make the model representative of actual IP network operation.

Bit Error Rate, Frame Error Rate. BER is a major indicator of the QoS and is related to the average SIR per received bit, or E_b/I_0. The relationship between BER and E_b/I_0 is one-to-one and is determined by the radio resources (RRs) of channel coding, modulation, receiver sensitivity, processing gain, macro-diversity, and transmit output power in the network's terminals as well as by channel fading statistics and interference in the mobile environment. BER can be directly related to the RRs of downlink transmit power from the BSs of the active set to the MS, and uplink transmit powers from the MSs to a target BS. Respectively, these power outputs also cause mutual interference in the downlink and uplink. Another RR factor of BER is the processing gain from the use of variable spreading codes or, equivalently, the number of transport channels to provide increased effective bandwidth. Another RR factor is FEC coding gain. Coding gain enables the transmitter to reduce power and achieve the same error rate. Selection of adaptive antenna beamforming within a cell can simultaneously increase the gain factors of BER and reduce the co-channel interference. The use of voice activity monitoring (VAM) increases the RR utilization based on the activity factor in speech. In the idle periods of speech for one connection, the RRs can be reallocated to provide service to other connection requests. This reduces the average signal power of all users from a uniform population and ensures, based on the weak law of large numbers, that the interference is nearly average most of the time.

Delay Limits. Class 2 and Class 3 services can be queued during periods of deep fading, high interference, or other channel degradation, to allow allocation of more RRs to higher-priority services. Delay and delay variation are caused by the number of handoffs that occur over the call's duration, by handoff failures, and by queueing delays in the case of retransmitted packets for packet data services. The queueing delay at the mobile can be controlled to not exceed a target delay limit (DL) for each MS. Other types of delay are assumed relatively small or allocated to other network mechanisms. DL is denoted either by a maximum tolerable delay, Λ_j for user j, or by the average allowable delay, $\overline{\Lambda}_j$.

Call Dropping and Blocking. The delays and outages at the network level are due to handoff decisions in the presence of dynamic loading by new and existing user requests for the finite RRs available at the network nodes. Frequent call handoffs may greatly increase the signaling load on the switch processors to negotiate the same QoS on the new connection path. Under these conditions, if queueing of non-real-time traffic fails due to queue overloads and excessive delays, one or more active services in the user request may have to be dropped to release more RRs for higher-priority service requests. The call processor will quickly become the network bottleneck, resulting in longer handoff times and possible handoff dropping. In the model, if there are no feasible RR allocations to service the handoff request, the request is delayed (suspended) or terminated. Calls arriving to the network are blocked from entry for the same reasons. Outages, due to handoff dropping, can lead to premature call termination and are considered worse than new call blocking.

In W-CDMA, a number of options are available to integrate multimedia services: (1) trade off processing gain for an increased information rate in the same spread bandwidth and (2) pair up basic data channels until the required information rate is obtained. The phrase "basic channel" refers to the CBR transmission with the highest processing gain. The Radio Resource Manager (RRM) in the Radio Network Controller (RNC) fully controls the choice of appropriate coding scheme, interleaving parameters, and rate-matching parameters.

The media access control (MAC) protocol regulates the data stream delivered to the physical layer over the transport channels. If the response to the MS's query carries different services, e.g., a real-time service and delay-insensitive data service, each service is assigned a different set of transport formats. As for a single service, the MS may use any transport format assigned for real-time services, whereas it may only use the transport formats for the non-real-time data service. The MS is assigned a specific output power/rate threshold. The aggregate output power/rate will never exceed the threshold. Thus, the transport formats used for data service fluctuate adaptively to the transport formats used for voice.[5]

2.3. Network architecture

The network consists of up to $N_{\max}$ users and M_1 fixed processing nodes or stations. Service requests from the MSs can come to the IM subnetwork from a variety of mobile computer platforms as shown in Figure 1. The active set size, *AS*, the number of receive signals from different BSs monitored by each MS to determine the handoff decision is assumed fixed.

A mobile user j can be represented by the mobile user profile at time n by the vector:

$\boldsymbol{X}_j(n) = [\boldsymbol{RSS}_{j,1}(n), \ldots, \boldsymbol{RSS}_{j,AS}(n); \ell_j(n); \boldsymbol{v}_j(n); \boldsymbol{p}_j(n); \rho_{j,1}(n),..., \rho_{j,S}(n)]^T$, where at time n, $\mathrm{RSS}_{j,i}(n)$ is the measured received signal strength from station i of the active set at MS j (for inactive or non-monitored BS i, $\mathrm{RSS}_{j,i}(n)=0$); $\ell_j(n)$ is the approximate location of MS j as measured by the positioning technique in the system; $v_j(n)$ is the speed of MS j determined as the scalar of the velocity vector; $p_j(n)$ is the output power of MS j; and $\boldsymbol{\rho} = [\rho_{j,1}(n),..., \rho_{j,S}(n)]^T$ with $\rho_{j,s}(n)$ is the rate or other QoS measure of service s to which MS j has subscribed, out of S services available simultaneously in the network, with inactive service or service completion denoted by $\rho_{j,s}(n) = 0$. All information about the incoming call must be provided to the mobile agents at the receiver processing platforms.

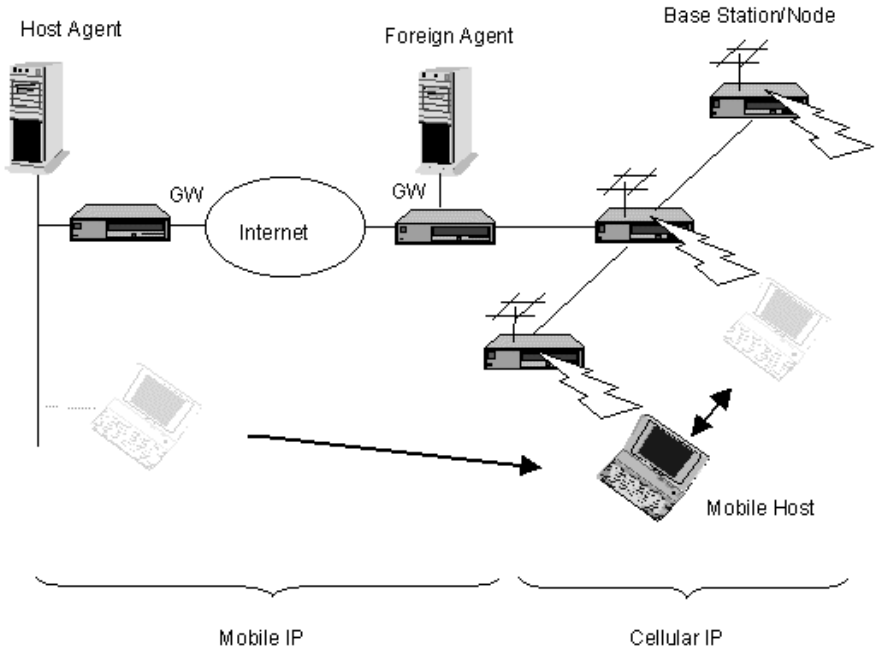

Figure 1. Typical mobile IP and cellular IP network architecture

2.4. Mobile software agent features

Agent communication languages allow agents to share information and send messages to each other. Agent architecture, on the other hand, encompasses network architecture and software architecture that support agent computing.

Agent Tcl, a mobile-agent system providing navigation and communication services, security mechanisms, and debugging and tracking tools. The system allows agent programs move transparently between computers. A software technologies, Telescript, with safety and security features, and the mobile agent architecture, MAGNA, have been evaluated in the literature.[6] A mobile agent technique to achieve load balancing in telecommunications networks is proposed.[7] The mobile agent programs discussed can travel along network nodes to suggest routes for improved communications. Mobile service agent techniques and the corresponding architectural principles as well as requirements of a distributed agent environment for UMTS are discussed in [8].

3. ELEMENTS OF THE AGENT COMMUNICATION NETWORK

Agents communicate with each other since they can assist one another. For example, agents sharing the same search query should be able to pass query results to each other so that redundant computation is avoided. The WM-ACN supports this agent communication feature. Each node in the ACN represents an agent on a network processing node with computational resources; each link represents a logical network connection (or agent communication link). Since agents with the same goal desire to share results of their search with each other, they are modeled as a complete graph. Therefore, an ACN of agents that share different goals is a graph of complete graphs.

Some key terms and definitions are introduced. A *host station* (or simply *station*) is a networked workstation or server on which agents reside. A *query station* is a station where a user releases a query or request for service that requires achieving a set of goals. A station can contain multiple agents. Similarly, an agent can pursue multiple goals. An *agent society* (or *society*) is a set of agents fully connected by a complete graph, with a common goal associated with each agent in the society. A common goal belonging to different agents may have different priorities. An agent society with a common goal of the same priority is called a *species*. Since an agent may have multiple goals, it is possible that two or more societies (or species) intersect or overlap. A *communication cut set* is a set of agents that belong to two distinct agent societies, which share one or more common goals. Removal of all elements of a communication cut set results in the separation of the two distinct societies. An agent in a communication cut set is called an *articulation agent*. Since agent societies (or species) are represented by complete graphs and these graphs have communication cut sets as intersections, articulation agents can be used to model a shortest network path between a query station and the station where an agent can satisfy its goal. In addition, an articulation agent can hold a *repository*, that is, a database containing the network communication status of links of an agent society. Network resources can thus be evaluated when an agent checks its surviving environment to decide its next evolutionary step.

In the context of the IM subnetwork of a 3G wireless system, the following identification is made between the elements of an ACN and the elements of wireless multimedia. Goals correspond to the information and associated services contained in the user request. As such, goals depend on the attributes or factors that define the QoS of the service, i.e., establish the minimum acceptable performance corresponding to a given service type, i.e., voice, data, file transfer, video. An agent society or species corresponds to an application class that provides a service type to a subscribed set of users. A communication cut set corresponds to the set of agents that support two or more services in response to two or more requests during overlapping time periods. A host station corresponds to a information-processing platform, located at a fixed or mobile network node, on which applications reside to support one or more IM services. A query station is a fixed or mobile terminal from which a subscriber to network services initiates a request that established the goal of the query agents. The repository can be thought of as the radio network resource database, location management database, etc., in which the updated status of wired and wireless communication links is maintained. Mobile agents are the enablers of the requests for multimedia services and their major objective is to satisfy the end-to-end QoS requirements of all user requests through the adaptive process of radio resource allocation (RRA).

In the formal notation of the terms, definitions and correspondences introduced, symbols used in the structure of the algorithms. In the following definitions, "≡' denotes "is defined as" and "$\mathbf{P}\ X$" denotes a set of object X:

$$Host\_Station \equiv URL \times Resources \times \mathbf{P}\ Agent$$

$$Resources \equiv Bandwidth \times Power \times Voice\ Monitoring \times Antenna\ Control \times CPU \times Memory \times Channels \times Applications$$

$$Agent \equiv \mathbf{P}\ Goal(QoS\ Factors) \times Evolution\ Policy$$

$$Goal \equiv Query\_Return\_URL \times Service\ Request \times Service\ Type \times Priority$$

$$Agent\_Society \equiv \mathbf{P}\ Agent$$

$$Species \subset Agent\_Society$$

A *Host_Station* has a uniform resource locator, i.e., URL or IP address, at which there resides applications for information services. A host station has system resources in the object *Resource*, and can contain some agents in the set $\mathbf{P}$ *Agent*. *Bandwidth* × *Power* × *Voice Monitoring* × *Antenna Control* represents the portion of the radio resources available to a station. *CPU* represents the computational power of a station. *Memory* represents the storage of the station, and includes both the primary and auxiliary memory. Applications are programs that enable services on a station. Each *Agent* has some *Goals* and an *Evolution Policy*, which is a set of application-dependent factors on which the agent depends to perform its evolution computation. *Query_Return_URL* is the URL where an agent should return its query results. *Service Request* is an application-dependent specification that represents a user request activating the agent. *Service Type* represents one or more of multimedia services that are in the user request and supported by the network. *Priority* is an integer that represents the priority of a goal and is related to the ordering the execution of service requests according to the QoS factors by service type. The larger the integer, the higher the goal or service priority. *Agent_Society* is a set of agents with a common goal. *Species* is an *Agent_Society* with the same goal priority.

Common object-oriented notation is used in the agent evolution algorithms to represent a component or attribute of an object. As an example, if agent *A* is used, then *A.Goal* represents the goals of that agent, where *A* is unique in the agent society or species to which it belongs. Before describing the evolution algorithms, the concepts of evolution states and the

graph model of the Food Web for species are first introduced.

4. AGENT EVOLUTION STATES

An agent evolves in reaction to an environment, in response to other agents, and in communications with other agents. The process of agent evolution is expressed in terms of the internal states of agent. An agent is in one of the following states after it is born and before it is killed or dies for reasons discussed in the following.

- Searching: the agent is searching for a goal.
- Suspending: the agent is waiting for enough resources in its environment in order to search for its goal.
- Dangling: the agent losts its goal of surviving, it is awaiting a new goal.
- Mutating: the agent is changed to a new species with a new goal and the agent survives in a new host station.

An agent is born (activated) to a *searching state* to search for its goal, i.e., an information-bearing service. In the Food Web analogy, all creatures must have goals, such as, the search for food. However, if the surviving environment, herein, a host station, does not contain enough of the resource(s), the agent may transition to a *suspending state*, equivalent to the hibernation of a creature. The searching process will be resumed when the environment has the available resources. But, if the environment lacks the required resources for a prolonged time period, the agent may be killed. When an agent finds its goal, the agent will pass the search results to other agents of the same society or species. Other agents will then abort their search, since the goal has been achieved, and transition to a *dangling state*. There is an inherent time limit for the period an agent can remain in the dangling state before it dies. On the other hand, the agent may be reassigned to a new goal associated with a different host station, which is a new destination where the agent must travel. In this situation, the agent is in a *mutating state* and is reborn to search for the new goal. In order to maintain agent activity, in a distributed multimedia retrieval and computing environment, message passing is used as the mechanism to control agent state transitions.

5. FOOD WEB AND GRAPH MODELS OF AGENT SOCIETIES

Agents can suspend/resume their activity or even kill other agents. A general policy is required to determine which agent is killed. In this construction, a species is a set of agents having the same goal with the same priority. Two or more societies are differentiated on the basis of the priority of the goal, which relates to the QoS of the service. A direct graph is constructed to model the dependency among species. This digraph is called a species food web. Each node in the graph represents a species. All species of a connected food web, or, equivalently, the graph model of a food web, possess the same goal. This model assumes the possibility that different users at different host stations of the network may issue the same query. Each directed edge of the digraph has an origin, representing a species of a higher goal priority and terminates at node (species) for which the goal has a lower priority. Since an agent, and consequently, a species can have multiple goals, each goal of an articulation agent should have an associated food web.

Each food web describes the goal priority dependencies of species. From a food web can be derived a *niche overlap graph*. In a natural ecosystem, two or more species have an ecological *niche overlap* if and only if they are competing for the same resource. A *niche overlap graph* can be used to represent the competition among species. The niche overlap graph is used to decide agent evolution policy and to estimate the effect when certain service-dependent factors are changed in an ACN. Based on the niche overlap graph, the strategies in algorithms are able to rearrange evolution policies to optimize the performance efficiency of the agents. This concept is analogous to the process that enables recovery from perturbations in natural ecosystems.

6. AGENT EVOLUTION COMPUTING IN WIRELESS MULTIMEDIA NETWORKS

The algorithms presented in this paper use the agent evolution state diagram and the niche overlap graphs for the purpose of agent evolution computing. First, some heuristic approaches are discussed to explain the fundamental concepts of agent searching and agent distribution. A set of agent evolution computational algorithms over a WM-ACN are then presented.

6.1 Agent searching and agent cloning

An agent exists to search for its goal. At the same time, since the searching process is distributed over the network stations, an agent desires to find a destination location on which to clone itself. Searching and cloning exist in a *co-routine* relationship. A co-routine can be considered a pair of processes. While one process serves as a producer, another serves as a

consumer. When the consumer uses up a resource, the consumer is suspended. When this occurs, the producer is activated and produces or allocates the resource until the resource reaches an upper limit. The producer is then suspended and the consumer resumes. If the searching process is a consumer, then the cloning process is producer who provides new URLs and associated resources. The following algorithms describe agent searching and cloning processes.

Algorithm 1. Search and Clone co-routines

```
Algorithm Search (G):
given a goal G
        repeat
                if goal G is found, then
                terminate Search
        else
                if URL_queue or Resource is empty, then
                        suspend Search until Clone returns
                else
                        search on a URL for goal G
                        and delete the URL from the queue

Algorithm Clone:
        repeat
        if URL_queue is full, then
                suspend Clone until Search returns
        else
                find out and place next URL with Resource  in the URL_queue
```

The **Search** and **Clone** co-routines use a queue to store URLs and assume the existence of a database where a current account of available network resources is stored. When the URL queue is empty, the algorithm **Search** is suspended until **Clone** returns. Otherwise, a URL in the queue is used to propagate the agent. Algorithm **Clone** collects new URLs via a search engine until the URL queue is full. However the operation of these co-routines is not efficient, since **Search** or **Clone** wait for each other until the URL queue is full or the URL queue or the Resource set is empty. This disadvantage can be removed using a concurrent algorithm of the two processes.

Algorithm 2. Concurrent Search and Clone algorithm

```
Algorithm Search_Clone(G):
given a goal G
        cobegin
                process Search:
                    repeat
                        if goal G is found, then
                            terminate Search_Clone
                        else
                            if URL_queue and Resource are not empty, then
                                search on a URL for goal G
                                and delete the URL
                process Clone:
                    repeat
                        if URL_queue is not full and the Resource is not empty, then
                            put the next URL in the URL_queue
        coend
```

The concurrent algorithm searches and propagates simultaneously when both the queue is not empty and resources are available or when the queue is not full. When these conditions are met, the two processes are performed concurrently. When the mobile agent, implemented either in the concurrent algorithm or co-routine algorithm, travels to a station, a local URL queue and resource database is used and the computation proceeds autonomously, in a decentralized manner. The above two approaches describe the relation between searching and cloning of agents. However, there is no communication among agents in either approach. All agents compete for the same goal and multiple copies of the same result will be sent back to the station originating the query. This situation both wastes CPU time and network resources. To overcome this

disadvantage requires an agent communication network in the multimedia IP network where agents evolve.

6.2 Agent evolution computing over a WM-ACN

The co-routine and concurrent algorithms discussed in earlier sections are created in the context of a single station. However, agent evolution on the WM-ACN involves asynchronous computations. Agents reside on distributed stations and interact with each other. The search and cloning processes of an agent may operate in a co-routine manner on a station. However, different agents are run concurrently on the same or distinct stations. Algorithm **Agent_Search** is the starting point of agent evolution. If the network resources meet a basic requirement, related to the minimum QoS for the service underlying the response to the query, the algorithm activates an agent in the searching state. If the search process finds its goal, e.g., the requested information or service is found, goal termination results in a dangling state of all agents in the same society, including the agent that finds the goal. At the same time, the search result is sent back to the original query station. Suppose that the goal cannot be achieved at a single station, the agent is cloned in another station following the routine of agent propagation. The **Agent_Clone** algorithm is then used. On the other hand, the agent may be suspended or even killed based on the availability of appropriate network resources to support the response to the query. Some auxiliary algorithms, based on digital versions of Kohonen's self-organizing feature maps (SOFMs), are used to allocate RRs to support the QoS of the services activated by the user queries.

Algorithm 3. Agent Search algorithm

```
Algorithm Agent_Search(A, G, X):
    given a goal G to agent A on station X of society S
    if Resource_Available(A, G, X) > acc_requirement(Q(G)) then
        agent A searches for G in its station X
        if G is found, then
            A sends an abort message to agents in S
            A sends search result to query station
            Agent_Search is complete
        else
            call Agent_Clone(A, G, S)
            terminate Agent_Search
    else if Resource_Available(A, G, X) > min_requirement(Q(G)) then
        call Agent_Suspend(A, G, X)
    else
        call Agent_Kill(A, G, X)
```

In the above algorithm, *acc_requirement*, the acceptable level of the resource(s) must be greater than or equal to *min_requirement* to satisfy the quality of service $Q(G)$ of the goal G. In this way, different degrees of the same service type are allowed in adaptation to the available amount of required network resource(s). The resource availability parameter depends on the service type and the agent evolution policy, as defined in the algorithm **Resource_Available**.

Algorithm 4. Agent Cloning algorithm

```
Algorithm Agent_Clone(A, G, S):
    given a source agent A searches for goal G of society S
    use search engine to find a new URL
        on an arbitrary station X that may contain goal G
    if station X has an agent A' then
        if goal of A' contains G then
            let S' be the society associated with G
                where A' belongs
            union societies S' and S
        else
            assign G to A'
            make A' join G
        call Agent_Search(A', G, X)
    else
        copy a new agent A'' of A on station X
        make A'' join society S
        call Agent_Search(A'', G, X)
```

Agent cloning is effected by the **Agent_Clone** algorithm. When the cloning process finds new URLs to broadcast an agent, two strategies can be employed. The first is to broadcast the agent to all URLs found by one search engine. But, in view of the network resources available, a second strategy could determine the common URLs found by two or more search engines. The cloning agent checks whether there is another agent at the destination URL or station. If there is, the algorithm checks whether the agent already at the URL shares the same goal with the agent to be cloned. If the two agents share the same goal, there is no need to clone another copy of the source agent. Thus, the goal can be computed by the agent at the destination URL. To maintain the structure of the ACN, this decision requires the union of the two societies to which the two agents belong. However, if the two agents do not have the goal in common, to save computational resources, the agent at the destination URL may be asked to assist in the search of the source agent's goal. The latter situation forces a reorganization of the society to which the source agent belongs. The action taken keeps the number of agents in the ACN to a minimum. Whether the source and destination agents share the same goal, the **Agent_Search** algorithm is again used to search for the goal. When there is no agent running on the destination station, the number of agents on the WM-ACN is increased by duplicating an agent on the destination URL. The society is then reorganized to reflect the increase and the **Agent_Search** algorithm is called again.

6.3. Auxiliary algorithms for agent evolution

Auxiliary algorithms are called within the search and cloning algorithms that are crucial to an understanding of the evolutionary dynamics of agent states within the wireless IP network. Modifications of these subordinate routines allow specific representation of the mobile agents' support of multimedia services in different operating scenarios. The first routine describes how agents transfer to and from the *suspending* state.

Algorithm 5. Agent Suspend algorithm

Algorithm **Agent_Suspend**(*A*, *G*, *X*):
given a goal *G* to agent *A* on station *X*
wait until **Resource_Available**(*A*, *G*, *X*) > *acc_requirement*(*G*)
call **Agent_Search**(*A*, *G*, *X*)

The second routine describes how agents are killed or die in the ACN.

Algorithm 6. Agent Kill algorithm

Algorithm **Agent_Kill**(*A*, *G*, *X*):
given a goal *G* to agent *A* on station *X*
terminate agent *A* on station *X*

There are several variations to the third auxiliary routine, since it relates to the complexity of the simultaneous distributed support of the different services associated with the goal of user requests and the maintenance of their QoS through the allocation of network resources. The simplest version of the **Resource_Available** algorithm applies to an IP network where one grade of service is provided as the result of an agent satisfying the goal associated with a user query.

Algorithm 7. Resource Available algorithm – simple combinatorial version

Algorithm **Resource_Available**(*A*, *G*, *X*):
given a goal *G* to agent *A* on station *X*
switch *A.Policy*
case *discrete_sim* $\wedge$ *network_bound* then
Available = *X.Resource.Network*
case *discrete_sim* $\wedge$ *cpu_bound* then
Available = *X.Resource.CPU*
case *discrete_sim* $\wedge$ memory_bound then
Available = *X.Resource.Memory*
case *discrete_sim* $\wedge$ *cpu_bound* $\wedge$
memory_bound then
Available = *X.Resource.CPU* $* w_1$ + *X.Resource.Memory* $* w_2$
case *discrete_sim* $\wedge$ …
Available = …

Algorithm 7. Resource Available algorithm – simple combinatorial version (continued)

```
        case internet_sim
            Available = resource available on X
        if G.Priority is low then
            Available = Available ∗ r
```

where w_1 and w_2 are convex weights, i.e., $w_1 \geq 0$ and $w_2 \geq 0$ with $w_1 + w_2 = 1$. In the simple **Resource_Available** algorithm, a restrictive description of an agent policy, *A.Policy*, is presented. If the goal's priority, *G.Priority*, is low, the constant r, with $0<r<1$, used to represent the fact that resources are reserved for other agents.

In general, as the number of resources increases to meet the diversity of multimedia services associated with the goals generated by simultaneous user requests at the distributed terminals of a wired/wireless IP network, the resulting complexity and dimensionality overwhelms the simple combinatorial approach of the **Resource_Available** algorithm. The approach to agent evolution must be expanded to encompass the size of the dynamic resource allocation problem. The use of a discrete-form of the Kohonen SOFM has been applied before by this author to the problem of allocating network resources to satisfy the dynamic QoS requirements of calls in a multimedia cellular network.[9] An application of this approach is invoked to replace the **Resource_Available** algorithm wherever it is called by other agent evolution algorithms.

6.4 Self-organizing feature map for dynamic resource allocation

The impact of the adaptive radio resource (RR) elements of the network model on interference and the dependency new agent goals on network conflict or interference suggest representing their effect on the state of mobile agents as a composite mapping, $\Psi = \Phi \bullet \Gamma$, on an $R \times M$-dimensional lattice. The composite mapping relates the radio resource allocation (RRA) to cell assignments of new and handoff calls, through the interference that the assignment generates in the cells. The concept of the mobile agents "competing" for network RRs to satisfy the service requirements in the goals from mobile user service profile vector $X_j(n)$ in time slot n suggests application of the SOFM approach to the problem of agent evolution policies based on adaptive RRA. The approach modifies Kohonen's SOFM to solve a discrete-space optimization problem of organizing the service demands in user profile vectors into a pattern of cell assignment among the lattice of RRs.[10] Development of a static RRA (SRRA) and extensions to DRRA problems are discussed in a previous work.[9] All feasible solutions to the RRA problem lie at the vertices of an $\aleph$ - dimensional hypercube, where $\aleph = R \cdot M$ and R is the available number of RR and M is the number of distinct components: distributed processing nodes, computer platforms, or base stations that comprise IP subnetwork. Note that $\aleph$ is the dimension of the domain of Ψ. The image of the vertices also intersects the constraint hyperplane defined by the limits on the RRs at network components. Since each entry ρ_{ij}, of the service profile vector of each user can be assumed integer-valued for all i and j, the image of the RR constraints set can be shown to form an integral polytope. Consider the neurons on this hypercube, defined as

$$x_{ij,\mathbf{r}} = \begin{cases} 1, \text{if cell } j \text{ in cell layer } i \text{ is assigned RR vector } \mathbf{r}, \\ 0, \text{otherwise}, \end{cases}$$

for j = 1, ⋯, N; and $\mathbf{r}$. Let $\boldsymbol{X}$ denote the $\aleph$ - dimensional array of these variables. Normalizing the range of values for each RR and the interference bounds to the interval [0,1], the set of RRs and its Γ-image in the interference range are each contained in unit hypercubes. A vertex is approached continuously from within the unit hypercube, starting from a point on the constraint hyperplane and inside the hypercube. This represents a feasible, non-integer solution to the RRA problem. The continuous variable approach in the interior of the hypercube is denoted by $w_{r,ij}$, so that, for a quality metric Q, $Q(\boldsymbol{W}) = Q(\boldsymbol{X})$ at the vertices. The value $w_{r,ij}$ represents the *probability* that the variable in position $(\mathbf{r}, i, j)$ of the array $\boldsymbol{X}$ is activated. The vector $\mathbf{r}$ is integer-valued, an index into the lattice of allowable RRAs. Kohonen's self-organization is applied to the array of *synaptic weights*, $\boldsymbol{W}$. This modification permits the SOFM to solve discrete-space optimization problems.

The structure of the discrete-space SOFM consists of an input layer of N nodes, and an $R \times N$-dimensional array of output nodes. The output nodes correspond to the solution array of discrete-valued RRAs, while the input layer represents the N BS coverage areas in the W-CDMA network. The weight connecting input node j to node $\mathbf{r}$ of the output array of nodes is given by $w_{r,j}$. A processing station in which an assignment of $\mathbf{r}$ is required is presented to the network through the input layer at node j. Physically, the user request is presented to the network at BS j. The nodes of the output layer compete with each other to determine which subarrray of the solution array to meet the QoS requirements of the input with minimal impact on the cost potential. The synaptic weights are then *adapted* to indicate the RRA decision using the neighborhood topology.

Consider the case where RRA $\boldsymbol{\rho}$ is required at BS i^* due to assignment of user service profile X_j. An input array $\boldsymbol{x}$ is presented to the network with a "1" in position (i^*, j) and 0 elsewhere. For each node $\boldsymbol{\rho} = (\rho_1, \rho_2, \cdots, \rho_R)$ of the outer layer, the value V_{r,i^*j}, the cost to the objective function of RRA $\boldsymbol{r}$ to BS i^*, is computed. The *cost potential* $V_{\boldsymbol{r},i^*j}$ of node $\boldsymbol{r}$ for a given input array $\boldsymbol{x}$ ($x_{ij} = 0$, $\forall i \neq i^*$, $x_{i^*j} = 1$) is defined by

$$V_{\boldsymbol{r},i^*j} \equiv \sum_{i=1}^{M} \sum_{\boldsymbol{s}\in\Re} P_{i^*,i,j,(\|\Psi(\boldsymbol{r})-\Psi(\boldsymbol{s})\|+1)} w_{\boldsymbol{s},ij} \tag{1}$$

where interference caused by the RR assignment for MS j is represented by proximity indicator or weight $P_{i,k,j,d+1}$, where $d = \|\Psi(\boldsymbol{r}) - \Psi(\boldsymbol{s})\|$ is the distance or "cost" difference in cell-assignment space between the images of RRAs $\boldsymbol{r}$ and $\boldsymbol{s}$. If $\Psi(\boldsymbol{r}) = \Psi(\boldsymbol{s})$, then the associated cost would be at a maximum, with cost decreasing until the two cell and RR assignments are sufficiently separated, so that the resulting interference and contention for resources are each below design threshold values. The array $\boldsymbol{P}$ is defined as

$$P_{k,i,j,d+1} = \max(0, P_{k,i,j,d-1}),\ d = 1, \cdots, N-1; \quad P_{k,i,j,1} = c_{ji},\ \forall\ k,\ i \neq k; \quad P_{k,k,\,j,1} = 0,\ \forall\ k, j. \tag{2}$$

The *dominant node*, $\boldsymbol{m}_0$, of the outer layer is the node with minimum cost potential V_{r,i^*j} for a particular input vector. In terminology, $V_{\boldsymbol{m}_0,i^*j} \leq V_{r,i^*j}$ for all nodes $\boldsymbol{r}$ and fixed i^*. The *neighborhood* of the dominant node $\boldsymbol{m}_0$, is the set of nodes $\boldsymbol{m}_1$, $\boldsymbol{m}_2$, $\cdots$, $\boldsymbol{m}_{\eta_{i^*}}$, ordered according to $V_{\boldsymbol{m}_0,i^*} \leq V_{\boldsymbol{m}_1,i^*} \leq V_{\boldsymbol{m}_2,i^*} \leq \cdots \leq V_{\boldsymbol{m}\eta_{i^*},i^*}$, where η_{i^*} is the size of the neighborhood in the SOFM for BS i^*. Dominant nodes and their neighborhoods are thus determined by competition among agents' goals according to the objective function, and the weights are modified according to Kohonen's weight adaptation rules within the dominant neighborhood.[10] The size of the dominant neighborhood depends upon which network component is receiving goal generated by the request of the user with profile X_j and the level of service in $\boldsymbol{\rho}_j$.

When weight updating is complete, the array $\boldsymbol{W}$ has been moved in a direction that may be away from the constraint hyperplane, resulting in an infeasible solution. In the next step, the weights of the nodes outside the dominant neighborhood organize themselves around the modified weights, so that $\boldsymbol{W}$ remains a feasible solution to the RRA problem during the update. This step can be performed by a hill-climbing HNN or HC-HNN. Representing the weight matrix $\boldsymbol{W}$ as a vector $\boldsymbol{w}$, $\boldsymbol{w}$ is considered a vector of states of a continuous HNN. The HNN performs random and asynchronous updates on $\boldsymbol{w}$, *excluding* the weights in the dominant neighborhood, to minimize the energy function:

$$\boldsymbol{E} \equiv \left\|\boldsymbol{w} - (\wp\boldsymbol{w} + \boldsymbol{\tau})\right\|^2 \tag{3}$$

where $\wp$ is the projection onto the constraint hyperplane defined by multidimensional RR limits and $\boldsymbol{\tau} = (\boldsymbol{I} - \wp)\boldsymbol{T}$, where $\boldsymbol{I}$ is the identity operator and $\boldsymbol{T}$ is the vector of upper limits on the resources. The energy function (3) is expressed in terms of a solution vector $\boldsymbol{x}$, constructed from the solution array $\boldsymbol{X}$, by ordering the elements $x_{i,k,r}$ according to an integer ordering of RR space indices and number of the S service classes.

The SOFM-based RRA algorithm updates within one network time slot after the last random request from a mobile user or completion of the corresponding and subsequent handoff attempt. The initial assignment of node k is determined by the position ℓ_j and speed v_j in the user profile X_j. Each update determines a new dominant node and its neighborhood of nodes and modifies their synaptic weights. The procedure is repeated until the SOFM weights stabilize to a feasible 0-1 solution for cell membership in all cell layers. This stable solution is a local maximum to the RRA optimization problem.

As the algorithm converges, the magnitude of weight modifications and the size of the neighborhoods are decreased. Initially, the size of the neighborhood for each subarray of $\boldsymbol{W}$, given by $\eta = (\eta_1, \eta_2, \cdots, \eta_M)$ is large, but is decreased incrementally until $\eta_j = \|\hat{\boldsymbol{\rho}}_j\|$, the total level of service demand at BS j in all M cells. Since weight modifications depend on the order in which the connection requests are input, the SOFM approach is inherently stochastic.

The following procedure for the SOFM algorithm can be applied to the RRA problem in mobile agent communication networks to enable IM service requests. (It is assumed that these user service requests and service completions have exponential interarrival times of parameter λ and μ, respectively.)

Algorithm 9. SOFM Resource Allocation algorithm

Algorithm **SOFM_Resource_Allocation**:

1. Initialize the weight vectors of the network as $\boldsymbol{w}_{kj,\boldsymbol{r}} = \left\|\boldsymbol{\rho}_k\right\| / \boldsymbol{R}_{\max}$, which yields an initial feasible, possibly non-integer solution.
2. For a service request from MS j at time slot n, select the processing node at station k based on $\ell_j(n)$ and speed $v_j(n)$ in the user profile X_j. Represent this requirement as the input array $\boldsymbol{x}$. Find the position k^* (BS coverage area) which is active, i.e., $x_{k^*,l} = 1$.
3. Calculate *cost potential* $V_{\boldsymbol{r},k^*l}$ for each index $\boldsymbol{r}$ in the output layer array according to (1).
4. Determine the dominant node, $\boldsymbol{m}_0$, by competition such that $V_{\boldsymbol{m}_0,k^*l} = \min V_{\boldsymbol{r},k^*i}$, $\forall \boldsymbol{r} \in \Re$, and identify its neighboring nodes $\boldsymbol{m}_1, \boldsymbol{m}_2, \cdots, \boldsymbol{m}_{\eta_{k^*}}$, where $\eta_{k^*} \geq \rho_{k^*}$ is the size of the neighborhood for input RR requirement at k^* for service class s.
5. Update synaptic weights in neighborhood of dominant node according to the rule

$$\Delta \boldsymbol{w}_{k^*j,\boldsymbol{r}} = \alpha(\eta, n)[\boldsymbol{e} - \boldsymbol{w}_{k^*j,\boldsymbol{r}}] \quad \forall \boldsymbol{r} \ni V_{\boldsymbol{r},k^*j} < V_{\boldsymbol{m}_{\eta_{k^*}},k^*j} \text{ where}$$

$$\varphi(\eta, n) = \frac{\varphi(n)\tau_{k^*}}{\left\|\rho_{k^*}\right\|} \exp\left[\frac{-\left|V_{m_0,k^*j} - V_{\boldsymbol{r},k^*j}\right|}{\sigma(n)}\right] \tag{4}$$

   which is a modified version of Kohonen's SOFM *slow* updating rule, where φ and σ are monotonically decreasing and positive functions of sampled time, $\boldsymbol{\tau}$ is a normalized weighting vector used in tie-breaking for a network node. For all other weights outside the neighborhood being updated, $\Delta \boldsymbol{w}_{k^*\boldsymbol{j},\boldsymbol{r}} = 0$. Weights are updated as $\boldsymbol{w}_{kj,\boldsymbol{r}} \leftarrow \boldsymbol{w}_{kj,\boldsymbol{r}} + \Delta \boldsymbol{w}_{kj,\boldsymbol{r}}$.
6. The weights will no longer lie on the constraint hyperplane, so an HC-HNN is applied to return to a feasible solution. The array $\boldsymbol{w}$ is modified around the weight adaptations of the SOFM into order that both constraints on resources and minimum requirements for the QoS attributes of service requested are met. If either the resource constraints or the minimum QoS cannot be satisfied, the agent with the goal of enabling the service request is suspended or killed.
7. Repeat Step 2 until RR requirements in all nodes have been selected as input vectors to the SOFM. This forms one period of the algorithm. The procedure is repeated for K periods. In each subsequent period, φ and σ are decreased according to any monotonically decreasing function.
8. Repeat Step 2 until $\left\|\Delta \boldsymbol{w}_{\boldsymbol{r},kj}\right\| \cong 0, \forall \boldsymbol{r}$, k and j. This condition is considered stable convergence of the synaptic weights for a given neighborhood size. Decrease the neighborhood sizes η_k linearly for all k.
9. Repeat Step 8 until $\eta_j = \left\|\hat{\boldsymbol{\rho}}_j\right\|$, for each network node or platform j, $j = 1, \cdots, M$.

For the multiservice demand array $\boldsymbol{\rho}$, the normalized weighting vector $\boldsymbol{\tau}$ is a heuristic used to damp oscillations in the algorithm updates.[11] Each element in the vector $\boldsymbol{\tau}$ is normalized. Following [12], SOFM parameters can be selected heuristically, with $\boldsymbol{K} = 10$,

$$\varphi(0) = \min_{1 \leq i \leq M, 1 \leq j \leq N_i,} (\rho_{ij}), \ \varphi(n+1) = 0.9\varphi(n),$$

$$\sigma(0) = 9, \ \sigma(n+1) = 0.9\sigma(n),$$

$$\eta_k(0) = \left\|\hat{\rho}_k\right\| + \lfloor M/K \rfloor, \quad \eta_k(n+1) = \eta_k(n) - 1.$$

Since feasibility is always restored during the second stage of the SOFM, any rearrangement of the agents assigned to fulfill user requests to enable a new service request is automatic subject to the resource constraints and minimum QoS requirements. If no rearrangement is possible, either the SOFM cannot converge to a feasible set of RRAs, a feasible rearrangement may be found by allowing service support at reduced QoS levels, or the agent assigned to the request is suspended until a feasible rearrangement is possible, or the agent is killed. In the latter case, the previous state of the ACN reinstated.

6.5 Revisions to agent suspend and kill algorithms

The above algorithms describe how the state of any type of agent evolves. The primary factor that determines how much agents interact depends on the level of network resources available to support the QoS of the service(s) underlying the mobile user's request. In a WM-ACN based on the 3G W-CDMA context, scenarios exist where agents suspend or even kill each other, as described previously. The niche overlap graphs of each goal are involved in these scenarios. The **Agent_Suspend** and **Agent_Kill** algorithms are revised to account for the niche overlap graphs. In the extended **Agent_Suspend** algorithm, if there is a goal with a lower priority than the goal of the searching agent, a suspend message is sent to the goal to delay its search. In the W-CDMA context, the lower priority may stem from the relatively lower delay sensitivity or other signal quality measure required by the service request of the goal. The searching agent may resume after the goal suspension, since network resources may be released as a result. Under the same circumstances, the extended **Agent_Kill** algorithm instead sends a kill message to the goal with a lower priority. The network resources available are determined, based on communications on the real-time local utilization from agents that monitor and transfer this information as they move from processing node to processing node of the distributed network. The available resources are compared to the minimum requirements to support the service QoS of the goal of the searching agent. If resumption of the search is feasible due to the determination of available resources, the **Agent_Search** algorithm is invoked; otherwise, the WM-ACN should terminate (kill) the searching agent.

Table 9. Extended Agent Suspend algorithm

```
Algorithm Agent_Suspend(A, G, X):
    given a goal G to agent A on station X
    check the niche overlap graph of G
    for each goal G' in the graph that
            with a priority lower than G
        send a suspend message to G' to delay search
    wait until Resource_Available(A, G, X) > acc_requirement(G)
    call Agent_Search(A, G, X)
```

Table 10. Extended Agent Kill algorithm

```
Algorithm Agent_Kill(A, G, X):
    given a goal G to agent A on station X
    check the niche overlap graph of G
    for each goal G' in the graph with a priority lower than G
        send a kill message to G' to terminate search
    if Resource_Available(A, G, X) > min_requirement(G)
        call Agent_Search(A, G, X)
    else
        terminate agent A on station X
```

7. CONCLUSIONS

In this work, the generic approach of a competitive Food Web model of agent communication behavior in wired IP applications is extended in three ways to create a WM-ACN. The model is modified to encompass user service requests for multimedia services on an IP network. The model is shown capable of representing the dynamics of a wired/wireless network architecture. The original available resource algorithm, used to determine if the network can support an agent's goal, is augmented with a discrete-space Kohonen SOFM to dynamically allocate radio resources to meet the QoS attributes delineated for the IM services by 3G W-CDMA system requirements. The result of these extensions is an evolutionary computational model for agent evolution in an IM network of a 3G wireless system. The main concepts presented are: (1) a model for the evolution of societies of mobile agents based on competitive interaction in a natural ecosystem; (2) a set of generic algorithms for the distributed computing of agent programs, extensible to a wide variety of mobile services and network architectures; and (3) a theoretical basis for future simulations in Java or a similar language to support the theory.

As suggested in [2], other extensions to the evolutionary model remain to be explored. One is based on agents learning from each other, i.e., exchanging information of mutual value in achieving the agents' goals. The state diagram of agent evolution is augmented with a new "learning" state. While in the "dangling" state, the agent can communicate to other agents. Computing methods can be repeated from other agents. And the agent transitions to the mutating state to wait for

another new goal. Moreover, when a computing platform at a network node does not have sufficient network resources, an agent in the suspending state can change its evolutionary policy to enter the environment prior to its transition to active searching. These are some examples of the information that the agents can learn. As an extension of the cloning process, two agents on a network computing node that share a common goal can be combined to form a new agent, i.e., the union of agents. The resulting agent may have more goals compared to the parent agents. The union or marriage of agents can be used to represent a hybrid service request or request for two or more simultaneous services in the IM network. An agent union or offspring state could further augment the agent evolution state diagram.

The agent evolution model discussed is limited to the support of the multimedia service requests of mobile users active in the network. The call control signaling, call admission and congestion management, and security mechanisms among the 3G network processing elements have not been addressed. These aspects of network operation can be also handled in future extensions of the model by the introduction of new types of agents to effect these network management functions in cooperation with the mobile agents enabling multimedia service requests. Simulations, using specific 3G IM operating parameters, QoS attributes and use scenarios described in W-CDMA technical reports, must also be performed to verify and refine the algorithms of agent evolution.

ACKNOWLEDGMENTS

The author wishes to thank FIT for its support and to commend the members of the special mobile groups (SMGs) of the 3rd Generation Partnership Project (3GPP) for their continuing development of next-generation wireless concepts that can be realized by means of the adaptive provisioning of multimedia services.

REFERENCES

1. R. Gray, D. Kotz, G. Cybenko, and D. Rus, "Mobile agents: motivations and state-of-the-art systems," Tech. Rpt. 2000-365, Thayer Sch. of Eng., Dept. of Comp. Sci., Dartmouth College, Hanover, NH., Apr. 19, 2000. URL ftp://ftp.cs.dartmouth.edu/TR/TR2000-365.ps.Z
2. T. K. Shih, "Agent communication network – a mobile agent communication model for Internet applications," *Proc. IEEE Int. Symp. Comp. and Commun., 1999*, pp. 425-431, Jul. 1999.
3. P. Chaudury, W. Mohr and S. Onoe, "The 3GPP Proposal for IMT-2000," *IEEE Commun. Mag.*, vol. 37, no. 12, pp. 72–81, Dec. 1999.
4. S. Acampora and M. Naghshineh, "Control and quality-of-service provisioning in high-speed microcellular networks," *IEEE Personal Commun. Mag*., 1994.
5. K. Das and S. D. Morgera, "Interference and SIR in integrated voice/data wireless DS-CDMA networks – a simulation study," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 1527–1538, Oct. 1997.
6. M. K. Perdikeas, F. G. Chatzipapadopoulos, and L. S. Venieris, "An evaluation study of mobile agent technology: standardization, implementation and evolution," *IEEE Int. Symp. on Multimedia Comp. and Sys., 1999*, vol. 2, pp.287-291, Jun. 1999.
7. E. Kovacs, K. Rohrle, and B. Schiemann, " Adaptive mobile access to context-aware services," *Proc. First Int. Symp. on Agent Sys. and Appl., 1999, and Third Int. Symp. on Mobile Agents*, pp. 190-201, Oct. 1999.
8. L. Hagen, M. Breugst, and T. Magedanz, "Impacts of mobile agent technology on mobile communication evolution," *IEEE Personal Commun. Mag*., pp. 56-69, Aug. 1998.
9. W. Hortos, "Cascaded neural networks for sequenced propagation estimation, multiuser detection, and adaptive radio resource control of third-generation wireless for multimedia services," *Appl. and Sci. of Computational Intel. II, Proc. of SPIE*, vol. 3722, pp. 261-275, Orlando, FL, Apr. 1999.
10. T. Kohonen, "Self-organized formation of topologically correct feature maps," *Bio. Cybern*., vol. 43, no. 1, pp. 59-69, 1982.
11. S. Abe, "Convergence acceleration of the Hopfield neural network by optimizing integration step sizes," *IEEE Trans. Syst. Man and Cybern.-Pt.B*, vol. 26, no. 1, pp. 194-201, 1996.
12. K. N. Sivarajan, R. J. McEliece, and J. W. Ketchum, "Channel assignment in mobile radio," in *Proc. 39th IEEE Veh. Technol. Soc. Conf.*, pp. 846-850, San Francisco, CA, 1989.