Emergent Mobile Services

Michael Meisel
UCLA
meisel@cs.ucla.edu

Vasileios Pappas
IBM Research
vpappas@us.ibm.com

Petros Zerfos
IBM Research
pzerfos@us.ibm.com

Lixia Zhang
UCLA
lixia@cs.ucla.edu

***Abstract*—Delivering a robust, high-performance application service over the Internet requires careful configuration and maintenance – different functional components of the service, such as caching, data processing, or data storage components, must be deployed at appropriate locations based on the resource constraints of the network and the receiving hosts. Although this manual configuration approach works over the relatively static topology of the Internet, the highly dynamic nature of mobile, ad-hoc networks would quickly render an initially well-configured service unusable. We confront this problem by exploring *service emergence*, a new design paradigm for self-organizing services, where individual nodes *automatically* adjust the roles they play in the system without any centralized control. From simple, localized interactions between neighboring nodes, a functional system emerges that meets the desired service goals and requirements. This paper presents a roadmap for designing emergent services. We first describe an example surveillance service, which provides both motivation and grounding for service emergence. Based on the study of biological systems, which exhibit intrinsic emergent behaviors, we extract a few guidelines for use in our designs. We then discuss one preliminary design attempt, along with evaluation criteria for this and other emergent service designs.**

I. Motivations

Mobile devices have become increasingly powerful over time. Today, they have hardware and software capabilities that are on par with those of the desktop computers of less than a decade ago [1]. It is highly likely that in the near future they will be able to run applications as complex as the ones currently running on the Internet. These mobile devices will be used not only to send and receive data as they are today, but also to perform more advanced functions, so that a collection of these mobile devices can provide advanced application services among themselves, without depending on any fixed infrastructure support. Such self-organizing, self-serving capability is especially important in military environments, characterized by the high demand to support diverse sets of applications in the absence of any fixed infrastructure. In this paper, we explore the challenges involved and new directions to explore in the quest to provide robust, complex application services within mobile networks.

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

Although the current underlying hardware technology makes it feasible for handheld devices to provide robust, complex application services, such as streaming media delivery, without Internet infrastructure support, the current method of service deployment is a fundamental barrier to achieving this goal. Current service deployments rely heavily on human intervention to configure the various components of a service, which makes such methods adequate only for a specific, static topology. Highly dynamic mobile networks, especially infrastructure-less, ad-hoc networks, are subject to frequent disconnections and fluctuating levels of available energy, bandwidth, processing and storage. In this context, the human intervention approach does not scale; the amount of re-configuration effort that is required to orchestrate the various service components becomes prohibitively high, as the network topology is constantly changing.

Emergent behavior can serve as a new paradigm for providing scalable service deployment and configuration in highly dynamic, mobile networks. Emergent behavior (or service *emergence*) refers to the collaboration of service components in order to elicit capabilities in the overall system that are far beyond those of the individual components [2] [3]. Emergence results from the self-organization of individual components, where each component acts autonomously based on what it learns from the environment and from highly localized interactions with other components. This can lead to a scalable design for complex services that are capable of automatically adapting to unanticipated changes.

In this work, our contribution is an exploration of how service emergence might be realized in mobile networks. We present a streaming-media-based surveillance service as a concrete example and case study for emergent mobile services (Section II). In understanding the abundant self-organizing, emergent behaviors observed in biological systems (Section III), we derive a set of guidelines that we believe can help us to program individual components in a mobile system, so that a desired overall system function can be achieved as emergent behavior resulting from individual components' local actions (Section IV). As a proof-of-concept, we also present one preliminary design for the surveillance service that adheres to these guidelines (Section V).

Although several research efforts have been devoted to self-organizing designs in recent years (e.g. [4] [5] [6] [2]), we believe our direction is unique in several ways. The most fundamental of these is that our design aims to provide robust services in a highly dynamic environment by using *identically* programmed components, and by enabling each component to automatically adjust its role based on the overall service goals. This distinguishes our efforts from other self-organizing systems, such as those based on mobile agents [3], where agents with *predefined roles* move between nodes.

Furthermore, we plan to explores a novel approach to designing individual components of a self-organizing system through *reverse engineering*. Whereas previous self-organizing system designs have first proposed the decision-making scheme for individual components and then observed the consequent behavior, we start with the desired emergent results and then try to understand what kinds of component decisions can lead to these results. We believe that this new approach can lead to a much deepened understanding on emergence in distributed systems.

One fundamental challenge in our new approach is evaluating the success of an emergent algorithm, which we discuss in Section VI. Further comparison with related work is provided in Section VII, and discussions on several open issues are presented in Section VIII.

II. An Example Service

To better illustrate the challenge of providing complex, robust services in mobile wireless networks, we present a practical example service: live video surveillance.

A typical usage scenario for this service is as follows. A commander in a hostile area wishes to view surveillance video from a particular geographical location, such as a particular street corner. A number of wireless-enabled and video-capture-capable resources are present in the area. Many of these resources, such as those carried by ground troops and installed in vehicles, are highly mobile. However, most, if not all, of these resources are involved in more important missions than surveillance. As a result, the service must be designed with the expectation that node mobility patterns will be unrelated to the surveillance mission.

In the presence of some shared, fixed infrastructure, such as satellite connectivity, this type of service can currently be implemented using a traditional client-server model. Cameras on handheld devices and mounted on vehicles, as well as stationary cameras, capture the live events. These video feeds are uploaded to remote servers, and then redistributed to the interested clients (such as the commander in the scenario described above) through the shared infrastructure network. If any clients have only low-bandwidth connections to the network, the server can transcode the video to a lower bitrate. However, this approach is completely untenable for networks where fixed infrastructure is unreliable or simply nonexistent.

A much more viable alternative in this scenario would be to provide the service from within an ad-hoc network formed by the mobile nodes. As hardware and software in mobile wireless devices becomes more and more powerful, these devices are increasingly able to function as general-purpose computing devices. We can take advantage of these new capabilities, building in the functions required to provide complex services

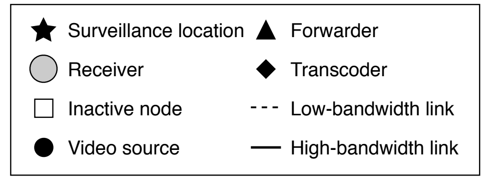


Fig. 1. The legend for the other figures in this paper.

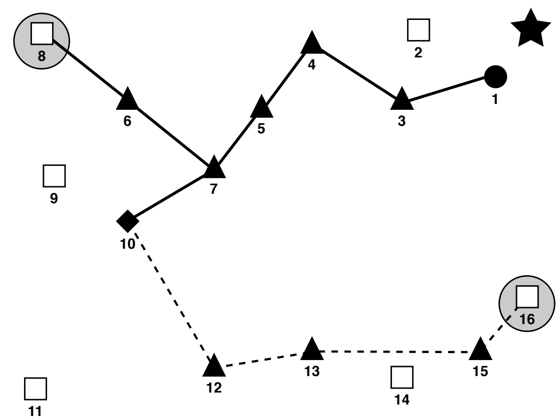


Fig. 2. An example network showing appropriate roles for each node.

– functions we would normally associate with fixed infrastructure servers. This includes resource-intensive tasks, such as transcoding video. We call these functions *roles*, and provide each node with the ability to perform any one of them. As a result, access to the surveillance service can be as robust to mobility, interference, and damage as the ad-hoc network itself. Additionally, latency may improve due to the reliance on exclusively local resources, without the use of any remote servers.

Let us examine how this ad-hoc surveillance service might work in greater detail. We define the scenario as follows. A node's possible roles are capturing video, transcoding video data, and forwarding video data. Any node can perform any *one* role at a time. However, the bandwidth on each link may vary. Only one node acts as the video source. We use a simple model for link bandwidth, where a link can be placed in one of two categories: short-range, high-bandwidth links, which can carry the full-bitrate video feed from the capture device, and longer-range, low-bandwidth links, which can only support video that has been transcoded to a lower bitrate. Since transcoding may degrade video quality or increase latency, the service should try to avoid using transcoders whenever possible. In other words, it is preferable for the nodes to be configured in such a way that they provide a path from source to receiver using only high-bandwidth links. Otherwise, to avoid wasting resources, the transcoder should be placed as close to the source as possible, and *all* links from transcoder to receiver can be low-bandwidth links.

Figure 2 shows a snapshot in time of an example network of 16 nodes, along with an example of a good selection of roles for each node, given this configuration. Let us examine the choices of roles more closely. Nodes 8 and 16 want to receive the surveillance video feed. Node 1 is closest to the location under surveillance, so it should provide the video feed. Many (but not all) of the nodes must be enlisted as forwarders to get the video to the proper receivers. Notice that the distances between the nodes allow for a high-bandwidth connection between node 8 and the source, but there is no link from node 10 to node 16 that can support a high-bandwidth connection, since the nodes are too far apart. Thus, a transcoder will be required to get the video feed to node 16. The transcoder should be placed as close to the source as possible, but it cannot interrupt the full-bitrate connection between the source and node 8, so it should be placed at node 10. Between the transcoder and node 16, there is no need for high-bandwidth links, so low-bandwidth links are used exclusively.

III. Understanding Emergence

The inaugural issue of the journal *Emergence* contains a pertinent definition of the term: "the arising of novel and coherent structures, patterns and properties during the process of self-organization in complex systems. [7]" These structures, patterns, and properties are "novel" in the sense that they are not present in the individual components of the system, but only in aggregate; they are only present when the system is viewed as a whole. Furthermore, they "arise" without any central coordination. Emergent properties are commonly found in biological systems, where a number of individual cells or organisms, by simply reacting to their immediate environment, produce a larger system with its own complex set of behaviors. One pertinent example of emergence in a biological system is the process of morphogenesis.

Morphogenesis is the process by which an organism develops its particular shape. That is, the process by which all humans come to be shaped like humans, frogs like frogs, and so on. In multicellular organisms, this shape emerges from millions of individual cells, each of which must form into the correct type of cell at the correct location in the body. At some point in this process, each cell decides on its fate – whether it will be a neuron, a blood cell, a skin cell, or so on. This part of the process is called *cell determination*.

Cell determination is possible due to the fact that each cell contains the organism's entire genome, which includes the instructions for developing into all possible cell types. Each cell must only determine which portion of these instructions to follow; though a cell in the brain contains the instructions for becoming a skin cell, it will choose to only follow the instructions for becoming a neuron.

Each cell knows what type of cell to become based only on signals from its immediate environment. It does not coordinate with any central authority, nor does it have any awareness of what other cell types are developing at other locations in the body. Yet, these simple, localized decisions result in morphogenesis, the emergence of a consistently shaped, coherent whole.

A. Revisiting Our Example

We would like mobile services to follow a model that is analogous to morphogenesis. Whereas multicellular organisms have different types of cells, mobile nodes can have different roles. All of the nodes must take on an appropriate role in order to provide a functional service. The formation of the correct overall shape of an organism is analogous to meeting overall service goals.

Returning to the video surveillance service example from Section II, at least one node must perform the role of video capture, some may need to transcode video data, and others may need to simply forward video data (without changing its bitrate). If every node has the capability to provide any of these roles, then each node can choose to take on whichever role it deems necessary based on the state of its environment at the time. For the surveillance service to emerge, each node must take on an appropriate role such that the service goals are met.

Given the snapshot of the network shown in Figure 2, it would be simple for a human service designer to determine the appropriate roles for each node and configure them manually. However, recall that we are considering services for highly mobile networks, where the location of each node is constantly in flux. If only a few of the nodes were to move, as shown in Figure 3(a), then the current roles are no longer appropriate – no video is provided to node 16, and potentially extraneous resources are being expended on a redundant feed to node 8. Clearly, the nodes have to choose new roles, at a rate far faster than a human service designer could, in order to continue to provide an effective surveillance service.

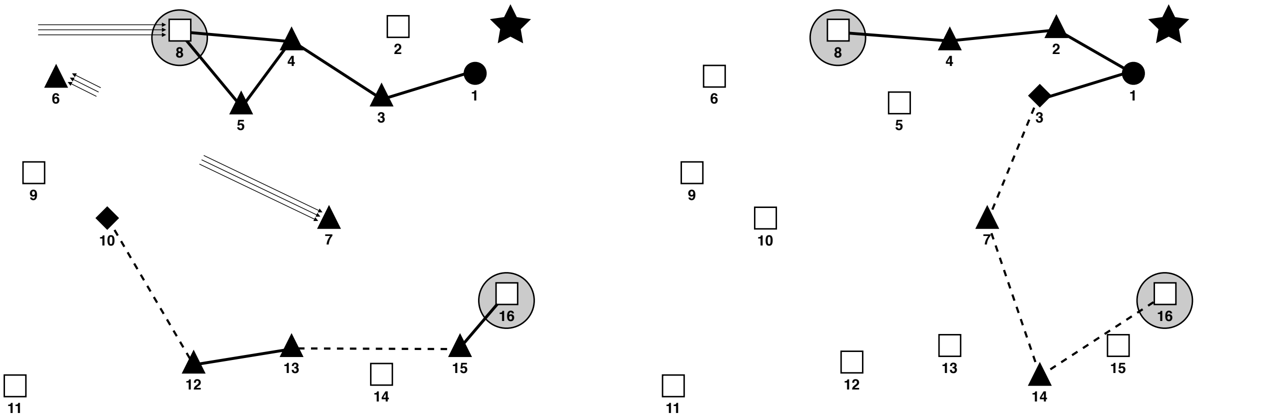
B. Emergence for Mobile Services

Our ultimate goal is to design a system whereby, much like the cells of a multicellular organism, each node knows how to determine its own role, and the overall result is a coherent service. This means the nodes must be able to determine appropriate roles for themselves in any static situation, such as the one shown in Figure 2. However, unlike a multicellular organism, whose shape stops changing once it is fully developed, mobile networks are constantly changing shape. Thus, nodes must also be able to continuously readjust their roles, ensuring that the service goals continue to be met in the face of mobility. For example, given the movement of nodes 6, 7, and 8 shown in Figure 3(a), the nodes should be able to choose new roles akin to those shown in Figure 3(b).

The major challenge of this approach is to determine how nodes will make these decisions. In Section IV, we distill some guidelines for addressing this challenge from our study of emergent behavior in a multitude of biological systems. We explore how these guidelines might be applied in Section V.

IV. Learning from Biological Emergence

Given a potentially large number of mobile nodes, our design goal is to forge the desired application service from the system-wide emergent behavior of the uncoordinated local decisions made by each of the nodes. This requires us to design



(a) Node Movement

(b) New Roles

Fig. 3. The example network from Fig. 2: (a) immediately after three nodes have moved, rendering the roles inappropriate, and (b) with more appropriate roles.

a program that, when installed and run on the nodes, will enable the system to achieve this goal. Thus, the first challenge that we are faced with becomes apparent: how can we derive the program that should be installed on individual nodes from the description of the desired application service?

Research on emergent behavior states that one cannot know what an emergent system will ultimately look like by simply examining the details of its individual components [8]. We argue that the converse is also true: the details of the desired emergent system do not tell us directly how each element should behave.

By examining the behavior of emergent biological systems, we make the following observations, which will serve as our guidelines for accomplishing the task at hand:

1) Use identical components to build complex systems. Cells all start out identical, but, as they form complex systems, they differentiate, taking on different roles based on feedback from the environment. Intuitively, we can argue that this is the most robust way to build a complex system. Had the cells started out different from the outset, nature would have to determine *a priori* how many copies of each different cell type are needed, and in what proportion. The result could be a poor fit for some particular environment.

2) Achieve robustness through redundancy. In general, in biological systems, no single element can impact the overall system outcome because of the high degree of redundancy. Having all cells be versatile in the functions that they can carry out also contributes to increased robustness, as regardless of which cell fails, another can take on its role.

3) Maintain the system at the desired state through local feedback. Biological systems exhibit dynamic behavior because they exist in a dynamic world. Both external changes (such as changes in the ambient temperature) and internal changes (such as tissue damage) are common. To keep the system operating correctly, individual components adjust their behavior based on the feedback that they obtain from their environment. The aggregate result is a system that maintains its desired state under constantly changing conditions.

In a similar spirit, mobile, infrastructure-less networks are also continuously changing, dynamic systems. Therefore, to maintain the desired overall system state, we must also incorporate into the system design adequate feedback channels, and program the rules for individual components to react to feedback signals.

4) Perform completely localized decision making. In reacting to changes, individual cells have no global knowledge. In fact, they do not even have global awareness – cells have no notion of what other cells might exist in the system beyond the vicinity of their direct neighbors. An individual component has at most two types of communication channels available: direct communication with its immediate neighbors, and indirect communication via the environment. We can speculate on the basic reason why biological systems work in such a localized manner: if every component wishes to interact with every other component in a large system, then the resulting system simply does not scale. There are two main reasons for this:

- Such interactions would require too many communication channels [2].
- If every node were to have global awareness, too much state would have to be maintained at each node.

Our design should also operate in this localized manner in order for it to scale to a potentially large number of mobile nodes.

5) Improve the overall system design through long-term feedback. The diverse array of biological systems that exist today are the result of random mutations that have taken place over a very long period of time. Darwinian evolution has pruned off sub-optimal mutations over the course of many millennia.

To build self-organizing artifact systems and bootstrap them in a short time scale, we, as the designers, must provide this long-term feedback loop, in place of a pure evolutionary process. If we believe that biological systems never stop evolving, this implies that our design cannot be completed once and for all, and instead must be tuned continuously.

V. Designing for Emergence

In this section, we present a concrete example of how the guidelines in the previous section can be applied to mobile services. In our preliminary work, we have begun developing and simulating algorithms that adhere to these guidelines. We present our most successful attempt to date as a proof-of-concept.

This algorithm is for a simplified version of the surveillance service presented in Section II. In this simplified scenario, nodes must choose to become a video source, a video forwarder, or take no role at all. We do not include the transcoder role as of yet, though we believe this algorithm could be easily extended to support it (and arbitrary other roles, as well). Furthermore, we assume there is only a single location that clients are interested in surveilling, and all nodes within range of that location will take on the source role.

A. A Proof-of-Concept Algorithm

To explain our algorithm, let us follow it in action, beginning with the nodes that are interested in receiving a video stream. These node, which we will call *clients*, broadcast requests for a video source. Other nodes always flood these requests through the network. To help nodes to distinguish between different requests, each request contains a random identifier (RID).

Eventually, the video source receives these requests, and begins to broadcast data packets containing the video stream. The video source places the set of RIDs for the requests that it is responding to in the data packet. Paired with each RID is a *gradient level*, a real number between 0 and 100. The packets originated by a video source will always have all gradient levels set to 100 (the maximum value).

The gradient levels in a packet decrease as it travels along various paths through the network. The gradient levels are decreased based on another value, the *cost metric*. The cost metric measures the cost incurred by transmitting the packet over a single link. Each link can have a different cost, as long all costs are non-negative. Examples of cost metrics are geographical distance traveled, the amount of signal attenuation, and the energy expended in receiving the packet.

When another node receives a data packet, it stores the set of tuples $\{(r, g_r, m) \mid r \in R\}$, where R is the set of all request identifiers in the packet, g_r is the incoming gradient level for request identifier r, and m is cost metric for the link the packet was transmitted over. Nodes keep these union of all of these sets of tuples for each packet that has arrived within a short time window.

If the receiving node is a forwarder, it will re-broadcast the data packet, decreasing the gradient levels. It will determine its own gradient level ℓ_i for each request identifier i:

$$\ell_i = max(\{g_r e^{-m} \mid (r, g_r, m) \in T, r = i\})$$

where T is the set of all tuples currently stored by the node. These ℓ_i values are the decreased gradient levels placed in the re-broadcasted packet. Note that the use of the function e^{-x} produces a value that is strictly greater than 0, regardless of the number of hops traversed, or the magnitude of the values of m.

If the receiving node is idle (it has no role), it may become a forwarder if any of these ℓ_i values have changed recently. This means that, when a new video stream enters the network, it will initially be flooded to all of the nodes.

Eventually, the data will reach the clients, which generate *acknowledgement (ACK)* packets. The purpose of ACK packets is to serve as a positive feedback mechanism. The more often a forwarder receives an ACK, the more likely it will maintain that role. Forwarders that do not receive ACKs will eventually drop the role and become idle.

ACK packets only acknowledge the RID that the receiving client originated. This RID, i, is placed in the ACK packet. ACK packets also contain the minimum and maximum gradient levels currently stored by the receiving client for RID i. That is, ACK packets contain the tuple (g_i^{min}, g_i^{max}), where:

$$g_i^{min} = \min(\{g_r \mid (r, g_r, m) \in T, r = i\})$$

$$g_i^{max} = \max(\{g_r \mid (r, g_r, m) \in T, r = i\})$$

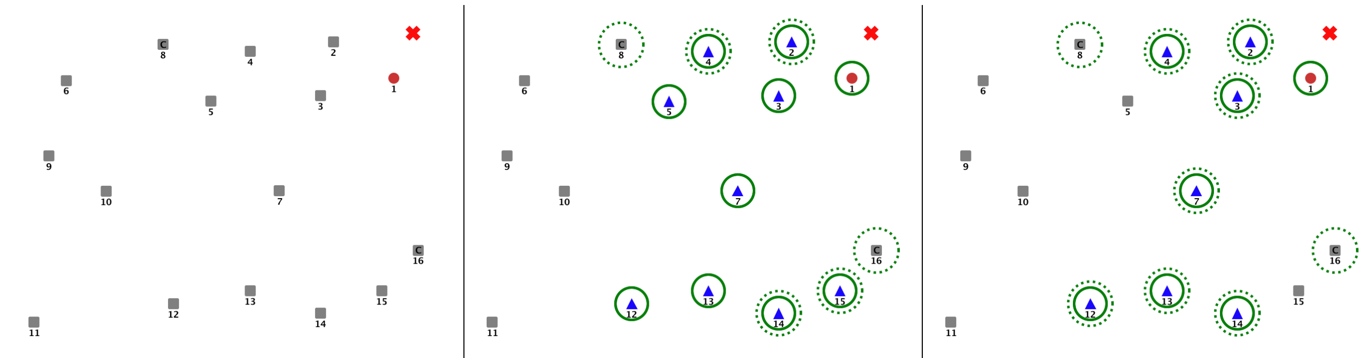
Receivers of the ACK then compare this range to their own ℓ_i value. Receivers may decide to drop the ACK with some probability p_{drop}, where:

$$p_{drop} = \left(\frac{g_i^{max} - \ell_i}{g_i^{max} - g_i^{min}} \right)^n$$

when $g_i^{min} \neq g_i^{max}$, and $p_{drop} = 0$ otherwise. n is some positive real number. The result is that the node with the highest gradient level ($\ell_i = g_i^{max}$) will never drop the ACK, while the node with the lowest gradient level ($\ell_i = g_i^{min}$) will always drop it (as long as $g_i^{min} \neq g_i^{max}$). The purpose of n is to determine how likely it is that a node with $g_i^{min} < \ell_i < g_i^{max}$ will forward the ACK. The more likely that nodes with an ℓ_i value in this range will forward an ACK, the more likely that forwarders that are not on the best path will receive positive feedback, creating more redundancy in the system. Values of n closer to 0 produce less redundancy, while values of n closer to 1 produce more redundancy. Values of $n > 1$ are possible, but not recommended. In our experiments, we set $n = \frac{1}{3}$.

B. A Proof-of-Concept Example

We have implemented the algorithm described in the previous section in our own simulator. Figure 4 shows the output of applying this algorithm to the same network shown in Figure 3. In this figure, nodes with no role are represented by gray, rounded rectangles, and the clients (nodes 8 and 16) are marked with a "C". The solid, inner circles that surround some nodes represent that the node is broadcasting a data packet, while the dotted, outer circles represent that the node is broadcasting an ACK packet. Figure 4(a) shows the network before any packets have been transmitted. Figure 4(b) shows the network just after the data packets from the video



(a) Time Step 0 (b) Time Step 10 (c) Time Step 30

Fig. 4. The algorithm from Sec. V-A simulated on the example network from Fig. 3 in our custom simulator. The network is show (a) before, (b) during, and (c) after role selection has converged.

source have entered the network and changed gradient levels, resulting in an initial flood of the connected portion of the network. Figure 4(c) shows the network after the ACKs have caused the network to settle into a stable state. Note that the role selection is quite similar to Figure 3(b), except for the absence of transcoders and the addition of some redundancy.

C. Summary

In short, our proof-of-concept algorithm works as follows. Data packets (containing surveillance streams) are generated and broadcast by nodes performing the source role. Forwarders always re-broadcast incoming data packets. Initially, a new video stream will briefly flood through the network. Each data packet carries a numerical gradient level, which represents the cost of the path the packet has traversed thus far. Acknowledgement packets (ACKs) are originated by clients interested in receiving surveillance streams. ACKs are forwarded by a forwarder with a probability dependent upon the highest gradient level the forwarder has seen. ACKs serve as a feedback mechanism (as per guideline 3 in Section IV), ensuring that forwarders that are along reasonable paths between a source and a client remain in their current role. Forwarders that do not receive positive feedback will eventually drop their roles and become idle. All nodes run the same algorithm (as per guideline 1), and can take on any role (as per guideline 2). Nodes are *not* required to have any individual addresses (as per guideline 4) – only broadcast packets are used.

Note that, if the cost metric used to determine the cost of each link is geographical distance, this algorithm, in its current form, reduces to a routing algorithm between source and clients. This routing algorithm would bear a strong resemblance to a number of existing routing protocols for ad-hoc and sensor networks, such as ARA [9], AntHocNet [10], and GRAB [11]. Though we readily admit these similarities to previous work, our intention is not to be imitative. Rather, we believe that these similarities lie in the fact that the we have explicitly developed and applied guidelines extracted from biological systems. Previous work has implicitly adhered to these guidelines, whether due to direct homage to a biological systems [9], [10] or similar abstractions [11].

Regardless, this algorithm achieved promising results in our initial simulations. A detailed performance evaluation will appear in future work. The next section proposes some metrics for such a performance evaluation.

VI. Evaluation Metrics

Understating whether a particular emergent algorithm leads to the desired system is crucial when designing a self-organizing system. The system has to be evaluated using a number of metrics in order to understand whether a particular emergent behavior produces the expected results. In this section, we present four generic metrics that can be used when evaluating the fit of a self-organizing algorithm to its service goals. These metrics may not be considered equally important in all types of applications; their relative ordering will depend on the particular application's needs and priorities.

A. Scalability

Scalability is the first metric we will consider in the design of a self-organizing system. More specifically, we define scalability as the ability of the self-organizing system to produce the same emergent behavior independently of it size. Size can be a function of the number of nodes, the number of users, the number of services, the area within which the network is deployed, or the scale of the the workload characteristics of the application (i.e., an underloaded versus an overloaded network). Achieving scalability across all of these dimensions may not be possible for some applications. As such, depending on the application, the scalability metric may take into account different sets of dimensions where each dimension may be weighted differently. Note that the service quality of the emergent system (as defined below) may be dependent upon the scale of the system. Thus, the scalability metric is highly coupled with the service quality metric.

B. Speed of Convergence

Speed of convergence is another metric to consider in the design of a self-organizing system. Speed of convergence refers to the time required for the system to achieve the desired behavior when starting from some initial or 'suboptimal' configuration. When the systems is considered to have converged is dependent upon the particular application. For some applications, it may make sense to only consider the system to have converged when it has reached some 'optimal' configuration. For other applications, a stable, working configuration may be sufficient, even if it is somewhat suboptimal. Furthermore, in many cases, an emergent behavior may not always lead to the expected optimal configuration, and may not even stabilize to just one final configuration. Note that, in these latter cases, the definition of the convergence speed is tightly coupled with that of service quality.

C. Efficiency

Efficiency is the metric that captures the amount of additional resources that are consumed due to the various self-organizing processes of the system. Some examples of efficiency, or lack thereof, are:

- the amount of traffic overhead introduced due communications overhead between the various self-organizing components,
- the amount of additionally energy spent due to redundancy introduced by the self-organizing algorithms, and
- the amount of additional storage needed in order for the nodes to be able to implement all possible roles.

Efficiency is a relative metric in that it is compared to a baseline. For example, the efficiency of a self-organizing system can be compared against a hypothetical system that achieves the same performance against a set of other metrics (i.e., scalability, speed of convergence, and/or service quality).

D. Service Quality

Finally, service quality is a metric that captures the ability of the self-organizing system to deliver an acceptable service. This metric is the most application-specific of the four listed here, given that the definition of service quality depends on the goals that the application is trying to achieve. Some examples of service quality are the following:

- **availability**, the ability of the system to continuously provide the service,
- **resiliency**, the ability of the system to adapt to the dynamics introduced by the highly mobile nature of the network, and
- **optimality**, the ability of the system to provide the best possible service for the given amount of resources that it consumes.

Apart from these generic types of service quality, a service designer may define more application-specific metrics. For instance, in the case of our video surveillance application, the quality of the transported video is a crucial service quality metric. Clearly, service quality metrics are tightly coupled with all other metrics.

VII. RELATED WORK

Self-organization principles applied to highly dynamic computing systems have been the focus of recent research efforts in the MANET and P2P fields. In the MANET routing domain, protocols such as AntHocNet[10] and SAMPLE [12] have used biological metaphors to identify robust and efficient routes for packet delivery. The inspiration for AntHocNet comes from the process of stigmergy employed by ants in their quest to identify shorter (or higher quality) paths while foraging for food. The SAMPLE protocol makes use of collaborative reinforcement learning to coordinate the solution to discrete optimization problems in multi-agent systems, in order to optimize desirable system properties such as throughput or robustness. Both of these protocols apply the principles of short- and long-term feedback. However, their framework does not emphasize the possibly different roles that the nodes can play in the MANET. Instead, they construct a single network service (routing), which is homogeneous in functionality across nodes.

Localized structures and positive feedback loops are self-organizing properties also exhibited by P2P systems such as Freenet [13], whose similarities to the emerging behavior and mechanisms of MANET routing services are further highlighted in [4]. However, as in the case of self-organizing MANET routing protocols, the network nodes are rather single-purpose, in contrast to the different roles and functionalities encountered in more complex services such as streaming media distribution, as considered in our work.

In the nascent field of modeling and composition of mobile web services, adaptation by continuous monitoring of the environment (often referred to as "execution context") has recently been studied, as well as composite services that are further decomposed into sets of more primitive ones offered by a plurality of nodes [14], [15], [16], [17]. These services require a role selection process to be in place. Various approaches and formalisms are employed for the description, coordination and adaptation of the mobile services, however the main assumption of self-organizing systems – that of completely localized decision making – is typically not met. The orchestration, that is, the binding of the composite service specification to nodes for execution, monitoring and supervision, is performed by an orchestration service that assumes access to global knowledge of the environment, the nodes, and their capabilities.

Both autonomous operation based on a partially observable execution environment and adaptive behavior that makes use of feedback have been advocated for use in multi-agent systems [18], [5]. Key abstractions such as agents, interactions, and organizational structures are used for designing and engineering complex software systems. These abstractions are more natural and closer to the conceptual model of the programming task under examination than existing object-oriented or procedural approaches. The multi-agent field embodies several of the concepts presented in this paper. However, we should note that sharing global knowledge in a large software system is rather straightforward and imposes minimal overhead, compared to a similar task in a MANET environment, in which the underlying *physical* infrastructure is significantly more dynamic.

Design of network services based on principles inspired by the operation of biological and social systems is also described in the adaptive networking architecture for service emergence (ANA-SE) [3], [19]. As in multi-agent systems, the main abstraction in this proposal is autonomous entities called cyber-entities. Emergent services created from such autonomous cyber-entities are able to interact with one another and adapt to the dynamics of the environment. The main difference between ANA-SE and our approach is that the former models services with agents that move around between nodes, so when they move their intelligence/memory comes with them, whereas in our work we decompose the services into roles, and let the nodes choose what role to play.

VIII. Discussion

A. Security

Since all nodes will run the same program, that program can include security measures. For example, in our surveillance service (see Section II), if encrypted data transmission is desirable, public keys could be used as request identifiers (see Section V-A). Request identifiers are already created by the client node that the data is destined for – the same node that would need to decrypt encrypted data.

Furthermore, new roles could be added to an emergent service specifically to increase security. Whereas the traditional Internet service model generally involves dedicated security appliances, such as firewalls and network intrusion detection systems, roles in an emergent service are only used when needed. Thus, the overhead for adding new roles for security could potentially be less in emergent services.

B. Applications

We envision this new approach of service composition for mobile networks to be broadly applicable in many different domains. Examples include military mobile ad-hoc networks, emergency disaster recovery networks, last-mile services in lesser-developed countries, and other wireless networks where the infrastructure support is limited or non-existent compared to the number and the density of the devices using the network.

References

[1] "Smartphone and OS markets," ABI Research, http://www.abiresearch.com/products/market_research/Smartphones, Tech. Rep., 2008.

[2] F. Dressler, "Self-organization in ad hoc networks: Overview and classification," Univ. of Erlangen, Dept. of Computer Science, Tech. Rep., 2006.

[3] T. Itao, T. Nakamura, M. Matsuo, and T. Suda, "Adaptive networking architecture for service emergence," *IEEE/IPSJ International Symposium on Applications and the Internet Workshops*, vol. 0, p. 9, 2001.

[4] B. Biskupski, J. Dowling, and J. Sacha, "Properties and mechanisms of self-organizing manet and p2p systems," *ACM Trans. Auton. Adapt. Syst.*, vol. 2, no. 1, p. 1, 2007.

[5] N. R. Jennings, "An agent-based approach for building complex software systems," *Commun. ACM*, vol. 44, no. 4, pp. 35–41, 2001.

[6] D. Miorandi, L. Yamamoto, and P. Dini, "Service evolution in bio-inspired communication systems," *ITSSA*, vol. 2, no. 1, pp. 51–60, 2006.

[7] J. Goldstein, "Emergence as a Construct: History and Issues," *Emergence*, vol. 1, no. 1, pp. 49–72, 1999.

[8] S. A. Kauffman, *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity*. Oxford, UK: Oxford University Press, 1995.

[9] M. Güneş, U. Sorges, and I. Bouazizi, "ARA – The Ant-Colony Based Routing Algorithm for MANETs," in *Proc. IEEE ICPPW*. Los Alamitos, CA, USA: IEEE Computer Society, Aug 2002, pp. 79–85.

[10] G. Di Caro, F. Ducatelle, and L. M. Gambardella, "AntHocNet: An Ant-Based Hybrid Routing Algorithm for Mobile Ad Hoc Networks," *PPSN VIII*, pp. 461–470, 2004.

[11] F. Ye, G. Zhong, S. Lu, and L. Zhang, "GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks," *Wirel. Netw.*, vol. 11, no. 3, pp. 285–298, 2005.

[12] E. Curran and J. Dowling, "Sample: Statistical network link modelling in an on-demand probabilistic routing protocol for ad hoc networks," in *International Conference on Wireless on Demand Network Systems and Service*, 2005, pp. 200–205.

[13] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," in *Designing Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science. Springer, 2001, pp. 46–66.

[14] M. de Leoni, M. Mecella, and G. D. Giacomo, "Highly dynamic adaptation in process management systems through execution monitoring," in *Business Process Management*, ser. Lecture Notes in Computer Science. Springer, 2007, vol. 4714, pp. 182–197.

[15] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "A framework for qos-aware binding and re-binding of composite web services," *J. Syst. Softw.*, vol. 81, no. 10, pp. 1754–1769, 2008.

[16] G. De Giacomo, M. de Leoni, M. Mecella, and F. Patrizi, "Automatic workflows composition of mobile services," in *International Conference on Web Services (ICWS)*, July 2007, pp. 823–830.

[17] Z. Maamar, Q. Z. Sheng, and B. Benatallah, "On composite web services provisioning in an environment of fixed and mobile computing resources," *Information Technology and Management, Special Issue on Workflow and e-Business*, vol. 5, no. 3-4, pp. 251–270, 2004.

[18] P. F. Carles Sierr and N. R. Jennings, "A service-oriented negotiation model between autonomous agents," in *Collaboration between Human and Artificial Societies*, ser. Lecture Notes in Computer Science. Springer, 1999, vol. 1624, pp. 201–219.

[19] T. Nakano and T. Suda, "Self-organizing network services with evolutionary adaptation," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1269–1278, 2005.