Towards Adaptable Ad Hoc Networks: the Routing Experience *

Cesar A. Santivanez[1] and Ioannis Stavrakakis[2]

[1] Internetwork Research Department,
BBN Technology, Cambridge, MA 02138, USA,
csantiva@bbn.com
[2] Department of Informatics and Telecommunications,
University of Athens, 15784 Athens, Greece
ioannis@di.uoa.gr

Abstract. Network users not only demand new and versatile application support by the networks but they themselves are becoming part of the network (network routers, caches, processors, etc) by contributing their resources to it and being engaged in ad hoc networking structures. As the large and diverse user population becomes more and more part of the networking infrastructure it is clear that networks will be dominated by a new type of network nodes which are much more nomadic, diverse and autonomic than in traditional networks, creating a fairly diverse – in size and characteristics – networking environment. For instance, low cost/high availability/convenience of wireless devices are expected to lead to the deployment of a plethora of wireless networks for diverse applications: from rescue missions to military communications, from collaborative computing and sensor networks to web browsing and e-mail exchange to real time voice and video communications. Each with different constraints and requirements. And, for each type of application there is also a high degree of variability in the networking context: from a low mobile network of a few nodes to a highly mobile network with thousands of nodes.

This high degree of variability in the networking environment calls for a new design paradigm where network elements (nodes) should be able to adapt to totally different scenarios, engaging in a different behavior depending on the situation. Thus, next generation networks should be able to learn their environment/context and adapt their behavior accordingly in order to achieve their goals. In this paper we introduce some key mechanisms required to enable broad adaptability. Although these mechanisms are general and common to a large variety of tasks/services (e.g. service discovery, location management, cooperative computing, clustering, etc.) we will discuss them in the context of the routing service, leveraging our past experience on the area. This will allow us to ground the discussion in concrete terms and the reader to better visualize the concepts.

Key words: ad hoc, wireless, autonomic, adaptability, routing.

* This work is funded in part by the EU-funded project ACCA and the EU Network of Excellence E-NEXT

1 Introduction

In traditional networks, network nodes were carefully designed to support specific network capabilities and were deployed and controlled by the owner(s) of the infrastructure. The ever-increasing user population was clearly located at the periphery of these networks and received a service tightly prescribed by the specific network.

Nowadays, the networking landscape is changing dramatically. The numerous network users not only demand new and versatile application support by the networks (e.g., mobility, multimedia, etc) but they themselves are becoming part of the network (network routers, caches, processors, etc) by contributing their resources to it and being engaged in ad hoc networking structures. As the diverse user population becomes more and more part of the networking infrastructure it is clear that networks are bound to be dominated by a new type of network nodes which are much more nomadic, diverse and autonomic.

In the wireless domain, WiFi (802.11-based) networks have opened up the way to high-speed wireless support of autonomic, nomadic users. In addition to the proliferation and enhancement of these networks with numerous extensions (QoS, ad hoc and other capabilities), other ways of networking such nodes requiring higher transmission rates and more demanding application support (mobile and sensor ad hoc networks) have emerged. The resulting rapid deployment of (increasingly autonomic) network elements (nodes) is leading to the formation of large and ever increasing multi-hop wireless networks. Indeed, wireless ad hoc networks of thousands of nodes are already in the designing phase for the USA military's Joint Tactical Radio System (JTRS) Cluster 1 program [1]. However, a large number of nodes coupled with the inherent limits of the end-to-end throughput of ad hoc networks lead to potential bandwidth starvation, unless extra care is taken to develop highly scalable algorithms. For some services/tasks, such as routing in homogenous networks [6], even the best solutions cannot prevent bandwidth starvation when the network size exceeds a certain size (the "curse of dimensionality").

To complicate matters further, the environment a node may encounter may be quite heterogeneous. The scenario a node may encounter and the corresponding best solution may vary greatly, not only in space and time, but also among different nodes sharing the same position (e.g. some nodes may be highly mobile while others may be static, while another pair of nodes may be moving together - i.e. relative mobility is zero). We refer to this as the "curse of diversity".

Finally, the objective of the network may also vary greatly: from allowing peer-to-peer communication among each pair of nodes to detecting moving objects and report it to a central server (sensor networks) to finding the closer provider of a service (e.g. find the closer free parking space) to propagate traffic condition information and negotiate vehicle speeds on an Autonomous Vehicle Network (AVN). Some of these goals are more challenging than others. For example, peer-to-peer communication in an homogeneous network is not scalable with respect to the number of nodes. In the other hand, finding the closer parking lot or locally coordinating among neighboring cars both scale well with network

size. Obviously, an application-driven adaptive solution that target the particular task at hand is required, since a general "one-size-fit-all" solution will be inefficient (an overkill) for some practical applications.

In view of the above, it is clear that network nodes are likely to become part of fairly diverse – in characteristics and size – networking environments. The only way nodes belonging to a **L**arge-scale, **A**utonomic, **D**iverse and **A**dhoc **(LADA)** network can cope effectively with such situations is by being able to learn their environment/context and adapt their behavior accordingly in order to achieve their goals (e.g. routing, service advertisement, content distribution, etc.).

In this paper we introduce some key mechanisms required to enable broad adaptability for LADA networks. Although these mechanisms are general, and common to a large variety of tasks/services (e.g. service discovery, cooperative computing, clustering, etc.), we will discuss them in the context of the routing service, leveraging our past experience on the area. This will allow for a more grounded, concrete discussion helping the reader to better visualize the concepts.

This paper is organized as follows. The next section discusses a framework for routing adaptability and introduces two key mechanisms: limited information dissemination (LID) and pattern extraction (PE). The next two sections that follow, discuss these two mechanisms in more detail in the general context of adaptability (i.e. not only for routing). The final section presents some conclusions and suggestions for future research work.

2 A Framework for Multi-mode Routing

Traditional routing protocols for mobile ad hoc networks are usually designed with a particular environment in mind and fail to adapt to the wide range of environments present in an ad hoc network. Because of the wide diversity of the conditions that may be encountered in an ad hoc network it seems that it would be difficult to effectively route information by engaging a single type of protocol. Instead, a multi-mode protocol should be developed which applies the appropriate "mode" or protocol that is determined to be effective at a given point in time and for the appropriate subset of the network. Thus, a multi-mode routing protocol should adapt itself to the present network conditions taking into consideration the traffic levels and patterns (i.e. the application-driven objectives), as well as the mobility patterns (i.e. environment constraints). In order to identify and utilize the network conditions, the multi-mode routing protocol has to rely on some structure-learning/engaging algorithms that extract the network state information (defined in terms of proper metrics) and, based on it, decide on the proper mode to apply to reach each destination.

In [2, 3] the framework for a multi-mode routing protocol shown in Fig. 1 was introduced. This framework proposes that a multi-mode routing protocol – running simultaneously at each node – consists of three elements: two complementing structure-learning/engaging modules that provide network state infor-

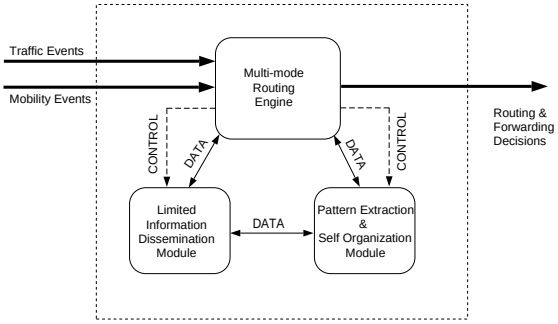

Fig. 1. Multi-mode Routing Protocol Framework.

mation to the third module, the *multi-mode routing engine*, which decides on the mode to apply based on the state of (parts of) the network.

The *multi-mode routing engine* receives information about traffic events (new session requests, or reception of packets to be forwarded to their destination) as well as mobility events (as, for example, nodes displacement and/or link creation/breakage) and passes this information to the structure-learning/engaging modules. Based on this information as well as exchanges among peer modules in neighboring nodes (for example, Link State Update – LSU – messages), the structure-learning/engaging modules obtain some information that defines the state of the network. This information is then passed to the multi-mode routing engine, which uses it to decide the proper routing modes to engage. The behavior of the structure-learning/engaging modules is not fixed but governed by parameters that are defined by the multi-mode routing engine. Thus, the function of the modules is to provide information to the multi-mode routing engine, which controls the modules as well as the final routing mode for each particular packet/destination.

The first of the modules, the *limited information dissemination* module, is responsible for implementing the principle: "the closer you are, the more information you have". This module is in charge of providing detailed information about nodes close by, as well as rough and maybe outdated information about nodes far away. This information may be disseminated in a number of ways. To focus the discussion, LSU messages were chosen as the bearer of the information. This choice was motivated by the fact that link state-based routing presents several desirable properties as for example: fast convergence, well-understood dynamics, loop freedom, etc. However, we should keep in mind that alternatively distance vectors or other metrics (position, service advertisement/description, etc.) may be used as information bearer, and therefore, different algorithms may be executed in the *limited information dissemination* module.

The LID algorithm limits the depth of LSU propagation, avoiding congesting the network with excessive routing overhead in networks with high rate of topological change. Because of the LID algorithm, every node will have good knowledge about the state of its closer links and of far away stable links. This

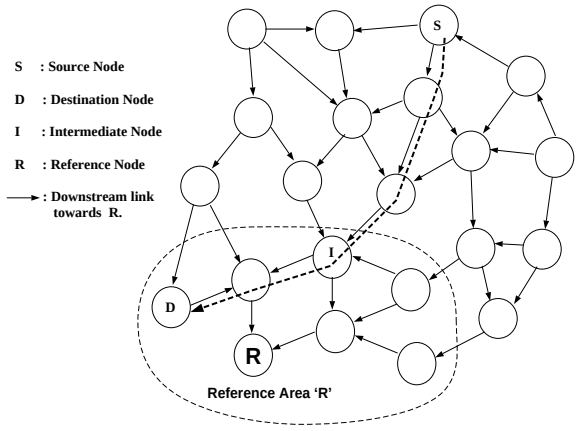

Fig. 2. Routing using the Reference Node/Area (RN/RA) concepts.

information will be used by the multi-mode routing protocol to construct links toward close destinations and even to destinations far away in the presence of stable links. When the LID algorithm is applied to a network with a low rate of topological change the result would be the same as if standard link-state algorithm were applied. When the LID algorithm is applied to a network with a high rate of topological change nodes will have detailed information for nodes close to them, without incurring excessive network overhead. This information needs to be combined with some rough information about how to route packets to nodes far away. This information may be provided by some complementary algorithm as the Self-Organizing algorithm discussed below. A family of efficient LID algorithms with good characteristics is presented in Sect. 3.1.

The second structure-learning/engaging module, is the *Pattern Extraction (PE) and Self Organization (SO)* module. The SO algorithm provides an efficient mechanism for reaching destinations far away for which the LID module fails to provide a route. Specifically, the SO algorithm tries to reduce the number of broadcasts required by a route discovery or flooding algorithms by providing pre-calculated routes *toward* some destinations that are *likely* to be involved in new communication sessions. For those routes to be useful, the cost associated with their maintenance should be less than the expected gain of using these routes. The SO algorithm bases its decisions on the traffic as well as mobility patterns of the nodes. It attempts to choose Reference Nodes (RN) and around them Reference Areas (RA) such that the expected number of new sessions having a destination inside the reference area (Gain, G) be maximized. This gain (G) has to be compared against a threshold (the cost of tracking the RNs plus the – hopefully one-time – location management cost) to decide whether it is worth creating routes toward a particular RA. Finally, the SO algorithm either provides information about links toward the RAs (see Fig. 2), or an indication of the highly mobile status of (some of) the destinations.

The gain function used by the SO algorithm is equal to the expected number of broadcasts saved (with respect to reactive route discovery flooding) if the node computing the gain function were to become a RN including (some of) its k-neighbors in its RA. A broadcast will be saved if the destination of a new session is inside a RA. Let node A be a potential RN and let $V(A,t)$ and $G(A,t)$ be the set of k-hop neighbors and the gain function of node A at time t, respectively. Each node $i \in V(A,t)$ has two parameters : $S_i(A,t)$ (probability that node i will stay inside node A's k-hop neighborhood in the immediate future) and $R_i(t)$ (expected number of new sessions having node i as destination in the immediate future). Then, the gain function $G(A,t)$ is defined as:

$$G(A,t) = \sum_{i \in V(A,t)} S_i(A,t) R_i(t)$$

Different approaches can be considered to estimate the values of $S_i(A,t)$ and $R_i(t)$, depending on the desired amount of complexity. We chose the following estimator:

$$\hat{S}_i(A,t) = (1-\rho) \sum_{j=0}^{\infty} \rho^j Asso(A,i,t-j\Delta t)$$

Where $Asso(A,i,t)$ is a function representing the instant association between nodes A and i at time t, and $0 < \rho < 1$ is the forgetting factor. The forgetting factor determines the extent of the "memory" of the estimator. Larger values of ρ will imply long memory and therefore slow reaction to instantaneous variations. Indeed, the *rising time* (i.e. time required to reach 63% of the desired value when the quantity to estimate is constant) associated with this estimator is $\Delta t/|ln(\rho)|$, which is close to $\Delta t/(1-\rho)$ for values of ρ close to 1. On the other hand, smaller values of ρ will reduce the memory length and result in a faster reaction to changes, but at the same time it will increase the probability of false alarms (i.e. estimating there is an association between two nodes where there is none). $\hat{S}_i(A,t)$ is easily computed by means of the following recursion:

$$\hat{S}_i(A,t+1) = \rho \hat{S}_i(A,t) + (1-\rho) Asso(A,i,t+1)$$

Thus, to compute the gain function, a node only needs to keep the past value of $\hat{S}_i(A,t)$ and the current value of $Asso(A,i,t+1)$ for each node i (k-hop neighbor). The association function $Asso(A,i,t)$ is chosen to be a function of the distance $d(A,i)$ between nodes A and i at the current time t, which node A can compute based on the topology information provided by the LID module for nodes close by. If node i is not reachable using node A's topology table entries it is assumed that $d(A,i) = \infty$. Finally, for the estimation of $R_i(t)$, feedback from the upper layer should be employed. Note that the task of estimating these quantities is performed by the so-called Pattern Extraction (PE) sub-module inside the SO module.

If all the nodes are assumed to have the same traffic patterns (i.e. $R_i(t) =$ constant), then the Pattern Extraction (PE) sub-module will attempt to find

the mobility pattern of the network. In particular, $\hat{S}_i(A,t)$ represents an estimation of the long-term association of nodes several hops away, as for example the association among nodes in the same group in group mobility scenarios[3]. Although it is possible that the mobility pattern of a network be totally random, that is not usually the case. Human mobility, for example, is based on groups (forming clouds) or follows some patterns (streets, highway, searching, etc.). Even automata mobility is shaped by the function they are executing and therefore there is some degree of spatial/temporal correlation. The self-organizing algorithm will attempt to find (or select) the mobility "leaders" (nodes around which others node move). For example, in networks formed by cars in a highway, the cars in the intermediate position would be the best candidates for mobility "leaders". However, node mobility is not the only factor to take into account. Even more important is the traffic pattern of the nodes. There is no need to pre-calculate routes for nodes that are not going to communicate at all, whereas there maybe other nodes that may need to be contacted frequently due to their mission (coordinator, server, etc.). For the latter nodes it should be highly desirable to have routes readily available saving the network from otherwise almost certain broadcasts. By considering the values of $R_i(t)$ when computing the gain function, the SO behavior is application-driven.

Finally, it was pointed out that a RA will be created only if it is effective. For networks (or some nodes) with high mobility rate or low traffic demand it may not be effective to create them. To forward packets to those nodes route discovery will be used. Similarly, if the routes toward the destination are invalidated too quickly, or if the traffic per session is low to the point that simply flooding the packets is expected to be more effective, then flooding will be used.

Summarizing, thanks to the information provided by these structure learning/engaging modules, each node's multi-mode routing engine will have knowledge of the state of some links (the closer ones and even some links far away that are stable), as well as links towards some regions of the network (RAs) together with information regarding the location (i.e. RA membership) of some destinations. Based on this information the multi-mode routing engine may select its "mode" of operation. Possible decisions include the use of a pre-calculated path of stable links (if available and if stable links are not congested); the use of links toward the destination node's RA expecting that the packet at some point will find a node with knowledge of routes toward the destination as shown in Fig. 2 (this routing mode resembles the Landmark Routing[7, 8] philosophy); the use of a query or a broadcast packet to get the destination node's location information; or simply use a combination of route discovery/flooding.

It should be noted that although the SO algorithm, the creation of reference areas, and the specific "modes" of operation of the multi-mode routing protocol are all particular to the problem of routing, the Limited Information Dissemination (LID) and Pattern Extraction (PE) algorithms have a much broader applicability as enabling blocks for self-adaptation for different services/tasks

[3] Since this pattern extraction is the key to the development of adaptable algorithms, it will be further discussed in Sect. 4

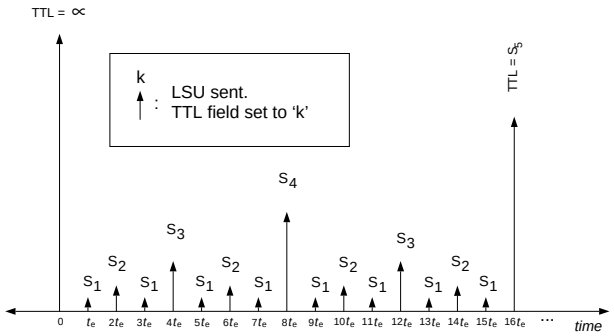

Fig. 3. LSU generation/dissemination process under FSLS

besides routing. We will discuss them in more detail - in the context of broad adaptability - in the next sections.

3 Limited Information Dissemination (LID)

From the start, from the selection of the bootstrapping mechanism to be employed, a node needs some level of awareness of global information. The most basic global information is the network size. For example, employing global advertisement as a bootstrapping mechanism may not be recommendable if the network size is too large. Subsequently, more refined pieces of (global) information may be required in order to make other decisions. The Limited Information Dissemination (LID) module takes care of providing each node with this info.

3.1 A Family of Efficient, Flexible and Adaptable LID algorithms

In our multi-mode routing approach, a general family of LID techniques referred to as Fuzzy Sighted Link State (FSLS) was employed. The general approach to information dissemination is shown in Fig. 3 (recall that the basic information-bearing element is the Link State Update, or LSU). Every t_e seconds a LSU is sent to nodes up to s_1 hops away, every $2t_e$ seconds a LSU is sent to nodes up to s_2 hops away, each $4t_e$ seconds a LSU is sent to nodes up to s_3 hops away, and so forth. The values of the $\{s_i\}$ sequence depends on the scenario. For example, for the case of routing for flat homogeneous networks, it was shown in [4] that the optimal sequence (referred to as HSLS) was $s_i = 2^i$. Not only that, but employing HSLS achieves the best scalability properties among known routing protocols. Indeed, that result showed that having different levels of information awareness (from fine grained local information to low resolution global information) pays off.

HSLS's effectiveness is due to its exploitation of the locality of the effect of most link changes. Indeed, for hop-by-hop routing, the extent of a node's decision

is limited to choosing the best next hop for a path among its one-hop neighbors. It turns out, that for large scale networks and destinations far away, the probability of making a good next hop decision is related to the angular displacement of the destination with respect to the node making the next hop decision. For example, if the node thinks that the destination is in the "North" direction, it will relay packets to this destination to its northernmost neighbor, and this decision will remain valid as long as the destination remains "on the north". Since this angular displacement depends on the ratio between node movement since last update (latency time times node speed) and the distance to this node, we conclude that the probability of making a bad next hop decision depend on the ratio between latency-of-link-state-information and distance. For uniformly distributed traffic, the best solution is obtained by keeping this probability of mistake bounded (almost constant) and this results in the HSLS schedule.

However, if the traffic distribution is not uniform, e.g. the traffic tends to be localized, different schedules can be considered. One particular schedule that is useful when the traffic is localized - as well as in order to provide pertinent information to the pattern extraction (PE) module - is the Near Sighted Link State (NSLS) schedule. In NSLS, all the event-driven LSUs are distributed only to nodes at a distance of k hops or less. That is, in NSLS $s_i = k$ for all k. These LSUs are complemented by periodic (long interval, seldom sent) global LSUs. Thus, the particular schedule to use for the $\{s_i\}$ sequence will be determined - among other things - by the range of impact of information changes (link state changes in the case of routing), the expected (average) cost of the mistakes induced by the inaccurate information, and the requirements of the PE module.

Among the pieces of information that can be extracted from a topology table filled by a LID module, even when the data is outdated, are:

- A rough estimate of the network size. The network size, which is not supposed to change frequently or dramatically, is basically needed to make the small/large network classification and decide on the methods (modes) of operation. For example, if the network is detected to be large, then global flooding should be avoided as a service advertisement mechanism.
- A Close/Far classification for each destination. This information is useful to determine the mode to engage for each.
- A Sparse/Dense classification of the network. If the network is regarded as dense, corrective actions need to be taken (e.g. topology control, use of multi-point relays, setting of parameters at the MAC layer, etc.).
- A Slow/Fast moving classification for each destination, based on the variation of its distance to other nodes over time.
- Provide the Pattern Extraction module with the information necessary to estimate the degree of association between this node and the other nodes in the network (see next section).
- For the routing service, provides the ID of a next hop towards each destination. Depending of the schedule of LSUs employed, this next hop decision may have a high probability of success.

3.2 Some Adaptation Services Enabled by a LID Mechanism

As it was explained before, LID is not limited to bearing link-state information, or to the routing service. For example, if geo-location information is available, the node position can be advertised instead of the LSU. Reactive routing protocols can then send Route REQuests (RREQ) to a specific region of the network where the destination is highly likely to be located, instead of sending the RREQ packets to the entire network. Another example is service advertisement. It was already pointed out that a service advertisement approach based on nodes sending global advertisement would not work for large networks, since they will consume most of the bandwidth in such advertisements due to the broadcast storm problem. Thus, a more sensible approach is to send local advertisement and to progressively increase the depth of propagation of such messages. This way, upon bootstrap a node will advertise its services to nodes, say, 2 hops away. It will wait some time and re-send the advertisement to nodes 4 hops away, and so forth. While doing so, since it will also be receiving advertisements from other nodes it will start to learn how big the network is and will adjust its timer/advertising schedule accordingly. Eventually, all the nodes in the network will learn of the server after a time proportional to the network size and its distance to the server, avoiding having the network collapse due to a broadcast storm during initialization.

Another example of the applicability of LID techniques can be found in the area of dynamic spectrum allocation, where a node's transmission frequency is not fixed beforehand, but it is computed on-the-fly based on sensing of the environment and the regulatory policy in effect. Such nodes are being under development as part of the DARPA's neXt Generation (XG) program[5]. Once the nodes have estimated the characteristics of the electromagnetic spectrum around them (e.g. the presence of incumbent nodes with primary rights over a particular bandwidth in that area) they will try to schedule (both in time and in frequency) their transmissions. To this end, nodes will require detailed information about nodes close by (say, up to 2 hops away) to perform distributed MAC scheduling algorithms. At the same time though, the nodes may greatly benefit from loose information about spectrum availability of nodes far away. This loose information (lower granularity, dividing the frequency in big chunks of spectrum and reporting aggregate usage over them) will be used to find "sweet spots" of global spectrum availability. These sweet spots will be the best candidates to look for a global coordination channel that all nodes are tuned to, and therefore can be used to broadcast packets to all nodes in transmission range, not only to those nodes *known* to be neighbors. Such a coordination channel is necessary to perform functionalities such as neighbor discovery/link setup and link maintenance (e.g. when a link becomes invalidated due to the arrival of a primary node with exclusive rights of usage over parts of the link's spectrum).

As we may see, LID is not only useful for routing, but it is a more general design paradigm that is neither local nor global. The basic tenet of it is that *some* global information is better than none. Designing algorithms based only on local information is equivalent to trying to find your way out of a forest by

walking (and watching) at the ground level. Using traditional centralized (or decentralized) algorithm where full global information is available is equivalent to climbing the highest peak available to figure out the way out of the forest. Using FSLS is similar to the sensible approach of climbing, from time to time, a small hill to get a sense of the surroundings and make the decision on the path to follow next, until the next hill. Note that the height of the hill that need to be climbed may also depend of the scenario (e.g. height of the trees) and therefore the FSLS must be adaptive to the scenario (feedback control loop).

We can easily see how the LID concept can be useful for a variety of autonomic systems. Take for example the case of an automatic vehicle network. A vehicle will need to exchange detailed information with vehicles close by in order to - among other things - avoid crashing with them. However, as the distance to the vehicles increases, only rough pieces of information are needed. For example, a vehicle may only be interested in the total number of nodes far away and their average speed, as to determine the likeliness of a traffic jam. Once again, the specifics of the information being propagated, the algorithms being run over the data and the best information propagation dissemination schedule is dependent on the task at hand. For flat routing, the best solution can be found in [4].

4 Pattern Extraction

Whether patterns are observable in a network or not, depends on the scale (space/time) we use to observe it. For example, let's consider a network formed by pedestrians and cars in a city. If we zoom out and see the network from the outer space, the entire network would look as a single group, well contained inside the city boundaries. On the other hand, if we look at the network from the ground, as seen by a pedestrian user with a very limited transmission range, the network would appear totally chaotic, with new neighbors appearing and disappearing. Obviously, both observations (city level and pedestrian level) would be of little use. However, a more sensible approach would be to look at the network as seen from a tall building. In this case, we could observe the different mobility patterns induced by the streets and nodes moving to similar destinations. We'll notice some cars trying to get out of the city and some trying to get in. We can group the cars according to their direction even though they momentarily move apart due to traffic conditions (traffic lights, etc.). The same observation can be made about the timescales. Cars whose trajectory may appear to have no connection while observed at a small timescale, may be discovered to be headed to a similar destination if observed over a longer period of time.

Thus, patterns are present in most networks. They are typically induced by the environment they operate in (e.g. cars on a highway, humans in a university, etc.) or the mission they fulfill (e.g. robots helping in disaster recovery operation, self-deploying sensors, etc.). In order to detect the useful patterns, we need to look beyond the one-hop neighborhood and determine the observation frequency(ies); that is, observe at the proper time/space scale.

While it may be easy to agree that patterns are present, the main question is *how hard are they to find*; that is, *will finding patterns be computationally feasible?* We answered this question for the case of routing for ad hoc networks under group mobility. We applied the associativity estimator $\hat{S}_i(A,t)$ presented in Sect. 2 to detect the group leaders in networks when group mobility is present. It should be noted that computing the estimator for a set of nodes k hops away ($k = 2$ in our experiments) did not incur significant processing overhead, since thanks to our recursive expression we basically needed to keep one place in memory for the estimator of each of the k-hop neighbors and run the update operation using the information about the nodes' current distance provided (as a by-product) by the LID module (that was computing the shortest path first tree for all the destinations). Thus, our estimator - while not optimal - showed the feasibility of detecting patterns at a low computational cost.

In Fig. 4 we show the results of applying the estimator to a 100-node network consisting of 5 groups. The estimator was used to determine the gain function (see Sect. 2 and [3]) of each node and choose the best candidates to become RNs (equivalent to cluster leaders). Ideally, the group leaders would be elected RNs and there should be as many RAs (equivalent to clusters) as mobility groups. In practice, however, estimation is not perfect. This is illustrated in Fig. 4, where the curve MoI (mobility only, ideal) represents the situation where the RNs are chosen based on the mobility pattern only, and they are chosen to be exactly the group leaders (ideal situation, where the identity of the group leaders is known beforehand). The curve MoC (mobility only, computed) represent the corresponding situation where only mobility patterns are taken into account and the estimator $\hat{S}_i(A,t)$ is used to select the best candidates to RNs. We can see that in general our estimation is not perfect and we loose some performance with respect to the ideal case, but still we obtain a good solution at a reasonable (computational) cost. It is interesting to note that there is a particular case (low mobility) when the estimator failure to properly detect the group leaders actually improves performance. This is due to the fact that during the simulation lifespan two groups were close to each other giving the impression of being only one. The estimator chose one node in the intersection of these two groups as the RN. And, since there was slow mobility, during the simulation lifespan the two groups acted as one. Choosing the wrong group leader actually helped performance. After a while, though, after the groups grew apart, there would have been a penalty for the bad selection, although it may not be big enough to counter-balance the gain from grouping the two sets of nodes together for a long period of time. This once again raises two important points: (i) timescales are important when detecting patterns, and (ii) we shouldn't loose sight that our goal/objective is not to find the groups/group leaders but to effectively deliver packets to their destinations. Our gain function captures this goal (application-driven approach). Thus, what really matters is that – at the routing protocol timescale – the nodes present a pattern that can be exploited for effective data delivery, even though in the long run the pattern may not hold.

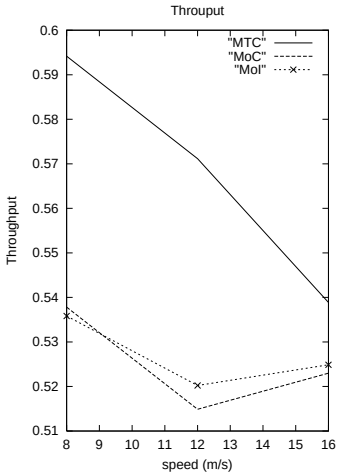

Fig. 4. Throughput for a 100 node network under different speeds using different estimators for choosing the reference nodes.

Finally, the remaining curve, MTC (mobility and traffic) represents the case when both mobility as well as traffic patterns are used. In this set of simulations, not all the nodes were destinations of active flows. Only a small group of nodes were actually involved in communications and they could successfully be grouped in two RAs. Thus, applying the application-driven self-organizing criteria that RAs are to be created only when it pays off, the SO algorithm decided to build/maintain 2 RAs only, reducing the protocol overhead and increasing the throughput. The simulation results show the evident superiority of considering the traffic patterns above mobility patterns.

The results shown in Fig. 4 refer to a particular system addressing the issue of multi-mode routing, and some specifics of the solution (gain function, reference area creation/tracking) are relevant only to the particular solution to the routing problem. The Pattern Extraction (PE) sub-module and the estimation being used, however, are much more general and applicable to a wide variety of problems. Basically, the key element to the success of the routing solution was the ability to detect associations between nodes at a bigger time scale than the immediate present and at a bigger space scale than the one-hop neighborhood. We were able to detect associations between nodes 2 or more hops away (as for example cars moving towards the same destination on a highway, where sometimes one of them passes the other and moves 2 or 3 cars away until the other ones catch up, and vice versa) and for a longer time period, of the order of the routing events. These associations, when present, form the backbone of the network. Thus, extracting patterns means to move beyond the search for stable links (one-hop associations) and start looking for stable associations at longer distances/time scales. These associations can then be exploited in a lot of different ways, depending on the task on hand. For example:

- In reactive routing techniques, we may prefer to choose paths where a subset of the nodes present strong associations between them. For example, let's assume that in the route $S-a-b-c-d-e-f-g-D$ nodes S and c present a strong association, as do nodes c and f, and nodes f and D. Thus, we may refer to nodes S, c, f, and D as "anchor" nodes since we may use them as anchors to maintain the route from S to D. For example, in case of a link breakage in the segment $S-a-b-c$, node S does not need to initiate a global repair of the route, but since node S knows that node c is likely to still be around, node S may issue a local request to built a new path to node c (instead of to the destination D). Once the path from S to c is repaired, with say a new segment $S-a'-b'-c$, the path to the destination becomes $S-a'-b'-c-d-e-f-g-D$. Thus, the use of anchor nodes (and pattern extraction) allows us to avoid global route repair localizing the effect of link breakage (a scalable approach).
- Knowledge of the underlying network patterns and backbone help to build stable structures, as for example grouping nodes in long-lived/stable clusters (or, as in our routing example, the reference areas). Reducing the instability of the structures built on top of a network (e.g., clusters) significantly reduces the overhead needed for repair/maintenance.
- Knowledge of patterns help to classify nodes according to their characteristics and to determine appropriate modes of operation for each node/region.

It can be seen that finding associations can significantly improve performance. However, it should be noted that finding these associations requires non-negligible time. This is true because, as mentioned earlier, the timescales at which we look (and care) for patterns is the same as the timescale of the application that will exploit the patterns (routing, in our experiments - case study). Thus, the convergence time for the estimator will be of the same order of magnitude of the network time scale, which is typically not small. For example, it will take a time in the order of $\Delta t/(1-\rho)$ seconds for the estimator $\hat{S}_i(A,t)$ (presented in Sect. 2) in our routing experiments to discover associations. The value of ρ cannot be too small since it this case the probability of false alarm will increase significantly and the structures formed based on the misunderstood patterns will not be stable (at the network time scale). As a consequence, initialization (initial discovery of patterns) will take a time in the order of the network timescales. Since this value is typically long, alternative techniques need to be used to provide service during this long initialization period. For example, flooding Route REQuest can be employed in the routing example to provide service to destinations until the pattern-based structures are present. Note that attempting to reduce this initial convergence time is likely to lower the quality of the estimator and decrease the network performance in the long run. Also, if the network patterns as observed at the network timescale start changing, it will take a non negligible time for the PE module to track these changes. In our experiments with networks exhibiting group mobility under realistic scenarios (speed, transmission range, etc.) corresponding to vehicles moving on the ground, it took several hundred seconds before patterns could be extracted and reference areas

could be formed. In the meantime, the multi-mode engine acted as if no pattern where present (worst case) resorting to outdated link state info or route request (depending of the destination) to deliver the packets.

5 Summary and Concluding Comments

As elaborated on earlier, network users not only demand new and versatile application support by the networks but they themselves are becoming part of the network. These users are becoming much more nomadic, diverse and autonomic, creating a fairly diverse in size and characteristics networking environment. Consequently, it is clear that future networks will increasingly:

- have ad hoc, changing and rather large structures,
- be designed for operation in diverse environments (heterogeneous) and
- consist of network nodes that will be both diverse and autonomic.

As a result, network nodes are likely to find themselves in (as well as contribute to shaping) large, ad hoc and quite diverse (LADA) networking environments. Unless carefully designed, the formed LADA networks would suffer greatly from the curse of dimensionality and diversity (and possibly an emerging one, the "curse of autonomicity") and would be fairly inefficient if functional at all. Autonomicity may be viewed as not only simply capturing a changing or variable behavior of nodes, but also autonomous behaviors that may be random or shaped by tasks, rules, policies or the environment.

Mechanisms that sense the network conditions and take decisions / adjust key parameters, have long been around (e.g., ethernet, etc). These mechanisms utilize some network state information (that they extract themselves or are being provided) and adjust their behavior / parameters. These "basic" adaptation mechanisms are completely inadequate for the large scale, ad hoc and heterogeneous LADA networking structures, composed by autonomic and diverse network nodes. Coping effectively with autonomicity, diversity and dimensionality that are inherently affecting the emerging networks requires a more comprehensive approach to adaptation than the "basic" one of the past.

Information dissemination is the cornerstone of an effective adaptable LADA network. Not only it will have to overcome the curse of dimensionality itself but also provide adequate information to enable the deployment of scalable (i.e., cope with the curse of dimensionality) and effective (multi-mode) network protocols. To facilitate the latter, the disseminated information should be sufficient to help other nodes **characterize the collectively shaped networking environment** as well as for **extracting behavioral and other patterns** (i.e., cope with the curse of autonomicity and diversity) in the network and feed adequately **self-organizing mechanisms**.

This paper has demonstrated the aforementioned advocated approach for LADA networks by applying it to large scale, adaptable, mobile, ad hoc networks. The effective operation of (autonomic) nodes in ad hoc networking environments relies strongly on their ability to adapt to it; that is, learn about

the specific environment and invoke the appropriate protocols. To enable this adaptation, a (scalable) Limited Information Dissemination algorithm is necessary to provide to the nodes local and global network information of various resolution levels. This information will be processed by the nodes, so that they detect key characteristics of the environment, possible organize themselves and finally invoke the proper protocol functionality. The aforementioned approach has been outlined (and applied successfully) for adaptive (multi-mode) routing in ad hoc networks, where the collected local and global information is processed by the Pattern Extraction and Self Organization algorithms. Other examples of adaptation services that require a Limited Information Dissemination protocol (such as dynamic spectrum allocation) are also presented (Sect. 3.2).

In addition to the framework for managing LADA networks proposed in this paper, this paper provides some ideas and algorithms for implementing specific functionalities of this framework, as well as attempts to bring out key issues that should be addressed to enable the proposed framework both for routing as well as other service support in LADA networks. Such issues include (scalable) modulation of the disseminated information in space and time to achieve diverse resolution for local and global information (LID algorithm), rules for information compression/aggregation/merging for scalable support of services other than routing, consideration of the appropriate time-scales for extracting behavioral and other patterns as well as algorithms to deliver them, rules for self-organization for scalable service provision, stability considerations of the adaptation strategies, etc.

References

1. The Joint Tactical Radio System (JTRS) program web page at http://jtrs.army.mil/
2. Santivanez, C., Stavrakakis, I.: A Framework for a Multi-mode Routing Protocol for (MANET) Networks. Proceedings of IEEE WCNC'99, New Orleans, LO, September 1999.
3. Santivanez, C.: A framework for multi-mode routing in wireless ad hoc networks: theoretical and practical aspects of scalability and dynamic adaptation to varying network size, traffic and mobility patterns. Doctoral thesis, Electrical and Computing Engineering Department, Northeastern University, Boston, MA, November 2001.
4. Santivanez, C., Ramanathan, S., Stavrakakis, I.: Making Link State Routing Scale for Ad Hoc Networks. Proceedings of MobiHOC'2001, Long Beach, CA, Oct. 2001.
5. DARPA's neXt Generation program: web page http://www.darpa.mil/ato/programs/XG/
6. Santivanez, C., McDonald, A. B., Stavrakakis I., Ramanathan, S.: On the Scalability of Ad Hoc Routing Protocols. Proceedings of IEEE Infocom'2002, New York, USA, June 2002.
7. P. F. Tsuchiya, P.F.: Landmark Routing: Architecture, Algorithms, and Issues. Technical Report MTR-87W00174, Cambridge, MA, MITRE Corporation, September 1987.
8. Pei, G., Gerla, M., Hong, X.: LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility. Proceedings of ACM Workshop on Mobile and Ad Hoc Networking and Computing MobiHOC'00, Boston, MA, August 2000.