Declarative Policy-based Adaptive MANET Routing

Changbin Liu* Ricardo Correa* Xiaozhou Li* Prithwish Basu† Boon Thau Loo* Yun Mao‡
*University of Pennsylvania †BBN Technologies ‡AT&T Labs - Research
{changbl, ricm, xiaozhou, boonloo}@seas.upenn.edu, pbasu@bbn.com, maoy@research.att.com

***Abstract*—This paper presents the design and implementation of declarative policy-based adaptive MANET routing protocols. Our work builds upon declarative networking, a recent innovation for building extensible network architectures using declarative languages. We make the following contributions. First, we demonstrate that traditional MANET protocols can be expressed in a compact fashion as declarative networks. We validate these declarative protocols via an experimental study on the ORBIT wireless testbed and a cluster-based emulation environment. Second, we demonstrate that policy-driven adaptation can be specified in a generic set of declarative rule-based policies that dictate the dynamic selection of different protocols based on network conditions. Third, we conduct extensive evaluation results of declarative policy-based adaptation of MANET routing on the ORBIT wireless testbed and the cluster-based emulation environment. Our experimental results show that the specified policies enable MANETs to dynamically hybridize a variety of routing protocols to achieve a good tradeoff in bandwidth utilization and route quality.**

I. Introduction

In the past decade there has been intense activity on the development of routing protocols for mobile ad hoc networks (MANETs). A wide variety of routing protocols have been proposed, all with their own strengths and weaknesses, and all with varying degrees of success. For example, reactive routing protocols such as DSR [8] and AODV [13] set up routing state on demand and hence are preferred for low traffic environments; proactive routing protocols such as OLSR [5] on the other hand expend network bandwidth to gather routing state with a purpose of amortizing this extra cost over multiple traffic flows – hence these are better for high traffic load environments, in general. Recently researchers have focused on the disruption tolerance aspects of MANETs that are at best intermittently connected, e.g., epidemic routing protocols.

Due to a wide range of variability in network connectivity and also a wide range of data traffic patterns, a *one-size-fits-all* MANET routing algorithm does not exist. Hybrid routing protocols attempt to address the above problem by combining features from various *pure* protocols, such as those of proactive or reactive types. While extant protocols in the hybrid category (e.g. [6], [17], [14], [7]) have systematic logic behind their design, they are still restrictive and are specified in a stove-piped manner. As a result, the *no-one-size-fits-all* argument also applies to these hybrid protocols. In reality, these protocols perform well only under certain conditions, and require additional heuristics to achieve good performance in scenarios where they are not designed for.

Ideally, one would like to have a platform to create highly customizable hybrid protocols by composing any number of known protocols, provided the policies, rules, and conditions for switching amongst them are clearly specified. To this end, we present our approach of using declarative languages to build policy-driven hybrid protocols. First, known protocols such as link state and epidemic protocol are specified in a database query style declarative language. Second, rule-based adaptation policies dictating when to use which protocols on what conditions are specified in the same language. Finally, the runtime system automatically compiles the protocols and policies into actual implementations.

The approach has the following benefits: (1) hybrid protocols written in declarative language are highly customizable, because protocols and policies are both specified in the same high-level language as first class concerns, suggesting opportunities for making finer-grained customizations on runtime adaptation; (2) the declarative framework enables quick prototyping and analysis of complex hybrid protocols in realistic environments in addition to network simulators. The protocol specifications are usually orders of magnitude shorter than the corresponding imperative implementations in languages like C/C++ or Java. Furthermore, shared protocol components can be reused and composed to create new hybrid protocols. Specifically, our contributions are as follows:

Declarative MANET routing protocol implementations: We demonstrate that MANET protocols such as various variants of Link State routing and Epidemic routing can be specified compactly as declarative networks [3], [11]. We validate these protocols by executing them on MANETs emulated on a testbed cluster and on 33 nodes (in 802.11 ad-hoc mode) on the ORBIT [1] wireless testbed.

Policy-based runtime adaptation: We demonstrate how policy-based decisions for creating hybrid protocols can be expressed in the same declarative language, and used to switch between protocols. For example, switching between different variants of link-state routing based on network dynamics and density, and switching between proactive and epidemic protocols in different parts of the network.

Experimental validation: We experimentally validate a variety of policy-based adaptive MANETs in dynamic settings on the ORBIT wireless testbed and in the cluster-based emulation environment. The results demonstrate that our approach enables MANETs to flexibly and dynamically adapt their routing protocols to achieve a good tradeoff in bandwidth utilization and route quality.

To our best knowledge, our work is one of the first attempts at evaluating a wide range of MANET protocols in combination on an actual wireless testbed. In addition to our contributions on policy-based adaptive MANETs, our work also demonstrates that declarative networking techniques can be used effectively to rapidly prototype, deploy, and compare across a variety of MANET protocols. Moreover, the declarative framework enables the ability to rapidly explore a wide range of deployment and implementation parameters necessary for tuning the performance of MANET routing protocols.

II. Background

The high level goal of *declarative networks* is to build extensible architectures that achieve a good balance of flexibility, performance and safety. Declarative networks are specified using *Network Datalog* (*NDlog*), which is a distributed recursive query language for querying networks.

Declarative queries such as *NDlog* are a natural and compact way to implement a variety of routing protocols and overlay networks. For example, traditional routing protocols such as the path vector and distance-vector protocols can be expressed in a few lines of code [3], and the Chord distributed hash table in 47 lines of code [11]. When compiled and executed, these declarative networks perform efficiently relative to imperative implementations.

NDlog is based on Datalog [15]: a Datalog program consists of a set of declarative *rules*. Each rule has the form `p :- q1, q2, ..., qn.`, which can be read informally as "`q1` and `q2` and `...` and `qn` implies `p`". Here, `p` is the *head* of the rule, and `q1, q2,...,qn` is a list of *literals* that constitutes the *body* of the rule. Literals are either *predicates* with *attributes* (which are bound to variables or constants by the query), or boolean expressions that involve function symbols (including arithmetic) applied to attributes.

Datalog rules can refer to one another in a mutually recursive fashion. The order in which the rules are presented in a program is semantically immaterial; likewise, the order predicates appear in a rule is not semantically meaningful. Commas are interpreted as logical conjunctions (*AND*). Conventionally, the names of predicates, function symbols, and constants begin with a lowercase letter, while variable names begin with an uppercase letter. Function calls are additionally prepended by `f_`. Aggregate constructs are represented as functions with attribute variables within angle brackets (<>). We illustrate *NDlog* using a simple example of two rules that computes all pairs of reachable nodes in a network:

```
r1 reachable(@S,N) :- link(@S,N).
r2 reachable(@S,D) :- link(@S,N), reachable(@N,D).
```

The rules `r1` and `r2` specify a distributed transitive closure computation, where rule `r1` computes all pairs of nodes reachable within a single hop from all input links (denoted by the `link`, and rule `r2` expresses that "if there is a link from `S` to `N`, and `N` can reach `D`, then `S` can reach `D`." The output of interest is the set of all `reachable(@S,D)` tuples, representing reachable pairs of nodes from `S` to `D`. By modifying this simple example, we can construct more complex routing protocols, such as the distance vector and path vector routing protocols.

NDlog supports a *location specifier* in each predicate, expressed with the `@` symbol followed by an attribute. This attribute is used to denote the source location of each corresponding tuple. For example, all `reachable` and `link` tuples are stored based on the `@S` address field. To support wireless broadcast, we have introduced a *broadcast location specifier* denoted by `@*` which will broadcast a tuple to all nodes within wireless range of the node where the rule is executed.

NDlog queries are compiled and executed as *distributed dataflows* by the query processor to implement various network protocols. These dataflows share a similar execution model with the Click modular router [9], which consists of elements that are connected together to implement a variety of network and flow control components. In addition, elements include database operators (such as joins, aggregation, selections, and projects) that are directly generated from the *NDlog* rules. Messages flow among dataflows executed at different nodes, resulting in updates to local tables, or query results that are returned to the mobile hosts that issued the queries. The local tables store the network state of various network protocols.

Predicates refer to tables which themselves are declared as soft-state with lifetimes. Event predicates (whose names start with an additional "e") are used to denote transient tables which are used as input to rules but not stored. For example, utilizing the built-in `periodic` keyword , node `X` periodically generates a `ePing` event every 10 seconds to its neighbor `Y` denoted in the `link(@X,Y)` predicate:

```
ePing(@Y,X) :- periodic(@X,10), link(@X,Y).
```

III. Declarative MANET Routing

In this section, we demonstrate how a variety of MANET protocols can be expressed using the declarative framework. Due to space constraints, we focus primarily on proactive protocols and one representative epidemic protocol. We have also experimented with the DSR reactive protocol. This section sets the stage for Section V where we discuss policies for hybridizing and switching between different protocols.

A. Proactive Protocols

A well studied proactive protocol is the link-state protocol, in which the entire topology is disseminated to all nodes in the network. We show first an example for network-wide flooding of link-state (LS) updates in traditional link-state, followed by two variants of link-state commonly used in MANET settings. Traditional dissemination of link state information is expressed by the following *NDlog* rules:

```
ls1 lsu(@S,S,N,C,S) :- link(@S,N,C).
ls2 lsu(@M,S,N,C,Z) :- link(@Z,M,C1),
          lsu(@Z,S,N,C,W), M!=W.
```

`lsu(@M,S,N,C,Z)` is a link state update (LSU) corresponding to `link(S,N,C)`, which indicates a link between node `S` and `N` with a cost of `C`. This LSU tuple is flooded in the network starting from source node `S`. During the flooding process, node `M` is the current node it is flooded to, while node `Z` is the node that forwarded this tuple to node `M`.

Rule `ls1` generates an `lsu` tuple for every link at each node. Rule `ls2` states that each node `Z` that receives an `lsu` tuple recursively forwards the tuple to all neighbors M except the node W that it received the tuple from. Datalog tables are set-valued, meaning that duplicate tuples are not considered for computation twice. This ensures that no similar lsu tuple is forwarded twice.

The above LS rules perform *triggered updates* continuously: whenever a link is added or deleted, a corresponding lsu is inserted or deleted locally, and then flooded to the entire network. As an alternative, one may prefer to implement link-state via *periodic updates* by modifying rule ls1 as follows:

```
ls1p lsu(@S,S,N,C,S) :- periodic(@S,10),
         link(@S,N,C).
ls2p lsu(@M,S,N,C,Z) :- link(@Z,M,C1),
         lsu(@Z,S,N,C,W), M!=W.
```

In rule ls1p we utilize the periodic keyword to flood once in every 10 seconds. In practice, a combination of triggered updates for timeliness and periodic updates for robustness are used. The declarative framework enables both approaches naturally via modifications to a single rule, demonstrating the power of declarative programming. In addition, *batched triggered updates* in which updates are batched and propagated at fixed intervals can also be concisely expressed within this framework. Once the entire network topology, i.e., all the links represented in lsu tuples, are available at each node, an additional 4 rules are required to compute the shortest paths with minimum cost C for each source S and destination D. The process is equivalent to executing the Dijkstra's algorithm locally.

Our example above utilizes unicast communication, where each link tuple results in an lsu tuple being sent via unicast to each neighbor. Using the broadcast location specifier @* described in Section II, the following rules broadcast link information to all neighbors within the wireless range of each node S:

```
ls1b lsu(@*,S,N,C,S) :- link(@S,N,C).
ls2b lsu(@*,S,N,C,Z) :- lsu(@Z,S,N,C,W).
```

Optimized Link-state Routing (OLSR): A well-known proactive MANET protocol is OLSR (Optimized Link State Protocol) [5]. OLSR ensures efficient flooding since only a subset of neighbors known as multipoint relays (MPR) keeps forwarding LSUs. The union of the neighbor sets of MPRs of any node X is equal to the set of 2-hop neighbors of X. The following rules olsr1-2 are modified from rule ls1-2 to implement OLSR-style flooding of LSUs:

```
olsr1 lsu(@S,S,N,C,S) :- periodic(@S,10),
         mprSelector(@S,N,C).
olsr2 lsu(@*,S,N,C,Z) :- lsu(@Z,S,N,C,W),
         mprSelector(@Z,M,C1), M==W.
```

The mprSelector predicate in rule olsr1-2 denotes the MPR selector tuples each storing multipoint relay selector node M for node Z (i.e. node M chooses node Z as one MPR node) . This predicate itself can be defined with additional rules to customize the definition of MPR. Compared to pure link-state protocol, in OLSR only mprSelector tuples instead of link are flooded as LSUs (as shown in rule olsr1), and only MPR nodes keep forwarding received LSUs (as shown in rule olsr2).

Hazy-sighted Link-state (HSLS): Hazy Sighted Link State routing (HSLS) [18] is a scalable LS routing variant for handling moderate to high rate of change in network topology. This protocol attempts to control the scope and frequency of its LSU flooding scheme based on the topology of the network. The basic principle of HSLS is that route calculation of a node should not be affected significantly by link dynamics due to mobility or failure in a portion of network that is far away from this node. Hence unlike the pure LS protocol which always performs network wide flood of all LSUs, HSLS sends LSUs to the 2^k hop neighbors of a node with a period equal to $2^k T_e$, where T_e is a nominal period. If link dynamics are high, pure LS starts thrashing because remote nodes could receive an LSU corresponding to a link that has long vanished.

Policy rules used in HSLS are expressed as follows:

```
hsls1 lsu(@S,S,N,C,S,TTL) :- periodic(@S,T),
         link(@S,N,C), T=f_pow(2,K)*Te,
         TTL=f_pow(2,K), K=range[1,10].
hsls2 lsu(@M,S,N,C,Z,K-1) :- lsu(@Z,S,N,C,W,K),
         link(@Z,M,C1), K>0, M!=W.
```

Rule hsls1 is periodically fired, and the period of execution depends on $2^K T_e$. Note that here we add one more attribute for lsu tuple, which is TTL used for controlling flooding scope. In declarative networking, it is easy to modify tuples, such as adding and deleting their attributes due to the need of different protocols. Rule hsls2 keeps forwarding LSUs if their TTL is larger than 0. Similar to LS, the HSLS rules can be modified to support triggered updates or batched triggered updates. If triggered updates are used, in order for all LSUs to reach every node, a periodic network-wide LSU flooding needs to be carried out based on the nominal period T_e.

B. Epidemic Protocols

Epidemic routing has been proposed for reliable delivery in intermittently connected MANETs (a class of disruption tolerant networks or DTNs). A key reliability component of such protocols is the summary vector exchange as illustrated by the rules e1-4 below:

```
e1 eBitVecReq(@Y,X,V):- summaryVec(@X,V),
         eDetectNewLink(@X,Y).
e2 eBitVecReply(@X,Y,V):- eBitVecReq(@Y,X,V1),
         summaryVec(@Y,V2),
         V=f_vec_AND(V1,f_vec_NOT(V2)).
e3 eNewMsg(@Y,I,S,D):- eBitVecReply(@X,Y,V),
         msgs(@X,I,S,D),
         f_vec_in(V,I)==true.
e4 msgs(@Y,I,S,D):- eNewMsg(@Y,I,S,D).
```

In rule e1, node X detects that a new link comes to be available, then it retrieves its local summaryVec table, consisting a bit vector where the ith bit denotes the receipt of the ith message, and then generates a eBitVecReq request to the neighbor Y connected by the new link. Upon receiving the request, node Y performs a bitwise AND operation (f_vec_AND) between the incoming summary vector V1 and the negation (f_vec_NOT) of local summary vector V2 to generate a new vector V which is sent back to X. This new vector `V` denotes messages seen by `X` but not `Y`. Rules `e3-4` then enables node `X` to filter local messages to be sent based on the bit vector `V` stored in the reply, which are then put in the local `msgs` table after transmission.

IV. Evaluation of Declarative MANET Routing

As a feasibility analysis of declarative MANETs, we perform an evaluation of three MANET protocols described in Section III: proactive LS, OLSR, and HSLS. We base our evaluation on the P2 declarative networking system [2], and validate the behavior of the declarative protocols in a cluster-based emulation environment and on the ORBIT wireless testbed [1].

A. Local Cluster Emulation

Our first set of experiments is carried out on a local cluster with 15 Pentium IV 2.8GHz PCs with 2GB RAM running Fedora Core 6 with kernel version 2.6.20, which are interconnected by high-speed Gigabit Ethernet. When running the protocols, we execute several P2 nodes on each cluster node, and together, these nodes execute declarative MANET routing protocols to compute the shortest paths in the network. In this setting, we emulate network connectivity by initializing the `link` table of each node such that every node has three neighbors. This setup allows us to validate the scalability trends (in terms of bandwidth utilization) of each protocol.

Figure 1 shows the per-node communication overhead of LS, OLSR, and HSLS protocols as the network size increases. All three protocols utilize periodic propagation of LSUs at 10 seconds interval (i.e. similar for the nominal period in HSLS). In our protocols, we measure the aggregate communication overhead required for each node to propagate its LSUs to other nodes.

We observe that the communication overhead increases linearly as the network size increases, a scalability trend that one would expect in link state protocols. Moreover, as expected, LS incurs the highest communication overhead, followed by OLSR, and HSLS. Intuitively, OLSR incurs lower communication overhead than LS since flooding is only performed via MPRs, whereas HSLS requires the least communication overhead as it sacrifices optimality for performance via the use of scoped flooding.

In addition to LS, OLSR, and HSLS, we have also implemented DSR and summary-based epidemic routing in 11 rules and 17 rules respectively. When executed in the same setup, these protocols exhibit expected protocol behavior and performance characteristics. For instance, in DSR, the per-node communication overhead increases linearly as the number of route requests increases. We notice a similar linear trend for DSR when we fix the number of queries, but increase the network size.

Overall, our results demonstrate that in a cluster-based emulation environment, declarative MANET implementations are able to achieve the same scalability trends as one may expect from the respective protocols being implemented.

B. ORBIT Wireless Testbed Evaluation

The cluster-based environment enables us to study the protocols within a controlled environment. To study declarative MANET protocols in an actual wireless environment, we evaluate two variants of link-state protocol (LS and HSLS) on the ORBIT wireless testbed [1]. The ORBIT testbed consists of machines with 1 GhZ VIA Nehemiah processors, 64KB cache, 512MB RAM, and supports two types of network adapters (Intel Pro-wireless 2915-based 802.11 a/b/g and Atheros AR5212-based 802.11 a/b/g). Nodes on the ORBIT testbed are placed a meter apart from one another in a grid and run with 1dBm transmit power.

Our evaluation is based on the P2 declarative networking system, with additional enhancements to support broadcast-based wireless communication. We utilize 33 testbed nodes within a $7m \times 5m$ grid area[1] for our experiments. Each ORBIT machine executes an instance of a P2 process. In the static network configuration, we generate random network topology, in which nodes are configured to communicate with 6-8 random neighbors. The neighborhood information is used to create the `link` table at each node. We utilize `iptables` to filter packets at the MAC layer so that each node only receives broadcasting messages from its designated neighbors.

Figure 2 shows the per-node bandwidth utilization (in KBps) measured from running the 33 nodes MANET executing either LS or HSLS protocols. By default, we configure all nodes in 802.11a ad-hoc mode with RTS/CTS for packets larger than 768 bytes and a retry number of 3. In both protocols, given our use of broadcast communication to disseminate LSUs to neighbors, RTS/CTS and retries are not invoked. We have selected 802.11a as it is less susceptible to interference on the ORBIT testbed compared to 802.11b.

The periodic flooding interval in LS is set to 30 seconds. Since not all nodes are started at the same time, the network-wide flood occurs over an interval of 20 seconds in the entire network, as shown by the per-node communication overhead peaks at 43-63KBps every 30 seconds. In contrast, HSLS incurs considerably less bandwidth due to the use of scoped flooding. Given a nominal period T_e of 30 seconds, the per-node communication overhead of LS and HSLS is 16KBps and 8KBps respectively. Moreover, our declarative HSLS implementation exhibits expected periodic scoped flooding behavior from the protocol specification. At regular intervals of 60 seconds, the per-node bandwidth utilization of HSLS peaks at 10KBps (for floods of TTL=2), whereas, at larger intervals of 120 seconds, the peaks are higher at 25 KBps. At the largest intervals, HSLS incurs the same overhead as LS for each network-wide flood. These results are consistent with what one would expect from the protocol behavior and performance of LS and HSLS.

In Section VI, we will present additional evaluation results for these protocols in a dynamic network environment and study the impact of adaptation policies on the performance of these protocols.

[1]Two of the nodes within our selected 7x5 grid were down at the time of the experiment.

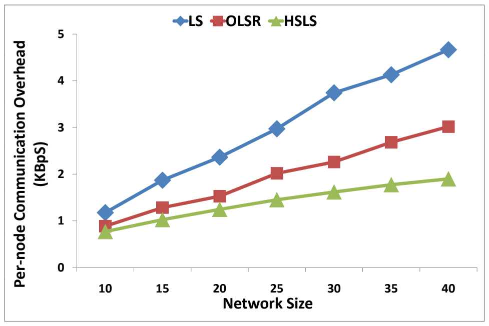

Fig. 1. Per-node Communication Overhead (KBps) for *LS*, *OLSR*, and *HSLS* as network size increases.

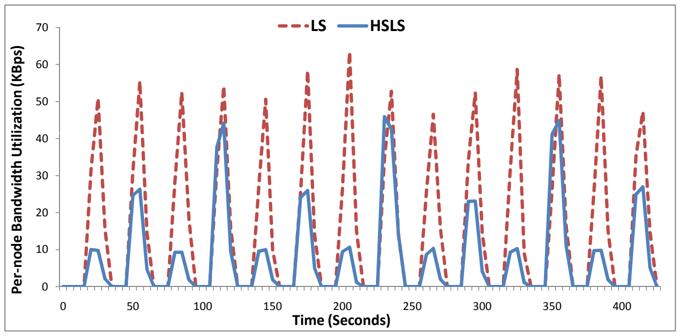

Fig. 2. Per-node Bandwidth utilization for Link-State (LS) and Hazy-Sighted LS (HSLS) on ORBIT.

V. Policy-based Adaptive MANET Routing

Building upon the basic declarative MANET protocols in the previous section, in this section, we demonstrate the construction of *adaptive hybrid* protocols by applying appropriate policies to available declarative protocol code. Such protocol behaviors are often necessary in MANETs for adapting to a wide range of network and traffic conditions. The policy-based examples described in this section are by no means exhaustive. We focus our examples and subsequent evaluation primarily on adapting a MANET protocol based on link dynamics and mobility. Other inputs to policies such as quality-of-service requirements, traffic patterns and network density are also possible and worth exploring.

Hybrid protocols attempt to combine the best features from various *pure* protocols and attempt to operate in one of the constituent modes depending on the network dynamics, traffic conditions, and other requirements such as reliability, security etc. If there were a perfect oracle that could learn about the entire network state, it would be easy to write switching rules for representing hybrid protocols in the sense that a node could decide to always run the optimal protocol that is most appropriate for the current scenario.

In the absence of this global information at every node in the network, it is still possible to write a generic set of policies that can allow run-time adaptation based on local state available at each node. However, characterizing the optimality and stability of such composite protocols in a general sense is an open and orthogonal problem. In this paper, we focus on demonstrating how hybrid composition could be facilitated with little effort using the declarative paradigm. Stability and optimality analysis is an interesting topic for future research.

A. Hybrid Link-state Protocols

The primary disadvantages of HSLS is that it sacrifices optimality in routing to the need for scalability. This is because it gathers imperfect information about the network topology and computes routes on this topology. Imperfect topology knowledge may result in computation of suboptimal routes, and this effect can be pronounced in somewhat dynamic but sparsely connected topologies.

There are a variety of metrics in which one can use to quantify the degree of mobility on a per-link basis in a network. The metric we explore is *average availability* (AA) [16], which is a metric of link to indicate its average fraction of time in that the link are available for use in the recent past. AA is calculated as the total time that a link's status is up divided by the total time since it was detected. This metric itself can be computed based on received LSUs, and is expressible using 5 declarative rules.

If the link AAs stop fluctuating wildly in most parts of the network as indicated by gathered LSUs, one may decide to switch from HSLS to pure LS routing since that may yield near-optimal routes with a lower *stretch*[2]. Moreover, a stable network will result in fewer triggered updates, ensuring that LS does not incur unnecessarily high control traffic.

Based on the HSLS rules presented earlier, one can further define a generic policy that allows us to switch between HSLS and LS based on the computed average AA of all links collected in the network. The average AA threshold below which to switch to pure LS is a configuration parameter that is set either by analysis or experimentation. This policy can be expressed by the following rules:

```
#include ls1, ls2, hsls1, hsls2
#define THRES 0.5
s1 averageLinkAvail(@M,AVG<AA>) :-
         lsu(@M,S,N,C,Z,TTL,AA).
s2 useHSLS(@M) :- averageLinkAvail(@M,AA), AA<THRES.
s3 useLS(@M) :- averageLinkAvail(@M,AA), AA>=THRES.
```

`#include` is a macro used to include earlier rules. Rule `s1` computes at node M the average AA of all links gathered from the LSUs that pass through M. Note that we add one additional attribute which is link `AA` for every LSU tuple here. Rules `s2-3` generate `useHSLS` and `useLS` predicates which are then added to rules `hsls1` and `ls1`, respectively.

Any form of protocol adaptation requires extensive experimentation and tuning, especially under a large number of network variables. A declarative protocol design is much more suited for that style than traditional protocol design due to its conciseness. To encode a new policy (e.g. use LS instead of HSLS when the network is sparse with high frequency of link updates), one only needs to modify the above rules to generate

[2]Given the source and the destination of a routing request, the stretch of the route is the ratio of the hop count of the path selected by the routing algorithm to that of the optimal path given by the oracle with complete and instantaneous knowledge of the entire network topology.

`useHSLS` and `useLS` without having to change the rules for the individual protocols themselves. Several metrics (in addition to link AA) can be easily composed declaratively. For instance, network density can be computed based on a rule that uses a combination of group-bys and aggregates (counts and average) across all links. The main point is not whether one policy is superior to another, but rather that the declarative framework makes such policy specifications concise and flexible.

B. *Hybrid Proactive-Epidemic*

As an alternative example, one can utilize a hybrid proactive-epidemic protocol, useful in a disruption-tolerant setting. This hybrid protocol switches between two modes of operation: (1) single path LS message forwarding in well connected parts of the network under low mobility, and (2) multi-path epidemic style message flooding in disrupted parts of the network under high mobility. In the simplest form, one can utilize the average link AA (presented in Section V-A) to decide at each node whether to use LS or epidemic based on a threshold. This policy is expressed using the following rules:

```
#include ls1, ls2, e1, e2, e3, e4
#define THRES 0.5
le1 averageLinkAvail(@M,AVG<AA>) :-
         lsu(@M,S,N,C,Z,TTL,AA).
le2 useEpidemic(@M,S,D) :- averageLinkAvail(@M,AA),
         message(@M,S,D), AA<THRES.
le3 useLS(@M,S,D) :- averageLinkAvail(@M,AA),
         message(@M,S,D), AA>=THRES.
```

In the above rules, rule `s1` computes the average link AA, which is then used by rules `s2-s3` to determine whether to use epidemic-based routing or LS accordingly. Interestingly, this policy requires that all nodes receive up-to-date LSUs – a challenge particularly in cases where the network may experience momentary partitions. One solution that we have explored (and validated successfully in later experimental section) is the use of epidemic flooding of the LSUs themselves.

The Anxiety-Prone Link-State (APLS) protocol [14] is a more sophisticated version of a proactive-epidemic policy that takes into account the end-to-end *cumulative AA* along a path from source to destination. The following set of rules demonstrate the ease at which these policies can be implemented:

```
#include ls1, ls2, e1, e2, e3, e4
#include bp1, bp2, bp3, bp4
#define THRES 1.2
pe1 useEpidemic(@M,S,D) :- bestPath(@M,S,D,P,C),
         C>THRES.
pe2 useLS(@M,S,D) :- bestPath(@M,S,D,P,C),
         C<=THRES.
```

In the rules above, at any node M, `bestPath` computes the cost of the shortest path between S and D (rules `bp1-4` is for computing best paths), which is then used by rules `pe1-2` to determine whether to use LS or epidemic routing. Intuitively, low values of path cost indicate that S and D are in a connected component whereas high values indicate that link availability is low, or LSU information is unavailable or stale, hence epidemic protocol is desired under those circumstances to improve delivery probability.

Note that while our first example illustrates switching the underlying dissemination scheme, the latter illustrates how to switch the route computation and forwarding. The declarative framework and compact specifications make it easy to write other (more intelligent) switching rules in this scenario.

VI. Evaluation of Policy-based MANET Routing

In this section, we perform an experimental evaluation to validate the benefits of policy-based adaptation via declarative networking. We utilize a combination of measurements obtained from a deployment on the ORBIT wireless testbed and emulation within a local cluster environment.

A. *ORBIT Experimental Setup*

We utilize the same ORBIT testbed described in Section IV-B, with additional mechanisms to emulate mobility. Given that nodes on the ORBIT testbed are static, we emulate random waypoint mobility as follows: each ORBIT node is initially assigned an x-y coordinate which is then updated based on the random waypoint model. Neighbors are then determined based on nodes that are at most 1.5 meters away in Cartesian distance based on the most recent coordinates. Similar to the earlier setup, we utilize `iptables` to filter packets at the MAC layer so that each node only receives broadcasting messages from its designated neighbors. The filters are updated as each node's neighborhood information is modified.

This approach enables one to dynamically adjust neighborhood information on ORBIT even though the nodes are physically static. This flexibility however comes at the expense of the increased likelihood of transmission collision (and hence dropped frames) since each node's neighbors may be not the ones that are physically closest on the grid. In all variants of link-state routing, to reduce the likelihood of collisions, we de-synchronize the time at which all nodes flood LSUs by spacing out the starting time of nodes. This reduces the peak bandwidth utilization when all nodes are sending LSUs to all other nodes. To reduce packet interference caused by potential simultaneous broadcast of LSUs, we add a random jitter of 0 to 0.1 seconds to every broadcast.

B. *Hybrid Link-state Protocol*

Our first evaluation study involves *Hybrid-LS*, a policy-driven link-state protocol described in Section V-A. *Hybrid-LS* utilizes traditional *LS* routing when the network is stable, and *HSLS* when unstable. We compare *Hybrid-LS* with two declarative implementations of link-state routing presented earlier: *LS* and *HSLS*.

In all link-state protocols, we utilize *batched triggered updates* introduced in Section III-A, which ensures that link updates are batched and propagated periodically. In other words, each node periodically batches up all link updates in the previous period, and sends out the corresponding LSUs. In our experiments, we set the propagation period to be 1 second, corresponding to the flooding period in *LS* and the nominal period T_e in *HSLS*. Interestingly, we observe from our ORBIT experiments that the choice of the propagation interval has a significant impact on packet loss (and hence route quality). While larger intervals potentially benefit from batching by packing several LSUs within a packet, sending many LSUs in a single batch may result in increased collision in the network.

In addition to periodic batched updates, there is network-wide refresh of link-state information performed periodically at intervals of 60 seconds. Correspondingly, LSUs that are not refreshed will time-out after 65 seconds. Given the triggered updates, this network-wide refresh is not strictly necessary. However, this ensures that all nodes eventually learn about the network topology via a soft-state refresh mechanism. This is particularly useful given that transient network partitions result in LSU packet losses during triggered updates. In addition, for *HSLS*, its scoped flooding prevents the protocol from learning about the network-wide topology necessary for computing average link AA (Section V-A). We note that our use of declarative programming ensures that selecting the flooding mechanism (periodic network-wide flood vs batched triggered) and tuning propagation intervals can be easily achieved by adding/modifying a few *NDlog* rules.

Mobility setup: To explore extremes in mobility patterns, we alternate at 60 seconds interval between three degrees of mobility using the random waypoint model: *stationary* stage in which all nodes stay in their current positions, *moderate* stage, in which nodes move at a moderate speed of 0.06 m/s, and *fast* stage, in which nodes move at 0.15 m/s. Figure 3 shows the mobility setup in terms of link events per second, where each event corresponds to an update (insertion followed by deletion) to the `link` table at the node whose neighborhood has been updated. On average, each node has 6-8 neighbors. We note that at moderate speed (the 2^{nd} and 4^{th} intervals), link events occur at a frequency of 6.5 events/second on average in the entire network. At fast speed (final interval), link events increase to 16.5 events/seconds.

Average link AA: The *link average availability (AA)* metric (first introduced in Section V-A) reflects the stability of each link, expressed as the fraction of time a link has been up in the recent past. Given that each node receives LSU updates from the network, each node can then compute the *network-wide* average availability, by averaging across all link AAs. The higher the average network-wide AA, the more stable the overall network is.

Figure 4 shows the average link AA (computed by averaging across the individual averages computed at each node) and validates that our mobility model is behaving as expected, and that the LSU propagation (and subsequent AA computations at each node) is able to reflect the current state of the network. We observe that when the network is stable, the average network-wide link AA is 1, and at periods of moderate speed, the average link AA can drop to as low as 0.88, only to recover to 1 when the network is stable again. At fast speed, we observe that link AA drops to 0.65. While the link AA is consistent across all three protocols, we observe that link AA is lower for *LS* under high mobility. This is due to the high frequency of collisions caused by excessive flooding of LSUs when *LS* is used. We will revisit this phenomenon when comparing packet losses across all three protocols.

Comparing *LS* and *HSLS*. One of the main advantages of *Hybrid-LS* over traditional *LS* and *HSLS* is that it attempts to find a good balance between communication overhead induced by LSUs and route quality. In the remaining of this section, we try to quantify these tradeoffs by first comparing *LS* and *HSLS* in terms of their communication overhead and route quality, and then contrast their performance against *Hybrid-LS*. Figures 5-7 compare *LS* and *HSLS* based on the per-node bandwidth utilization (KBps), and two measures of route quality (*stretch* and *validity*). The respective averages are summarized in Table I in the first and second columns.

Figure 5 shows per-node bandwidth utilization (KBps) obtained via *tcpdump* for *LS* and *HSLS* during the corresponding experimental run. Not surprisingly, *LS* incurs higher communication overhead compared to *HSLS*, averaging at 79.6KBps compared to 48.0KBps for *HSLS*.

While *LS* consumes more bandwidth compared to *HSLS*, it is able to compute higher quality routes. We consider two well-known notions of route quality: *route stretch* as defined in Section V-A, and *route validity*, a route is *valid* if at the time it is computed from the local LSUs, all the links that comprise the route are up.

Figure 6 compares the protocols by comparing the average stretch of the 5% longest routes. We choose to plot the stretch for 5% longest routes because stretch is typically a greater concern for longer routes, whereas routes that have low hop-counts are less impacted by stretch. We make the following observations. First, a perfect stretch of 1 is typically only achievable when all nodes are stationary. Second, at moderate speed, *LS* results in routes that are of lower stretch, rarely exceeding an average route stretch of 1.1, compared to *HSLS* which achieves an average as high as 1.35. In the *stationary* stage , both protocols eventually converge to a stretch of 1, though *LS* is able to recover much quickly. At high speed, we observe that the stretch of *LS* and *HSLS* is roughly equivalent, average at 1.28 and 1.36 respectively.

We make a similar observation for the route validity metric, as shown in Figure 7, where *LS* is able to achieve higher percentage of valid routes at moderate speed, and equivalent percentage of valid routes at high speed. Overall, our results indicate that at moderate speed, *LS* is a desirable protocol since it is able to compute high quality routes at relatively low bandwidth utilization. At high speed, the use of *LS* becomes counter-productive, as network-wide floods under churn result in high communication overhead, which significantly degrades the successful delivery of LSUs necessary for maintaining up-to-date routes. In this case, the *HSLS* protocol is far more desirable given that it achieves routes of equivalent quality with lower communication overhead and significantly higher packet delivery rates.

Benefits of *Hybrid-LS*: To validate that the specified policies of *Hybrid-LS* can indeed adapt between *LS* and *HSLS* based on computed network-wide average link AA, we perform a similar evaluation study on the ORBIT testbed by measuring the performance characteristics of *Hybrid-LS*. In this protocol, the specified policy sets the $THRES$ parameter described in Section V-A to 0.80. $THRES$ can be tuned either experi-

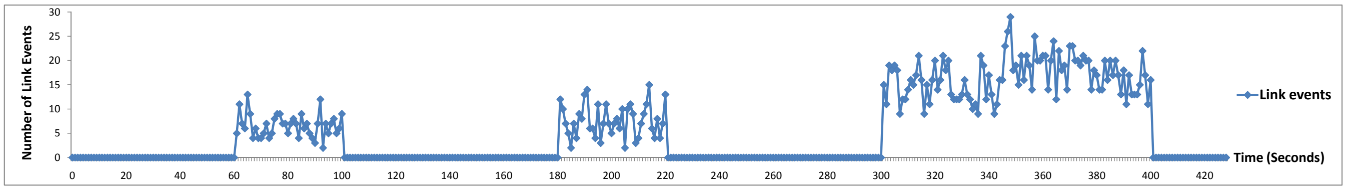

Fig. 3. Number of link events per second, with alternating 60 seconds periods of stability and mobility.

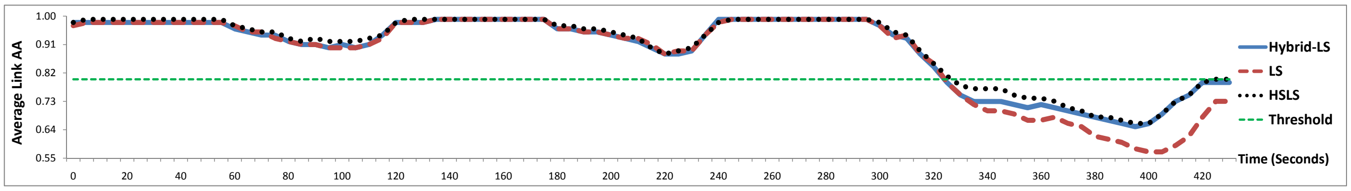

Fig. 4. Average network-wide link AA computed from LSUs for *LS*, *HSLS* and *Hybrid-LS*. The green line indicates the switching threshold $THRES$ by the adaptation policy.

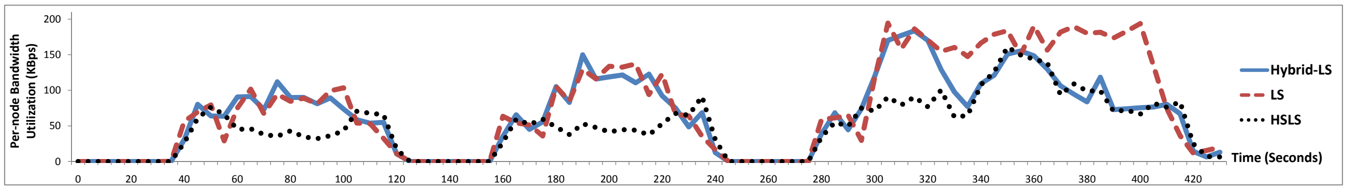

Fig. 5. Per-node communication overhead (KBps) for *LS*, *HSLS* and *Hybrid-LS*.

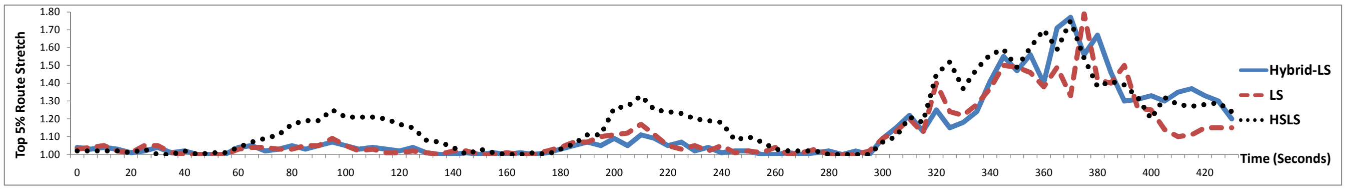

Fig. 6. Route stretch for top 5% longest routes for *LS*, *HSLS* and *Hybrid-LS*.

Mobility	Performance	LS	HSLS	Hybrid-LS
High	BW (KBps)	146.8	94.3	116.8
	Stretch	1.28	1.36	1.31
	Validity	38.4%	38.2%	42.5%
Moderate	BW (KBps)	79.6	48.0	81.8
	Stretch	1.07	1.17	1.05
	Validity	81.4%	74.2%	81.8%

TABLE I
PERFORMANCE COMPARISONS (BANDWIDTH UTILIZATION FOR TRIGGERED UPDATES, AVERAGE ROUTE STRETCH AND VALIDITY) AMONG *LS*, *HSLS* AND *Hybrid-LS* DURING MODERATE AND HIGH MOBILITY.

mentally or by analysis, and adaptive tuning of this value is an interesting avenue for future work.

Figure 8 shows the effects of this policy, by measuring the percentage of nodes using *HSLS* during the experiment. Not unexpected, *LS* and *HSLS* are insensitive to changing AA values, since they are not policy driven. On the other hand, we observe that in *Hybrid-LS*, nodes are able to quickly and successfully adapt according to the policy. As AA is above the threshold (shown in Figure 4), all nodes adapt their protocol to utilize LS. Conversely, after time 325 seconds, as the network becomes unstable, nodes start adapting to using *HSLS*. Since there is a delay in the arrival of LSUs at different nodes, some nodes are slower than others to adapt to the new protocol. Eventually, all nodes adapt to using the *HSLS* protocol. At the end of the experiment, as the network stabilizes, nodes begin to re-adapts to using *LS*. Overall, we note that *Hybrid-LS* is able to effectively adapt based on network conditions.

The third column of Table I summarizes the performance characteristics of *Hybrid-LS* in comparison with *LS* and *HSLS*. We note that *Hybrid-LS* is able to achieve a good balance between communication overhead and route quality. In periods where the network is in moderate mobility, *Hybrid-LS* adapts to a more aggressive flooding strategy used by *LS*. Hence, the protocol is able to achieve equivalent route quality compared to *LS*, and higher route quality compared to *HSLS*. For example, the average and peak route stretch of *Hybrid-LS* is 1.05 and 1.11 respectively. In contrast, *HSLS* results in an average and peak route stretch of 1.17 and 1.33.

In periods of high mobility, *Hybrid-LS* adapts to using the *HSLS* protocol. As a result, it is able to achieve equivalent route quality compared to *LS* and *HSLS*, while utilizing only a

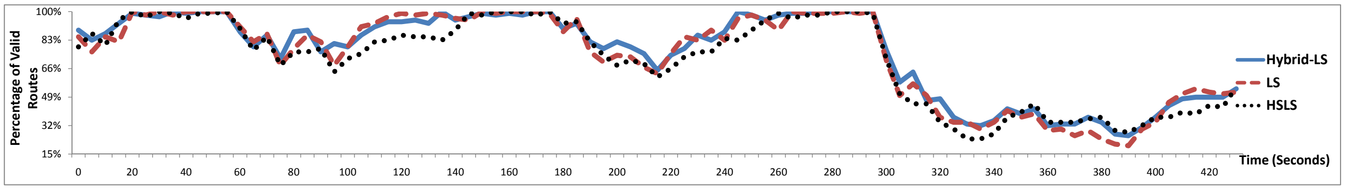

Fig. 7. Percentage of valid routes for *LS*, *HSLS* and *Hybrid-LS*.

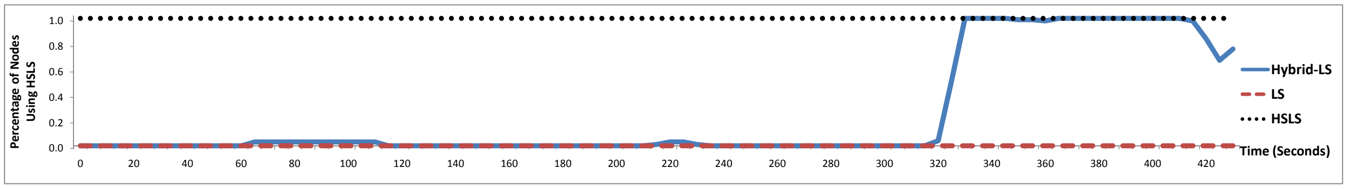

Fig. 8. Percentage of nodes using HSLS for *LS*, *HSLS* and *Hybrid-LS*.

fraction of the bandwidth that *LS* requires. By utilizing scoped flooding, we observe that the packet loss ratios of both *HSLS* and *Hybrid-LS* are significantly lower (12.2%) compared to *LS* (68.9%).

All in all, our results demonstrate that *Hybrid-LS* is able to achieve the desired protocols of both protocols given the current network conditions, by leveraging *LS*'s capabilities to achieve high quality routes under moderate mobility, and adapting to *HSLS*'s lower bandwidth utilization and higher packet delivery ratios at extreme rates of mobility.

C. Hybrid Proactive-Epidemic

Our next experiment evaluates another hybrid protocol, one that combines proactive routing and epidemic routing, as described in Section V-B. Due to the high bandwidth utilization of the store-and-forward mechanisms used in epidemic protocols, emulating mobility on the Orbit testbed in a similar fashion as our earlier experiments results in high packet losses and is hence infeasible. We instead carry out this experiment using the cluster-based emulation environment described in Section IV-A. We emulate a 35-node network, where each node is an instance of a P2 declarative networking engine executing different MANET rules and policies. Each one of these 35 nodes runs an instance of the P2 system, and is deployed on one of the 15 physical machines in the cluster (i.e. each physical machine executes 2-3 P2 instances).

In our cluster emulation, random waypoint model is also deployed. The 35 nodes are initially randomly placed within a $7m \times 5m$ arena, where we alternate at 60 seconds time intervals between moderate mobility (random waypoint with a speed of 0.03 m/s) and stability (speed of 0.001 m/s) in the network. In order to create an intermittently disconnected environment, the neighbor distance is reduced to 1.2 meters during the moderate mobility interval.

The hybrid policy *Hybrid-Epi* depends on a similar network-wide average link AA computed from LSUs. When AA threshold is below 0.9, epidemic protocol is utilized. Otherwise, traditional link-state is used. For comparison purposes, we have included also the baselines of *LS* and *Epidemic*. In periods of low connectivity with moderate mobility where nodes are moving at a speed of 0.03m/s, in order to get timely updates to link AAs, these three protocols all utilize epidemic forwarding of LSUs introduced in Section V-B. To measure packet delivery ratio, messages are forwarded from random sources to destinations every 5 seconds.

Comparing *LS* and *Epidemic*. Table II provides a summary of the average bandwidth consumption and average message delivery ratio in periods of low and high connectivity. Given that our setup involves nodes moving at a higher speed during periods of low connectivity, the bandwidth utilization of LSU dissemination is higher across all protocols during high connectivity periods. A packet is considered successfully delivered if it can reach its destination within a reasonable latency (set to 65 seconds). We make the following observations for the results. First, in periods of low connectivity, *LS* (3.2KBps) incurs far less traffic than *Epidemic* (14.8KBps). However, this comes at the expense of a lower message delivery rate (80.1% as opposed to 100%). This is also reflected in the message delivery ratio, where *Epidemic* is able to maintain high message delivery ratio regardless of network mobility. In periods of high connectivity with low mobility, *LS* clearly benefits from having low communication overhead (0.25KBps, primarily dominated by LSU updates) while achieving a high message delivery ratio of 99.4%. On the other hand, *Epidemic* achieves equivalent message delivery ratio at a much higher overhead in communication (8.3KBps).

Benefits of *Hybrid-Epi*. As summarized in Table II, the policies of *Hybrid-Epi* are adapted in order to achieve a good balance between communication overhead and packet delivery ratio. In periods of low connectivity, in order to ensure high packet delivery in the disconnected network, *Hybrid-Epi* utilizes an epidemic-style forwarding strategy, hence ensuring high packet delivery ratio (100%). However, in periods of high connectivity, because *Hybrid-Epi* switches to *LS* routing, it is still able to achieve high packet delivery ratio (97.5%) while maintaining lower overheads (0.24KBps). Interestingly, we observe that in the process of switching from one protocol to another, *Hybrid-Epi* incurs some packet losses (resulting in

2-3% reduction in average delivery ratios). To alleviate such losses, one possible policy extension that we are exploring is to for nodes to continue using *Epidemic* for an additional delay as their neighboring nodes adapt to use the *LS* protocol.

Overall, *Hybrid-Epi* achieves the best features of both protocols, by adapting between two protocols based on network conditions. In terms of packet delivery ratio, it generally has equivalent performance compared to *Epidemic*, but is able to avoid incurring the communication overhead in periods of high connectivity that is observed in *Epidemic*.

Connectivity	**Performance**	**LS**	**Epidemic**	**Hybrid-Epi**
Low	**BW (KBps)**	3.2	14.8	14.9
	Delivery Ratio	80.1%	100%	100%
High	**BW (KBps)**	0.25	8.3	0.24
	Delivery Ratio	99.4%	100%	97.5%

TABLE II
AVERAGE BANDWIDTH UTILIZATION (KBPS) AND MESSAGE DELIVERY RATIO IN PERIODS OF LOW AND HIGH CONNECTIVITY RESPECTIVELY.

VII. Related Work

Of particular relevance to our work is the extensive literature on adaptive MANET routing protocols. These include hybrid protocols such as Zone Routing Protocol (ZRP) [6], SHARP [17] and ZHLS [7]. The declarative framework presented in this paper does not preclude such hybrid protocols. In fact, the flexibility provided by declarative networking enables one to better explore the design space of such hybrid protocols.

Prior to this paper, declarative networking has been studied primarily in wired environments, such as IP routing [3] and overlay network construction [11]. Recent work [4] has also demonstrated the feasibility of using declarative techniques to program sensor network protocols. The MANET settings present new challenges posed by the presence of mobility in the network. In addition, the variability of wireless environment presents compelling motivation for the use of declarative framework for synthesizing a variety of protocols, and expressing policy decisions that enable one to adaptively select and compose protocols at runtime.

Reference [10] first proposes the use of declarative programming to prototype and adapt MANET routing protocols. This paper realizes the vision with a detailed design, analysis and implementation on the ORBIT wireless testbed and cluster-based emulation. We additionally flesh out several practical issues on deploying MANETs, and presents scenarios on policy-based adaptation which are validated via realistic experimentation on the ORBIT testbed. Finally, we have proposed and implemented extensions to the declarative networking language and runtime system in order to support broadcast-based wireless communication.

VIII. Conclusion

In this paper, we present a declarative framework on adaptive MANET routing protocols. We demonstrate that a variety of MANET routing protocols can be specified tersely using the *NDlog* declarative networking language. Furthermore, policy-based decisions can be expressed within the same declarative framework for runtime switching amongst protocols to create hybrid adaptive protocols. We experimentally validate a variety of declarative MANET routing protocols and policy-based adaptive protocols on the ORBIT wireless testbed and cluster-based emulation environment.

Our immediate steps include further experimentation on the ORBIT wireless testbed as well as emulation software [14] that enables us to evaluate our system in a variety of network settings. We are particularly experimenting with more complex policies that incorporate other metrics such as network density and traffic patterns.

We are currently in the process of developing a *declarative network simulator*, realized by integrating a declarative networking engine with the ns-3 [12] network simulator. Building upon our MANET deployment experiences on the ORBIT testbed, we are also exploring releasing a declarative MANET routing toolkit for use on the ORBIT testbed for rapidly prototyping, comparing and adapting across a wide range of MANET protocols.

IX. Acknowledgments

This work is supported in part by NSF grants CCF-0820208, CNS-0721845, CNS-0831376, and CNS-0845552.

References

[1] Orbit - wireless network testbed. http://www.orbit-lab.org/.
[2] P2: Declarative Networking. http://p2.cs.berkeley.edu.
[3] Boon Thau Loo et. al. Declarative Routing: Extensible Routing with Declarative Queries. In *SIGCOMM*, 2005.
[4] D. Chu, L. Popa, A. Tavakoli, J. M. Hellerstein, P. Levis, S. Shenker, and I. Stoica. The design and implementation of a declarative sensor network system. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 175–188, New York, NY, USA, 2007. ACM.
[5] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr). In *RFC 3626 (Experimental)*, October 2003.
[6] Z. J. Haas. A New Routing Protocol for the Reconfigurable Wireless Networks. In *IEEE Int. Conf. on Universal Personal Comms.*, 1997.
[7] M. Joa-Ng and I.-T. Lu. Peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1415–1425, 1999.
[8] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.
[9] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click Modular Router. *ACM TOCS*, 18(3), 2000.
[10] C. Liu, Y. Mao, M. Oprea, P. Basu, and B. T. Loo. A declarative perspective on adaptive manet routing. In *PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, pages 63–68, New York, NY, USA, 2008. ACM.
[11] B. T. Loo, T. Condie, J. M. Hellerstein, P. Maniatis, T. Roscoe, and I. Stoica. Implementing Declarative Overlays. In *SOSP*, 2005.
[12] Network Simulator 3. http://www.nsnam.org/.
[13] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999.
[14] Rajesh Krishnan et. al. The spindle disruption-tolerant networking system. In *MILCOM*, 2007.
[15] R. Ramakrishnan and J. D. Ullman. A Survey of Research on Deductive Database Systems. *Journal of Logic Programming*, 23(2), 1993.
[16] R. Ramanathan, R. Hansen, P. Basu, R. Rosales-Hain, and R. Krishnan. Prioritized epidemic routing for opportunistic networks. In *ACM MobiOpp '07*, pages 62–66, San Juan, Puerto Rico, 2007.
[17] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer. SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks. In *ACM MobiHoc*, 2003.
[18] C. Santivanez, R. Ramanathan, and I. Stavrakakis. Making link-state routing scale for ad hoc networks. In *ACM MobiHoc*, 2001.